****

**Fifth Semester Project**

**Web Mail Client:**

**Team Members:**

**Disha Shah**

**Nikki Punjabi**

**Zankhan Joshi**

**Zoebali Maknojia**

**(BCA STUDENTS)**

**ACKNOWLEDGEMENT**

This project has been a great mode of understanding and putting to practice what was covered during the academics. It has also been a valuable experience for us. It prepares us to apply ourselves better and become a successful IT professional. Of course, all this was not possible without the support and guidance of many people, and hence, we would like to take this opportunity to acknowledge their support towards us.

We are thankful to Dr. Vipul Kalamkar (Co-ordinator, B.C.A.) for his invaluable encouragement and inspiration.

We are also highly thankful to our project guide Mrs. Shefali Shah for her extreme patience, guidance and cordiality shown towards us which has helped in the process of development.

During this project we have received immense support and assistance from all the intellectual and helpful colleagues. It has modeled us intellectually and professionally. Our heartfelt thanks to all our colleagues and we wish them the best for their future.

**INDEX**

1. **UNIVERSITY PROFILE**

****

**1.1 Brief History of the Maharaja Sayajirao University**

In 1947, Maharaja PratapSingh Gaekwad pursued the idea of a separate University and appointed a Committee under the chairmanship of Shri K.H.Munshi. The committee submitted its report to the government of Baroda towardsthe end of the year 1948. And finally the Maharaja SayajiRao University was established on 26th of April, 1949.

The M.S.University of Baroda is a prestigious University in the Western India with whopping strength of the thirty five thousand students,twelve hundred teachers covering seventeen faculties and more than ninety departments/institutions and one thousand and nine hundred administrative staff. The M.S.U. provides education from Balwadi to higher Education.

**1.2 Objective of the University**

The main objective of establishing the Maharaja Sayajirao University of Baroda was to provide a distinct type of University – A teaching and residential University which should have complete freedom in all academic matters and would be free to institute new branches of studies suited to the needs and aspirations of the region in particular and of the country in general.

**1.3 Structure of the University**

The Maharaja University of Baroda is the Chancellor of the University. The Vice-Chancellor is the principal executive and academic officer of the University and is to be assisted in his work by Pro-Vice Chancellor or a Reactor. The Senate is the supreme governing body and the authority of the University and the Syndicate, its executive authority. Among the other authorities of the of the University is the council of Post-Graduate Studies and Research which has to deal with all matters relating to instruction, training and research in the various subjects taught in the University. The other authorities of the University are the Faculties and the Boards of Studies there under, which are principally responsible for making recommendations in the matter of the course of studies, textbooks and generally on the academic matters. The act deals with the residence of the students and lies down that all students of the University should reside in the hostels of the University or under conditions laid down by the relevant Ordinances. Provision has also been made in the act for the establishment of a Board as may be prescribed by the Statutes. Besides the usual fact Faculties of Arts, Science, Commerce, Medicine, Technology (including engineering), Law and Agriculture, the University is authorized to institute new Faculties of Education and Psychology, Home Science, Fine Arts and Social Work.

**1.4 Establishment of the University**

Under the Government of Baroda Legal Remeberancer’s Notification No.41 dated 30th April 1949, read with its amendment of the same date, all the above provisions of the Act, Except Section 5, which provided for the University of Bombay to this University, were brought into operation from the 30th April 1949, and thus the University was established from the date.

**1.5 History of Science Faculty**

In 1926, the then Viceroy and Governor General of India, Sir Issac Rufus laid the foundation stone of Sayaji Jubilee Institute under Baroda college, which soon became a leading institution of higher education and research in Science. With the establishment of the University in 1949, all the science departments of Baroda College were reconstituted and the Faculty of Science was created. Since 1949, the Faculty offers courses leading to B.sc., M.Sc. and Ph.D. degrees in various fields of science. Later the Faculty made significant strides in various disciplines of science that attracted students from all over India and other countries and soon it developed into a cosmopolitan center of higher education in science. The growth of the Faculty has thus complemented the pioneering efforts of the great visionary, the Maharaja Sayajirao Gaekwad, who had taken pains to improve science education in and Baroda.

**1.5 Overview of B.C.A.**

It is a three-year Programme leading to the degree “Bachelor of Computer Applications” (BCA). The course is aimed at developing the computer professionals in the wide range of application areas such as business sector, government sector, scientific research, medical science, social sciences, management, communication and various other fields of the IT boom. They give their students a platform to develop their software development skills practically as well as through research and frequent project works.

1. **INTRODUCTION TO THE PROJECT**

# 2.1Project Profile

|  |  |
| --- | --- |
| **Project Title :** | Webmail Client |
| **Objective :** | Personalize webmail client for an organization which is able to send and receive mails from inside and outside domain. |
| **Institution :** | Bachelors of Computer Applications Programme.  Faculty of Science,  Maharaja Sayajirao University of Baroda. |
| **Project Guide :** | Mrs.Shefali Shah |
| **Duration :** | 4 Months |
| **Programming Language :** | Java |

**2.2 Project Definition**

Webmail client is an application used for sending and receiving mails with or without attachments through a browser within or outside the domain.

**2.3 Project in brief**

Webmail runs on any computer which has an internet or intranet connection and a browser. The client does not need a computer with their mail application installed in it. This application is based on client-server architecture. Thus, it is called Webmail client. It enables user to send, save, receive or delete-mail. Mail from the user is sent to a central server through a webmail client and then the server routes the mail to its intended destination. The client can save the unsent mails in drafts. The client can send mails with or without attachments.

**2.4 Purpose**

The purpose of this webmail client is to make a webmail client for an organization such that they can send or receive mail with or without attachment from inside or outside domain without installing or configuring the application.

**2.5 Project scenario and scope**

Organization can do mailing activities inside or outside domain only after installing or configuring the application. Most of the organizations use Microsoft Outlook which needs to be installed and configured on each computer to be used. It becomes very tedious and difficult to install and configure in all the machines in the organization. Maintaining the application becomes unfeasible.

So in demand to bridge this gap and smoothen the functioning of mailing activities in an organization, this webmail client will be a boon. The webmail client can be accessed by anyone in the organization without any configuration. The client can send or receive mails easily within or outside the domain.

**2.6 Objectives of Project**

In today’s scenario, most of the organizations use applications that need to be configured and installed in all the machines of the organization. So the main objective is to develop a webmail client through which user can send or receive mails with or without attachments from inside or outside the domain.

The objectives of our application are:

* **Ease of use :**

Nowadays, everyone is comfortable with using internet. This webmail client works simply as a website which can be easily used by anyone who is comfortable using websites.

* **Speed :**

The webmail client is very light-weight. So, the browser loads it fast depending upon the internet speed.

* **Security :**

One of the prime aspects of any system is the security.

The webmail client authenticates the user before he starts his mailing activities. The webmail client is very secure in a way that if user forgets to sign out or doesn’t do any activity for 15 minutes then the screen would freeze and the user will have to login again. Thus, it is secured and minimizes the chance of anyone else mishandling the account.

* **MIME supported :**

User can send or receive mails with attachments. Attachments can be done through MIME. **Multipurpose Internet Mail Extensions** (**MIME**) is an Internet standard that extends the format of email to support text in character sets other than ASCII, non-text attachments, message bodies with multiple parts and header information in non-ASCII character sets. E-mails are sent through SMTP in MIME format. This feature is supported by Webmail client.

**2.7 Problem Domain**

In Today’s scenario almost all companies and industries use an application that needs to be configured or installed in every machine to do their mailing activities inside or outside domain. It becomes a very tedious task to configure and maintain in all machines.

In order to simplify and make the task easy, we have come out with a solution by providing a webmail client which can work on intranet as well as internet and thus send and receive mails from inside and outside the domain. User can also send or receive mails with any kind of attachments.

**2.8 Benefits of Webmail client**

* **Ease of access**

Webmail client just needs an internet connection. Thus, it gives flexibility to the user to work from anywhere.

* **Independent of server crashing**

Webmail client works on single server but if due to any reason the server crashes then there has to be a backup which could be easily done. Webmail client can be used on any other sub-parallel server if the main server crashes down by only changing the server address.

* **Light-weight application**

Webmail client is very light-weight. So, the browser loads it quickly depending on the intranet and internet speed.

* **Avoid misuse of user account**

The webmail client authenticates the user before he starts his mailing activities. The webmail client is very secure in a way that if user forgets to sign out or doesn’t do any activity for 15 minutes then the screen would freeze and the user will have to login again. Thus, it is secured and minimizes the chance of anyone else mishandling the account.

**2.9 Technology and Literature Review**

* **Java** as a programming language.
* Protocols implemented :
* SMTP :

**Simple Mail Transfer Protocol** (**SMTP**) is an Internet standard for electronic mail (e-mail) transmission across Internet Protocol (IP) networks.SMTP is specified for outgoing mail transport and uses TCP port 25.SMTP is last updated by RFC 5321, which includes the extended SMTP.

* IMAP :

Internet Message Access Protocol is an Internet standard for receiving mails from the server. The current version IMAP4rev1 is defined by RFC 3501. IMAP allows storing e-mail on the server and accessing it by sending a request to the server. It supports deletion of messages directly from server. Thus, maintaining mailbox will ne easier for user.

* MIME format is followed.

**Multipurpose Internet Mail Extensions** (**MIME**) is an Internet standard that extends the format of email to support text in character sets other than ASCII, non-text attachments, message bodies with multiple parts and header information in non-ASCII character sets. E-mails are sent through SMTP in MIME format. The goals of MIME were achieved by using RFC 1521 and RFC 1522. MIME defines several header fields like Content-type, Content-Transfer-Encoding which are very useful for developers.

* **Microsoft Word** for documentation.
* **E-Draw** for UML diagrams.

1. **Project Management**

**3.1 Software Development Model**

A process model is a development strategy is used to achieve a goal that satisfies the requirements abiding by the constraints. There are many types’ software process models like linear sequential model, water fall model, spiral model, V model, etc. But we have decided to use spiral model for our project.

This model is an iterative model and was devised to address the difficulties of sequential models. This model contains all activities involved in waterfall model, and also adds activities such as risk management, software reuse, and prototyping**.**

Why Spiral Model ?

* It is not possible to get design appropriately, right at the beginning.
* Requirements will be frozen.
* Revisiting to earlier stage is allowed to correct the mistakes, to take care of the new requirements.



**ABOUT SPIRAL MODEL**

In the model the first quadrant consists of determining the objectives, Requirements and constraints.

The second quadrant consists of Analyzing &evaluating the alternatives, identifying resolving the risks and designing the software.

The third quadrant consists of developing and verifying the product (Component of the Software).

The forth quadrant consist of planning the next Component.

Software development is risk driven development. If an activity is performed in such a manner that no risk occurs and if any risk occur then preparedness is there to solve the problem, such a way of performing an activity is said to be the risk driven way of performing the activity. The macro level understanding is same for round to round, but micro level planning may be different in rounds.

Requirement validation means to see that the requirement is solving a business problem or not.

**4.2 Project Plan**

Project Management is the ensemble of activities (such as tasks) concerned with successfully achieving a set of goals. This includes planning, scheduling and maintaining progress of the activities that comprise the project. Reduced to its simplest project management is the discipline of maintaining the risk of failure at as low a value as necessary over the lifetime of the project Risk of failure arises primarily from the presence of uncertainty at all stages of a project.

An alternate point of view is that project management is the discipline of defining and achieving targets while optimizing the use of resources (time, money, people, space, etc). In the case of Project Management, project plays an important role.

A Project Plan is “A formal, approved document used to guide both project execution and project control. The primary uses of the project plan are to document planning assumptions and decisions, facilitate communication among stakeholders, and document approved scope, cost, and schedule baselines. A project plan may be summary or detailed.”

* Process Impact:

This plan will be used to evaluate and manage the project. Key assumptions that affect the plan should be updated throughout the life-time of the project.

* Summary of the Project:

This system will enable the user to send or receive mails with or without attachments from inside or outside the domain.

* Summary of the Methodology:

To design this software we will use the object oriented design (OOD) methodology. Because it provides features like reuse, quality, an emphasis on modeling the real world, resistance to change, encapsulation and abstraction etc. Only the OOD provides the mechanism that enables the designer to achieve above with less complexity. We will also follow SMTP and IMAP protocols.

* How will changes be controlled?
* Requests for requirement changes will be tracked in the issue tracker.
* The change control will review requested changes and authorize work on them as appropriate.
* After the feature complete milestone, no new features will be added to this release.
* After the code complete milestone, no entirely new product source code will be added to this release.

**3.4 Risk Management**

Risk management is the identification, assessment, and prioritization of risks (as *the effect of uncertainty on objectives*, whether positive or negative) followed by coordinated and economical application of resources to minimize, monitor, and control the probability and/or impact of unfortunate events or to maximize the realization of opportunities.

**3.5 Risk Identification**

Risk identification is a systematic attempt to specify threats to the project plan (estimates, schedule, resource loading etc.). By identifying known and predictable risks, the project manager takes the first step towards avoiding them when possible and controlling them when necessary.

This project entails the following risks:

1. Dependant on single server.
2. Security issues if user forgets to log out.

Other various risks identified:

* Achieving Project scope.
* Communication problem.
* Risk of not meeting schedule.
* Integrating the functionalities.
* Client changing the requirements.

**3.6 Risk Analysis**

Project risk events consist of three factors:

* + - The nature of the project risk event
    - The likelihood or probability of its occurrence
    - The amount at stake

In the above section identified risk are discussed in detail below:

1. Dependent in a single server

Webmail client works on a single server. If due to some reason server crashes down the webmail client won’t work. This could prove fatal to the organization.

1. Security issues if user forgets to logout

If user forgets to logout, then someone else who sits on that machine can hamper or misuse the user’s account.

**3.7 Risk Planning**

In this section, we, as a team, offer a few solutions on the above discussed risks:

1. Dependant on single server :

Webmail client works on single server but if due to any reason the server crashes then there has to be a backup which could be easily done. Webmail client can be used on any other sub-parallel server if the main server crashes down by only changing the server address.

1. Security issues if user forgets to log out :

If user forgets to log out and if anyone else comes then he can misuse the user’s account. Thus to protect, the webmail client freezes if it is idle for 15 minutes. User can resume work only after refreshing the page and log in again.

1. **REQUIREMENT STUDY AND ANALYSIS**

**4.1 Study of the current System**

The Process of establishing the services that the user requires from a system and constraints under which it operates is developed. The requirements themselves are the description of the system services and constraints that are generated during the requirement’s engineering process.

**What is Requirement?**

It may range from a high-level abstract statement of a service or of a system constrains to a detailed mathematical function specification.

Requirement Elicitation also sometimes called as **requirement discovery**. It involves technical staff working with customer to find out about the application domain, the service that the system should provide and the system’s operational constraints. It may also involve end-users, managers, engineers, involved in maintenance, domain experts, trade, and etc. called **stakeholders**.

According to our perspective, three endemic syndromes complicate Requirement Elicitation.

1. The “**yes, but**” syndrome stems from human nature and the user’s inability to experience the software they might a physical device.
2. Searching for requirement s is like searching for “**Undiscovered Ruins**”; the more you find, the more you know remain.
3. The “**User and the Developer**” syndrome reflects the profound differences between the two, making communication difficult between the user and the developer.
   1. **User Characteristics**

User can be any organization that needs webmail client for their domain from where they can do mailing activities from within or outside the domain with ease.

Webmail client has been designed in such a way that any organization can personalize it and use it for their mailing activities. The end users of this application will be employees of an organization who will be able to send or receive mails with or without attachment from within or outside the domain over the internet.

**4.3 System Requirement**

* **Software Requirement**
  + **Development :**

|  |  |
| --- | --- |
| Programming Language…………………. | Java |

* **Implementation :**

Internet

Mail Server

**4.4 User Requirement**

* **Mail access from anywhere :**

In Today's scenario, most of the organizations use application that needs to be configured on every machine in an organization. Another limitation was that the user had to do work only at office and could not access it from home. Thus, user wants access to mailing activities from anywhere and without configuring anything.

* **Security :**

One of the prime aspects of any system is the security.

The webmail client authenticates the user before he starts his mailing activities. The webmail client is very secure in a way that if user forgets to sign out or doesn’t do any activity for 15 minutes then the screen would freeze and the user will have to login again. Thus, it is secured and minimizes the chance of anyone else mishandling the account.

* **Speed :**

The webmail client should be light weight so that it can be quickly loaded.

* **MIME supported :**

User can send or receive mails with attachments. Attachments can be done through MIME. **Multipurpose Internet Mail Extensions** (**MIME**) is an Internet standard that extends the format of email to support text in character sets other than ASCII, non-text attachments, message bodies with multiple parts and header information in non-ASCII character sets. E-mails are sent through SMTP in MIME format. This feature is supported by Webmail client.

* 1. **Functional Requirements**
* Sending mails with/without attachment.
* Receiving mails.
* Checking sent mails.
* Deleting mails.
* Restore mails.
* Searching mails.
  1. **Non – Functional Requirements**
* Ease of use :

Nowadays, everyone is comfortable with using internet. This webmail client works simply as a website which can be easily used by anyone who is comfortable using websites.

* Speed :

The webmail client is very light-weight. So, the browser loads it fast depending upon the internet speed.

* Security :

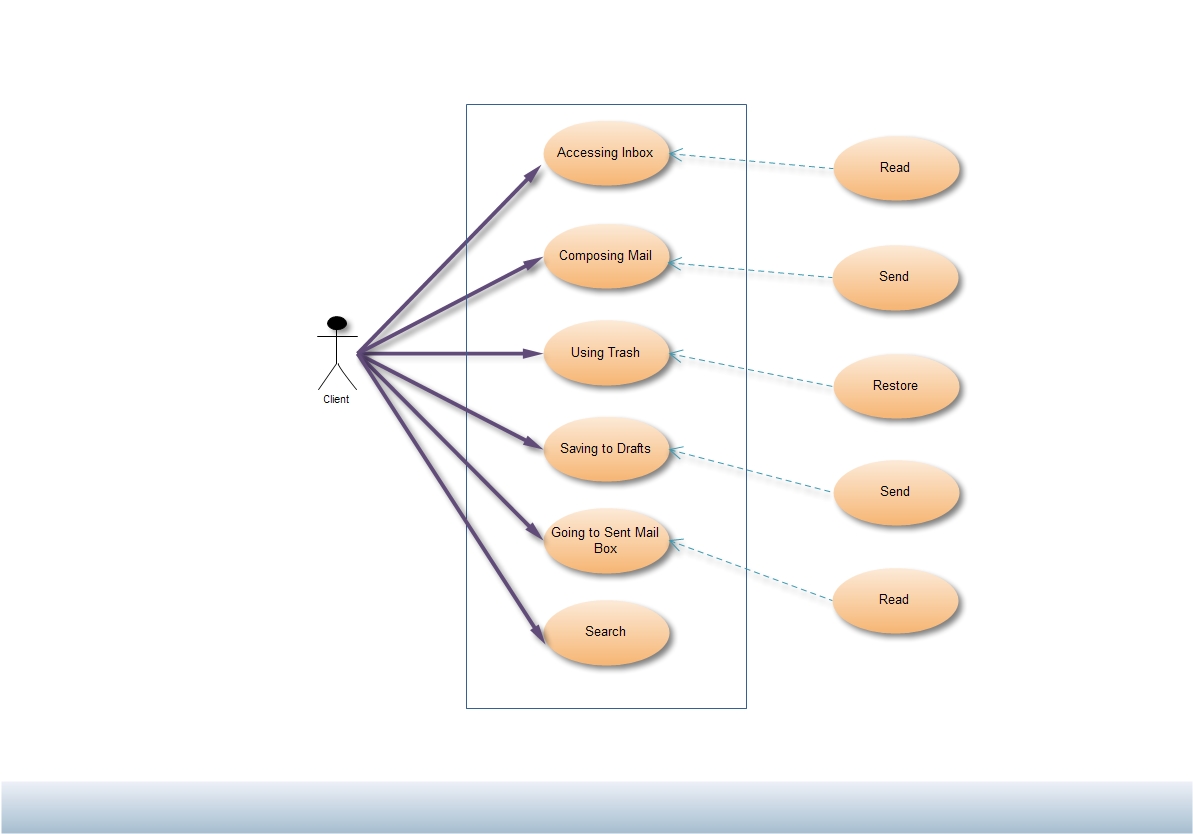
One of the prime aspects of any system is the security.

The webmail client authenticates the user before he starts his mailing activities. The webmail client is very secure in a way that if user forgets to sign out or doesn’t do any activity for 15 minutes then the screen would freeze and the user will have to login again. Thus, it is secured and minimizes the chance of anyone else mishandling the account.

* MIME supported :

User can send or receive mails with attachments. Attachments can be done through MIME. **Multipurpose Internet Mail Extensions** (**MIME**) is an Internet standard that extends the format of email to support text in character sets other than ASCII, non-text attachments, message bodies with multiple parts and header information in non-ASCII character sets. E-mails are sent through SMTP in MIME format. This feature is supported by Webmail client.

**5.Use-Case diagram**

****

The use-case diagram displays the functionality provided by the Application (System) to its users (actors).

In this, Webmail client is the system, which provides the following functionalities to its concerned users (in this case it’s anyone who visits this webmail client i.e Client).

1. **Accessing Inbox:**

After successful login, the application provides user accessibility to Inbox where all incoming mails will be populated. This activity extends other activity: Reading. Incoming mails can be read by the user.

1. **Composing mail:**

The application provides the user facility to compose mails. This activity extends other activity: Send. The composed mail with or without attachment can be sent to anyone inside or outside the domain.

1. **Using Trash:**

The application provides the user facility to delete mails. Deleted mails are stored in Trash. It might happen that the user might have deleted by mistake and might want to restore it. This restoration of deleted mails is supported by the application.

1. **Saving to Drafts:**

The application provides the user to save the unsent mails. User might want to send mail later for some reason then he can save that in Drafts. Thus, sending mail is extended activity of this activity.

1. **Going to Sent Mail Box :**

This application provides the user to read the sent mails. The sent mails are stored in Sent Mailbox.

1. **Search:**

This application provides the user to search mail anywhere in their mailbox.

1. **Restored Mail:**

This application provides the user to read the restored mail. The mails that are restored from the trash folder are sent to Restored Mail.

**5.1 Schedule**

|  |  |
| --- | --- |
| **Month** | **Activity** |
|  |  |
| August | Studying the Requirements. |
|  |  |
| September |  |
| 3rd week | Planning |
| 4th week | Analyzing the Functionalities. |
|  |  |
| October |  |
| 1st week | Analysis and Design |
| 2nd week | Implementation |
| 3rd week | Implementation |
| 4th week | Implementation |
|  |  |
| November |  |
| 1st week | Implementation |
| 2nd week | Implementation |
| 3rd week | Testing |
|  |  |
| December |  |
| 2nd week | Refactoring of project |
|  |  |

* Documentation has been done simultaneously.

**6. Glossary of Diagrams**

* 1. **Activity Diagram:**

The easiest way to visualize an Activity diagram is to think of a flowchart of a code. The flowchart is used to depict the business logic flow and the events that cause decisions and actions in the code to take place.

Activity diagrams represent the business and operational workflows of a system. An Activity diagram is a dynamic diagram that shows the activity and the event that causes the object to be in the particular state.

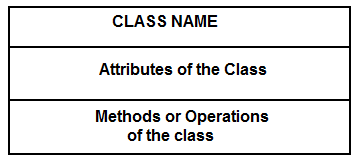
So, what is the importance of an Activity diagram, as opposed to a State diagram? A State diagram shows the different states an object is in during the lifecycle of its existence in the system, and the transitions in the states of the objects. These transitions depict the activities causing these transitions, shown by arrows.An Activity diagram talks more about these transitions and activities causing the changes in the object states. Let us take a look at the building blocks of an Activity diagram.

An Activity diagram consists of the following behavioral elements:

|  |  |
| --- | --- |
| Element and its description | Symbol |
| Initial Activity: This shows the starting point or first activity of the flow. Denoted by a solid circle. This is similar to the notation used for Initial State. | http://www.developer.com/img/articles/2003/08/11/UML07T1.gif |
| Activity: Represented by a rectangle with rounded (almost oval) edges. | http://www.developer.com/img/articles/2003/08/11/UML07T2.gif |
| Decisions: Similar to flowcharts, a logic where a decision is to be made is depicted by a diamond, with the options written on either sides of the arrows emerging from the diamond, within box brackets. | http://www.developer.com/img/articles/2003/08/11/UML07T3.gif |
| Merge Node :This node has several inputs and only single output.Its purpose is the merging of flows. The inputs are not synchronized. If a flow reaches such a node it proceeds at the output without waiting for the arrival of other flows. | http://sourcemaking.com/files/sm/images/uml/img_43.jpg |
| Final Activity: The end of the Activity diagram is shown by a bull's eye symbol, also called as a final activity. | http://www.developer.com/img/articles/2003/08/11/UML07T6.gif |

* 1. **Class Diagram**

The class diagram is the main building block in [object oriented](http://en.wikipedia.org/wiki/Object_oriented) modeling. It is used both for general [conceptual modeling](http://en.wikipedia.org/wiki/Conceptual_model) of the systematic of the application, and for detailed modeling translating the models into [programming code](http://en.wikipedia.org/wiki/Programming_code). The classes in a class diagram represent both the main objects and or interactions in the application and the objects to be programmed. In the class diagram these classes are represented with boxes which contain three parts:



A class with three sections.

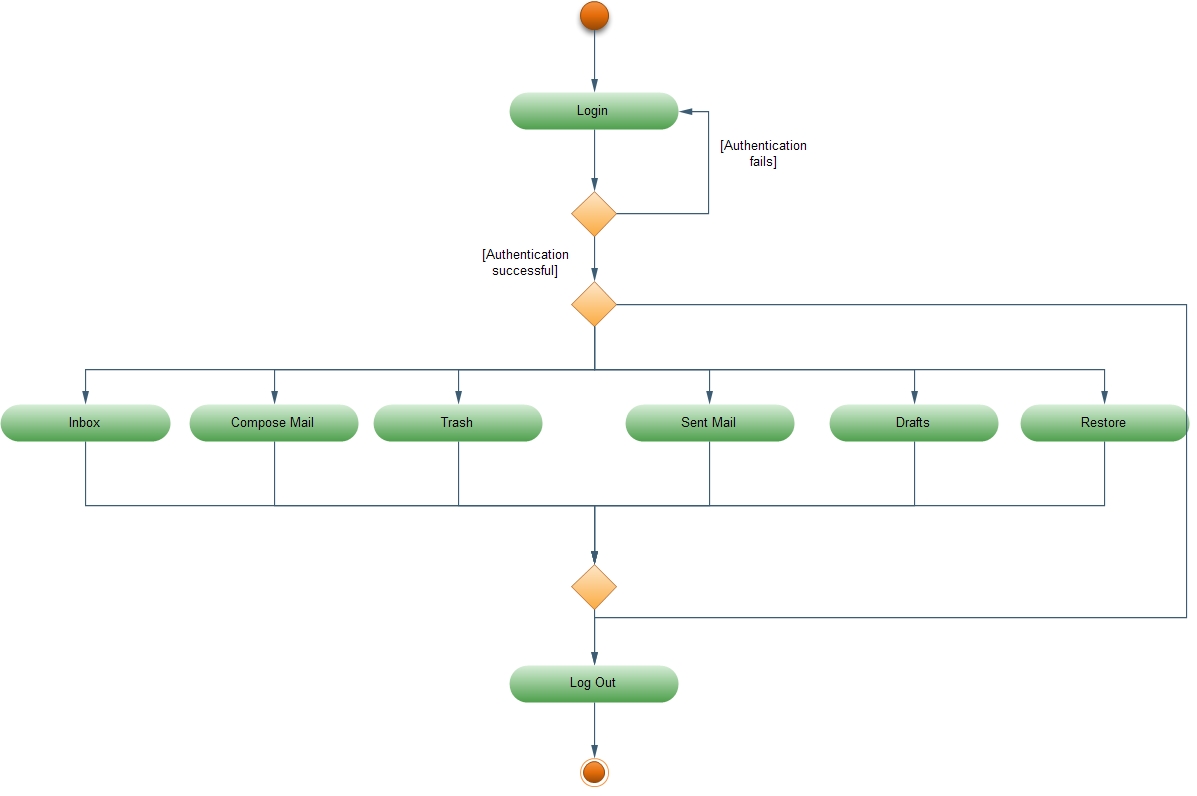
* The upper part holds the name of the class
* The middle part contains the attributes of the class
* The bottom part gives the methods or operations the class can take or undertake

In the system design of a system, a number of classes are identified and grouped together in a class diagram which helps to determine the statical relations between those objects. With detailed modeling, the classes of the conceptual design are often split in a number of subcl

**6.1.1 Webmail Client**

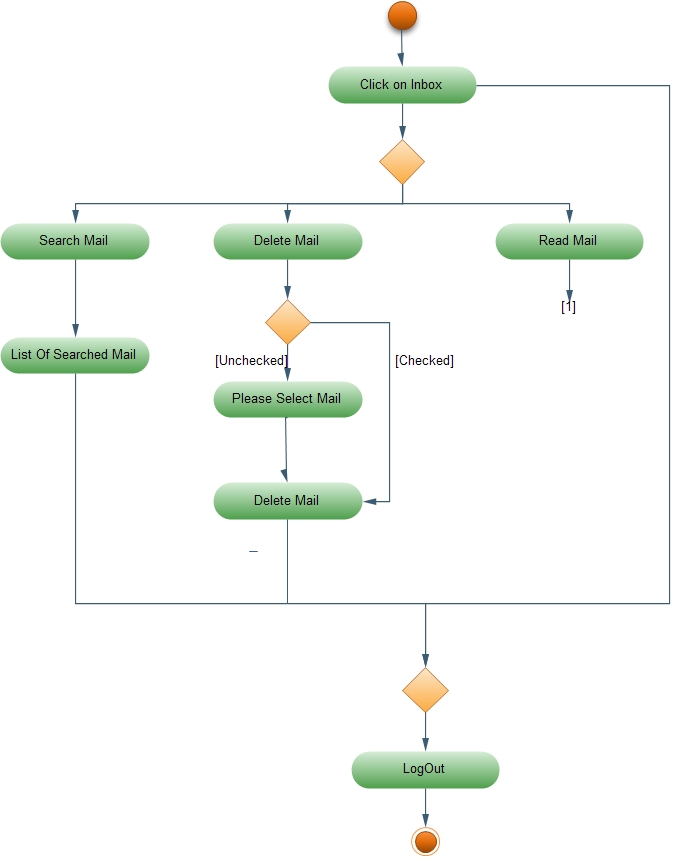
**Activity Diagram**

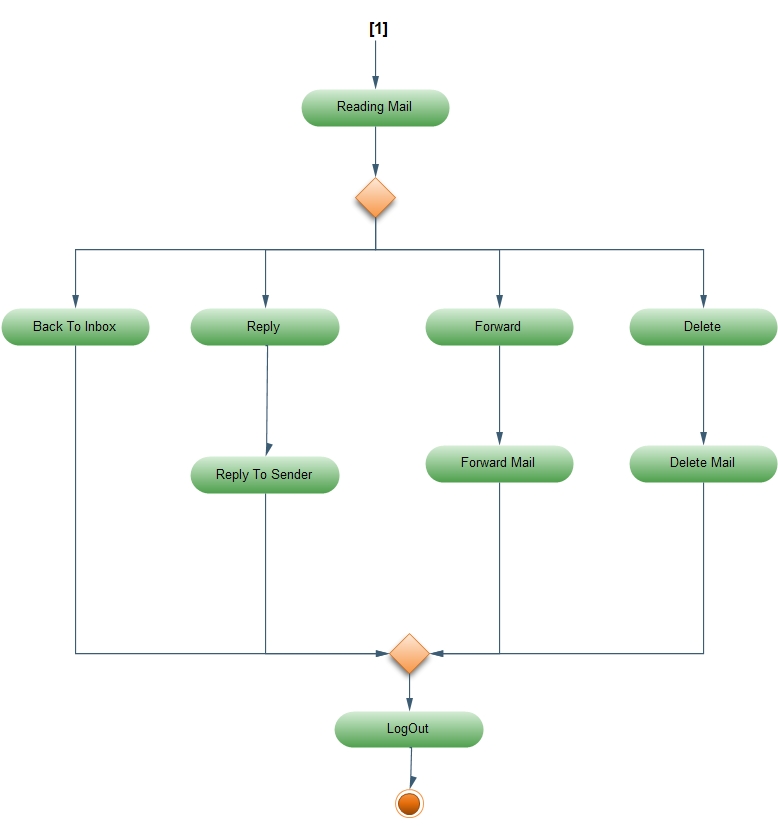
**Login**

****

**This login activity diagram gives description of user login. When the user opens the web mail client, webmail client asks for username and password. These credentials are sent to mail server to check if the user exists and if exists it opens his mailbox or else redirect to the same page. User can compose, read, sent or delete mails. On click of Inbox, all the incoming mails of the user are populated. On click of Compose mail, user can create new mail with or without attachment. If user doesn’t want to send then, he can save that mail. It is saved in Drafts. On click of Drafts, user can view those saved mails. Deleted mails by user in the same session are saved in Trash. On click of Trash, all the deleted mails by the user are populated. On click of Sent mail, all the sent mails by the user are displayed which can be read by the user. On click of restore user can view restored mails. User can log off from any page when clicked on Logout.**

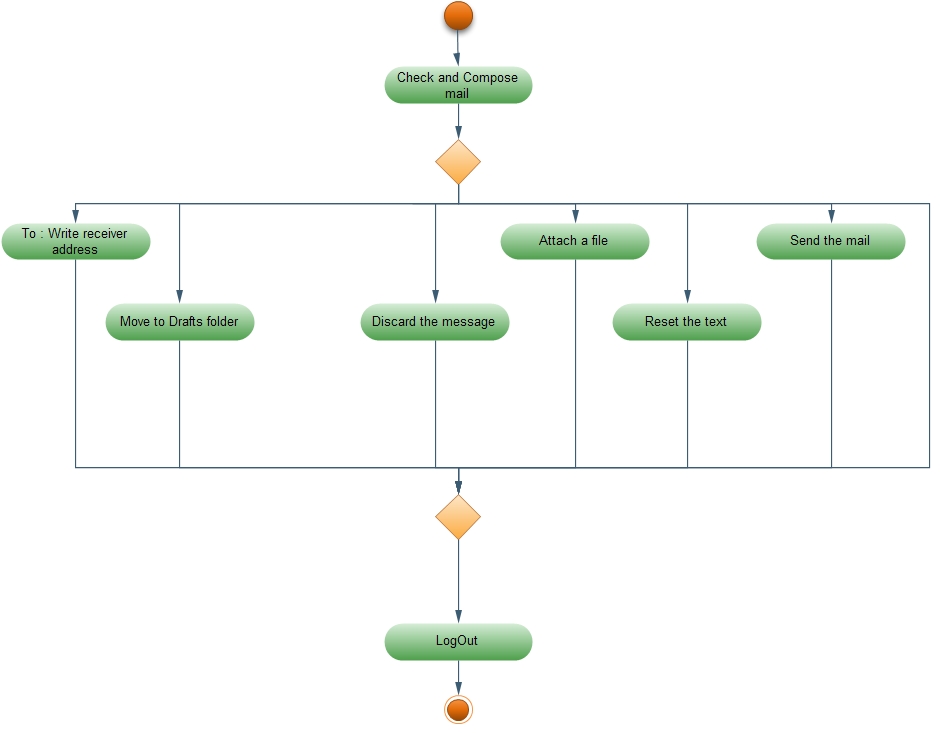
**Clicking on Inbox**



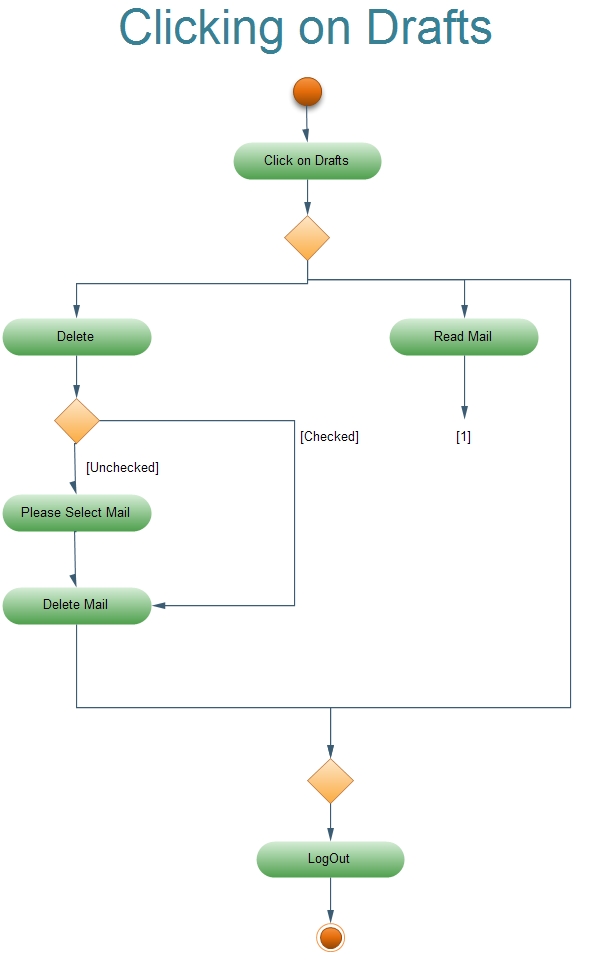
****

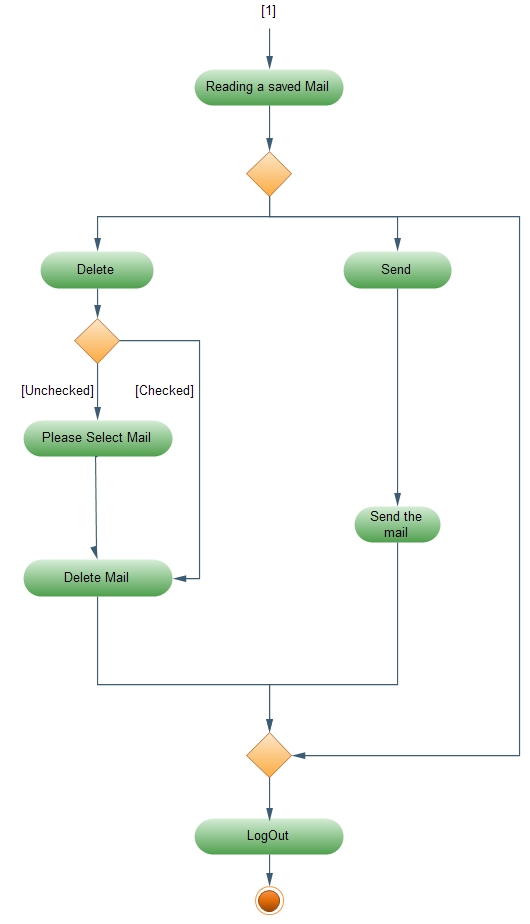
**On click of Inbox, all the received mails by the user are populated. User can read, search or delete mails to folder. On click of Search, request for the searched item will be sent to mail server and the list of matched mails will be populated. On click of Delete, Webmail client will check whether mails are checked or not. If checked, the selected mails will be deleted. If not, message will be displayed asking user to select mails to delete. User can read mails by clicking on it. After opening the mail, user can either reply, forward, delete or can go back to Inbox. On clicking Reply, webmail client will open new mail which will have the address of receiver in to: field. On clicking Forward, webmail client will open new mail with the same content. User has to just mention the addresses to whom the mail has to be forwarded. On clicking Delete, the current mail which user is viewing will be deleted. Users can logout by clicking on Logout.**

**Compose Mail**

****

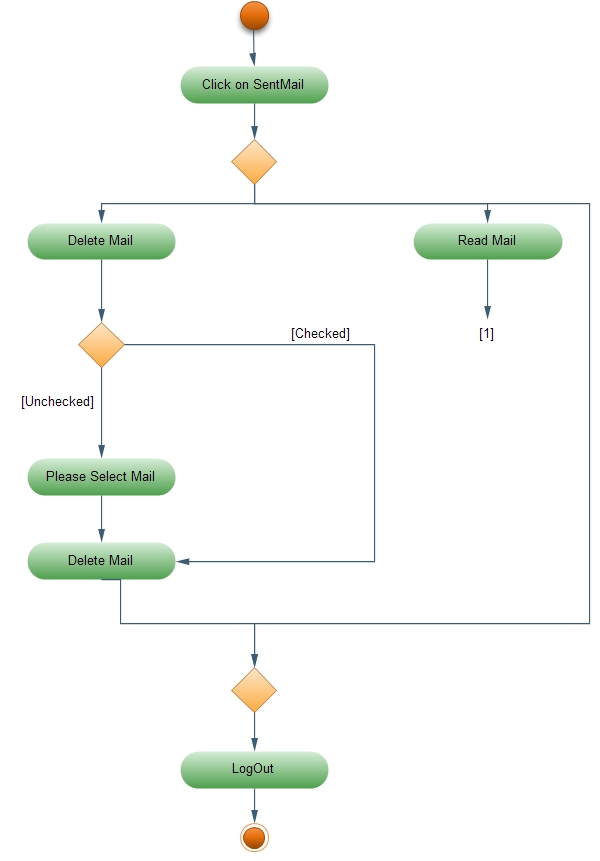
**On clicking Compose, user can compose a new mail. To send mail, it is mandatory for the user to fill to: textbox where the address of recipient client has to be written. On clicking Attach, user can attach file to the email. On clicking Send, the mail will be send to the intended recipients. On clicking Reset, user can reset all the fields in that mail. User can save the email to Drafts folder. User can also discard the email by clicking on Discard. User can log off by clicking on Logout.**

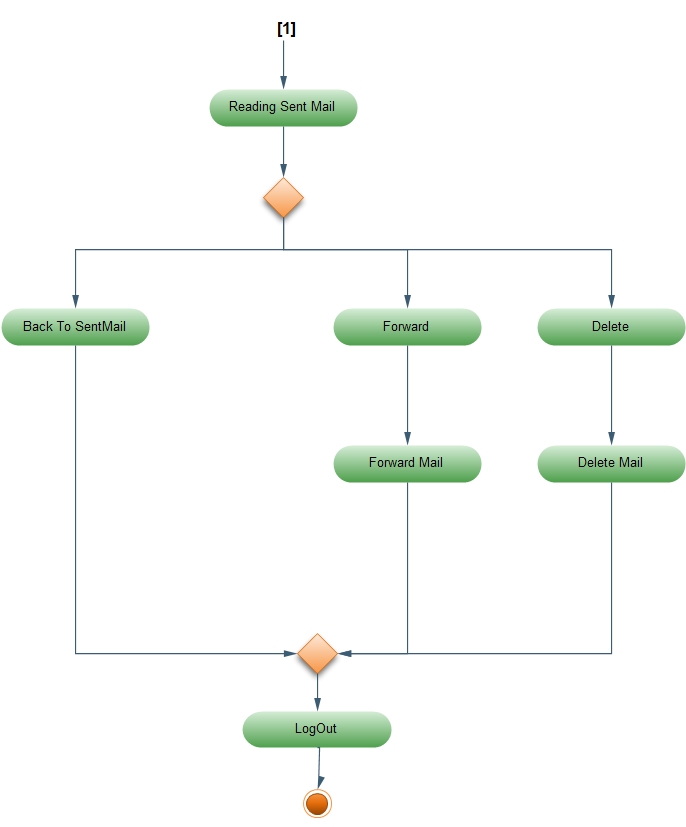
****

****

**On clicking Drafts, the saved and unsent mails will be populated. User can either read or delete the mail. On click of Delete, Webmail client will check whether mails are checked or not. If checked, the selected mails will be deleted. If not, message will be displayed asking user to select mails to delete. User can read mail by clicking on the mail, which he wants to read. After the mail opens, user can either send or delete mail. On clicking Send, the user will be able to compose the mail for sending in which all the original content of the mail would already be there. User can log off by clicking on Logout.**

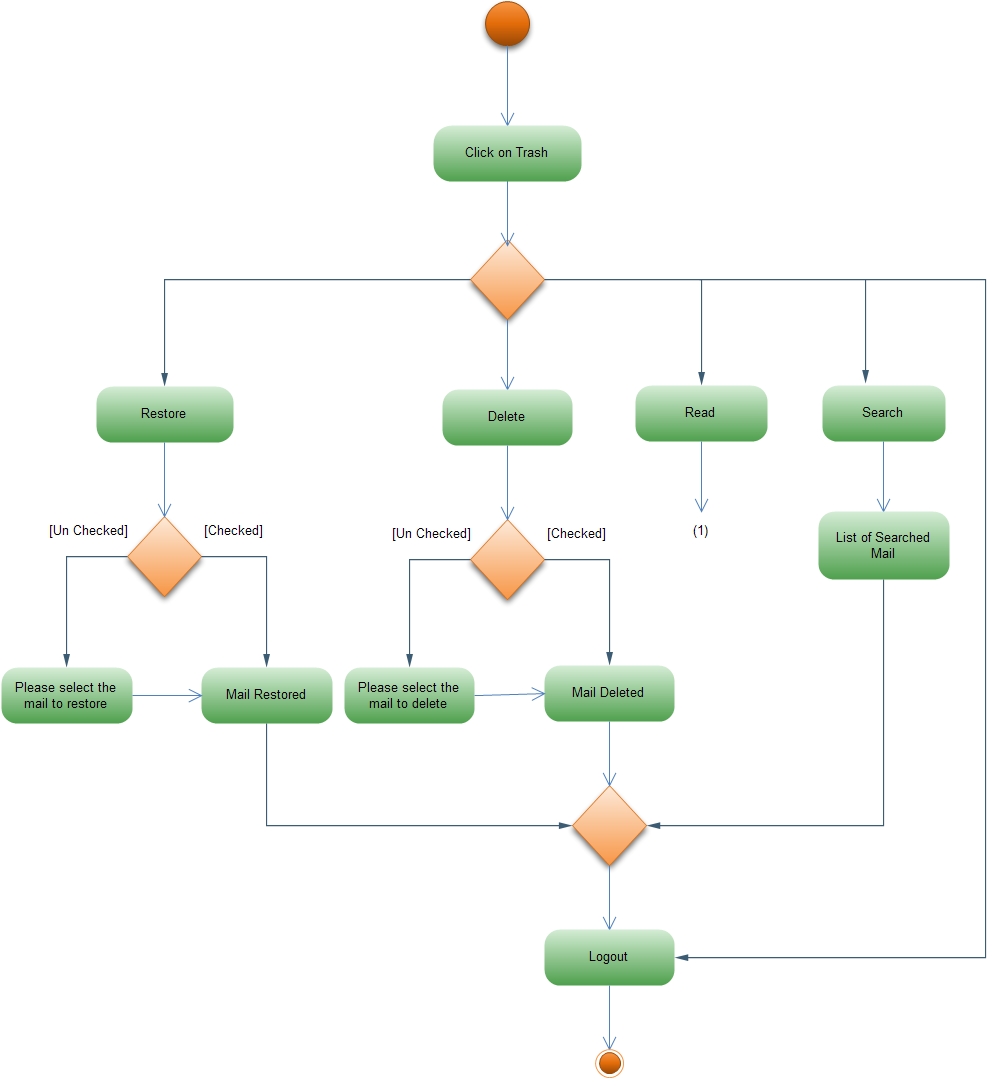
**Clicking on Sent Mail**

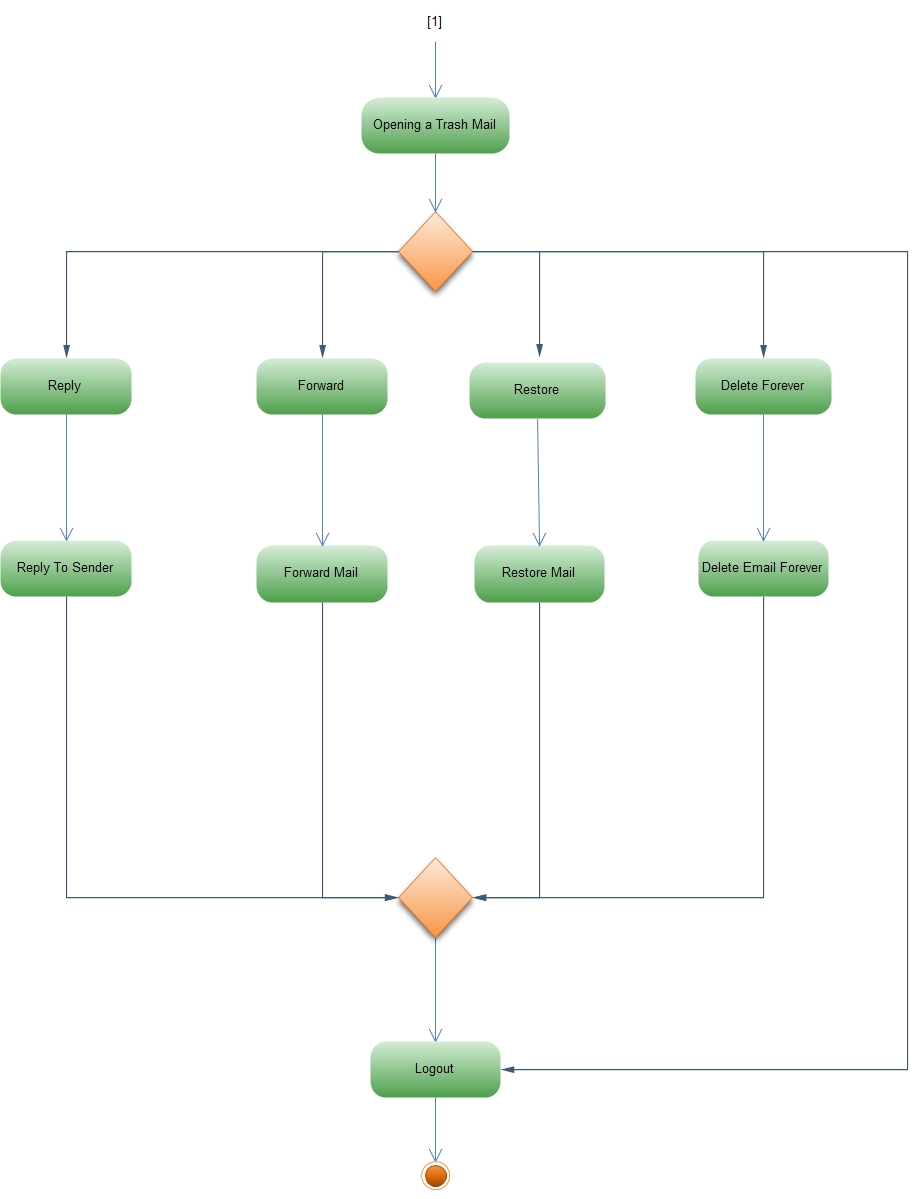
****

****

**On clicking Sent Mail, user can see all the mails sent by him. User can either read or delete mails. On clicking Delete, Webmail client will check whether mails are checked or not. If checked, the selected mails will be deleted. If not, message will be displayed asking user to select mails to delete. On clicking any mail, that mail will open and the user can read it. User can either forward or delete that opened mail. User can forward the mail to others by clicking on Forward. A new mail will be opened with the same content. User can then send that mail to the new specified recipients. User can log off by clicking on Logout.**

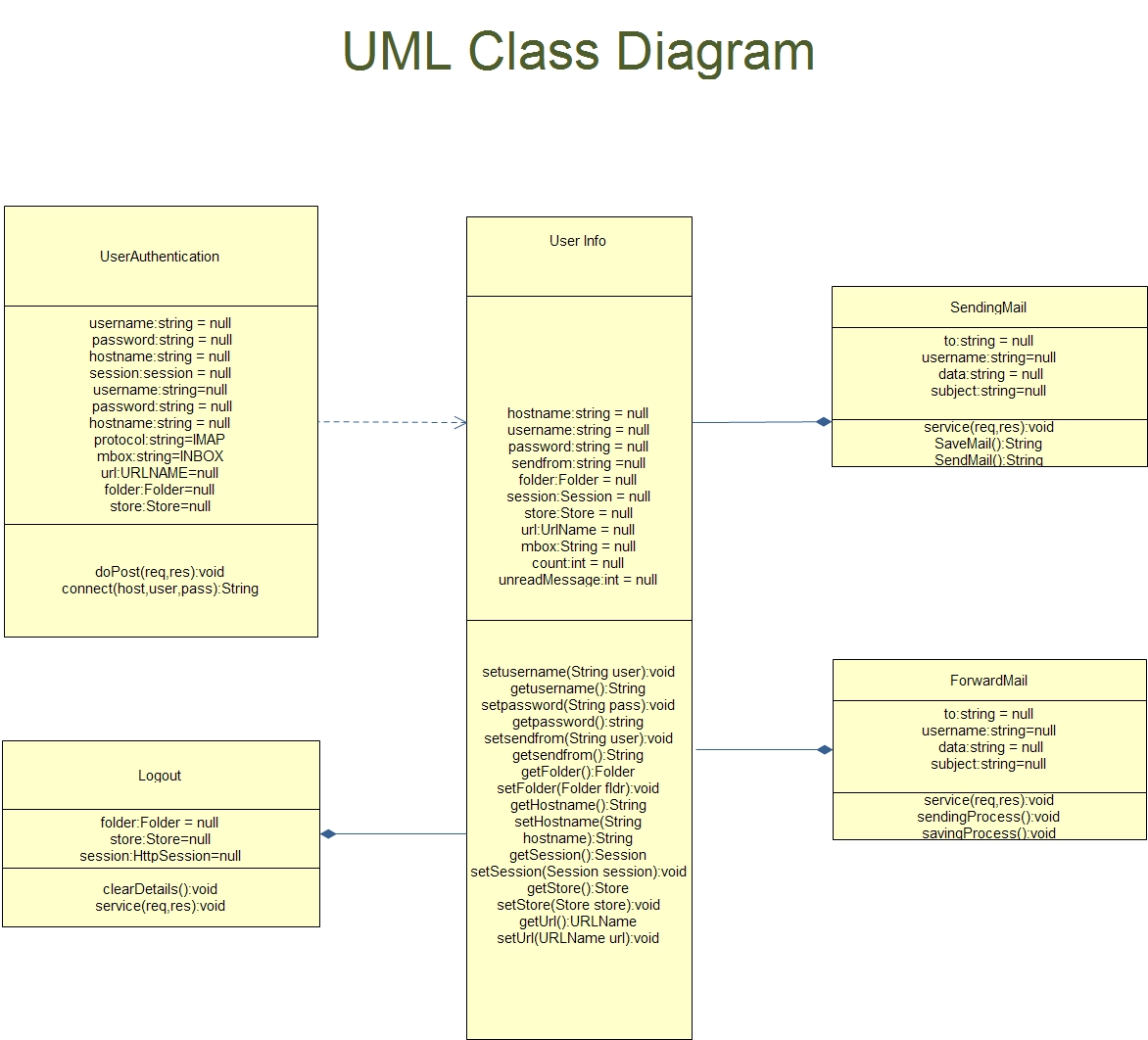
**Click on Trash**

****

****

**On clicking Trash, user can see all the deleted mails. User can search, read, delete permanently or restore the mail back to the restored folder. On clicking Search, request for the searched item will be sent to mail server and the list of matched mails will be populated. On clicking Delete, Webmail client will check whether mails are checked or not. If checked, the selected mails will be deleted permanently. If not, message will be displayed asking user to select mails to delete. When user clicks on Restore button the webmail client checks whether any mails are selected or not. If selected, the selected mails will be restored to Restored Mail Folder. If not, user will be asked to select mails to be restored. On clicking the mail, user can read that mail. User can then either restore or delete the opened mail. On clicking Delete, the opened mail will be deleted. On clicking Restore, the opened mail will be restored to Restored Folder. User can log off by clicking on Logout.**

**6.2.1 Webmail Client – Class Diagrams**

****

This Class diagram describes the interaction between the classes to classes.

When the user login’s, user will click on submit button. These value will be passed to UserAuthentication Class using onSubmit () Method. This Class the try to open the store of the specified user at specified hostname, if user’s store exist it will set the user required information to userinfo class which will be useful for displaying user’s store information. After setting this class will redirect user to home page. If user’s store is not found and login is unsuccessful user will be redirected to login page.

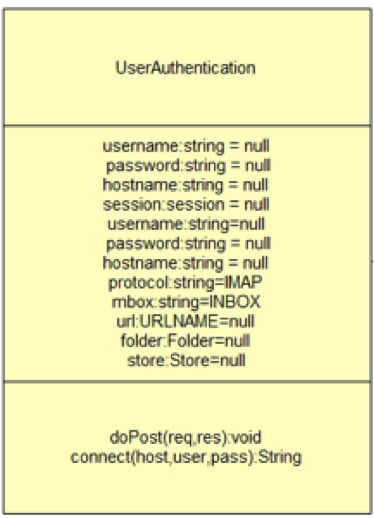
Whenever authentication is successful authenticator class will set the user’s information into userinfo class. So if any page have to display information about the user’s store based on user clicking it will collect the required information first from the userinfo class and will process the data and will give provide the required information to the user.If authentication fails authenticator will not set any values so all the values will be null.

If any user composes new mail then all the values written by an user will be sent to sendmail class so this class will take all the values including the attachment from the user. After that it will fetch the required values of the user from the userinfo class like from address and then it will send the mail.

As content of particular email is displayed user has option to click on reply mail. If user clicks on reply mail he will be redirected to composereplymail.jsp page so when user replies he can add edit the data add an attachment and will submit the values. Submitted values will be sent to SendingMail Class which will decode the multipart mail and will send the mail to the specified recipients.For sending the mail SendingMail class will require some information about user so it will fetch the required information from the userinfo class and send the mail.

Whenever message is displayed to the user he also has an option to forward the mail.if user clicks on forward button, user will be redirected to composeforwardmail.jsp page. Now user can edit the data and cannot add new attachment all the values will be sent to ForwardMail Class which will take the required information about the user from userinfo class and will send the forwarded mail to the specified recipients.

Whenever user is logged in user has an option to logout anytime. If user clicks the logout button logout class will take the user’s information from the userinfo class and will free up all the user set values after that it will redirect user towards login i.e. login.jsp page.

****

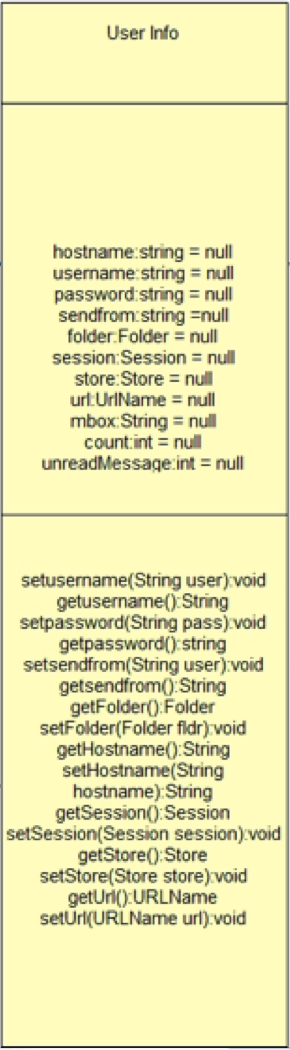
**Class :UserAuthentication**

public class **UserAuthentication** extends HttpServlet

This Class is used for authentication purpose.It will try to open the store according to the response it will open the page.

|  |  |
| --- | --- |
| **Method Summary** | |
| String | connect(hostname,username,password();  This Method will use these values for opening the user’s store. |

|  |  |
| --- | --- |
| **Java Methods** | |
| void | **doPost(**HttpServletRequest req, HttpServletResponse res)throws ServletException, IOException)  This method will process the request. |
| Void | [**setAttribute**](http://docs.oracle.com/javaee/1.3/api/javax/servlet/ServletRequest.html#setAttribute(java.lang.String, java.lang.Object))(java.lang.String name, java.lang.Object o)  Stores an attribute in this request. |
| [ServletContext](http://docs.oracle.com/javaee/5/api/javax/servlet/ServletContext.html) | [**getServletContext**](http://docs.oracle.com/javaee/5/api/javax/servlet/ServletConfig.html#getServletContext())()  Returns a reference to the [ServletContext](http://docs.oracle.com/javaee/5/api/javax/servlet/ServletContext.html) in which the caller is executing. |
| Session | [**getDefaultInstance**](http://javamail.kenai.com/nonav/javadocs/javax/mail/Session.html#getDefaultInstance(java.util.Properties))([Properties](http://java.sun.com/j2se/1.4.2/docs/api/java/util/Properties.html?is-external=true) props)  Stores an attribute in this request. |
| [Store](http://docs.oracle.com/javaee/5/api/javax/servlet/ServletContext.html) | [**getStore**](http://javamail.kenai.com/nonav/javadocs/javax/mail/Session.html#getStore(javax.mail.URLName))([URLName](http://javamail.kenai.com/nonav/javadocs/javax/mail/URLName.html) url)  Get a Store object for the given URLName. |
| [void](http://docs.oracle.com/javaee/5/api/javax/servlet/ServletContext.html) | [**connect**](http://javamail.kenai.com/nonav/javadocs/javax/mail/Service.html#connect(java.lang.String, java.lang.String, java.lang.String))([String](http://java.sun.com/j2se/1.4.2/docs/api/java/lang/String.html?is-external=true) host, [String](http://java.sun.com/j2se/1.4.2/docs/api/java/lang/String.html?is-external=true) user, [String](http://java.sun.com/j2se/1.4.2/docs/api/java/lang/String.html?is-external=true) password)  Connect to the specified address. |
| [Folder](http://docs.oracle.com/javaee/5/api/javax/servlet/ServletContext.html) | [**getFolder**](http://javamail.kenai.com/nonav/javadocs/com/sun/mail/imap/IMAPStore.html#getFolder(java.lang.String))([URLName](http://javamail.kenai.com/nonav/javadocs/javax/mail/URLName.html) url)  Get named folder. |
| [void](http://docs.oracle.com/javaee/5/api/javax/servlet/ServletContext.html) | [**open**](http://javamail.kenai.com/nonav/javadocs/javax/mail/Folder.html#open(int))(int mode)  Open this Folder. |
| [HttpSession](http://docs.oracle.com/javaee/6/api/javax/servlet/http/HttpSession.html) | [**getSession**](http://docs.oracle.com/javaee/6/api/javax/servlet/http/HttpServletRequest.html#getSession())()  Returns the current session associated with this request, or if the request does not have a session, creates one. |
| [RequestDispatcher](http://docs.oracle.com/javaee/1.3/api/javax/servlet/RequestDispatcher.html) | [**getRequestDispatcher**](http://docs.oracle.com/javaee/1.3/api/javax/servlet/ServletRequest.html#getRequestDispatcher(java.lang.String))(java.lang.String path)  Returns a [RequestDispatcher](http://docs.oracle.com/javaee/1.3/api/javax/servlet/RequestDispatcher.html) object that acts as a wrapper for the resource located at the given path. |
| [void](http://docs.oracle.com/javaee/1.3/api/javax/servlet/RequestDispatcher.html) | [**forward**](http://docs.oracle.com/javaee/5/api/javax/servlet/RequestDispatcher.html#forward(javax.servlet.ServletRequest, javax.servlet.ServletResponse))([ServletRequest](http://docs.oracle.com/javaee/5/api/javax/servlet/ServletRequest.html) request, [ServletResponse](http://docs.oracle.com/javaee/5/api/javax/servlet/ServletResponse.html) response)  Forwards a request from a servlet to another resource (servlet, JSP file, or HTML file) on the server. |

****

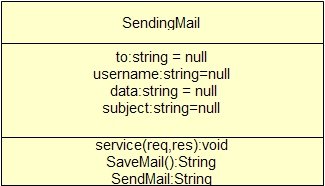
**Class :userinfo**

public class **userinfo**

This class will set the user attributes if user is authenticated.

|  |  |
| --- | --- |
| **Constructor Summary** | |
| userinfo(String username, String password)  sets username and password. |  |

|  |  |
| --- | --- |
| **Method Summary** | |
| void | setusername(String user)  Sets Username |
| String | getusername()  Return Username |
| Void | setpassword(String pass)  Sets Password |
| String | getpassword()  Return Password |
| Void | setsendfrom(String user)  Sets sendFrom |
| String | getsendfrom()  Return sendFrom |
| Void | setFolder(Folder fldr)  Sets folder |
| Folder | getFolder()  Return folder |
| Void | setHostname(String hostname)  Sets hostname |
| String | getHostname()  Return hostname |
| Void | setSession(Session session)  Sets session |
| Session | getSession()  Return session |
| Void | setStore(Store store)  Sets store |
| Store | getStore()  Return store |
| Void | setUrl(URLName url)  Sets url |
| URLName | getUrl()  Return url |

****

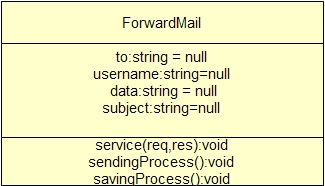
**Class :sendmail**

public class sendmail extends HttpServlet

This class will send mail or save mail to draft according to the user submitted value.

|  |  |
| --- | --- |
| **Method Summary** | |
| String | SendMail()  This method will send mail and return the message if sent or not. |
| String | SaveMail()  This method will save mail and return the message if sent or not. |

|  |  |
| --- | --- |
| **Java Methods** | |
| void | **doPost**(HttpServletRequest req, HttpServletResponse res)throws ServletException, IOException  This method will process the request. |
| Session | [**getInstance**](http://javamail.kenai.com/nonav/javadocs/javax/mail/Session.html#getDefaultInstance(java.util.Properties))([Properties](http://java.sun.com/j2se/1.4.2/docs/api/java/util/Properties.html?is-external=true) props)  Get a new Session object. |
| java.lang.object | [**getAttribute**](http://docs.oracle.com/javaee/1.4/api/javax/servlet/http/HttpSession.html#getAttribute(java.lang.String))(java.lang.String name)  Returns the object bound with the specified name in this session, or null if no object is bound under the name. |
| [object](http://docs.oracle.com/javaee/5/api/javax/servlet/ServletContext.html) | [**put**](http://docs.oracle.com/javase/1.4.2/docs/api/java/util/Hashtable.html#put(java.lang.Object, java.lang.Object))([Object](http://docs.oracle.com/javase/1.4.2/docs/api/java/lang/Object.html) key, [Object](http://docs.oracle.com/javase/1.4.2/docs/api/java/lang/Object.html) value)  Maps the specified key to the specified value in this hashtable. |
| java.lang.String | [**getName**](file:///C:\nikki\cos-26Dec2008\doc\com\oreilly\servlet\multipart\Part.html#getName())()  Returns the name of the form element that this Part corresponds to. |
| boolean | [**isFile**](file:///C:\nikki\cos-26Dec2008\doc\com\oreilly\servlet\multipart\Part.html#isFile())()  Returns true if this Part is a FilePart. |
| boolean | [**isParam**](file:///C:\nikki\cos-26Dec2008\doc\com\oreilly\servlet\multipart\Part.html#isParam())()  Returns true if this Part is a ParamPart. |
| java.lang.String | [**getContentType**](file:///C:\nikki\cos-26Dec2008\doc\com\oreilly\servlet\multipart\FilePart.html#getContentType())()  Returns the content type of the file data contained within. |
| java.lang.String | [**getFileName**](file:///C:\nikki\cos-26Dec2008\doc\com\oreilly\servlet\multipart\FilePart.html#getFileName())()  Returns the name that the file was stored with on the remote system, or null if the user didn't enter a file to be uploaded. |
| long | [**writeTo**](file:///C:\nikki\cos-26Dec2008\doc\com\oreilly\servlet\multipart\FilePart.html#writeTo(java.io.File))(java.io.File fileOrDirectory)  Write this file part to a file or directory. |
| [void](http://docs.oracle.com/javaee/5/api/javax/servlet/ServletContext.html) | [**setFrom**](http://javamail.kenai.com/nonav/javadocs/javax/mail/Message.html#setFrom(javax.mail.Address))([Address](http://javamail.kenai.com/nonav/javadocs/javax/mail/Address.html) address)  Set the "From" attribute in this Message |
| [void](http://docs.oracle.com/javaee/5/api/javax/servlet/ServletContext.html) | [**setRecipients**](http://javamail.kenai.com/nonav/javadocs/javax/mail/Message.html#setRecipients(javax.mail.Message.RecipientType, javax.mail.Address[]))([Message.RecipientType](http://javamail.kenai.com/nonav/javadocs/javax/mail/Message.RecipientType.html) type, [Address](http://javamail.kenai.com/nonav/javadocs/javax/mail/Address.html)[] addresses)  Set the recipient addresses. |
| [void](http://docs.oracle.com/javaee/5/api/javax/servlet/ServletContext.html) | [**setSubject**](http://javamail.kenai.com/nonav/javadocs/javax/mail/Message.html#setSubject(java.lang.String))([String](http://java.sun.com/j2se/1.4.2/docs/api/java/lang/String.html?is-external=true) subject)  Set the subject of this message. |
| [void](http://docs.oracle.com/javaee/5/api/javax/servlet/ServletContext.html) | [**setSentDate**](http://javamail.kenai.com/nonav/javadocs/javax/mail/Message.html#setSentDate(java.util.Date))([Date](http://java.sun.com/j2se/1.4.2/docs/api/java/util/Date.html?is-external=true) date)  Set the sent date of this message. |
| [void](http://docs.oracle.com/javaee/5/api/javax/servlet/ServletContext.html) | [**setText**](http://docs.oracle.com/javaee/1.3/api/javax/mail/internet/MimeMessage.html#setText(java.lang.String))(java.lang.String text)  Convenience method that sets the given String as this part's content, with a MIME type of "text/plain". |
| [void](http://docs.oracle.com/javaee/5/api/javax/servlet/ServletContext.html) | [**addBodyPart**](http://docs.oracle.com/javaee/1.4/api/javax/mail/Multipart.html#addBodyPart(javax.mail.BodyPart))([BodyPart](http://docs.oracle.com/javaee/1.4/api/javax/mail/BodyPart.html) part)  Adds a Part to the multipart. |
| [void](http://docs.oracle.com/javaee/5/api/javax/servlet/ServletContext.html) | [**setContent**](http://docs.oracle.com/javaee/1.3/api/javax/mail/internet/MimeMessage.html#setContent(javax.mail.Multipart))([Multipart](http://docs.oracle.com/javaee/1.3/api/javax/mail/Multipart.html) mp)  This method sets the Message's content to a Multipart object. |
| [void](http://docs.oracle.com/javaee/5/api/javax/servlet/ServletContext.html) | [**send**](http://javamail.kenai.com/nonav/javadocs/javax/mail/Transport.html#send(javax.mail.Message))([Message](http://javamail.kenai.com/nonav/javadocs/javax/mail/Message.html) msg)  Send a message. |
| [void](http://docs.oracle.com/javaee/5/api/javax/servlet/ServletContext.html) | [**appendMessages**](http://javamail.kenai.com/nonav/javadocs/javax/mail/Folder.html#appendMessages(javax.mail.Message[]))([Message](http://javamail.kenai.com/nonav/javadocs/javax/mail/Message.html)[] msgs)  Append given Messages to this folder. |
| [void](http://docs.oracle.com/javaee/5/api/javax/servlet/ServletContext.html) | [**saveChanges**](http://docs.oracle.com/javaee/1.3/api/javax/mail/internet/MimeMessage.html#saveChanges())()  Updates the appropriate header fields of this message to be consistent with the message's contents. |
| [void](http://docs.oracle.com/javaee/5/api/javax/servlet/ServletContext.html) | [**open**](http://javamail.kenai.com/nonav/javadocs/javax/mail/Folder.html#open(int))(int mode)  Open this Folder. |

****

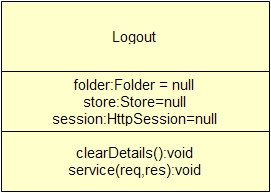
**Class :ForwardMail**

public class ForwardMail extends HttpServlet

This class will send mail or save mail to draft according to the user submitted value.

|  |  |
| --- | --- |
| **Method Summary** | |
| String | SendMail()  This method will send mail and return the message if sent or not. |
| String | SaveMail()  This method will save mail and return the message if sent or not. |

|  |  |
| --- | --- |
| **Java Methods** | |
| [void](http://docs.oracle.com/javaee/5/api/javax/servlet/ServletContext.html) | doPost(HttpServletRequest req, HttpServletResponse res)throws ServletException, IOException  This method will process the request. |
| Session | [**getInstance**](http://javamail.kenai.com/nonav/javadocs/javax/mail/Session.html#getDefaultInstance(java.util.Properties))([Properties](http://java.sun.com/j2se/1.4.2/docs/api/java/util/Properties.html?is-external=true) props)  Get a new Session object. |
| java.lang.object | [**getAttribute**](http://docs.oracle.com/javaee/1.4/api/javax/servlet/http/HttpSession.html#getAttribute(java.lang.String))(java.lang.String name)  Returns the object bound with the specified name in this session, or null if no object is bound under the name. |
| [object](http://docs.oracle.com/javaee/5/api/javax/servlet/ServletContext.html) | [**put**](http://docs.oracle.com/javase/1.4.2/docs/api/java/util/Hashtable.html#put(java.lang.Object, java.lang.Object))([Object](http://docs.oracle.com/javase/1.4.2/docs/api/java/lang/Object.html) key, [Object](http://docs.oracle.com/javase/1.4.2/docs/api/java/lang/Object.html) value)  Maps the specified key to the specified value in this hashtable. |
| java.lang.String | [**getName**](file:///C:\nikki\cos-26Dec2008\doc\com\oreilly\servlet\multipart\Part.html#getName())()  Returns the name of the form element that this Part corresponds to. |
| boolean | [**isFile**](file:///C:\nikki\cos-26Dec2008\doc\com\oreilly\servlet\multipart\Part.html#isFile())()  Returns true if this Part is a FilePart. |
| boolean | [**isParam**](file:///C:\nikki\cos-26Dec2008\doc\com\oreilly\servlet\multipart\Part.html#isParam())()  Returns true if this Part is a ParamPart. |
| java.lang.String | [**getContentType**](file:///C:\nikki\cos-26Dec2008\doc\com\oreilly\servlet\multipart\FilePart.html#getContentType())()  Returns the content type of the file data contained within. |
| java.lang.String | [**getFileName**](file:///C:\nikki\cos-26Dec2008\doc\com\oreilly\servlet\multipart\FilePart.html#getFileName())()  Returns the name that the file was stored with on the remote system, or null if the user didn't enter a file to be uploaded. |
| long | [**writeTo**](file:///C:\nikki\cos-26Dec2008\doc\com\oreilly\servlet\multipart\FilePart.html#writeTo(java.io.File))(java.io.File fileOrDirectory)  Write this file part to a file or directory. |
| [void](http://docs.oracle.com/javaee/5/api/javax/servlet/ServletContext.html) | [**setFrom**](http://javamail.kenai.com/nonav/javadocs/javax/mail/Message.html#setFrom(javax.mail.Address))([Address](http://javamail.kenai.com/nonav/javadocs/javax/mail/Address.html) address)  Set the "From" attribute in this Message |
| [void](http://docs.oracle.com/javaee/5/api/javax/servlet/ServletContext.html) | [**setRecipients**](http://javamail.kenai.com/nonav/javadocs/javax/mail/Message.html#setRecipients(javax.mail.Message.RecipientType, javax.mail.Address[]))([Message.RecipientType](http://javamail.kenai.com/nonav/javadocs/javax/mail/Message.RecipientType.html) type, [Address](http://javamail.kenai.com/nonav/javadocs/javax/mail/Address.html)[] addresses)  Set the recipient addresses. |
| [void](http://docs.oracle.com/javaee/5/api/javax/servlet/ServletContext.html) | [**setSubject**](http://javamail.kenai.com/nonav/javadocs/javax/mail/Message.html#setSubject(java.lang.String))([String](http://java.sun.com/j2se/1.4.2/docs/api/java/lang/String.html?is-external=true) subject)  Set the subject of this message. |
| [void](http://docs.oracle.com/javaee/5/api/javax/servlet/ServletContext.html) | [**setSentDate**](http://javamail.kenai.com/nonav/javadocs/javax/mail/Message.html#setSentDate(java.util.Date))([Date](http://java.sun.com/j2se/1.4.2/docs/api/java/util/Date.html?is-external=true) date)  Set the sent date of this message. |
| [void](http://docs.oracle.com/javaee/5/api/javax/servlet/ServletContext.html) | [**setText**](http://docs.oracle.com/javaee/1.3/api/javax/mail/internet/MimeMessage.html#setText(java.lang.String))(java.lang.String text)  Convenience method that sets the given String as this part's content, with a MIME type of "text/plain". |
| [void](http://docs.oracle.com/javaee/5/api/javax/servlet/ServletContext.html) | [**addBodyPart**](http://docs.oracle.com/javaee/1.4/api/javax/mail/Multipart.html#addBodyPart(javax.mail.BodyPart))([BodyPart](http://docs.oracle.com/javaee/1.4/api/javax/mail/BodyPart.html) part)  Adds a Part to the multipart. |
| [void](http://docs.oracle.com/javaee/5/api/javax/servlet/ServletContext.html) | [**setContent**](http://docs.oracle.com/javaee/1.3/api/javax/mail/internet/MimeMessage.html#setContent(javax.mail.Multipart))([Multipart](http://docs.oracle.com/javaee/1.3/api/javax/mail/Multipart.html) mp)  This method sets the Message's content to a Multipart object. |
| [void](http://docs.oracle.com/javaee/5/api/javax/servlet/ServletContext.html) | [**send**](http://javamail.kenai.com/nonav/javadocs/javax/mail/Transport.html#send(javax.mail.Message))([Message](http://javamail.kenai.com/nonav/javadocs/javax/mail/Message.html) msg)  Send a message. |
| [void](http://docs.oracle.com/javaee/5/api/javax/servlet/ServletContext.html) | [**appendMessages**](http://javamail.kenai.com/nonav/javadocs/javax/mail/Folder.html#appendMessages(javax.mail.Message[]))([Message](http://javamail.kenai.com/nonav/javadocs/javax/mail/Message.html)[] msgs)  Append given Messages to this folder. |
| [void](http://docs.oracle.com/javaee/5/api/javax/servlet/ServletContext.html) | [**saveChanges**](http://docs.oracle.com/javaee/1.3/api/javax/mail/internet/MimeMessage.html#saveChanges())()  Updates the appropriate header fields of this message to be consistent with the message's contents. |
| [void](http://docs.oracle.com/javaee/5/api/javax/servlet/ServletContext.html) | [**open**](http://javamail.kenai.com/nonav/javadocs/javax/mail/Folder.html#open(int))(int mode)  Open this Folder. |

****

**Class :LogoutServlet**

public class LogoutServlet extends HttpServlet

This Class will free all resources and redirect to login page.

|  |  |
| --- | --- |
| **Method Summary** | |
| void | clearDetails()  This method will free up all the resources. |

1. **Testing**

|  |  |  |
| --- | --- | --- |
| Action | Error | Valid |
| Username or password field blank | Please enter username/password | Login successfully |
| Username or password invalid | Invalid Username or Password | Login successfully |
| To field of compose mail blank | Enter Email-id in To field | No error |
| Subject field blank | Warning will display | No error |

* We checked this client on different computers which are connected through LAN.
* If user closes the browser without logging out session will be closed. So, user has to login again.
* It works properly on most popular browsers and operating systems.
* If user keeps Rookery on ideal state for 15 min application will be hanged. So, no one can miss use it.

1. **Limitations**

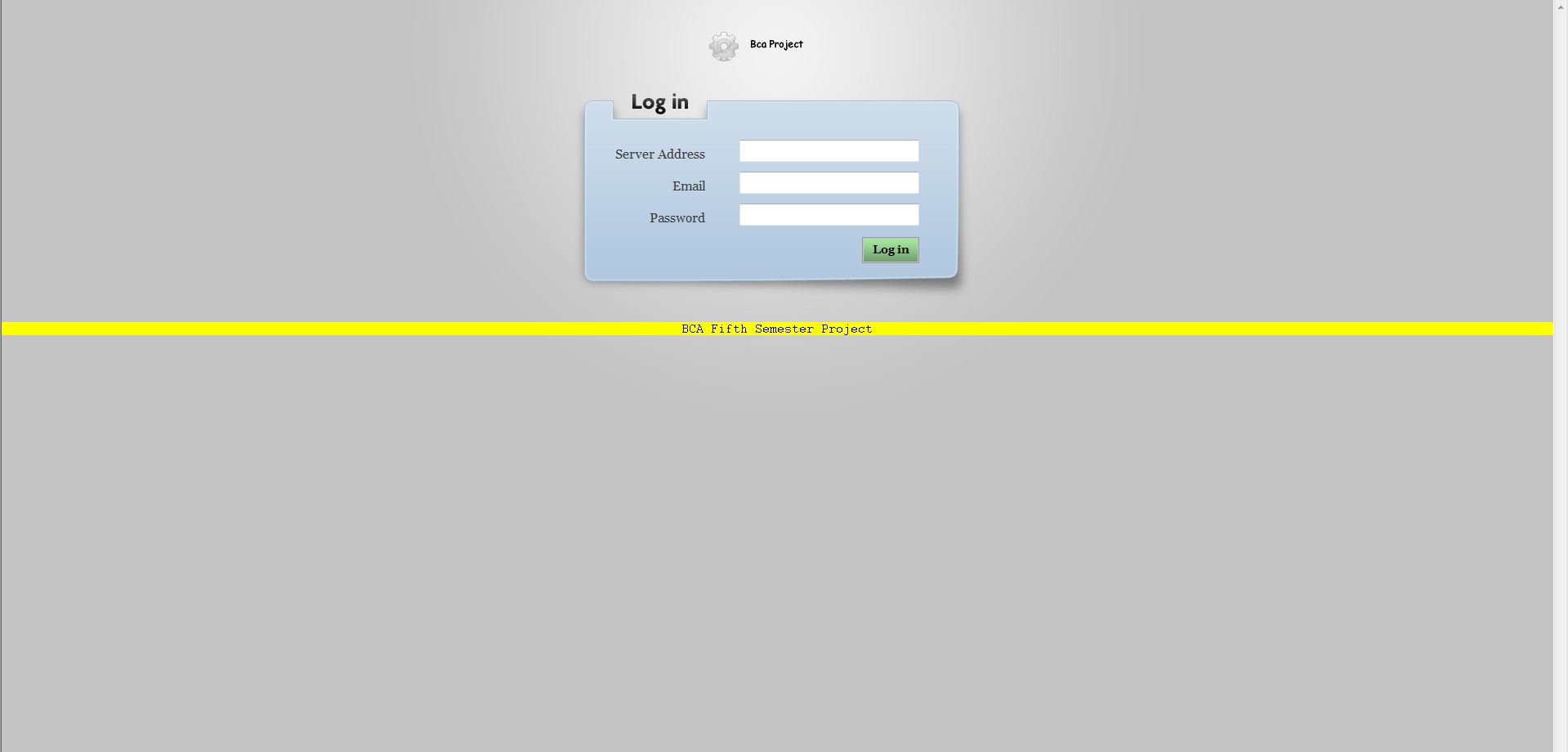
* Rookery can’t send email outside the domain right now because domain is not register.
* It can’t manage mails in different folders.
* It provides very basic functionalities compare to the other email services like Gmail and Yahoo mail.
* It can restore mail in one different folder.
* It is not as secure as Gmail and Yahoo mail.
* It requires email server which supports java mail API on server machine.
* It can send only one attachment file at a time.
* It can’t decode some types of file properly which sent by attachment.

1. **Enhancement**

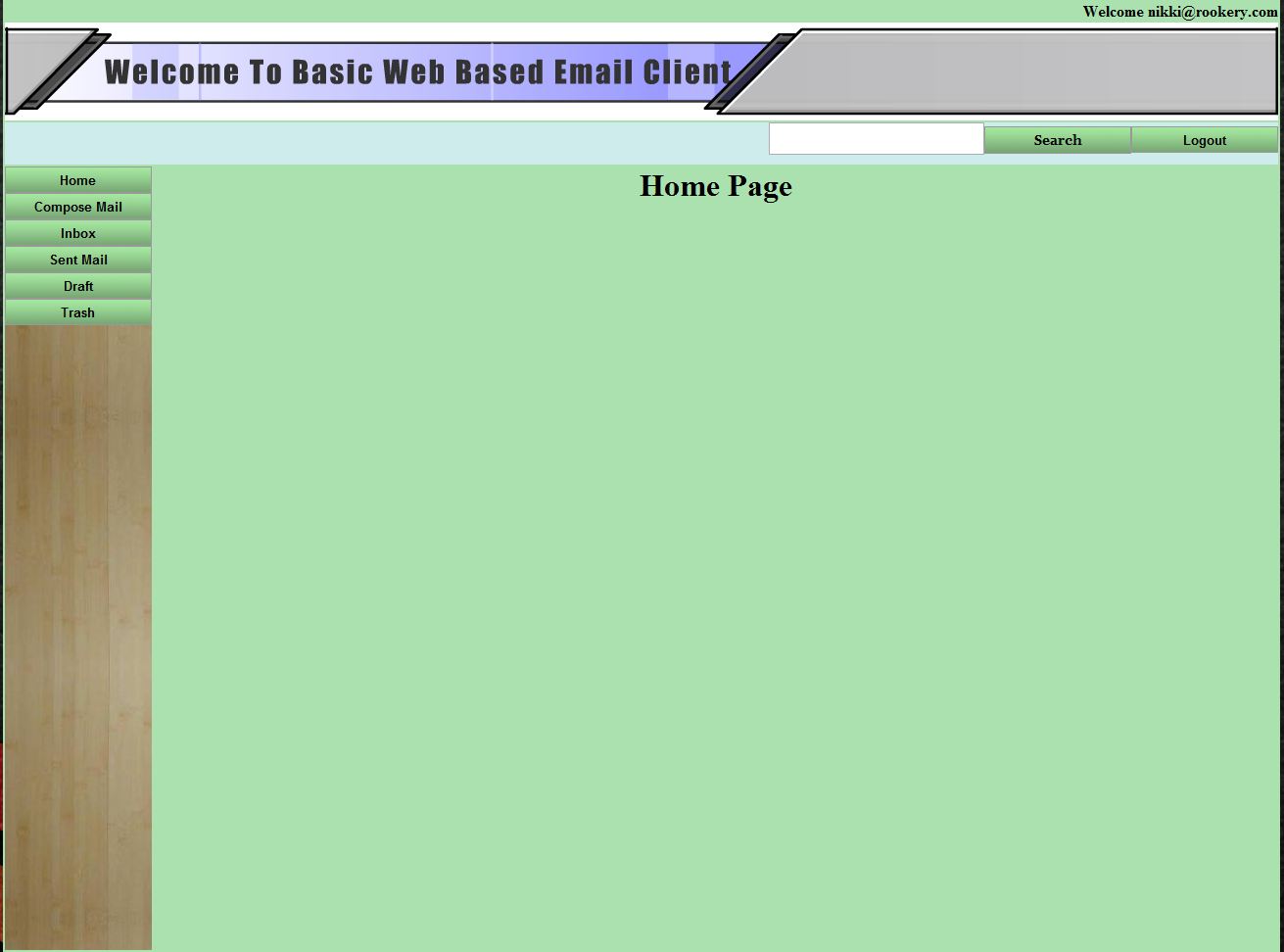
* In rookery we can implement many facilities like displaying news, weather forecast, score of matches, advertisements etc.
* We can deploy it on internet so anyone can use it.
* We can add many features like managing mail in folders, creating groups so can directly send mails to all group members.
* Managing contacts
* Online chatting facility
* Video calling like Skype

1. **Webmail Client – SCREEN SHOTS**

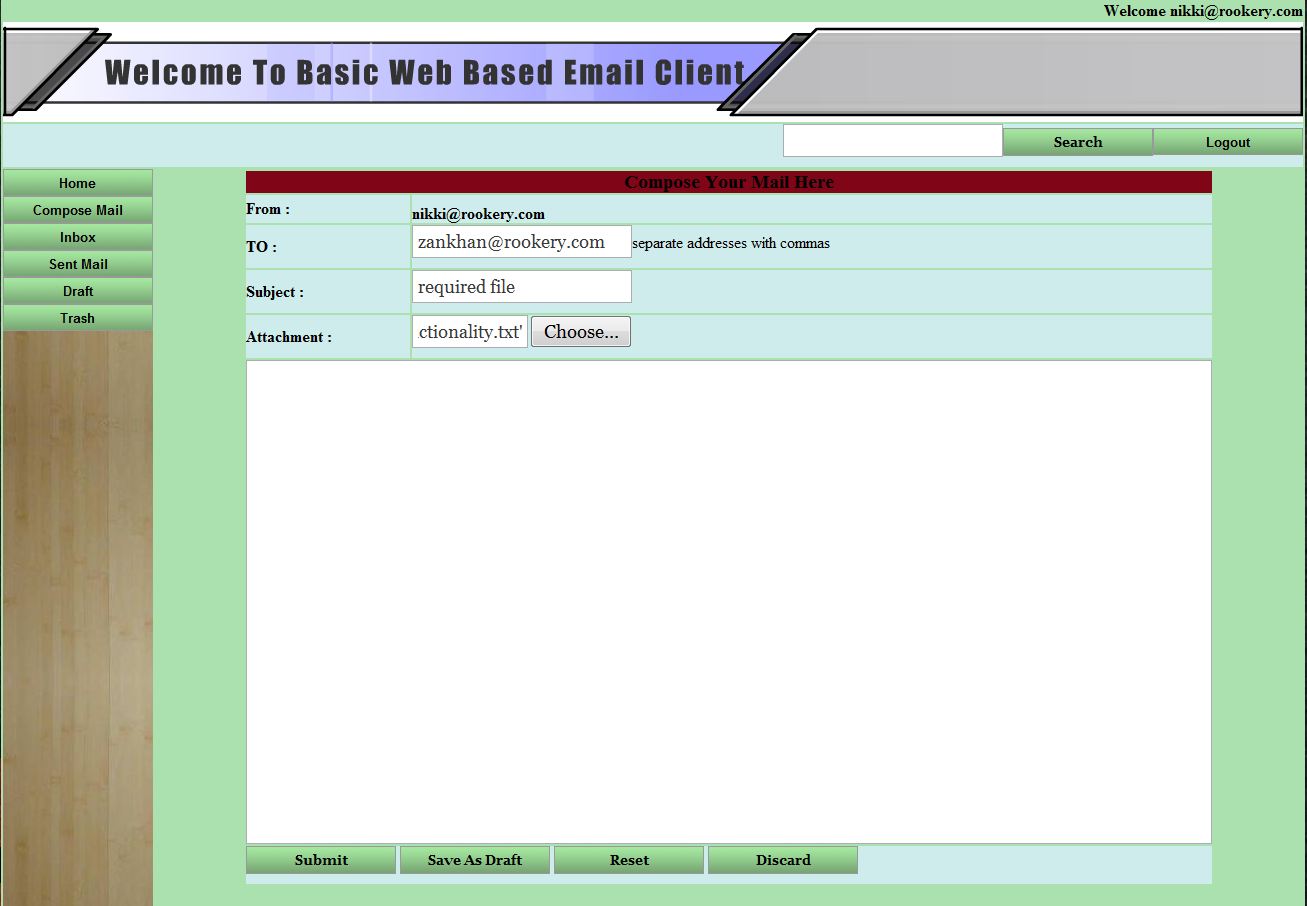
**Login Page**



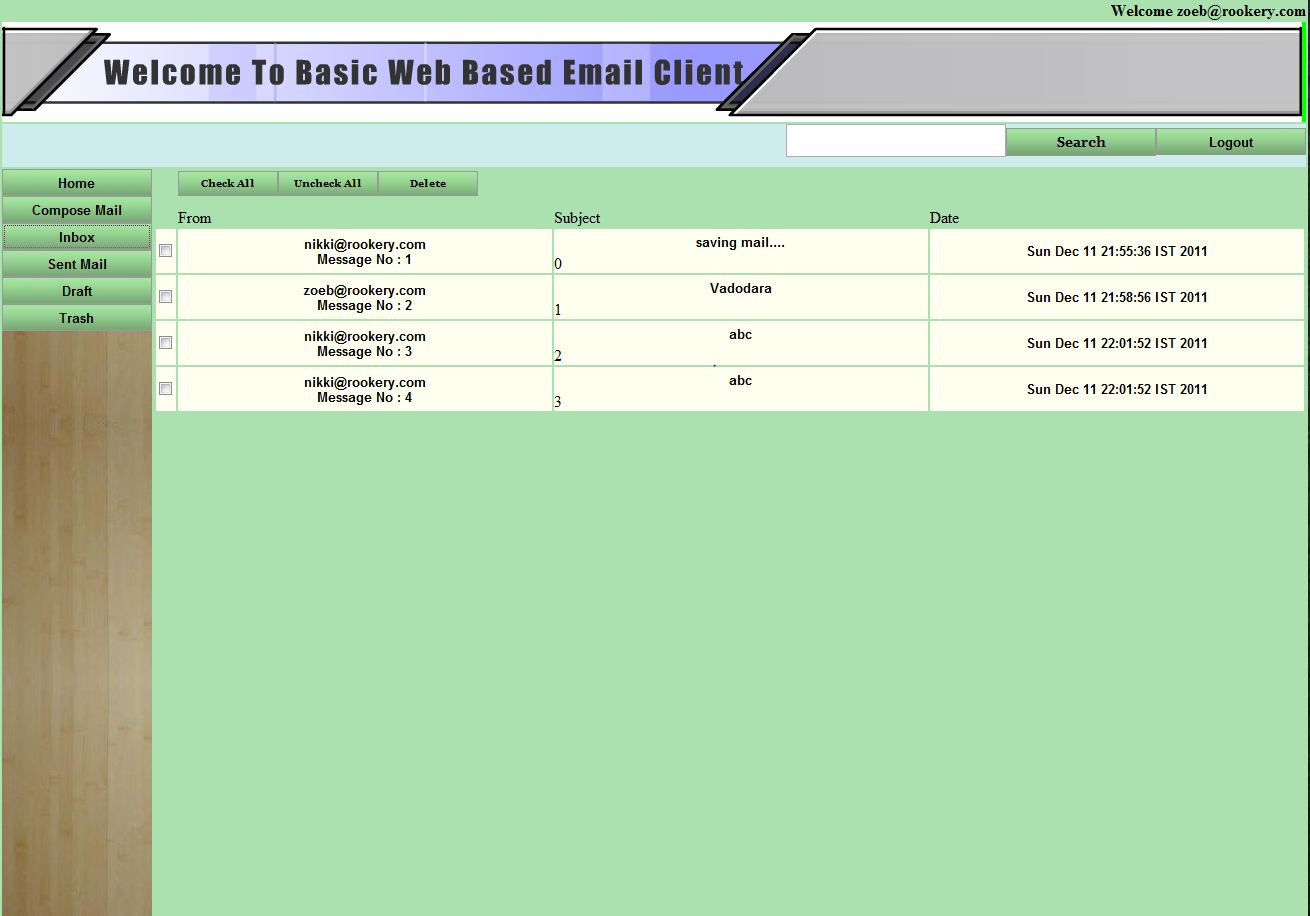
**Home Page**



**Compose MaIl Page**



**Inbox Page**



**Trash Page**

