A project report on

**COLLEGE MAILING SYSTEM**

**In**

**COMPUTER SCIENCE ENGINEERING**

Submitted By

**JASTI SREEPURNA(11W91A0575)**

**GUJJULA SAIRAM (11W91A0573)**

**VODNALA ASHWIN KUMAR (11W91A05B6)**

Under the Guidance of

**Mrs.E.SOUMYA**

(Asst.Prof CSE & IT DEPT)

****

DEPARTMENT OF COMPUTER SCIENCE ENGINEERIN

**MALLAREDDY INSTITUTE OF ENGINEERING & TECHNOLOGY**

(Affiliated to Jawaharlal Nehru Technological University)

2014 – 2015

**MALLAREDDY INSTITUTE OF ENGINEERING & TECHNOLOGY**

**DEPARTMENT OF COMPUTER SCIENCE ENGINEERING**

****

**CERTIFICATE**

This is to certify that the project report titled **COLLEGE MAILING SYSTEM** is being submitted by Jasti Sreepurna (11W91A0575), Gujjula Sairam (11W91A0573), Vodnala Ashwin kumar (11W91A05B6) in  **B.Tech IV I** **semester Computer Science and Engineering** is a record bonafide work carried out by them. The results embodied in this report have not been submitted to any other University for the award of any degree.

**Internal Guide Head of Department**

**External**

**DECLARATION**

I hereby declare that the project entitled **“COLLEGE MAILING SYSTEM”** submitted to Malla Reddy Institute Of Engineering and Technology, affiliated to Jawaharlal Nehru Technological University Hyderabad (JNTUH), for the award of the degree of Bachelor of Technology in Computer Science & Engineering is a result of original industrial oriented project done by us.

It is further declared that the project report or any part thereof has not been previously submitted to any University or Institute for the award of degree or diploma

**VODNALA ASHWIN KUMAR**

**(11W91A05B6)**

**CONTENTS**

Abstract

List of Figures

List of Tables

List of Screens

Symbols & Abbreviations

**1. INTRODUCTION…………………………………………………………** 1

1.1 Motivation 1

1.2 Problem definition 1

1.3 Objective of Project 1

1.4 Limitations of Project 1

1.5 Organization of Documentation 1

**2. LITERATURE SURVEY …………………………………………………2-3**

2.1 Introduction 2

2.2 Existing System 2

2.3 Disadvantages of Existing system 2

2.4 Proposed System 3

2.5 Conclusion 3

**3. ANALYSIS…………………………………………………………………4-11**

3.1 Introduction 4-10

3.2 Software Requirement Specification 10-11

3.2.1 User requirement 10

3.2.2 Software requirement 10

3.2.3 Hardware requirement 11

3.3 Conclusion 11

**4. DESIGN…………………………………………………………………..12-42**

4.1 Introduction 12

4.2 DFD / ER / UML diagram (any other project diagrams) 13-41

4.3 Module design and organization 42

4.4 Conclusion 42

**5. IMPLEMENTATION & RESULTS…………………………………….43-99**

5.1 Introduction 43-44

5.2 Sample Code 44-68

5.3 Explanation Of Key Functions 68-88

5.4 Method of Implementation 89-99

5.4.1 Tables 89-94

5.4.2 Output Screens 94-98

5.4 Conclusion 99

**6. TESTING & VALIDATION ……………………………………………100-110**

6.1 Introduction 100

6.2 Design of test cases and scenarios 101-106

6.3 Validation 107-109

6.4 Conclusion 110

**7. CONCLUSION……………………………………………………………111**

**REFERENCES………………………………………………………………112**

**ABSTRACT**

The Gmail Client which we have developed is basically fulfills the needs of college system in terms of assistance, sending notices to the college staffs, students, sending the daily updates to the people of the college, sending the results to the students which can be internal exams or external exams results through our mailing system. One may think we have so many kinds of mailing system like yahoo mailing, gmail mailing and like so, but it lacks any kind of central tracking or giving the control to someone who may be head from any kind of departments, or colleges, like principal or administrator, if using gmail or other cant get the complete control on others emails account as because if using this existing mailing system than the principal or administer also has to become one of the member of this mailing accounts. So that’s means being the member itself cant have control on other members accounts.

This central controlling of mailing system will be provided by our Gmail Client wherein we have divided mailing system on the based on two kinds of users , one is Administrator and the other one is user. Administrator will have complete control on users accounts(other than administrator), complete control in the sense like administrator can change the password of the users if he wants to restrict them from login into our mailing system and like so. The one of the important feature of our Gmail Client is that intramailing interaction can be done in an easier and in a convenient way.

**LIST OF FIGURES**.

* Waterfall Model
* Prototyping Methodology
* Use Case Diagram For Admin
* Use Case Diagram For User
* Sequence Diagram for Admin
* Sequence Diagram for User
* Class Diagram for Admin
* Class Diagram for user
* Collaboration Diagram for User
* Collaboration Diagram for Admin
* State Chart Diagram
* Activity Diagram for User
* Activity Diagram for Admin
* Component Diagram
* Deployment Diagram
* Object Diagram
* ER Diagram
* DFD Diagrams

**LIST OF TABLES**

1. Contact database table
2. Draft database table
3. Mail attachment database table
4. User mail database table
5. User registration database table

**ABBREVATIONS**

DFD : Data Flow Diagrams

UML : Unified Modelling Language

ADO.NET : ActiveX Data Object .Net

ASP.NET : Active Server Page.Net

**INTRODUCTION**

**1.INTRODUCTION**

**1.1MOTIVATION**

Providing the assistance to the college members such as sending notices to the college staffs, students, sending the daily updates to the people of the college, sending the results to the students which can be internal exams or external exams results is a great task. So this could be easily achieved through the mailing system in a secured way where each person could receive the mails.

* 1. **PROBLEM DEFINITION**

At present situations sending notices to each and every member of the college is very complex.All educational institutions are following the same procedure which includes the usage of papers to send notices. This has many drawbacks. But, in this is an online-era, where everything is done in a feasible manner.

**1.3 OBJECTIVE OF PROJECT**

The main Objective of our project college mailing system is to enhance the features of existing mailing system and utilize the existing system features such as mailing communication and also to import new features such as central control of our website by the admin of the organization such as company, college etc.

**1.4 LIMITATIONS OF PROJECT**

Study chats can be included relating to the subjects by the students and the professors.

**1.5 ORGANISATION OF DOCUMENTATION**

In this project documentation we have initially put the definition and objective of the project as well as the design of the project which is followed by the implementation and testing phases. Finally the project has been concluded successfully and also the future enhancements of the project were given in this documentation.

**LITERATURE SURVEY**

**2. LITERATURE SURVEY**

**2.1 INTRODUCTION**

The Gmail Client which we have developed is basically fulfills the needs of college system in terms of assistance, sending notices to the college staffs, students, sending the daily updates to the people of the college, sending the results to the students which can be internal exams or external exams results through our mailing system.

College Mailing System consists kinds of users, one is Administrator and the other one is user. Administrator will have complete control on users accounts(other than administrator), complete control in the sense like administrator can change the password of the users if he wants to restrict them from login into our mailing system and like so. The one of the important feature of our Gmail Client is that intranet mailing interaction can be done in an easier and in a convenient way.

**2.1 EXISTING SYSTEM**

Existing mailing System like gmail, yahoo mail and so, can be used within any department or college wherein all the members can pass outs mails to each other but the only thing is the lack of control on others members mail accounts by the head person in the department who also be the member. Being the head , he/she should access the other mail accounts as they have rights to access their accounts, as part of controlling the accounts. For instance head of the college or administrator using the existing system can send to the members of the existing system but cant have control on others accounts , as being the member itself cant get permission to view others mails accounts and control those accounts.

**2.2 DISADVANTAGES OF EXISTING SYSTEM**

As mentioned above there is lack of control on the others members mail accounts by the head person in the department who is also a member. This has many drawbacks in furture.

**2.3 PROPOSED SYSTEM**

This central controlling of mailing system will be provided by our Gmail Client wherein we have divided mailing system on the based on two kinds of users, one is Administrator and the other one is user. Administrator will have complete control on users accounts(other than administrator), complete control in the sense like administrator can change the password of the users if he wants to restrict them from login into our mailing system and like so. The one of the important feature of our Gmail Client is that intramailing interaction can be done in an easier and in a convenient way.

**2.4 CONCLUSION**  This project better suits for intra organization such Colleges, Company’s etc where they need efficient communication between users and effective control by the admin of the site.

**ANALYSIS**

**3. ANALYSIS**

**3.1 INTRODUCTION**

After analyzing the requirements of the task to be performed, the next step is to analyze the problem and understand its context. The first activity in the phase is studying the existing system and other is to understand the requirements and domain of the new system. Both the activities are equally important but the first activity serves as a basis of giving the functional specifications and then successful design of the proposed system. Understanding the properties and requirements of a new system is more difficult and requires creative thinking as well as understanding of existing system is also difficult. Improper understanding of present system can lead diversion from solution.

**3.1.1 Analysis Model**

The model that is basically being followed is WATER FALL Model which states that the phases are organized in a linear order. First of all, the feasibility study is done. Once that part is over, the requirement analysis and project planning begins. If system exists as a whole but modification and addition of new module is needed, analysis of present system can be used as basic model. The design starts after the requirement analysis is complete and the coding begins after the design is complete. Once the programming is completed, the testing is done. In this model the sequence of activities performed in a software development project are:

* Requirement Analysis
* Project Planning
* System Design
* Detail Design
* Coding
* Unit Testing
* System Integration & Testing

Here the linear ordering of these activities is critical. At the end of the phase, the output of one phase is the input to other phase. The output of each phase should be consistent with the overall requirement of the system. Some of the qualities of spiral model are also incorporated like after the people concerned with the project review completion of each of the phase the work done. Water fall model has been chosen because all requirements were known before and the objective of our software development is the computerization/automation of an already existing manual working system.

**ADVANTAGES:**

**1.** This model is very easy to use and implement.

**2.** Each phase is completed at a time and processed.

**3.** This model better works for smaller projects if only the requirements are well understood.

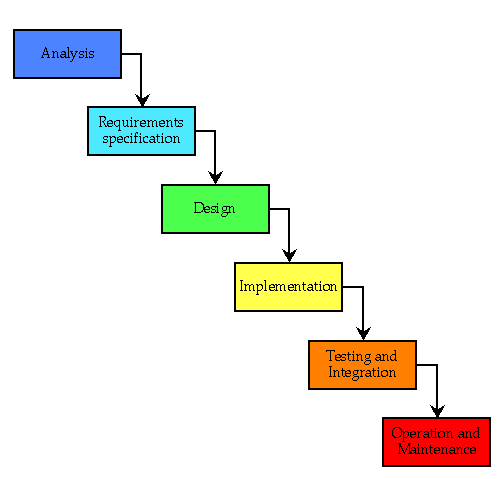
4. In each phase have deliverables and that must be reviewed.

**DISADVANTAGES:**

1. If the requirements are gathered are inaccurate then the final product is inaccurate and the error is known in the final phase of the model. Any sort of errors that cannot be detected in any previous phase.

2. For long, object-oriented, complex and ongoing projects it’s a poor model.

3. This model has high risks.



**Fig 3.1.1 Waterfall Lifecycle Model.**

**PROTOTYPE MODEL:**

In this model the requirements are gathered firstly, and the prototype is deployed according to the requirements. This prototype is a quick design which goes through the coding, design and testing. The phases are not done in detail. By seeing this prototype the client feels like a real system, so that the client understands the entire requirements of the systems.

**ADVANTAGES:**

**1.** During the development process the developers are interestingly engaged.

**2.** The prototype developed that is used by the users for well understanding of the methodology.

**3.** The user involvement is increased and improved.

**4.** The flaws and faults are identified early.

**5.** The user’s opinion about the product is known early which leads to an improved system.

**DISADVANTAGES:**

**1.** This model focuses on design quite than functionality.

**2.** The model is implemented firstly and then errors are evaluated later which becomes a complex process

**3.** The model is also known as throw-away prototype.

**4.** More time spent on development of the prototype that result in delay of the final product.

Requirements

Gathering

Quick Design

Refine Requirements

Build Prototype

Customer Evaluation of the prototype

Design

Implement

Test

Maintain

**.**

**3.1.2 Feasibility Report**

Preliminary investigation examine project feasibility, the likelihood the system will be useful to the organization. The main objective of the feasibility study is to test the Technical, Operational and Economical feasibility for adding new modules and debugging old running system. All system is feasible if they are unlimited resources and infinite time. There are aspects in the feasibility study portion of the preliminary investigation:

* Technical Feasibility
* Operation Feasibility
* Economical Feasibility

**TECHINICAL FEASIBILITY:**

Evaluating the technical feasibility is the trickiest part of a feasibility study. This is because, at this point in time, not too many detailed design of the system, making it difficult to access issues like performance, costs on (on account of the kind of technology to be deployed) etc. A number of issues have to be considered while doing a technical analysis.

**Understand the different technologies involved in the proposed system:**

Before commencing the project, we have to be very clear about what are the technologies that are required for the development of the new system.

**OPERATIONAL FEASIBILITY:**

Proposed projects are beneficial only if they can be turned into information systems that will meet the organizations operating requirements. Simply stated, this test of feasibility asks if the system will work when it is developed and installed. Are there major barriers to Implementation? Here are questions that will help test the operational feasibility of a project:

* Is there sufficient support for the project from management from users? If the current system is well liked and used to the extent that persons will not be able to see reasons for change, there may be resistance.
* Are the current business methods acceptable to the user? If they are not, Users may welcome a change that will bring about a more operational and useful systems.
* Have the user been involved in the planning and development of the project?

Early involvement reduces the chances of resistance to the system and in general and increases the likelihood of successful project. Since the proposed system was to help reduce the hardships encountered. In the existing manual system, the new system was considered to be operational feasible.

**ECONOMIC FEASIBILITY:**

Economic feasibility attempts 2 weigh the costs of developing and implementing a new system, against the benefits that would accrue from having the new system in place. This feasibility study gives the top management the economic justification for the new system.

A simple economic analysis which gives the actual comparison of costs and benefits are much more meaningful in this case. In addition, this proves to be a useful point of reference to compare actual costs as the project progresses. There could be various types of intangible benefits on account of automation. These could include increased customer satisfaction, improvement in product quality better decision making timeliness of information, expediting activities, improved accuracy of operations, better documentation and record keeping, faster retrieval of information, better employee morale.

**3.2 SOFTWARE REQUIREMENT SPECIFICATION**

**Purpose:** The main purpose for preparing this document is to give a general insight into the analysis and requirements of the existing system or situation and for determining the operating characteristics of the system.

**Scope:** This Document plays a vital role in the development life cycle (SDLC) and it describes the complete requirement of the system. It is meant for use by the developers and will be the basic during testing phase. Any changes made to the requirements in the future will have to go through formal change approval process.

The developer is responsible for:

* Developing the system, which meets the SRS and solving all the requirements of the system?
* Demonstrating the system and installing the system at client's location after the acceptance testing is successful.
* Submitting the required user manual describing the system interfaces to work on it and also the documents of the system.
* Conducting any user training that might be needed for using the system.
* Maintaining the system for a period of one year after installation.

**3.2.1 User Requirements**

* Computer

**3.2.2 Software Requirements**

1. Microsoft .Net framework 3.5
2. Microsoft Visual Studio 2008
3. Microsoft ASP.Net 3.5
4. Microsoft C#.Net language
5. Microsoft SQL Server 2005
6. ADO.NET
7. HTML

**3.2.3 Hardware Requirements**

Processor : Intel Pentium 4 or more

Ram : 1 GB or more

Hard disk : 40 GB hard disk recommended for primary partition.

**3.3 CONCLUSION**

In this phase, we understand the software requirement specifications for the project. We arrange all the required components to develop the project in this phase itself so that we will have a clear idea regarding the requirements before designing the project. Thus we will proceed to the design phase followed by the implementation phase of the project.

Design

**4. DESIGN**

**4.1 INTRODUCTION**

The design phase begins with the requirements specification for the software to be developed. Design is the first step to moving from the problem domain towards the solution domain. Design is essentially the bridge between requirement specification and the final solution for satisfying the requirements. It is the most critical factor effecting the quality of the software. The design process for software system has two levels.

1. System Design or Top level design

2. Detailed Design or Logical Design

**System Design:**

In the system design the focus on the deciding which modules are needed for the system, the specification of these modules and how these modules should be interconnected.

**Detailed Design:**

In detailed design the interconnection of the modules or how the specifications of the modules can be satisfied is decided. Some properties for a software system design are

* Verifiability
* Completeness
* Consistency
* Traceability
* Simplicity / Understandability

**4.2 DFD/UML DIAGRAM**

**UML diagrams**

Modelling is an activity that has been carried out over the years in software development. When writing applications by using the simplest languages to the most powerful and complex languages, you still need to model. Modelling can be as straightforward as drawing a flowchart listing the steps carried out by an application.

**Why do we use modeling?**

Defining a model makes it easier to break up a complex application or a huge system into simple, discrete pieces that can be individually studied. We can focus more easily on the smaller parts of a system and then understand the "big picture." Hence, the reasons behind modeling can be summed up in two words:

* Readability
* Reusability

**Readability:** Brings clarity—ease of understanding. Understanding a system is the first step in either building or enhancing a system. This involves knowing what a system is made up of, how it behaves, and so forth. Modelling a system ensures that it becomes readable and, most importantly, easy to document. Depicting a system to make it readable involves capturing the structure of a system and the behaviour of the system.

**Reusability:** Is the by product of making a system readable. After a system has been modelled to make it easy to understand, we tend to identify similarities or redundancy, be they in terms of functionality, features, or structure.The Unified Modelling Language, or UML, as it is popularly known by its TLA (three-letter acronym!), is the language that can be used to model systems and make them readable. This essentially means that UML provides the ability to capture the characteristics of a system by using notations. UML provides a wide array of simple, easy to understand notations for documenting systems based on the object-oriented design principles. These notations are called the nine diagrams of UML.Different languages have been used for depicting systems using object-oriented methodology. The prominent among these were the Rumbaugh methodology, the Booch methodology, and the Jacobson methodology. The problem was that, although each methodology had its advantages, they were essentially disparate. Hence, if you had to work on different projects that used any of these methodologies, you had to be well versed with each of these methodologies. A very tall order indeed! The Unified Modelling Language is just that. It "unifies" the design principles of each of these methodologies into a single, standard, language that can be easily applied across the board for all object-oriented systems. But, unlike the different methodologies that tended more to the design and detailed design of systems, UML spans the realm of requirements, analysis, and design and, uniquely, implementation as well. The beauty of UML lies in the fact that any of the nine diagrams of UML can be used on an incremental basis as the need arises. Considering all these reasons, it is no wonder that UML is considered "the" language of choice. UML does not have any dependencies with respect to any technologies or languages. This implies that you can use UML to model applications and systems based on either of the current hot technologies; for example, J2EE and .NET. Every effort has been made to keep UML as a clear and concise modelling language without being tied down to any technologies.

**INTRODUCTION TO UML:**

The Unified Modelling Language (UML) is a standard language for specifying, visualizing, constructing, and documenting the artifacts of software systems, as well as for business modelling and other non-software systems. The UML represents a collection of best engineering practices that have proven successful in the modelling of large and complex systems. The UML is a very important part of developing objects oriented software and the software development process. The UML uses mostly graphical notations to express the design of software projects. Using the UML helps project teams communicate, explore potential designs, and validate the architectural design of the software.

**Goals of UML**

The primary goals in the design of the UML were:

1. Provide users with a ready-to-use, expressive visual modeling language so they can develop and exchange meaningful models.
2. Provide extensibility and specialization mechanisms to extend the core concepts.
3. Be independent of particular programming languages and development processes.
4. Provide a formal basis for understanding the modeling language.
5. Encourage the growth of the OO tools market.
6. Support higher-level development concepts such as collaborations, frameworks, patterns and components.
7. Integrate best practices.

**Why we use UML?**

As the strategic value of software increases for many companies, the industry looks for techniques to automate the production of software and to improve quality and reduce cost and time-to-market. These techniques include component technology, visual programming, patterns and frameworks. Businesses also seek techniques to manage the complexity of systems as they increase in scope and scale. In particular, they recognize the need to solve recurring architectural problems, such as physical distribution, concurrency, replication, security, load balancing and fault tolerance. Additionally, the development for the World Wide Web, while making some things simpler, has exacerbated these architectural problems. The Unified Modelling Language (UML) was designed to respond to these needs.

**UML Diagrams**

The underlying premise of UML is that no one diagram can capture the different elements of a system in its entirety. Hence, UML is made up of nine diagrams that can be used to model a system at different points of time in the software life cycle of a system. The nine UML diagrams are:

**Use case diagram:**

The use case diagram is used to identify the primary elements and processes that form the system. The primary elements are termed as "actors" and the processes are called "use cases." The use case diagram shows which actors interact with each use case.

**Class diagram:**

The class diagram is used to refine the use case diagram and define a detailed design of the system. The class diagram classifies the actors defined in the use case diagram into a set of interrelated classes. The relationship or association between the classes can be either an "is-a" or "has-a" relationship. Each class in the class diagram may be capable of providing certain functionalities. These functionalities provided by the class are termed "methods" of the class. Apart from this, each class may have certain "attributes" that uniquely identify the class.

**Object diagram:**

The object diagram is a special kind of class diagram. An object is an instance of a class. This essentially means that an object represents the state of a class at a given point of time while the system is running. The object diagram captures the state of different classes in the system and their relationships or associations at a given point of time.

**State diagram:**

A state diagram, as the name suggests, represents the different states that objects in the system undergo during their life cycle. Objects in the system change states in response to events. In addition to this, a state diagram also captures the transition of the object's state from an initial state to a final state in response to events affecting the system.

**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, actions, transitions, initial and final states, and guard conditions.

**Sequence diagram:**

A sequence diagram represents the interaction between different objects in the system. The important aspect of a sequence diagram is that it is time-ordered. This means that the exact sequence of the interactions between the objects is represented step by step. Different objects in the sequence diagram interact with each other by passing "messages".

**Collaboration diagram:**

A collaboration diagram groups together the interactions between different objects. The interactions are listed as numbered interactions that help to trace the sequence of the interactions. The collaboration diagram helps to identify all the possible interactions that each object has with other objects.

**Component diagram:**

The component diagram represents the high-level parts that make up the system. This diagram depicts, at a high level, what components form part of the system and how they are interrelated. A component diagram depicts the components culled after the system has undergone the development or construction phase

**Deployment diagram:**

The deployment diagram captures the configuration of the runtime elements of the application. This diagram is by far most useful when a system is built and ready to be deployed. Now that we have an idea of the different UML diagrams, let us see if we can somehow group together these diagrams to enable us to further understand how to use them.

**UML Diagram Classification**—Static, Dynamic, and Implementation

A software system can be said to have two distinct characteristics: a structural, "static" part and a behavioral, "dynamic" part. In addition to these two characteristics, an additional characteristic that a software system possesses is related to implementation.Before we categorize UML diagrams into each of these three characteristics, let us take a quick look at exactly what these characteristics are.

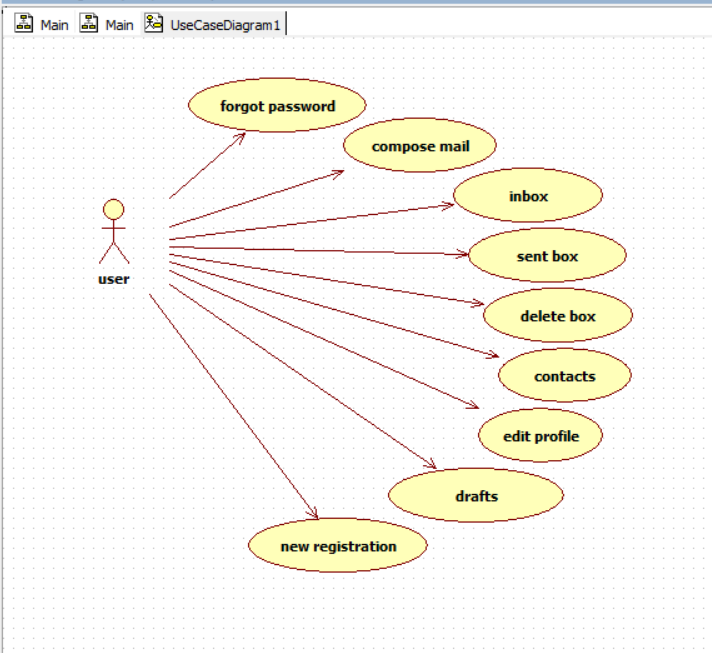
1. Static: The static characteristic of a system is essentially the structural aspect of the system. The static characteristics define what parts the system is made up of.
2. Dynamic: The behavioral features of a system; for example, the ways a system behaves in response to certain events or actions are the dynamic characteristics of a system.
3. Implementation: The implementation characteristic of a system is an entirely new feature that describes the different elements required for deploying a system.

The UML diagrams that fall under each of these categories are:

* **Static**
* Use case diagram
* Class diagram
* **Dynamic**
* Object diagram
* State diagram
* Activity diagram
* Sequence diagram
* Collaboration diagram
* **Implementation**
* Component diagram
* Deployment diagram

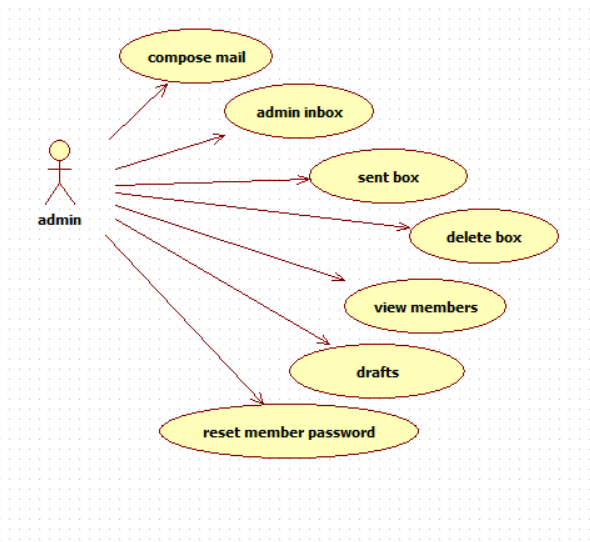
**USE CASE DIAGRAM:**

**I)User:**

****

**Fig:4.2.1.1 Use Case Diagram For User**

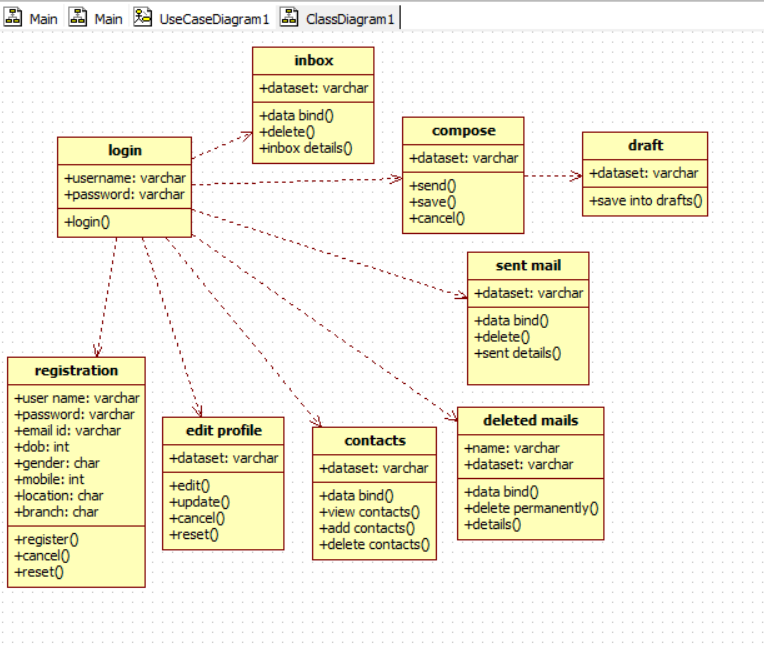
II)Admin:

****

**Fig:4.2.1.2 Use Case Diagram For Admin**

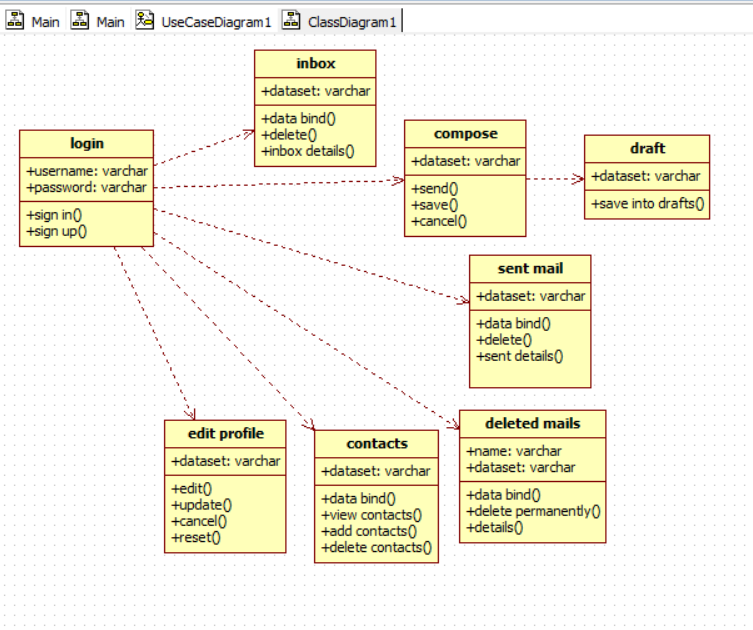
**CLASS DIAGRAM**

I)User:



**Fig:4.2.2.1 Class Diagram For User**

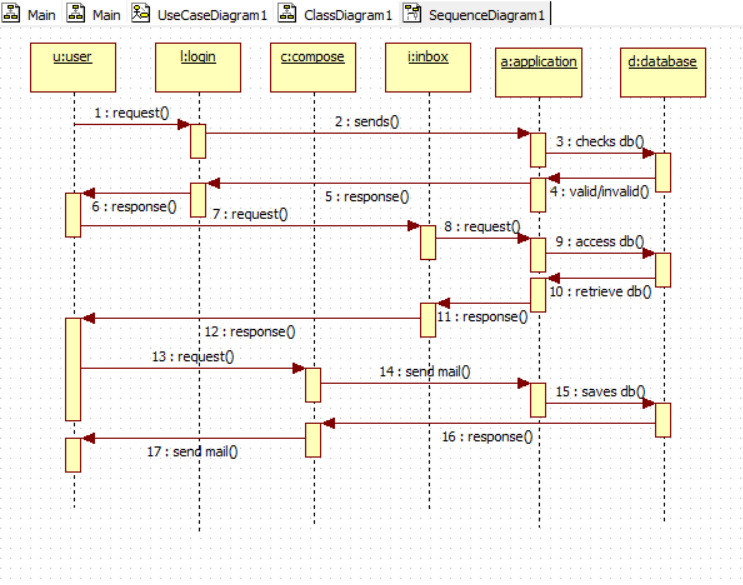
ii)Admin:



**Fig:4.2.2.2 Class Diagram For Admin**

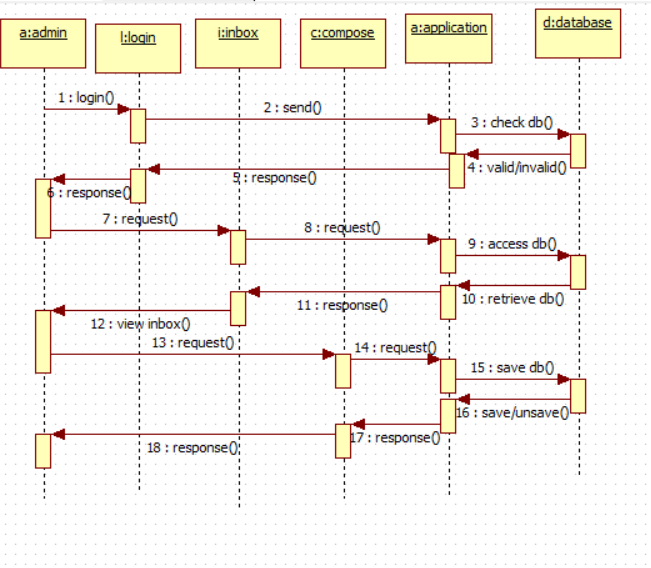
**SEQUENCE DIAGRAM:**

i)USER



**Fig:4.2.3.1 Sequence Diagram for User**

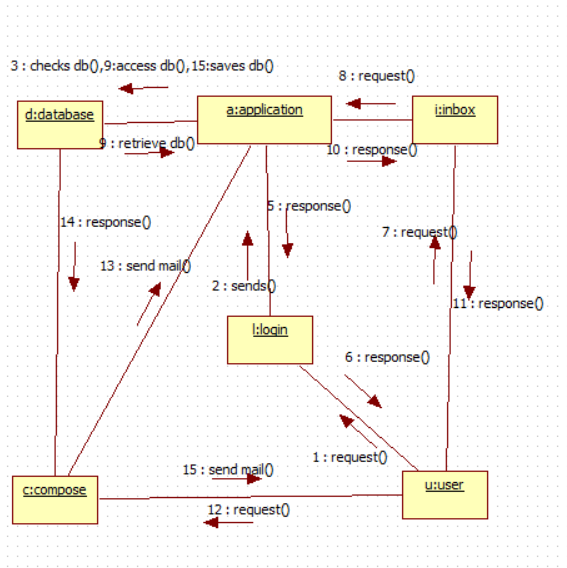
ii)ADMIN:



**Fig:4.2.3.2 Sequence Diagram for Admin**

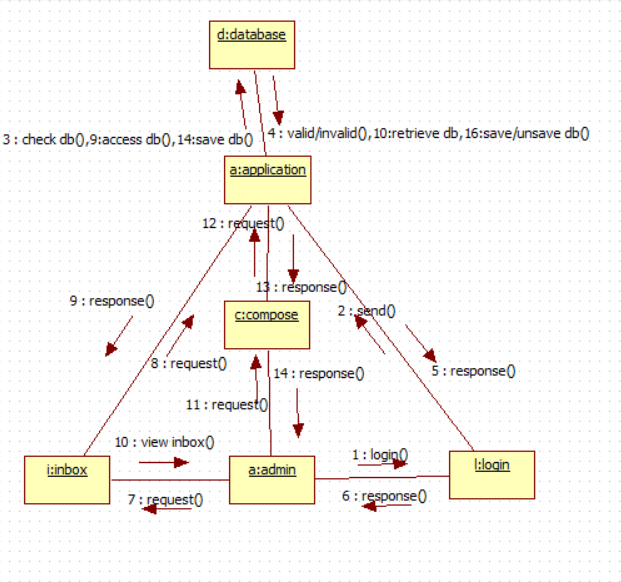
**COLLABORATION DIAGRAM**

i)User:



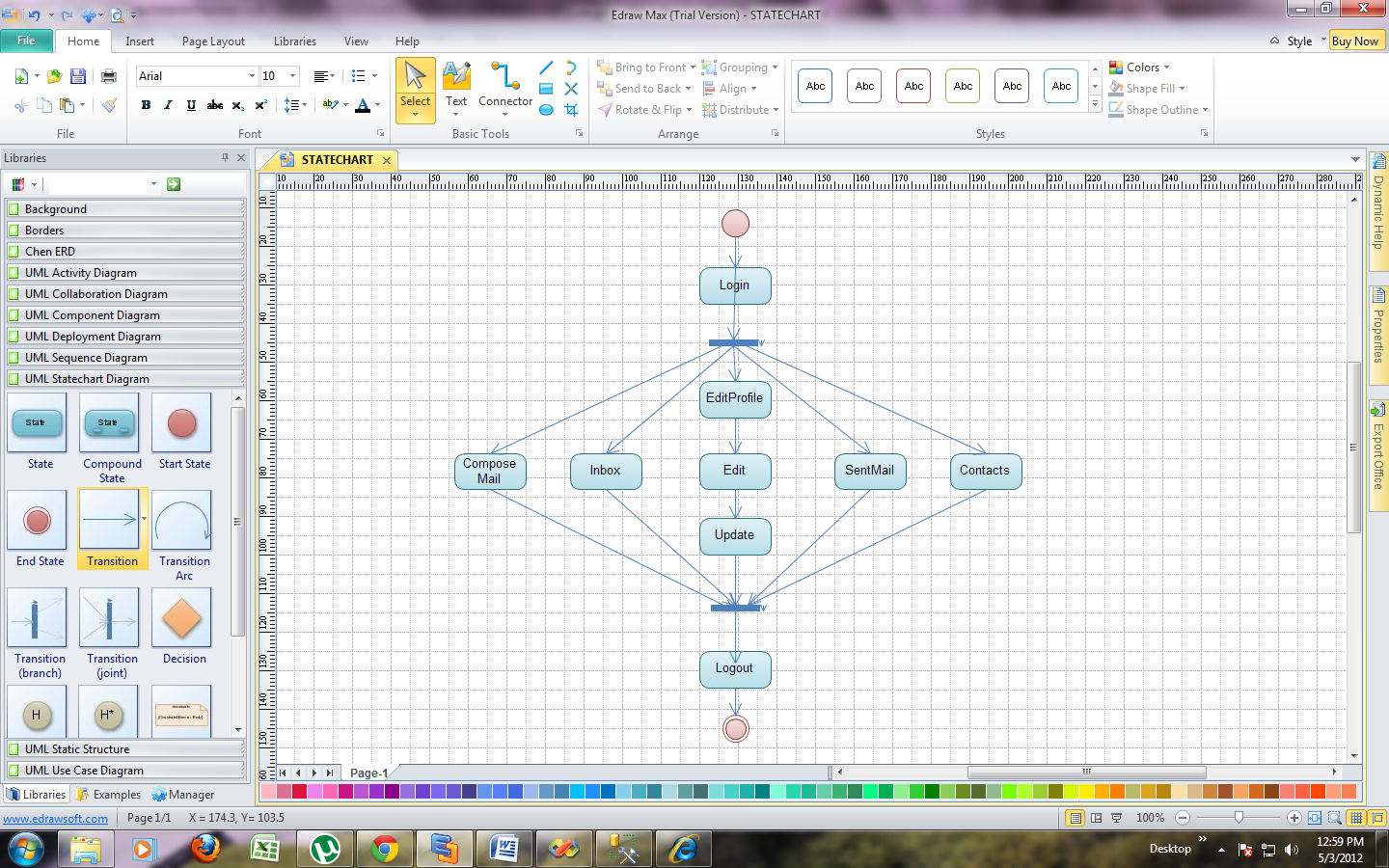
**Fig:4.2.4.1 Collaboration Diagram for User**

ii)Admin:



**Fig:4.2.4.2 Collaboration Diagram for Admin**

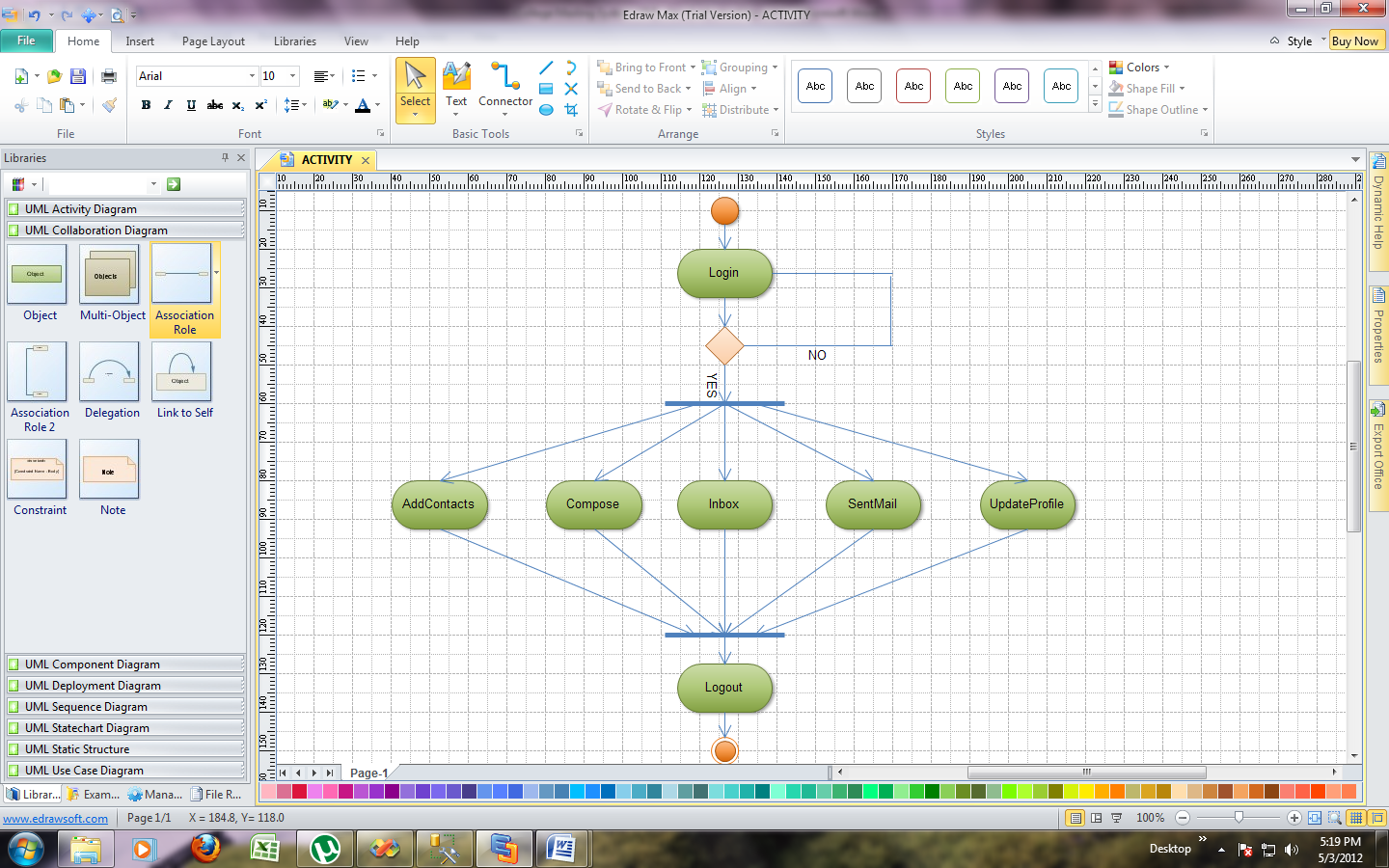
**STATECHART DIAGRAM**



**Fig:4.2.5 StateChart Diagram**

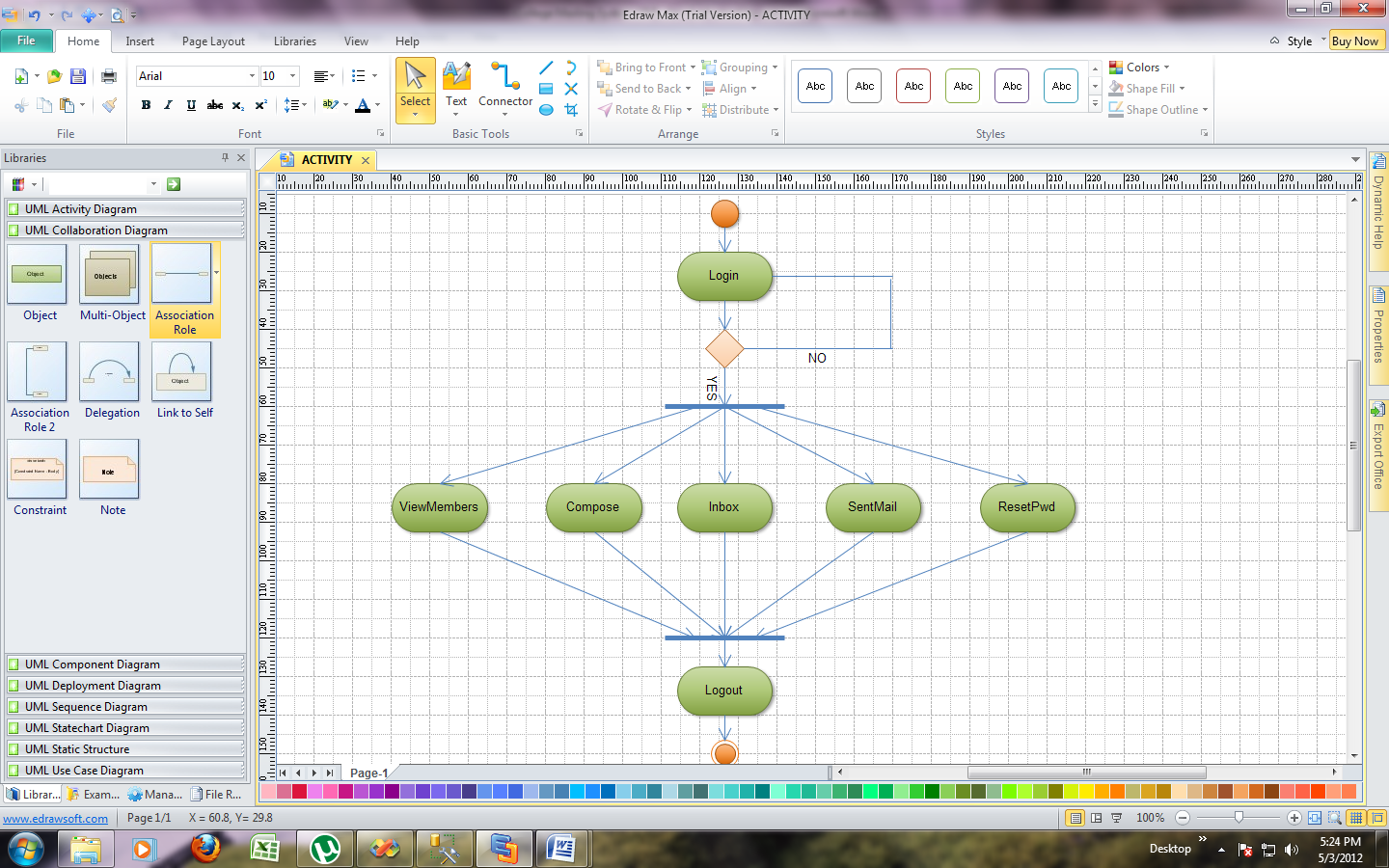
**ACTIVITY DIAGRAM**

i)User:



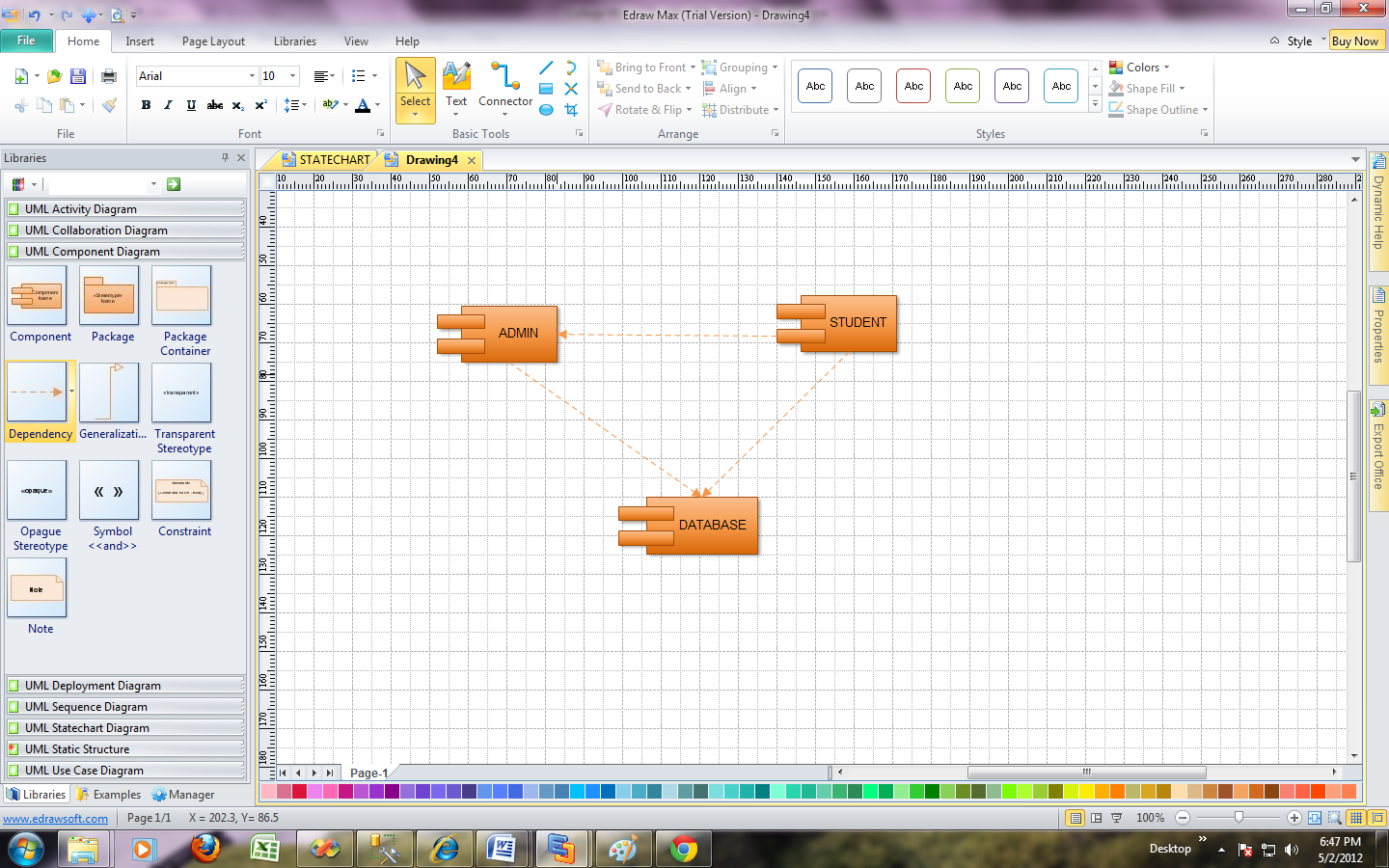
**Fig:4.2.6.1 Activity Diagram for User**

ii) Admin:



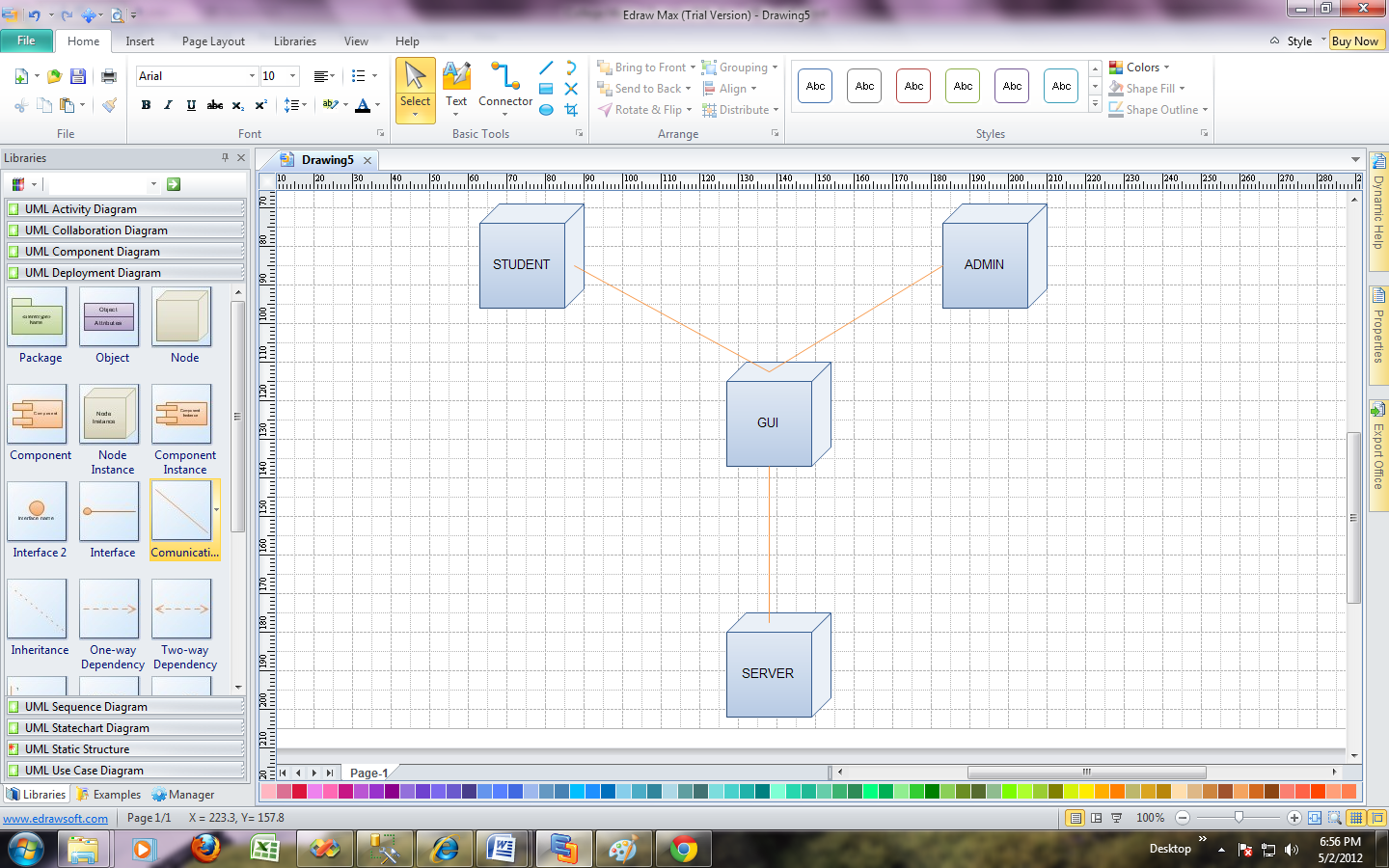
**Fig:4.2.6.1 Activity Diagram for Admin**

**COMPONENT DIAGRAM**

****

**Fig:4.2.7 Component Diagram**

**DEPLOYMENT DIAGRAM**



**Fig:4.2.8 Deployment Diagram**

**DFD Diagrams**

**Level 0**

**login**

**login**

**Level 1**

**Compose mail**

**login**

**Sent box**

**Inbox**

**Delete box**

**Compose mail**

**login**

**Sent box**

**Inbox**

**Delete box**

**Edit profile**

**Level 2:User**

**Add contacts**

**Compose mail**

**login**

**Sent box**

**Inbox**

**Delete box**

**Admin:**

**View members**

**Reset members**

**Compose mail**

**login**

**Sent box**

**Inbox**

**Delete box**

**Level3:**

**User**

**Edit profile**

**Add contacts**

Log

out

**Compose mail**

**login**

**Sent box**

**Inbox**

**Delete box**

**ADMIN**

**ViewMem**

**ResetMem**

**Compose mail**

Logout

**login**

**Sent box**

**Inbox**

**Delete box**

**4.3 Module Design and Organization**

In our application we have two modules student module and Admin module. Students will register into our application and make use of the mailing system. On the other hand admin have rights to control on our website and he/she can even resets the students passwords on temporary basics in order to restrict users from login into website , when students fails to fulfill the attendance or fees.

**User module:** User/Student will register to our website, once after login he performs the following tasks:

* Sending mails to the members and to the admin
* Receiving mails from members and from admin
* Deleting the mails
* Viewing sends items and deleted items
* Adding contacts

**Admin module:**

Administrator will have complete control on users accounts(other than administrator), complete control in the sense like administrator can change the password of the users if he wants to restrict them from login into our mailing system and like so.

**4.4 CONCLUSION**

In this way we can design the layout of the project which is to be implemented during the construction phase. Thus we will have a clear picture of the project before being coded. Hence any necessary enhancements can be made during this phase and coding can be started.

**IMPLEMENTATION AND RESULTS**

**5. IMPLEMENTATION AND RESULTS**

**5.1. INTRODUCTION**

The implementation part is the most important phase of the project. In this phase, we code the entire project in the chosen software according to the design laid during the previous phase. The code has to be in such a way that the user requirements are satisfied and also not complicated for the user i.e., the user interface or GUI has to be easy to navigate. The code should be efficient in all terms like space, easy to update, etc. In this manner, we can complete the coding part of the project and later it can be sent for testing before being delivered to the customer.

**Microsoft .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.

**5.2 SAMPLE CODE**

using System;

using System.Collections;

using System.Configuration;

using System.Data;

using System.Linq;

using System.Web;

using System.Web.Security;

using System.Web.UI;

using System.Web.UI.HtmlControls;

using System.Web.UI.WebControls;

using System.Web.UI.WebControls.WebParts;

using System.Xml.Linq;

using CollegeMailingSystemBAL;

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

{

CollegeMailingSystemBAL.UserRegEdit loginobj = new CollegeMailingSystemBAL.UserRegEdit();

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

{

}

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

{

loginobj.loginname = LoginUsernameTxtbx.Text;

loginobj.password = Loginpwdtxtbx.Text;

loginobj.branch = BranchLoginDDL.SelectedItem.Text;

Session["UserName"] = LoginUsernameTxtbx.Text;

DataSet ds = new DataSet();

if (LoginUsernameTxtbx.Text == "Admin")

{

Response.Redirect("AdminInbox.aspx");

}

else

{

ds = loginobj.login(loginobj.loginname, loginobj.password, loginobj.branch);

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

{

Session["UserName"] = loginobj.loginname;

Session["Password"] = loginobj.password;

Session["SenderId"] = ds.Tables[0].Rows[0]["Email"];

}

else

{

Response.Redirect("UserInbox.aspx");

}

}

//else

//{

// LoginUsernameTxtbx.Text = string.Empty;

// Loginpwdtxtbx.Text = string.Empty;

// BranchLoginDDL.Items.Clear();

// BranchLoginDDL.Items.Add("select");

// BranchLoginDDL.Items.Add("BTECH");

// BranchLoginDDL.Items.Add("MCA");

// BranchLoginDDL.Items.Add("MBA");

// BranchLoginDDL.Items.Add("MTECH");

// BranchLoginDDL.Items.Add("Admin");

// BranchLoginDDL.SelectedItem.Text = "select";

// usrloginresultlbl.Text = "Invalid User Login";

//}

}

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

{

Response.Redirect("Registration.aspx");

}

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

{

Response.Redirect("ForgotPassword.aspx");

}

protected void BranchLoginDDL\_SelectedIndexChanged(object sender, EventArgse)

{

}

}

Registration page:

using System;

using System.Collections;

using System.Configuration;

using System.Data;

using System.Linq;

using System.Web;

using System.Web.Security;

using System.Web.UI;

using System.Web.UI.HtmlControls;

using System.Web.UI.WebControls;

using System.Web.UI.WebControls.WebParts;

using System.Xml.Linq;

using CollegeMailingSystemBAL;

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

{

CollegeMailingSystemBAL.UserRegEdit UsrRegobj = new CollegeMailingSystemBAL.UserRegEdit();

// char gender = 'b';

int Usertype = 0;

string UsrRegResult;

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

{

}

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

{

Response.Redirect("Login.aspx");

}

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

{

// if(UsrRegRadiolistGen.SelectedItem.Selected==

UsrRegResultlbl.Visible = true;

//DateTime dob =Convert.ToDateTime(UsrRegDOBtxtbx.Text);

UsrRegResult = UsrRegobj.UserRegistration(UsrRegUserNameTxtbx.Text, UsrRegEmailTxtBx.Text, UsrRegpwdtxtbx.Text, Convert.ToChar(UsrRegRadiolistGen.SelectedItem.Value), UsrRegDOBtxtbx.Text, UsrRegMobTxtBx.Text, UsrRegDDLloc.SelectedItem.Text, UsrRegDDLSecrtyQues.SelectedItem.Text, UsrRegScrtyAnsTxBx.Text, UsrRegAltEmailTxtBx.Text, UsrRegDDLBrnch.SelectedItem.Text, Usertype);

if (UsrRegResult == "no")

UsrRegResultlbl.Text = " User Succussful Registration";

else

UsrRegResultlbl.Text = "Registration Fails";

}

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

{

UsrRegUserNameTxtbx.Text = string.Empty;

UsrRegEmailTxtBx.Text = string.Empty;

UsrRegpwdtxtbx.Text = string.Empty;

if (UsrRegRadiolistGen.SelectedItem == null) { }

else

UsrRegRadiolistGen.SelectedItem.Selected = false;

UsrRegDOBtxtbx.Text = string.Empty;

UsrRegMobTxtBx.Text = string.Empty;

UsrRegDDLloc.SelectedItem.Text = "select";

UsrRegDDLSecrtyQues.SelectedItem.Text = "select";

UsrRegScrtyAnsTxBx.Text = string.Empty;

UsrRegAltEmailTxtBx.Text = string.Empty;

UsrRegDDLBrnch.SelectedItem.Text = string.Empty;

}

}

**5.3** **EXPLANATION OF KEY FUNCTIONS**

**ADO.NET**

ADO.NET ARCHITECHTURE

DATA STORE

ADO.NET is an extension of the ADO data access model which consist of only the connected architecture. The Microsoft organization has realized the data related operations and have studied and analyzed different data related technologies among which they found ADO to be interesting , later on they extended the features of the ADO and defined own data related technology by refining the ADO and hence given the name as ADO.Net. The Microsoft organization , grouped some set of namespaces which can operates on data , and put together into technology called ADO.NET. The ADO.Net technology enhances the features of the ADO, which consist of only connected architecture where as in ADO.NET they have introduces a disconnected architecture.

The connected architecture important feature is the data provider, which consists of the four important objects namely connection, command, data reader and data adapter.

The connection object provides the connection to the data store nothing but to the back end database servers. The connection class consists of default constructors and parameterized constructors. The constructor takes arguments, which provides the connection to the back end servers. The arguments that the connection class constructors takes are Data Source , Database and security.

CONECTION

COMMAND

DATA READER

ACEESS

ORACLE

MSSQL

DATAVIEW

XML OUTPUT

DATAVIEW

DataTable2

DATA RELATION

DataTable1

DataRow/DataColumn

D

DATA

ADAPTER

**DATA PROVIDER**

**DATASET**

The first parameter Data Source represents the server name to which our application needs to be connected. That’s means from the available servers we need to select the particular data base server which can be done through the data source parameter. The second parameter indicates the database to which we are going to connect, that’s means in that particular database server, to which data base we want to connect can be done through the data base parameter. The third parameter security indicates, the security provided for the database server. If the server is running under windows authentication mode , than will use integrated security to be true that’s means no need to specify the user name and password explicitly why because the system will takes the prebuilt username and password which has been set for the system. On the other hand if the back end server Is running under sql authentication mode that will specify the username and password which has be set during the installation of the server, using the security parameter we can connect to the backend database server.

Also sqlconnection class consists of the methods such as open and close. The open method is used to open the connection to the database server. Whereas the close method is used to disconnect the connection from the server. Once the connection is opened while in the application use, the connection should be closed when the application terminates.

The another object of the dataprovider is the command object, using which one can write the queries in order to manipulate data in the database. Once the connection is opened ,the sqlcommand class make use of the connection and will operates on the database. The sqlcommand class will do manipulation using the queries or the stored procedures. Which has to be decided by the programmers whether they want to use the queries or the stored procedures using the method command type? If we want to use queries than we need to select the text query or else we need to select stored procedure option from the command type method.

Command object consists of three methods namely execute non query, execute reader and execute scalar. The execute non query will returns the integer values as an output which indicates how many records have been updated, or modified etc. the second method execute reader returns the complete records been affected by the operations , whereas execute scalar returns the first row first column value remaining will be neglected.

The third object in the connected architecture is the datareader ,which reads the data in a forward only mode , that’s means its retrieves the data from the data base server and forwards it to the application.

The another object is the dataadapter which acts like an interface or bridge between the connected architecture and disconnected architecture.

In the disconnected architecture the important feature is the dataset. It’s a collection of datatables and datarows and the datatables will be linked using the data relations . when the dataset need to be filled ,its request to the data adapter which in turn fills the dataset by making use of fill method. Of dataadapter.

Features of ADO.NET are as follows:

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

**Asp.net description**

**Server-side managed code**

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

XML Web services, an important evolution in Web-based technology, are distributed, server-side application components similar to common Web sites. However, unlike Web-based applications, XML Web services components have no UI and are not targeted for browsers such as Internet Explorer and Netscape Navigator. Instead, XML Web services consist of reusable software components designed to be consumed by other applications, such as traditional client applications, Web-based applications, or even other XML Web services. As a result, XML Web services technology is rapidly moving application development and deployment into the highly distributed environment of the Internet.

If you have used earlier versions of ASP technology, you will immediately notice the improvements that ASP.NET and Web Forms offers. For example, you can develop Web Forms pages in any language that supports the .NET Framework. In addition, your code no longer needs to share the same file with your HTTP text (although it can continue to do so if you prefer). Web Forms pages execute in native machine language because, like any other managed application, they take full advantage of the runtime. In contrast, unmanaged ASP pages are always scripted and interpreted. ASP.NET pages are faster, more functional, and easier to develop than unmanaged ASP pages because they interact with the runtime like any managed application.

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

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

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

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

**Active Server Pages.NET**

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

* **Enhanced Performance.** ASP.NET is compiled common language runtime code running on the server. Unlike its interpreted predecessors, ASP.NET can take advantage of early binding, just-in-time compilation, native optimization, and caching services right out of the box. This amounts to dramatically better performance before you ever write a line of code.
* **World-Class Tool Support.** The ASP.NET framework is complemented by a rich toolbox and designer in the Visual Studio integrated development environment. WYSIWYG editing, drag-and-drop server controls, and automatic deployment are just a few of the features this powerful tool provides.
* **Power and Flexibility.** Because ASP.NET is based on the common language runtime, the power and flexibility of that entire platform is available to Web application developers. The .NET Framework class library, Messaging, and Data Access solutions are all seamlessly accessible from the Web. ASP.NET is also language-independent, so you can choose the language that best applies to your application or partition your application across many languages. Further, common language runtime interoperability guarantees that your existing investment in COM-based development is preserved when migrating to ASP.NET.
* **Simplicity.** ASP.NET makes it easy to perform common tasks, from simple form submission and client authentication to deployment and site configuration. For example, the ASP.NET page framework allows you to build user interfaces that cleanly separate application logic from presentation code and to handle events in a simple, Visual Basic - like forms processing model. Additionally, the common language runtime simplifies development, with managed code services such as automatic reference counting and garbage collection.
* **Manageability.** ASP.NET employs a text-based, hierarchical configuration system, which simplifies applying settings to your server environment and Web applications. Because configuration information is stored as plain text, new settings may be applied without the aid of local administration tools. This "zero local administration" philosophy extends to deploying ASP.NET Framework applications as well. An ASP.NET Framework application is deployed to a server simply by copying the necessary files to the server. No server restart is required, even to deploy or replace running compiled code.
* **Scalability and Availability.** ASP.NET has been designed with scalability in mind, with features specifically tailored to improve performance in clustered and multiprocessor environments. Further, processes are closely monitored and managed by the ASP.NET runtime, so that if one misbehaves (leaks, deadlocks), a new process can be created in its place, which helps keep your application constantly available to handle requests.
* **Customizability and Extensibility.** ASP.NET delivers a well-factored architecture that allows developers to "plug-in" their code at the appropriate level. In fact, it is possible to extend or replace any subcomponent of the ASP.NET runtime with your own custom-written component. Implementing custom authentication or state services has never been easier.
* **Security.** With built in Windows authentication and per-application configuration, you can be assured that your applications are secure.

#### Language Support

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

**What is ASP.NET Web Forms?**

The ASP.NET Web Forms page framework is a scalable common language runtime programming model that can be used on the server to dynamically generate Web pages.

Intended as a logical evolution of ASP (ASP.NET provides syntax compatibility with existing pages), the ASP.NET Web Forms framework has been specifically designed to address a number of key deficiencies in the previous model. In particular, it provides:

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

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

An ASP.NET page can be created simply by taking an existing HTML file and changing its file name extension to .aspx (no modification of code is required). For example, the following sample demonstrates a simple HTML page that collects a user's name and category preference and then performs a form postback to the originating page when a button is clicked:

ASP.NET provides syntax compatibility with existing ASP pages. This includes support for <% %> code render blocks that can be intermixed with HTML content within an .aspx file. These code blocks execute in a top-down manner at page render time.

**Code-Behind Web Forms**

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

**Introduction to ASP.NET Server Controls**

In addition to (or instead of) using <% %> code blocks to program dynamic content, ASP.NET page developers can use ASP.NET server controls to program Web pages. Server controls are declared within an .aspx file using custom tags or intrinsic HTML tags that contain a **runat="server"** attribute value. Intrinsic HTML tags are handled by one of the controls in the **System.Web.UI.HtmlControls** namespace. Any tag that doesn't explicitly map to one of the controls is assigned the type of **System.Web.UI.HtmlControls.HtmlGenericControl**.

Server controls automatically maintain any client-entered values between round trips to the server. This control state is not stored on the server (it is instead stored within an **<input type="hidden">** form field that is round-tripped between requests). Note also that no client-side script is required.

In addition to supporting standard HTML input controls, ASP.NET enables developers to utilize richer custom controls on their pages. For example, the following sample demonstrates how the **<asp:adrotator>** control can be used to dynamically display rotating ads on a page.

1. ASP.NET Web Forms provide an easy and powerful way to build dynamic Web UI.
2. ASP.NET Web Forms pages can target any browser client (there are no script library or cookie requirements).
3. ASP.NET Web Forms pages provide syntax compatibility with existing ASP pages.
4. ASP.NET server controls provide an easy way to encapsulate common functionality.
5. ASP.NET ships with 45 built-in server controls. Developers can also use controls built by third parties.
6. ASP.NET server controls can automatically project both uplevel and downlevel HTML.
7. ASP.NET templates provide an easy way to customize the look and feel of list server controls.
8. ASP.NET validation controls provide an easy way to do declarative client or server data validation.

**5.4 METHOD OF IMPLEMENTATION**

**Database design**

Database design is the process of producing a detailed [data model](http://en.wikipedia.org/wiki/Data_model) of a [database](http://en.wikipedia.org/wiki/Database). It is the logical design of the base data structures used to store the data.

**5.4.1 FORMS/TABLES:**

|  |  |  |
| --- | --- | --- |
| **Field Name** | **Data Type** | **Constraints** |
| ContactId | int | Unchecked |
| ContactName | varchar(500) | Checked |
| Address | varchar(2000) | Checked |
| PhoneNo | varchar(50) | Checked |
| Contactsemail | varchar(1000) | Checked |
| UserId | int | Checked |

Fig:5.2.1. Contact Db Table

|  |  |  |
| --- | --- | --- |
| **Field Name** | **Data Type** | **Constraints** |
| DraftId | int | Unchecked |
| Toaddr | varchar(300) | Checked |
| Ccaddr | varchar(300) | Checked |
| Subject | varchar(500) | Checked |
| Attachment | varchar(500) | Checked |
| Mailcontent | varchar(8000) | Checked |
| SendId | varchar (100) | Checked |
| UserId | int | Checked |

Fig:5.2.2. Draft Db Table

|  |  |  |
| --- | --- | --- |
| **Field Name** | **Data Type** | **Constraints** |
| MailAttachId | Int | Unchecked |
| MailId | int | Checked |
| Attachment | varchar(MAX) | Checked |

Fig:5.2.3. Mail Attachment db Table

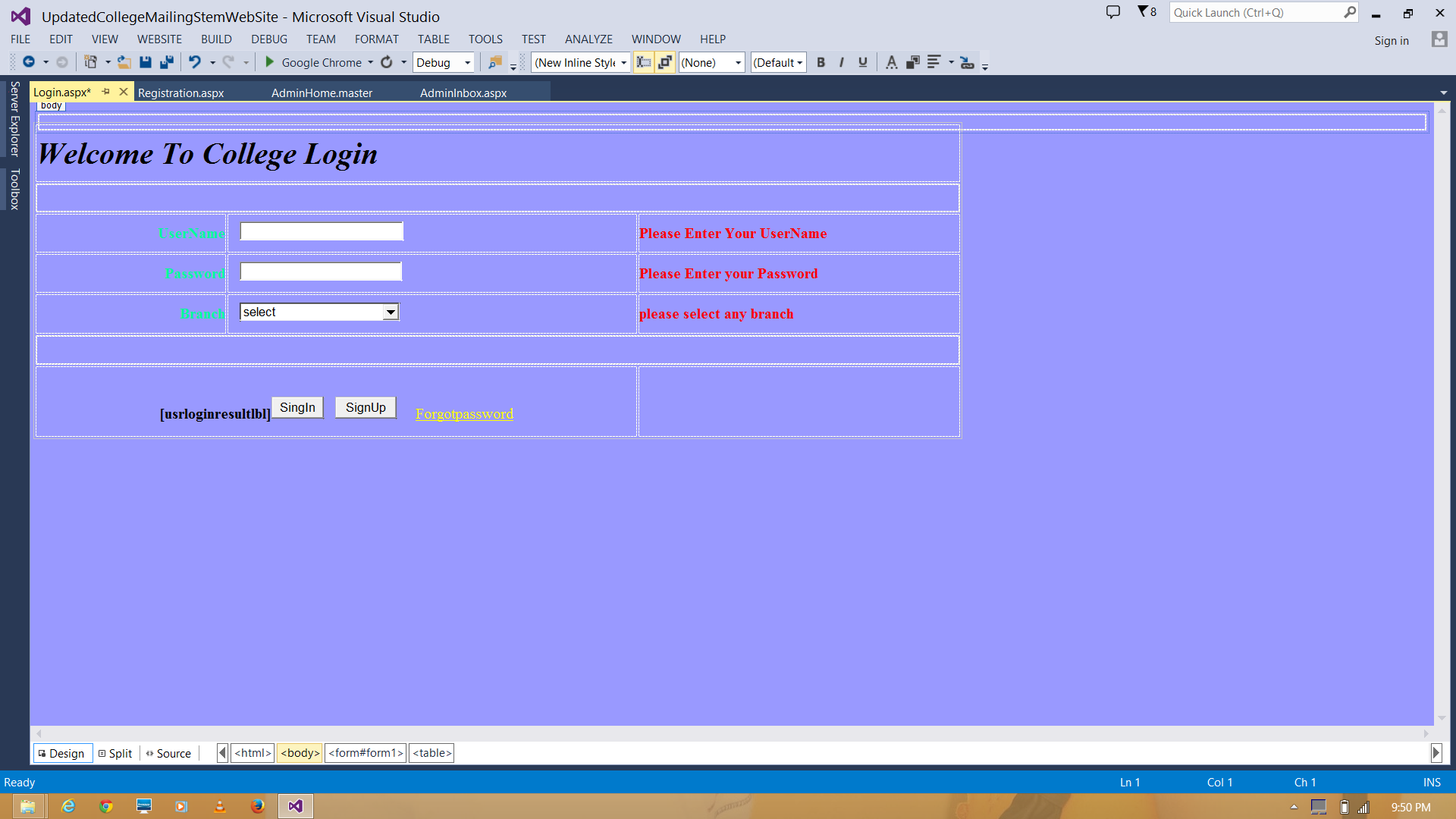
|  |  |  |
| --- | --- | --- |
| **Field Name** | **Data Type** | **Constraints** |
| MailTo\_List | varchar(1000) | Checked |
| CC\_List | varchar(1000) | Checked |
| SentId | varchar(1000) | Checked |
| status | char(10) | Checked |

|  |  |  |
| --- | --- | --- |
| **Field Name** | **Data Type** | **Constraints** |
| UserId | int | Unchecked |
| UserName | varchar(50) | Checked |
| Email | varchar (50) | Checked |
| Password | varchar(1000) | Checked |
| Gender | char(10) | Checked |
| DOB | datetime | Checked |
| Mobile | varchar (50) | Checked |
| Location | varchar (50) | Checked |
| SecurityQuestion | varchar (50) | Checked |
| SecurityAnswer | varchar (50) | Checked |
| AlternativeEmail | varchar (50) | Checked |
| Branch | varchar (50) | Checked |
| Usertype | int | Checked |
| UserStatus | int | Checked |

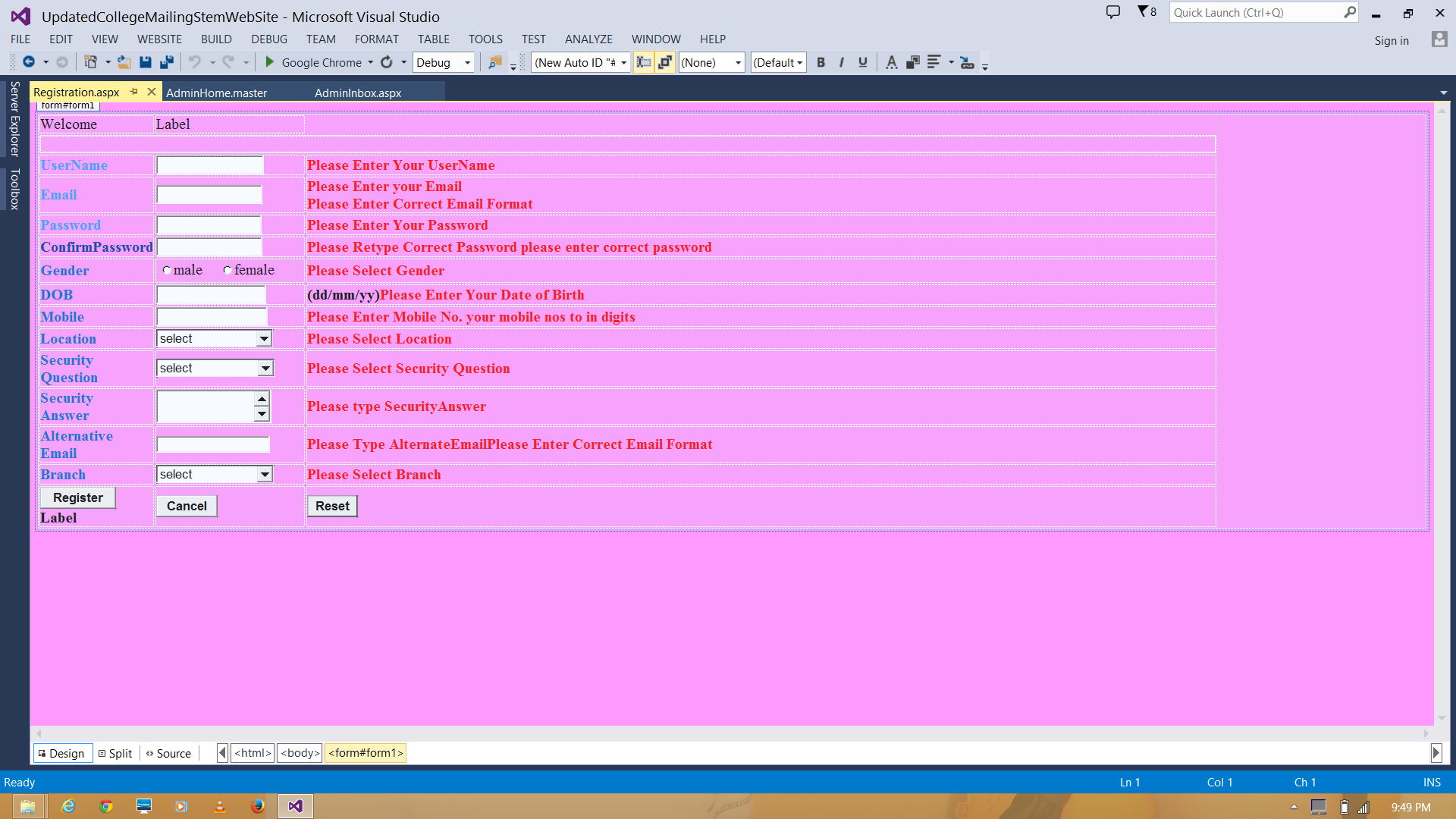
Fig:5.2.6. User Registration Db Table

**5.4.2.OUTPUT SCREENS**

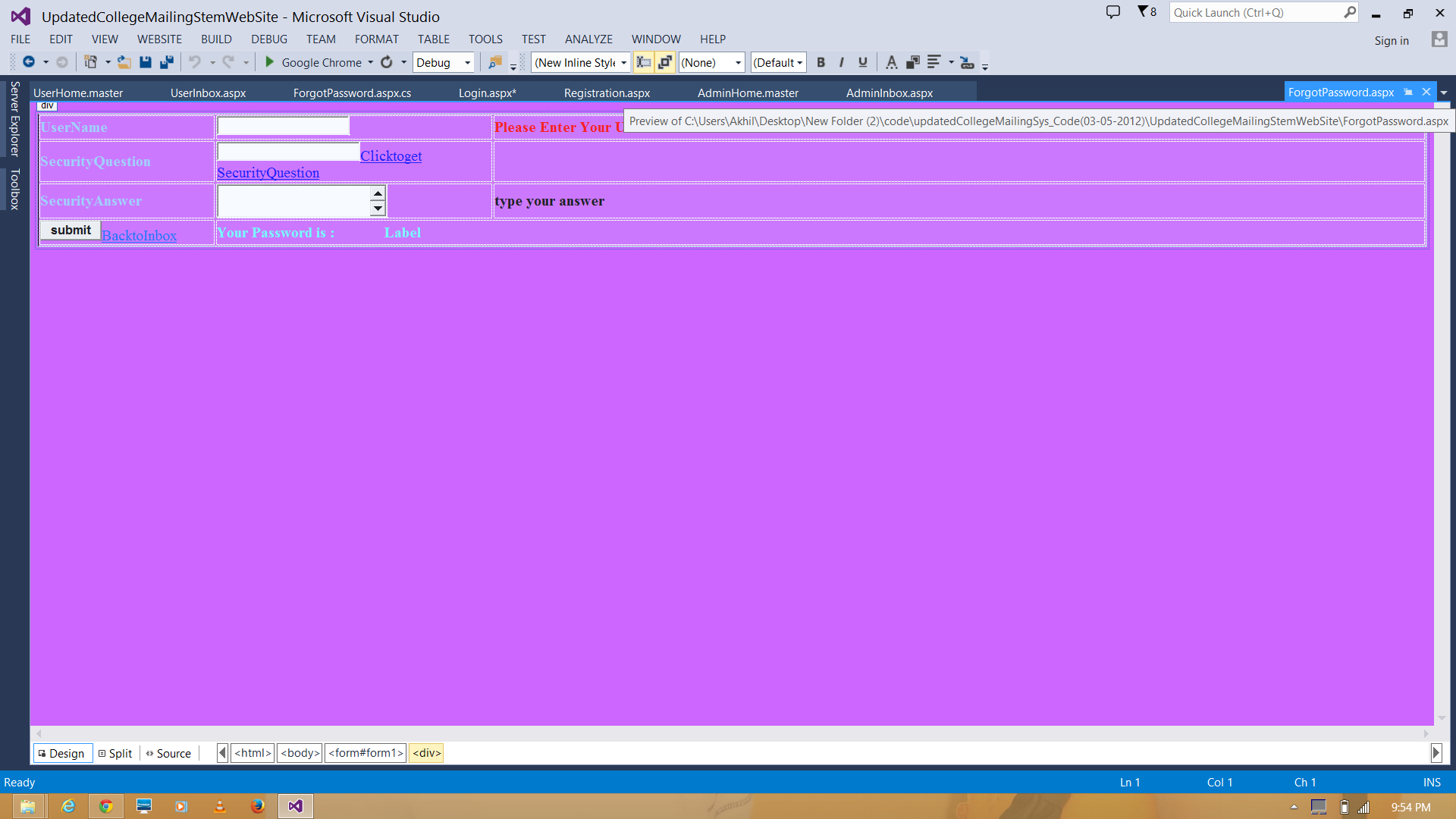
**Login page:**

****

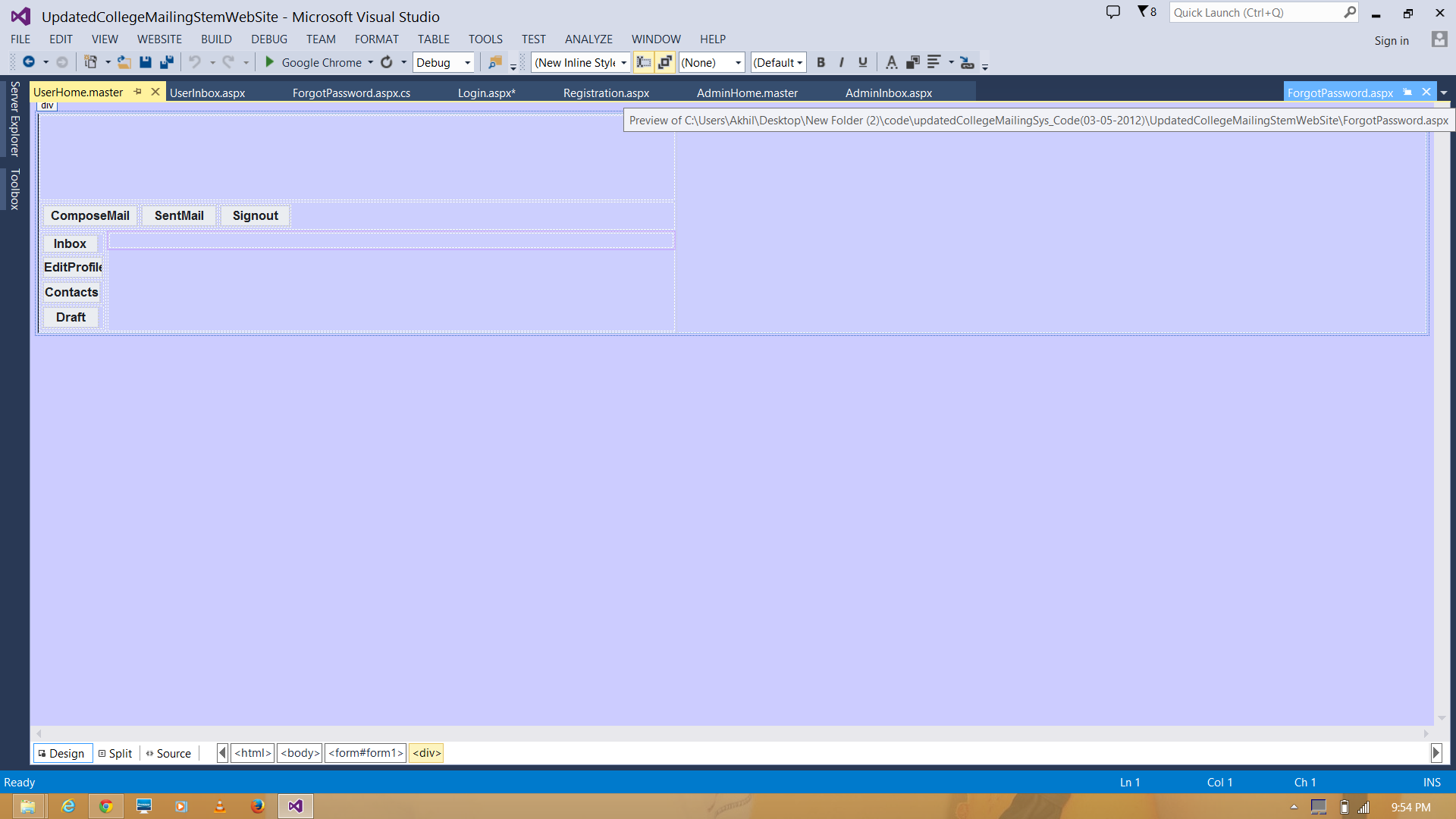
**Registration page:**



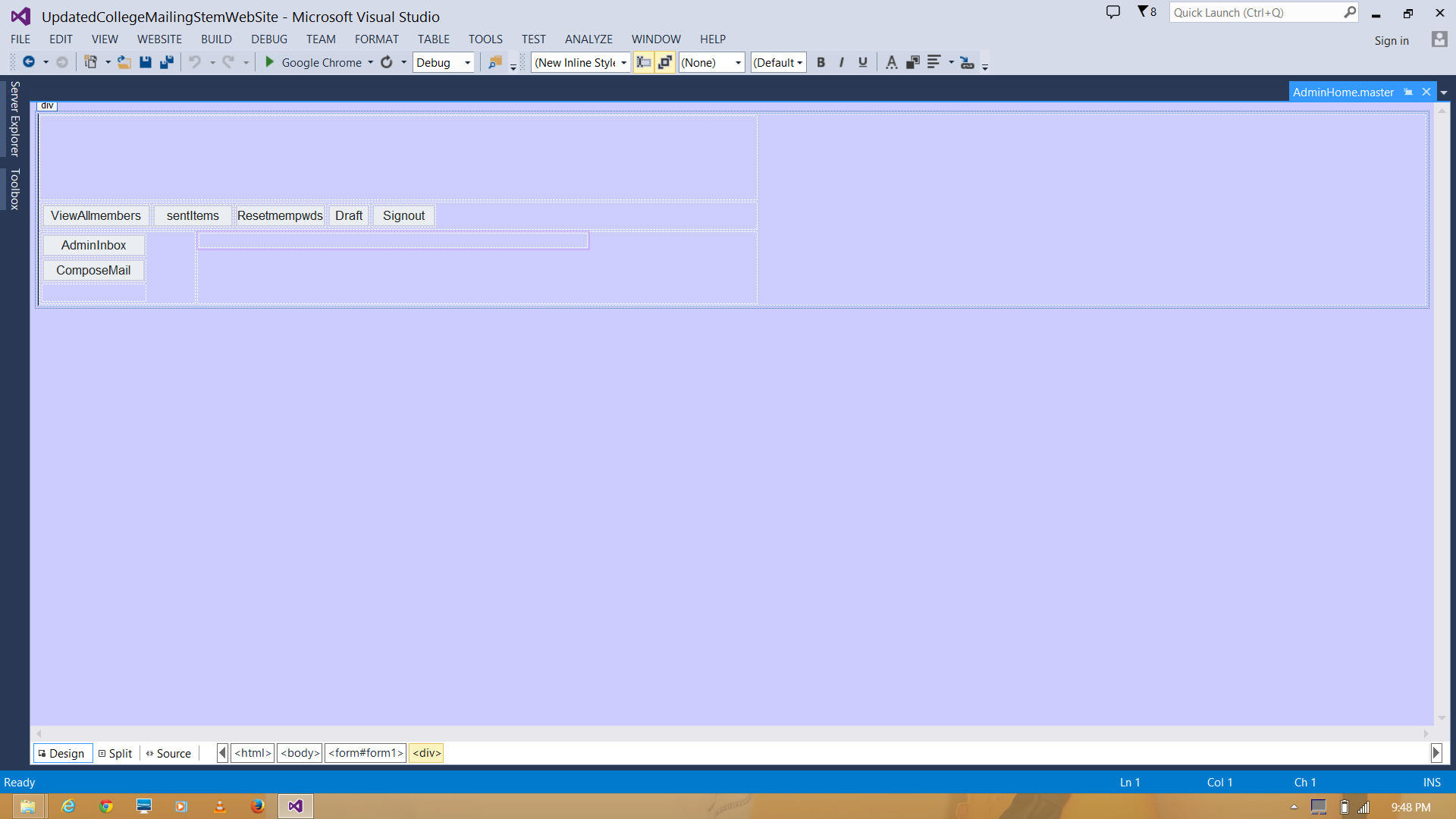
**Forgot password page:**

****

**User home page:**

****

**Admin home page:**

****

**5.3 CONCLUSION**

In this way we implemented the project successfully with the help of .NET framework for an easy interaction of the user with the interfaces. We proceed to the next phase i.e., testing which is very important before delivering the project.

**TESTING AND VALIDATION**

**6. TESTING AND VALIDATION**

**6.1 INTRODUCTION**

Software testing is a critical element of software quality assurance and represents the ultimate review of specification, design and coding. In fact, testing is the one step in the software engineering process that could be viewed as destructive rather than constructive.

A strategy for software testing integrates software test case design methods into a well-planned series of steps that result in the successful construction of software. Testing is the set of activities that can be planned in advance and conducted systematically. The underlying motivation of program testing is to affirm software quality with methods that can economically and effectively apply to both strategic to both large and small-scale systems.

The following are the Testing Objectives:

* Testing is a process of executing a program with the intent of finding an error.
* A good test has a high probability of finding an as yet undiscovered error.
* A successful test is one that uncovers an as yet undiscovered error.

**6.2 DESIGN OF TEST CASES AND SCENARIOS**

The objective is to design tests that systematically uncover different classes of errors and do so with a minimum amount of time and effort. Testing cannot show the absence of defects, it can only show that software defects are present.

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

**Black Box Testing:**Black box testing treats the software as a "black box," without any knowledge of internal implementation. Black box testing methods include: equivalence partitioning, boundary value analysis, all-pairs testing, fuzz testing, model-based testing, traceability matrix, exploratory testing and specification-based testing.

**Advantages and Disadvantages :**

The black box tester has no "bonds" with the code, and a tester's perception is very simple: a code must have bugs. Using the principle, "Ask and you shall receive," black box testers find bugs where programmers don't. But, on the other hand, black box testing has been said to be "like a walk in a dark labyrinth without a flashlight," because the tester doesn't know how the software being tested was actually constructed. That's why there are situations when

(1) a black box tester writes many test cases to check something that can be tested by only one test case, and/or

(2) some parts of the back end are not tested at all.

Therefore, black box testing has the advantage of "an unaffiliated opinion," on the one hand, and the disadvantage of "blind exploring," on the other.

**White Box Testing :**White box testing, by contrast to black box testing, is when the tester has access to the internal data structures and algorithms (and the code that implement these)

Types of white box testing :

* API testing - Testing of the application using Public and Private APIs.
* Code coverage - creating tests to satisfy some criteria of code coverage. For example, the test designer can create tests to cause all statements in the program to be executed at least once.
* Fault injection methods.
* Mutation testing methods.
* Static testing - White box testing includes all static testing.

**Code Completeness Evaluation:**White box testing methods can also be used to evaluate the completeness of a test suite that was created with black box testing methods. This allows the software team to examine parts of a system that are rarely tested and ensures that the most important function points have been tested.

Two common forms of code coverage are:

* Function coverage, which reports on functions executed
* Statement coverage, which reports on the number of lines executed to complete the test.

They both return coverage metric, measured as a percentage.

**Grey Box Testing :**Recent years the term grey box testing has come into common usage. This involves having access to internal data structures and algorithms for purposes of designing the test cases, but testing at the user, or black-box level. Manipulating input data and formatting output do not qualify as "grey-box," because the input and output are clearly outside of the "black-box" that we are calling "the software under test." (This distinction is particularly important when conducting integration testing between two modules of code written by two different developers, where only the interfaces are exposed for test.) Grey box testing may also include reverse engineering to determine, for instance, boundary values or error messages.

Acceptance Testing :

Acceptance testing can mean one of two things:

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

**Regression Testing**: Regression testing is any type of software testing that seeks to uncover software regressions. Such regressions occur whenever software functionality that was previously working correctly stops working as intended. Typically regressions occur as an unintended consequence of program changes. Common methods of regression testing include re-running previously run tests and checking whether previously fixed faults have re-emerged.

**NON FUNCTIONAL SOFTWARE TESTING:**

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

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

Testing can be done on the following levels:

* Unit testing tests the minimal software component, or module. Each unit (basic component) of the software is tested to verify that the detailed design for the unit has been correctly implemented. In an object-oriented environment, this is usually at the class level, and the minimal unit tests include the constructors and destructors.
* Integration testing exposes defects in the interfaces and interaction between integrated components (modules). Progressively larger groups of tested software components corresponding to elements of the architectural design are integrated and tested until the software works as a system.
* System testing tests a completely integrated system to verify that it meets its requirements.
* System integration testing verifies that a system is integrated to any external or third party systems defined in the system requirements.

Before shipping the final version of software, alpha and beta testing are often done additionally:

* Alpha testing is simulated or actual operational testing by potential users/customers or an independent test team at the developers' site. Alpha testing is often employed for off-the-shelf software as a form of internal acceptance testing, before the software goes to beta testing.
* Beta testing comes after alpha testing. Versions of the software, known as beta versions, are released to a limited audience outside of the programming team. The software is released to groups of people so that further testing can ensure the product has few faults or bugs. Sometimes, beta versions are made available to the open public to increase the feedback field to a maximal number of future users.

Finally, acceptance testing can be conducted by the end-user, customer, or client to validate whether or not to accept the product. Acceptance testing may be performed as part of the hand-off process between any two phases of development.There are a number of common software measures, often called "metrics", which are used to measure the state of the software or the adequacy of the testing.

**A Sample Testing Cycle:**

Although variations exist between organizations, there is a typical cycle for testing:

* Requirements analysis: Testing should begin in the requirements phase of the software development life cycle. During the design phase, testers work with developers in determining what aspects of a design are testable and with what parameters those tests work.
* Test planning: Test strategy, test plan, test bed creation. A lot of activities will be carried out during testing, so that a plan is needed.
* Test development: Test procedures, test scenarios, test cases, test datasets, test scripts to use in testing software.
* Test execution: Testers execute the software based on the plans and tests and report any errors found to the development team.
* Test reporting: Once testing is completed, testers generate metrics and make final reports on their test effort and whether or not the software tested is ready for release.
* Test result analysis: Or Defect Analysis, is done by the development team usually along with the client, in order to decide what defects should be treated, fixed, rejected (i.e. found software working properly) or deferred to be dealt with at a later time.
* Retesting the resolved defects. Once a defect has been dealt with by the development team, it is retested by the testing team.
* Regression testing: It is common to have a small test program built of a subset of tests, for each integration of new, modified or fixed software, in order to ensure that the latest delivery has not ruined anything, and that the software product as a whole is still working correctly.
* Test Closure**:** Once the test meets the exit criteria, the activities such as capturing the key outputs, lessons learned, results, logs, documents related to the project are archived and used as a reference for future projects.

**6.2 Design Of Test Cases/Scenarios:**

Admin/User Login Page

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **FUNCTION** | **EXPECTED REUSLTS** | **ACTUAL RESULTS** | **LOW PRIORITY** | **HIGH PRIORITY** |
| Should check for if username field is given or not | If the usename field is not given then should popup error message | If username field is empty it is showing the error message | yes |  |
| Should check for if password field is empty or not | If the password field is given empty then it should show error | If password field is given empty then it is displaying error | yes |  |
| Authentication | If given credentials are valid then redirect to respective page | Redirected to the respective page | -------------------- | Yes |
| Authentication | Should display error message | Displayed error message if the user credentials are not valid | -------------------- | Yes |

**6.3 VALIDATION**

The terms verification and validations are used interchangeably we will describe both these methods. Verification is the process of determining whether or not the products of given phase of software development fulfill the specifications established in the previous phase. These activities include proving and reviews.

Validation is the process of evaluating the software at the end of software development process, we find how well the software satisfies the requirement specifications.The requirement of the software starts with requirement document and requirement specifications without errors and specifying client’s requirements correctly.

The validation process of evaluating the developed system at the end is to ensure that it must satisfy all the necessary requirement specification. Requirement verification also checks the factors as completeness, consistency and testability of the requirements**.**

As we all know that testing plays a crucial role in evaluation of the system. That is in order to know whether the system working properly or not. In other words we can say that in order to know whether the system which we have developed will give the expected output or not can be know by doing the testing. Testing phase comes after coding phase .

Usually organizations or the software developing companies use different types of testing strategies in order to evaluate the performance of a system. Also it gives the output which provides clear information regarding the project or system , whether the project which we have developed will going to give the expected output or not , that is whether the system fails or succeed in the market.

We have many types of testing such as unit testing, integration testing, system testing, black box testing, white box testing and regression analysis testing and so on.In our project Secure Cryptographic messaging we are using unit testing, integration testing , and system testing.

Unit testing is the one in which each entity or objects in the module will be tested . Once the entity is evaluated to be tested successfully than will move further with the another kind of testing. That’s is once unit testing is done with all modules, than integration testing will be done, on the every module or on group of two or three modules. Finally system testing will be done , in which all the modules of a system will be tested at once , there by getting the overall performance of a system that means we can conclude the result on the entire system whether our system is working as per our requirements or as per our expectations or not. The advantage of developing or testing modules wise is that , we can reduce the effort, cost and time. Because if we are testing module wise than we can know clearly which module is working fine and which module is not working , thereby the module which is not working perfectly can be evaluated once again by going necessary modifications unlike the system being tested on a whole , where if any errors comes in than the entire system need to be tested or evaluated which consumes more effort , time and cost.

**6.4 CONCLUSION**

In this way we also completed the testing phase of the project and ensured that the system is ready to go live. Thus we developed a new technology so that every user can receive and send the mail the mails.

**CONCLUSION**

**7.CONCLUSION**

This project better suits for intra organization such Colleges, Company’s etc where they need efficient communication between users and effective control by the admin of the site.

FUTURE ENHANCEMENT :

As a future enhancement, in our application we can include study chats relating to the subjects by the students and the professors

**REFERENCES**

1. Jeff Poise (2002). Programming Microsoft .NET. Microsoft Press.

2. Asp.net general information and tutorials: http://www.asp.net

3. Building an ASP.NET Intranet, Wrox publication Jeff Poise (2002). Programming Microsoft .NET. Microsoft Press.

* Asp.net general information and tutorials: <http://www.asp.net>
* Building an ASP.NET Intranet, Wrox publication
* Microsoft developer’s references and help: <http://msdn.microsoft.com/asp.net/>
* Design Patterns C# by Steven metsker.
* The Unified Modeling Language- By Grady Booch
* Software Engineering- By R.S Pressman