

# Data Leakage Detection

## Major Project Report

Submitted by

Ankit Kumar Tater	:	08J41A1206
Y. V. Pradeep Kumar Reddy	:	08J41A1235
Pradeep Saklani	:	08J41A1236

Under the Guidance

of

Mr. V. Satish Kumar

Assistant Professor

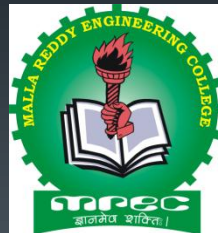
B.Tech(CSIT), M.Tech (CSE)

to

Jawaharlal Nehru Technological University Hyderabad

in partial fulfillment of the requirements for the award of degree of

## BACHELOR OF TECHNOLOGY in INFORMATION TECHNOLOGY



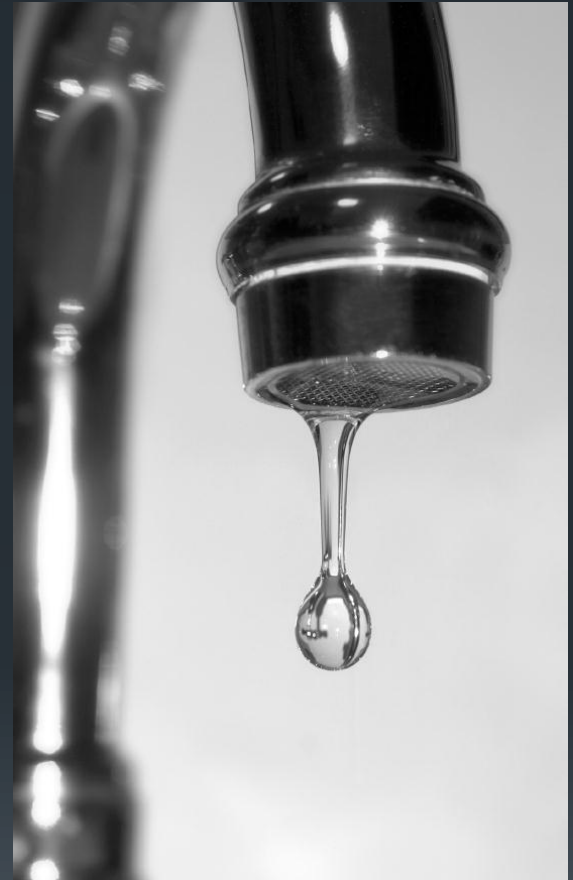
**DEPARTMENT OF INFORMATION TECHNOLOGY**  
**MALLA REDDY ENGINEERING COLLEGE**

Maisammaguda Dhulapally, (post Via) Hakimpet,  
Hyderabad - 500014

**April 2012**

# Agenda

- INTRODUCTION
- ISSUES
- SCOPE
- ANALYSIS
- DESIGN
- IMPLEMENTATIONS




# Introduction



In the course of doing business, sometimes sensitive data must be handed over to supposedly trusted third parties.

Our goal is to detect when the distributor's sensitive data has been leaked by agents, and if possible to identify the agent that leaked the data.

- 
- A data distributor has given sensitive data to a set of supposedly trusted agents (third parties). Some of the data is leaked and found in an unauthorized place (e.g., on the web or somebody's laptop).
  - The distributor must assess the likelihood that the leaked data came from one or more agents, as opposed to having been independently gathered by other means. We propose data allocation strategies (across the agents) that improve the probability of identifying leakages.
  - These methods do not rely on alterations of the released data (e.g., *watermarks*). In some cases we can also inject “realistic but fake” data records to further improve our chances of detecting leakage and identifying the guilty party.



**Employees**

Development  
Chains



**Business Partners**

Supply Chains    Off-shoring  
Business Hubs    Outsourcing



**Customers**

Demand  
Chains



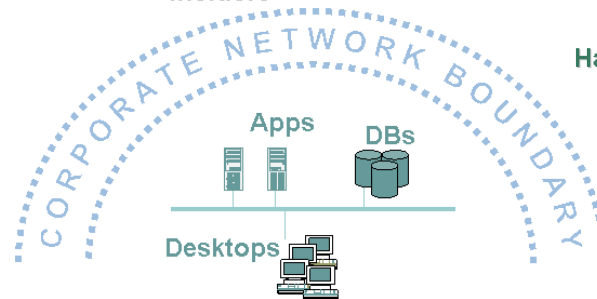
**Inadvertent  
Violators**



**Malicious  
Insiders**



**Hackers**



# Types of employees that put your company at risk.



- The security illiterate
  - Majority of employees with little or no knowledge of security
  - Corporate risk because of accidental breaches
- The gadget nerds
  - Introduce a variety of devices to their work PCs
  - Download software
- The unlawful residents
  - Use the company IT resources in ways they shouldn't
  - i.e., by storing music, movies, or playing games
- The malicious/disgruntled employees
  - Typically minority of employees
  - Gain access to areas of the IT system to which they shouldn't
  - Send corporate data (e.g., customer lists, R&D, etc.) to third parties

# Issues

- We develop a model for assessing the “guilt” of agents. We also present algorithms for distributing objects to agents, in a way that improves our chances of identifying a leaker.
- Finally, we also consider the option of adding “fake” objects to the distributed set.
- Such objects do not correspond to real entities but appear realistic to the agents.
- In a sense, the fake objects acts as a type of watermark for the entire set, without modifying any individual members. If it turns out an agent was given one or more fake objects that were leaked, then the distributor can be more confident that agent was guilty.

# Scope

- There are conventional techniques being used and these include technical and fundamental analysis.
- The main issue with these techniques is that they are manual, and need laborious work along with experience.



# Implementation



The system has the following

- Data Allocation
- Fake Object
- Optimization
- Data Distributor



## Data Allocation :

- The main focus of our project is the data allocation problem as how can the distributor “intelligently” give data to agents in order to improve the chances of detecting a guilty agent.



## Fake Object :

Fake objects are objects generated by the distributor in order to increase the chances of detecting agents that leak data. The distributor may be able to add fake objects to the distributed data in order to improve his effectiveness in detecting guilty agents. Our use of fake objects is inspired by the use of “trace” records in mailing lists.



## Optimization :

- The Optimization Module is the distributor's data allocation to agents has one constraint and one objective. The distributor's constraint is to satisfy agents' requests, by providing them with the number of objects they request or with all available objects that satisfy their conditions. His objective is to be able to detect an agent who leaks any portion of his data.



## Data Distributor :

- A data distributor has given sensitive data to a set of supposedly trusted agents (third parties). Some of the data is leaked and found in an unauthorized place (e.g., on the web or somebody's laptop). The distributor must assess the likelihood that the leaked data came from one or more agents, as opposed to having been independently gathered by other means.



## Proposed System

- In the Proposed System the hackers can be traced with good amount of evidence. In this proposed system the leakage of data is detected by the following methods viz., generating fake objects, Watermarking and by Encrypting the data.



## Existing System

- The Existing System can detect the hackers but the total no of cookies (evidence) will be less and the organization may not be able to proceed legally for further proceedings due to lack of good amount of cookies and the chances to escape of hackers are high.

# Modules

- Admin module
- User module



# Admin Module

- Administrator has to logon to the system.
- Admin can add information about a new user.
- Admin can add/view/delete/edit the user details.
- Admin can create user groups and place users in it.

# User Module

- A user must login to use the services.
- A user can send data sharing requests to other users.
- A user can accept/reject data sharing requests from other users.
- A user can trace the flow of its data i.e. can see what all users possess its data.

# Software Requirements



Operating System	:	Any Graphical OS
Technologies	:	Java (JSP, JDBC)
Back end	:	MS Access/SQL Server

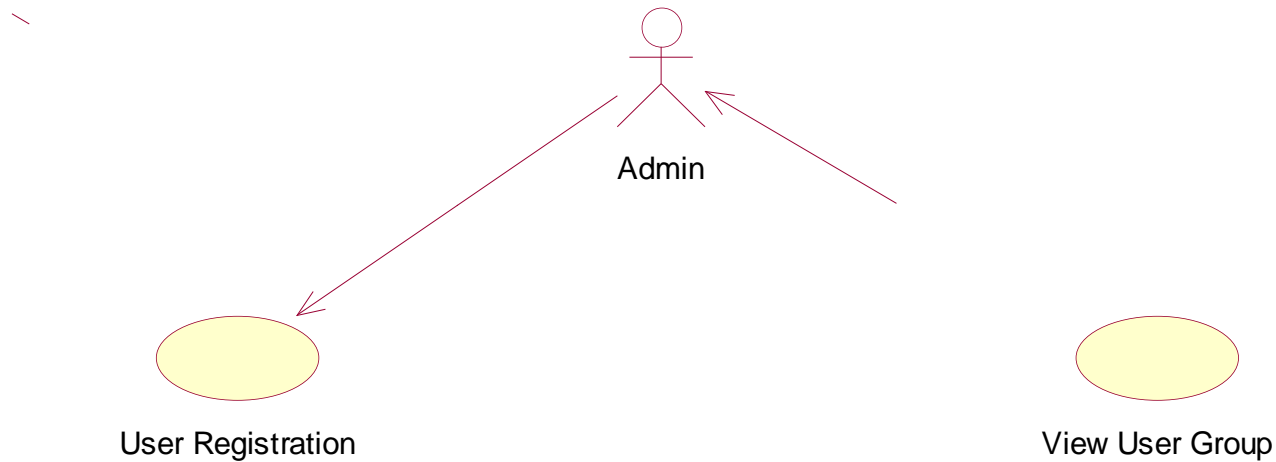
# Hardware Requirements

Processor	:	Pentium – IV
Hard Disk	:	40 GB
Ram	:	512 MB

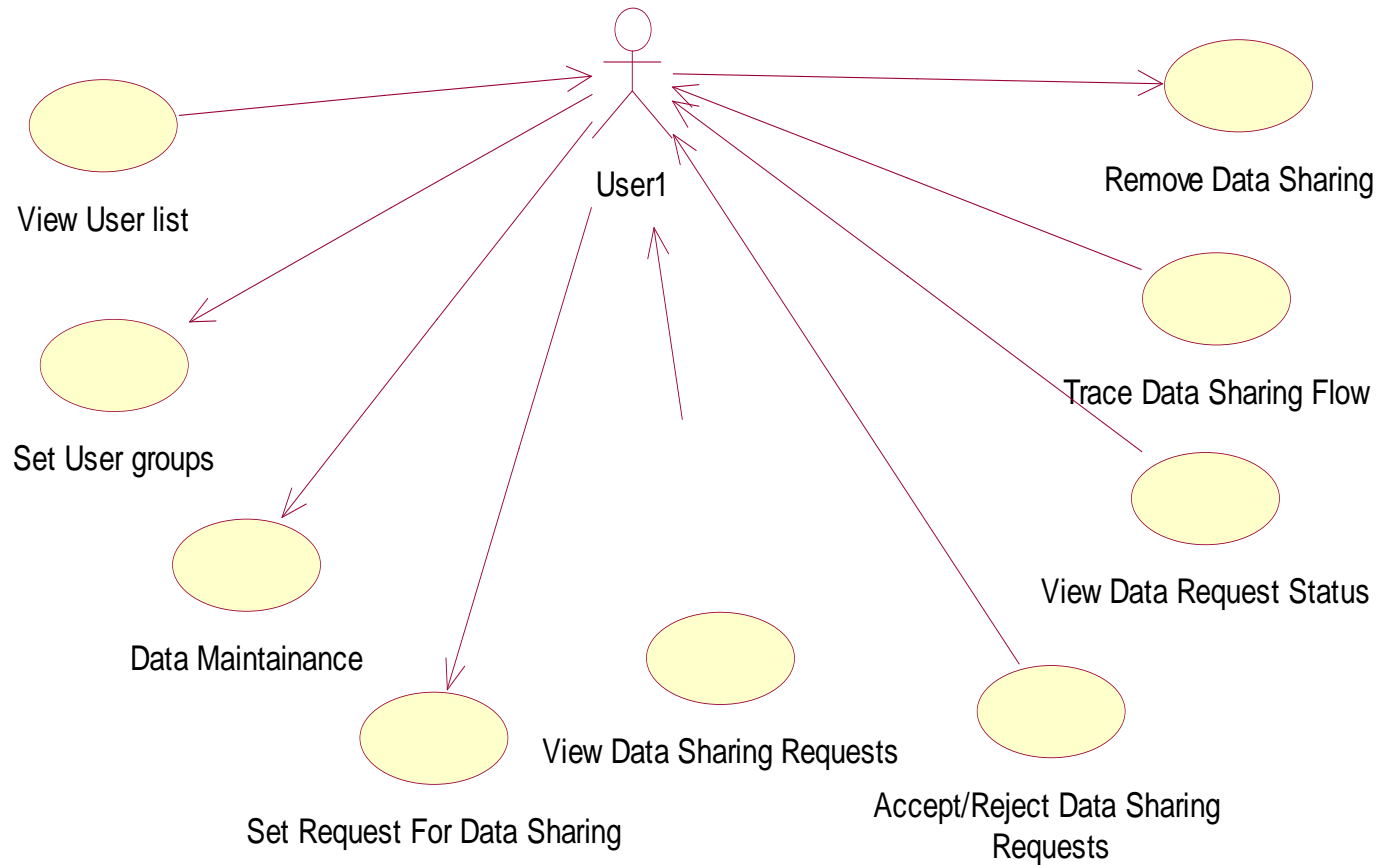
# Use case diagram for data leakage detection



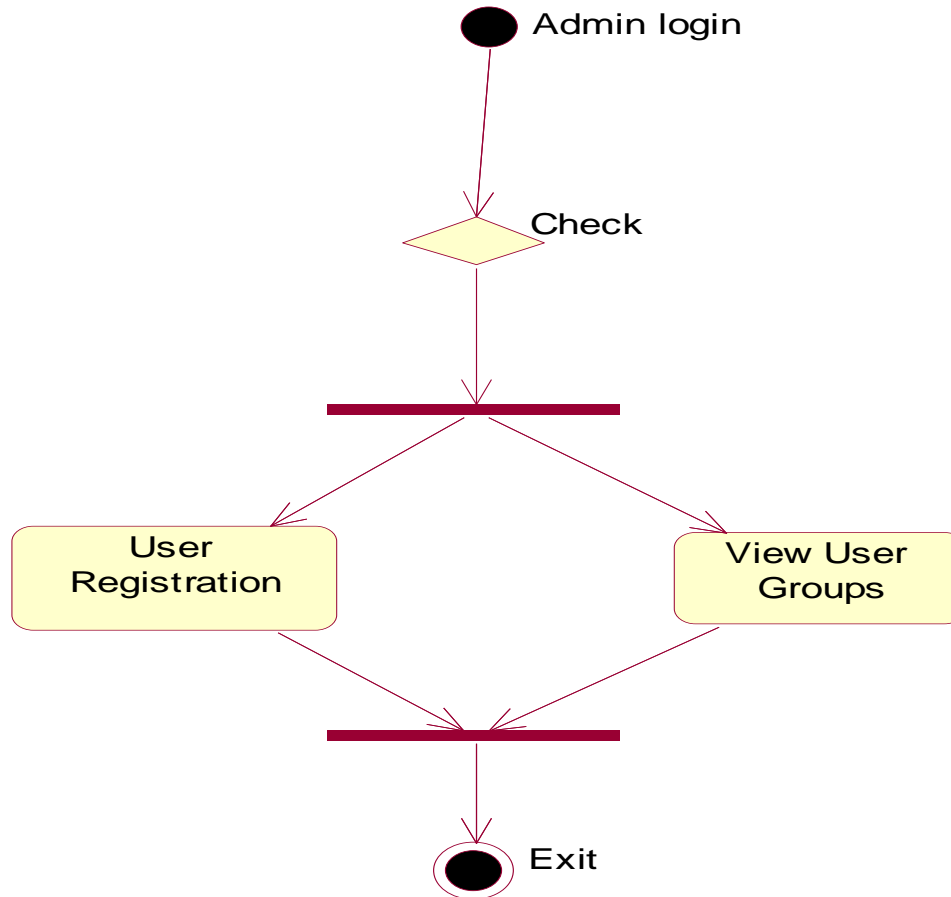
# Use case for Admin



# Use case for User

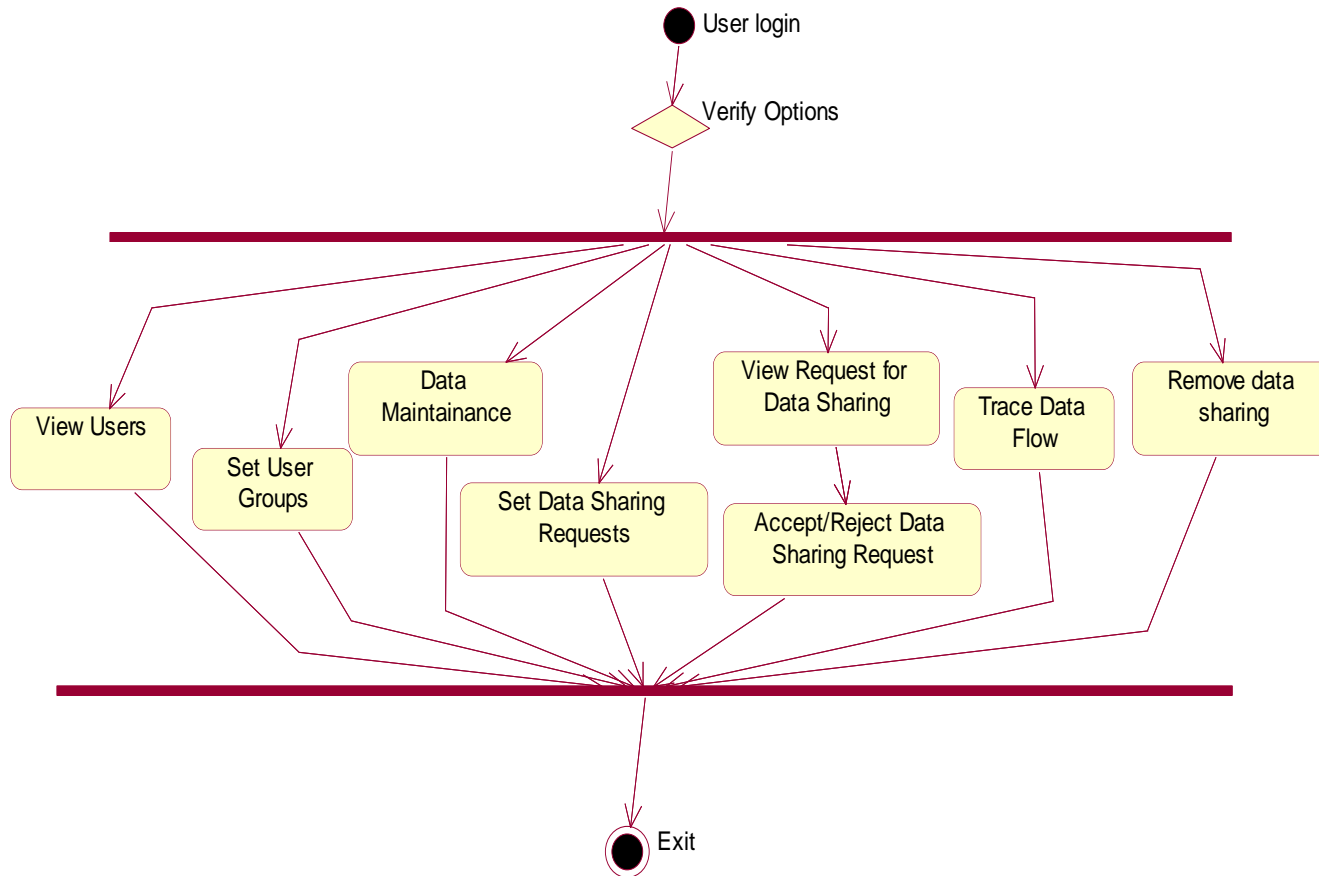


# Activity Diagram for Admin login

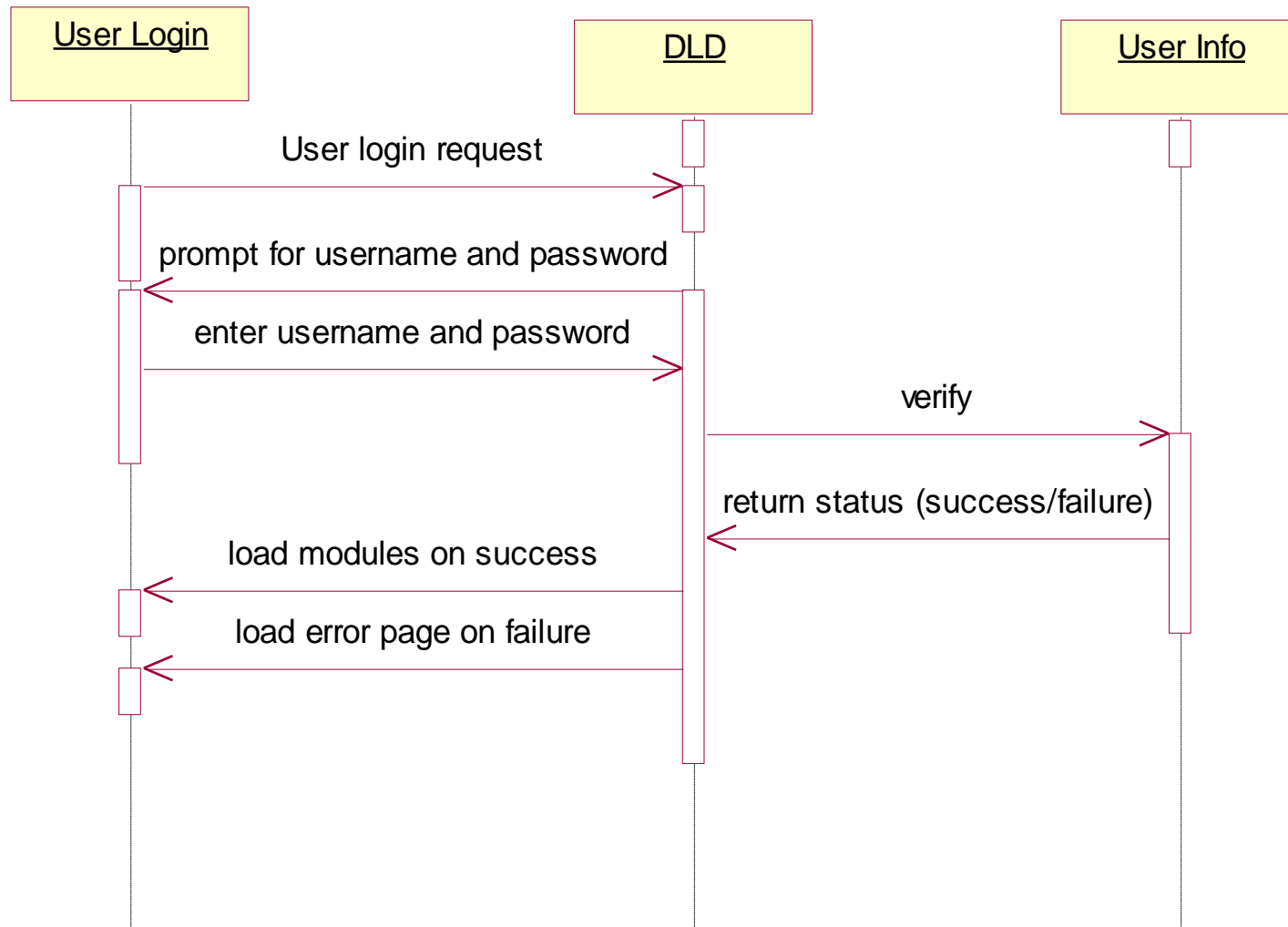




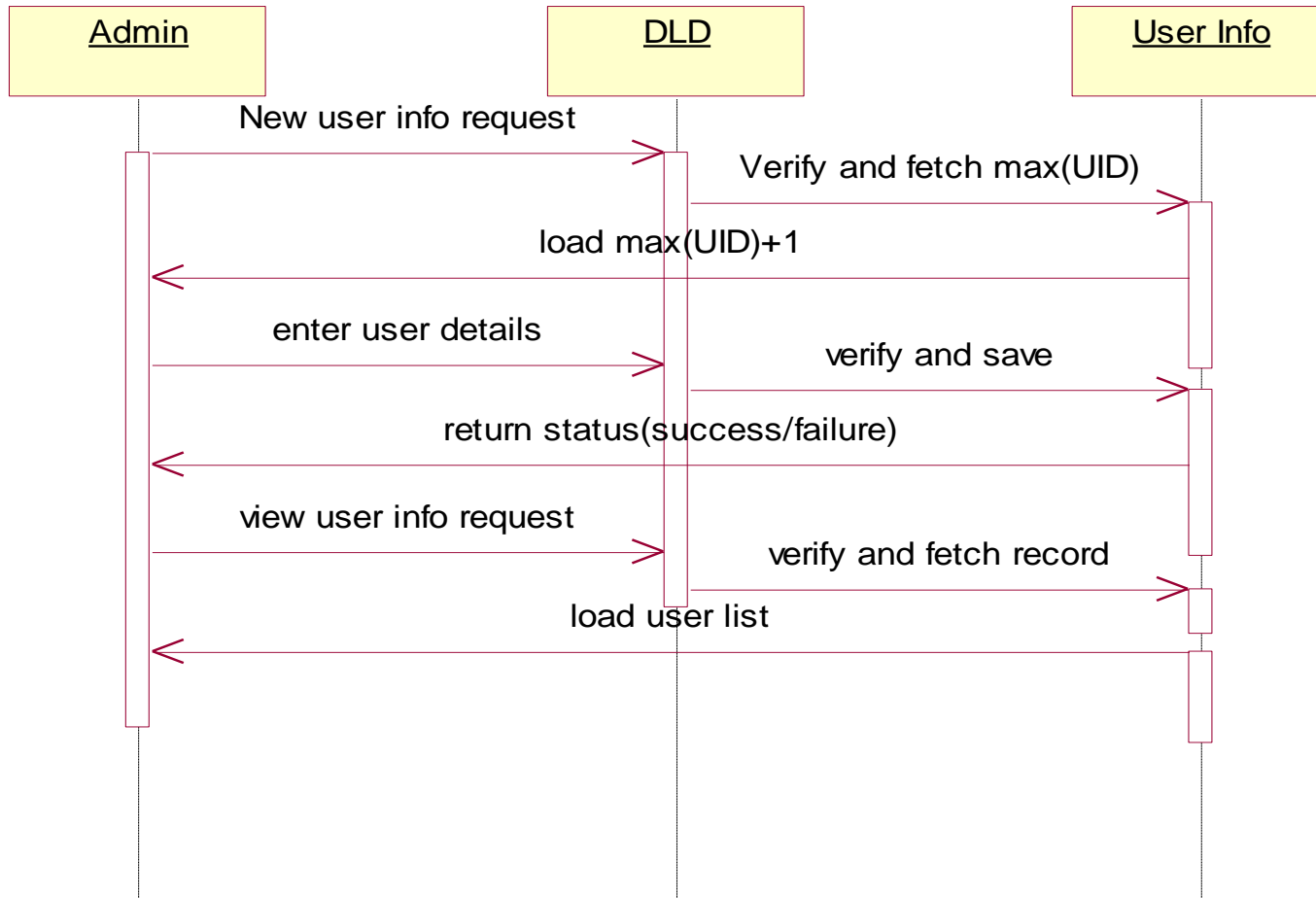
# Activity for User Login



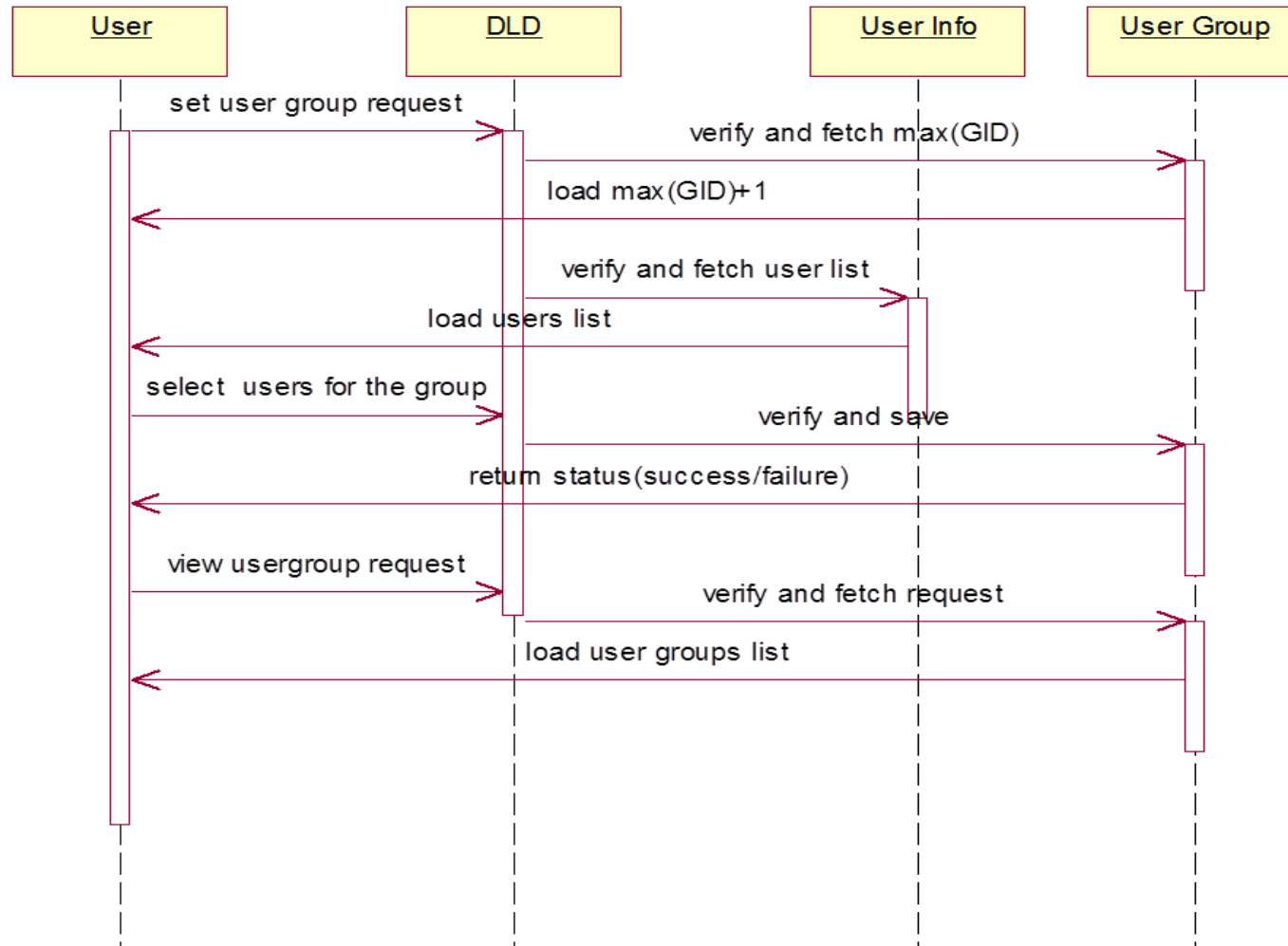
# Sequence Diagram for User Login



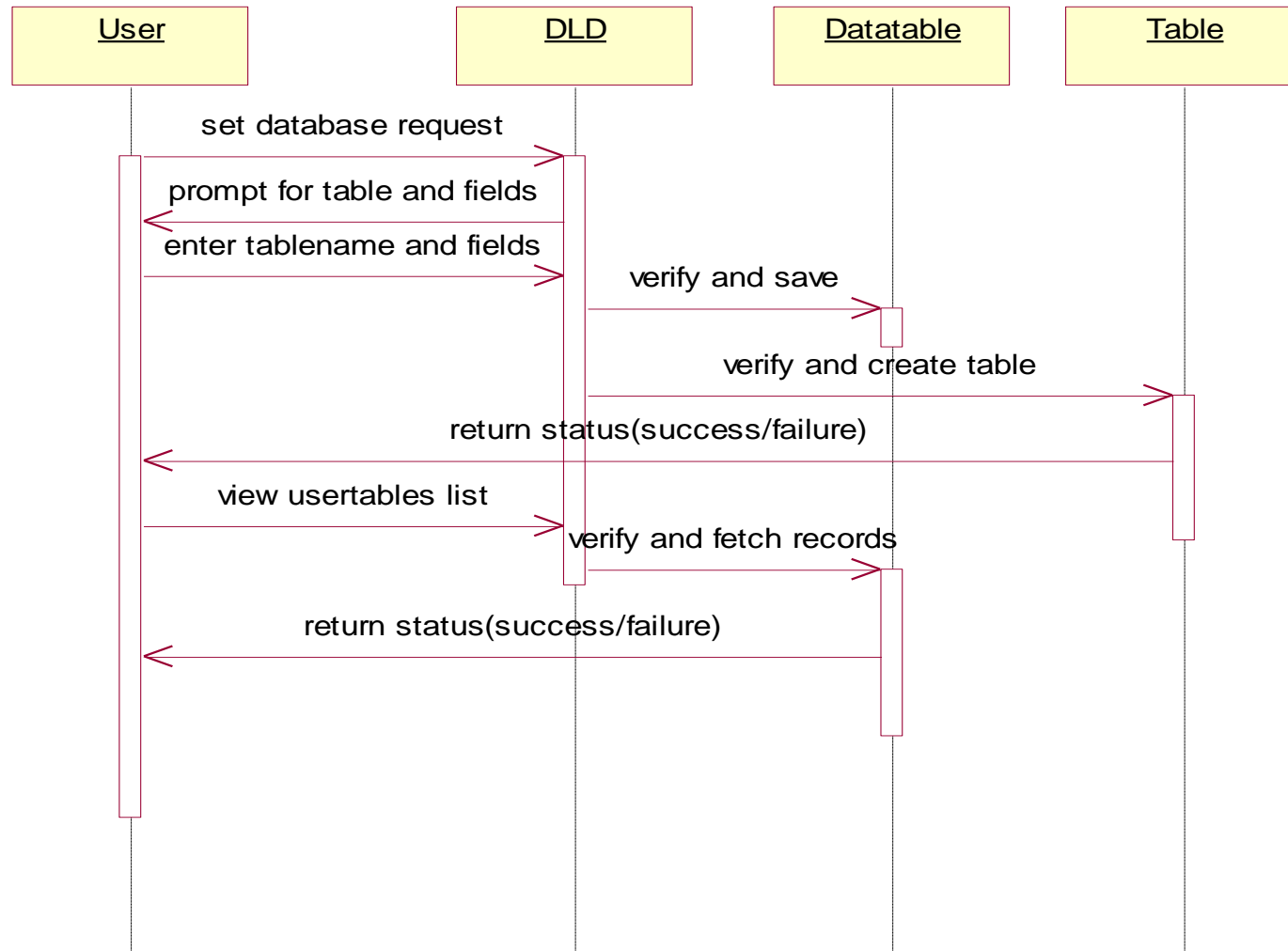
# Sequence Diagram for User Info



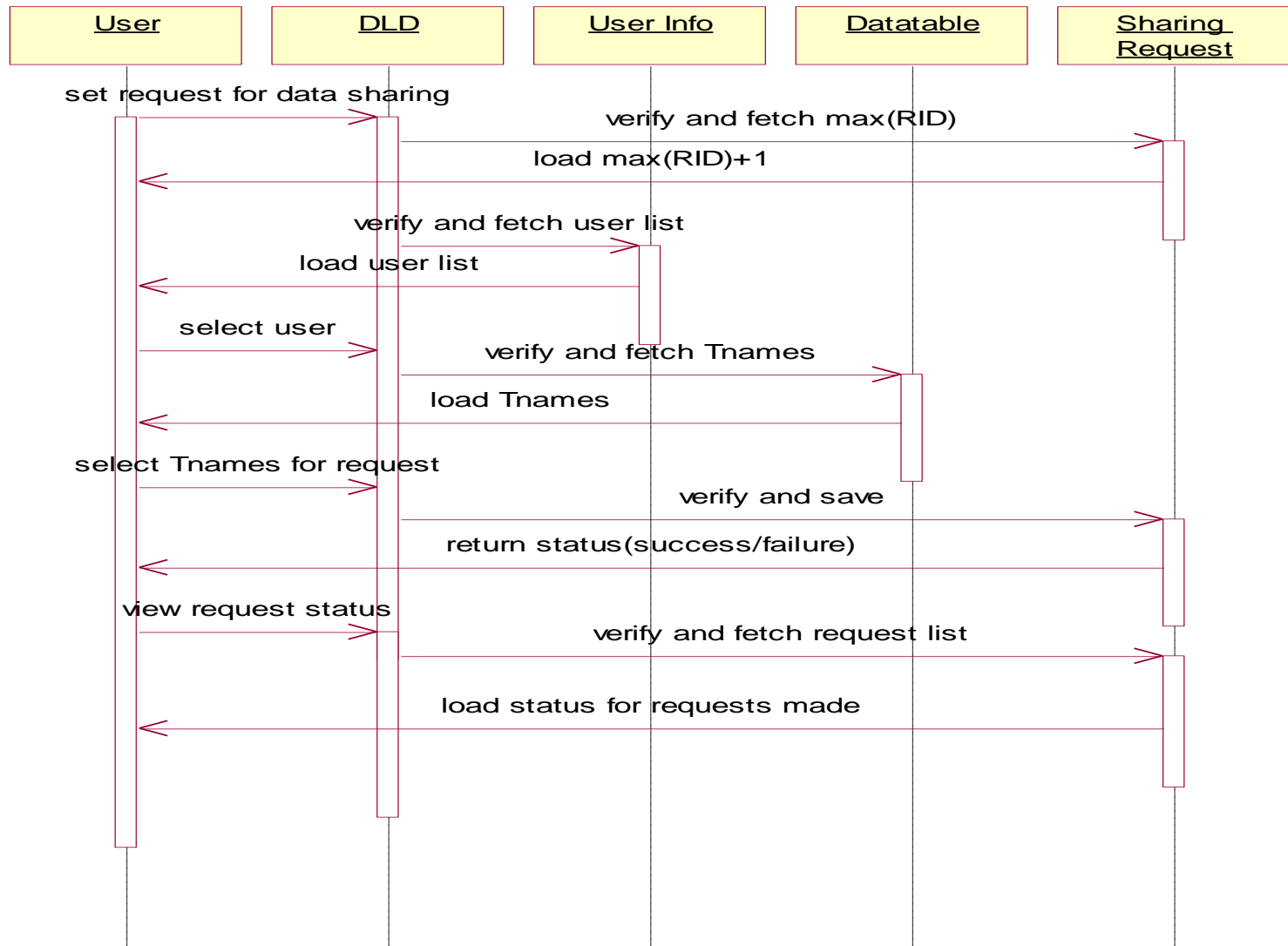
# Sequence Diagram for User Group



# Sequence Diagram for Datatable Maintenance



# Sequence Diagram for Data Sharing Requests



# Admin Login Form



A screenshot of a Java Swing window titled "LOGIN FORM". The window has a standard Mac OS X-style title bar with a red close button, a yellow maximize button, and a green minimize button. The main content area is white and contains two text input fields. The first field is labeled "UserName :" and the second is labeled "PassWord :". Below the input fields are two buttons: "LOGIN" and "EXIT". The window is set against a solid blue background.

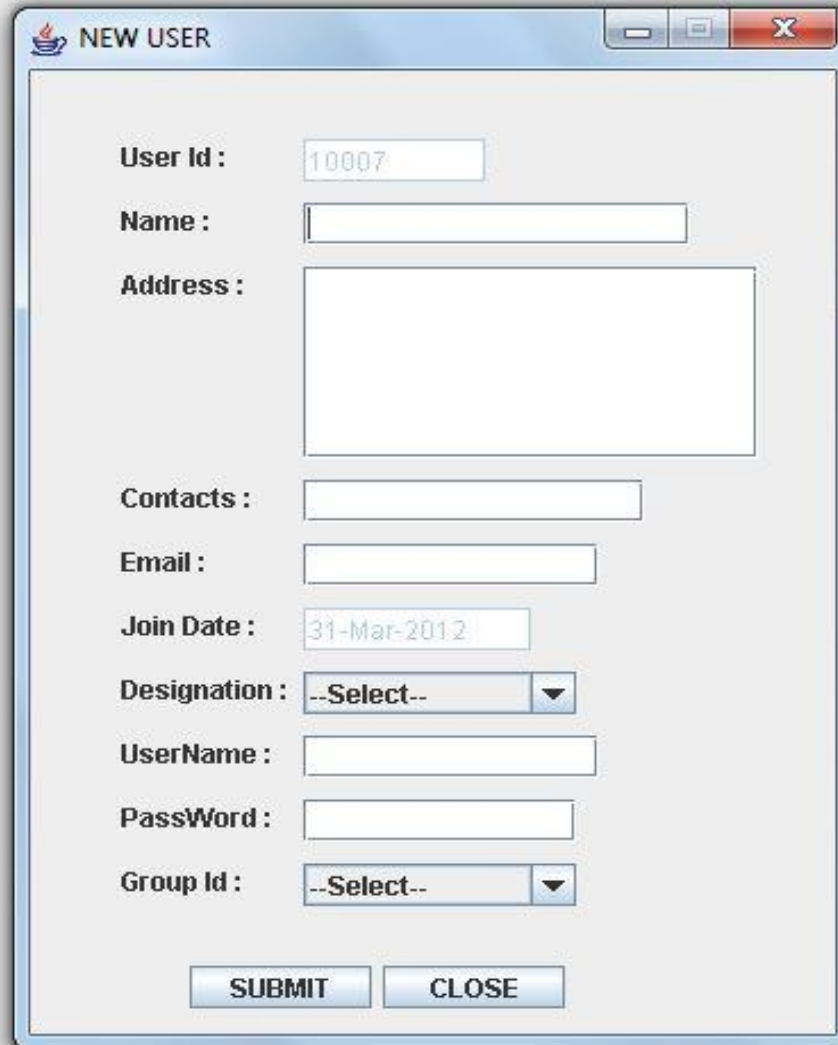
**LOGIN FORM**

**UserName :**

**PassWord :**

**LOGIN** **EXIT**

# Add New User



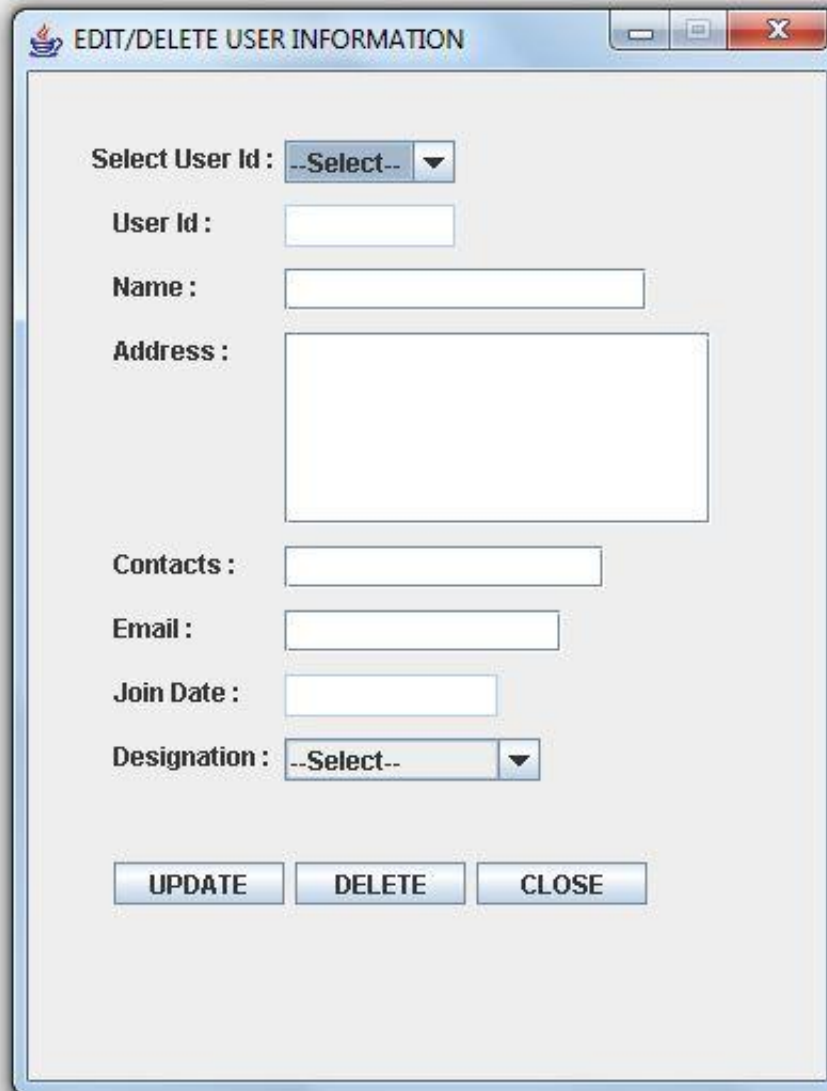
A screenshot of a 'NEW USER' dialog box. The dialog has a title bar with a coffee cup icon, the text 'NEW USER', and standard window controls (minimize, maximize, close). The form contains several input fields: 'User Id' with the value '10007', 'Name' (empty), 'Address' (empty text area), 'Contacts' (empty), 'Email' (empty), 'Join Date' with the value '31-Mar-2012', 'Designation' (dropdown menu showing '--Select--'), 'UserName' (empty), 'PassWord' (empty), and 'Group Id' (dropdown menu showing '--Select--'). At the bottom are 'SUBMIT' and 'CLOSE' buttons.

User Id :	10007
Name :	
Address :	
Contacts :	
Email :	
Join Date :	31-Mar-2012
Designation :	--Select--
UserName :	
PassWord :	
Group Id :	--Select--

SUBMIT CLOSE



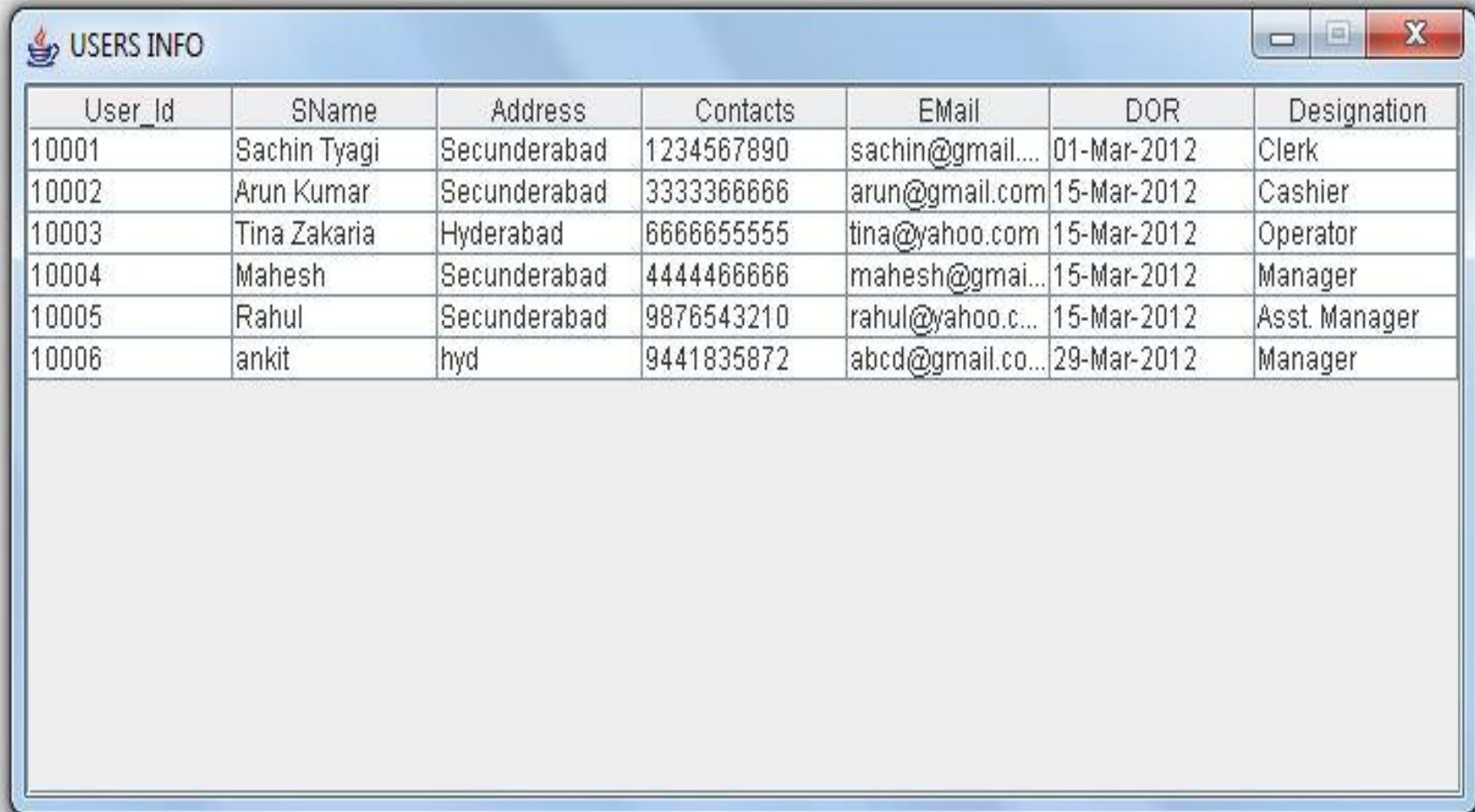
# Edit/Delete User Information



The image shows a Java Swing window titled "EDIT/DELETE USER INFORMATION". The window contains a form with the following fields and controls:

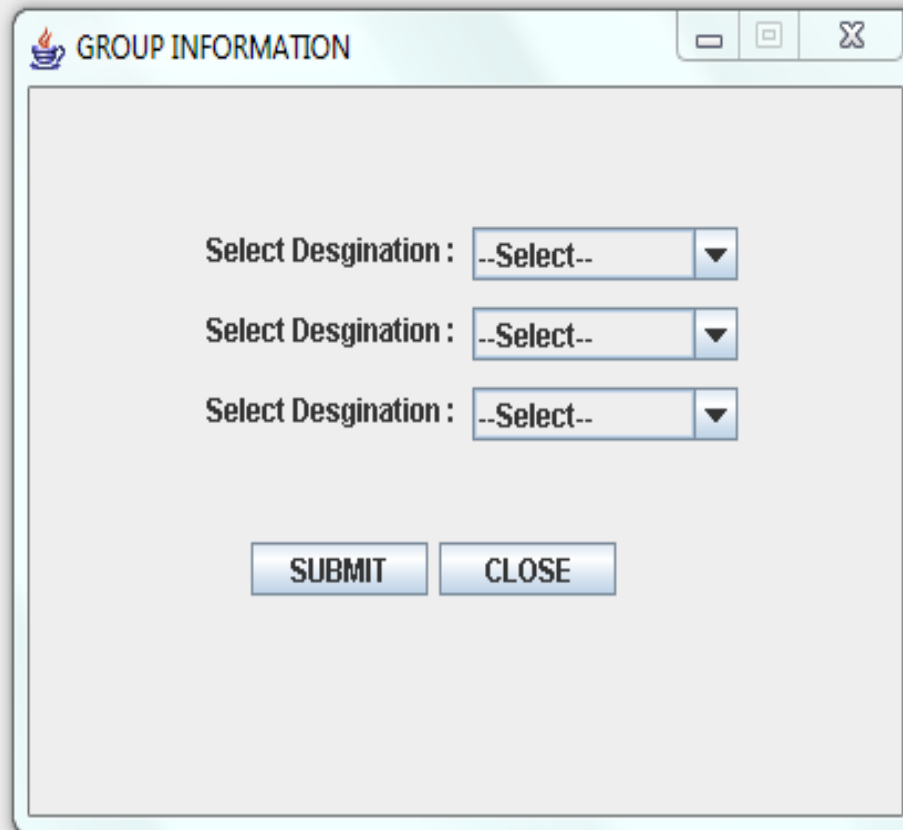
- Select User Id :** A dropdown menu with "--Select--" as the selected option.
- User Id :** A text input field.
- Name :** A text input field.
- Address :** A large text area for multi-line input.
- Contacts :** A text input field.
- Email :** A text input field.
- Join Date :** A text input field.
- Designation :** A dropdown menu with "--Select--" as the selected option.
- Buttons:** Three buttons labeled "UPDATE", "DELETE", and "CLOSE" are located at the bottom of the form.

# View User Information



User_Id	SName	Address	Contacts	EMail	DOR	Designation
10001	Sachin Tyagi	Secunderabad	1234567890	sachin@gmail...	01-Mar-2012	Clerk
10002	Arun Kumar	Secunderabad	3333366666	arun@gmail.com	15-Mar-2012	Cashier
10003	Tina Zakaria	Hyderabad	6666655555	tina@yahoo.com	15-Mar-2012	Operator
10004	Mahesh	Secunderabad	4444466666	mahesh@gmai...	15-Mar-2012	Manager
10005	Rahul	Secunderabad	9876543210	rahul@yahoo.c...	15-Mar-2012	Asst. Manager
10006	ankit	hyd	9441835872	abcd@gmail.co...	29-Mar-2012	Manager

# Set Groups



GROUP INFORMATION

Select Desgination : --Select-- ▼

Select Desgination : --Select-- ▼

Select Desgination : --Select-- ▼

SUBMIT CLOSE

# View Groups



Group_Id	Members
1	Manager,Asst. Manager,Clerk
2	Manager,Clerk,Cashier
3	Asst. Manager,Cashier,Operator
4	Manager,Asst. Manager,Cashier

# User Login



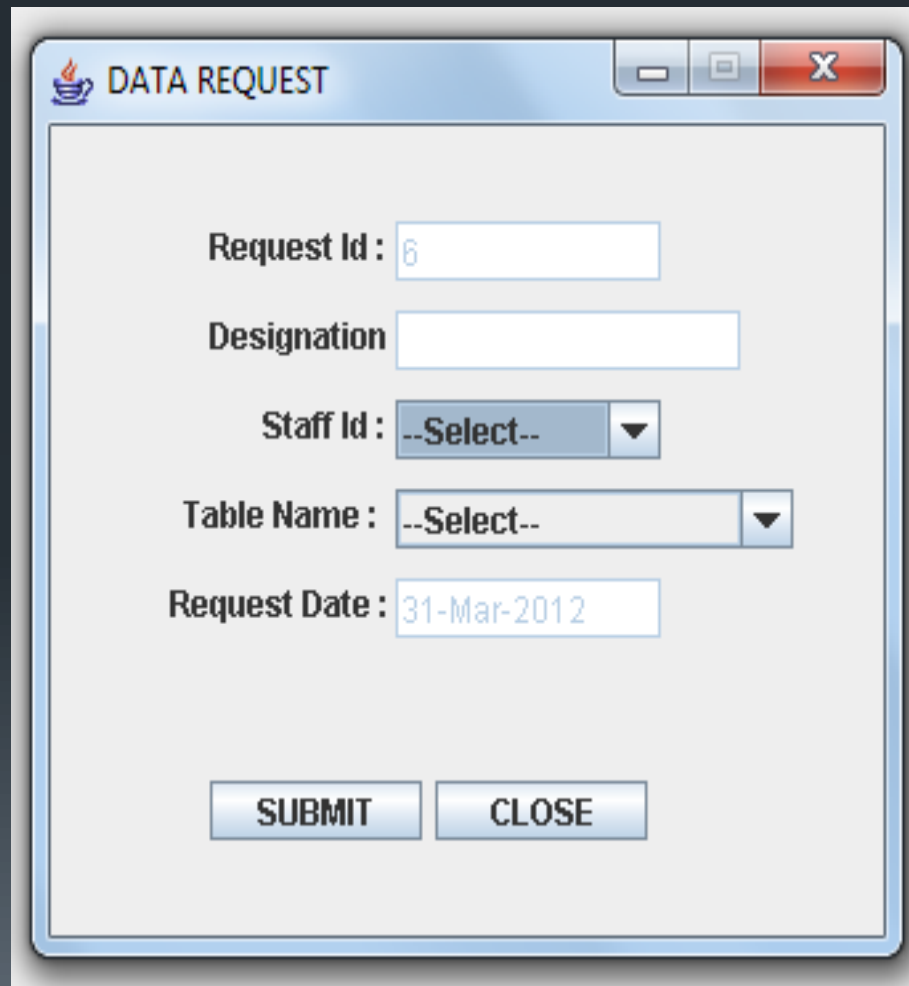
STAFF LOGIN FORM

UserName :

PassWord :

LOGIN EXIT

# Request for Data Sharing



**DATA REQUEST**

Request Id : 6

Designation

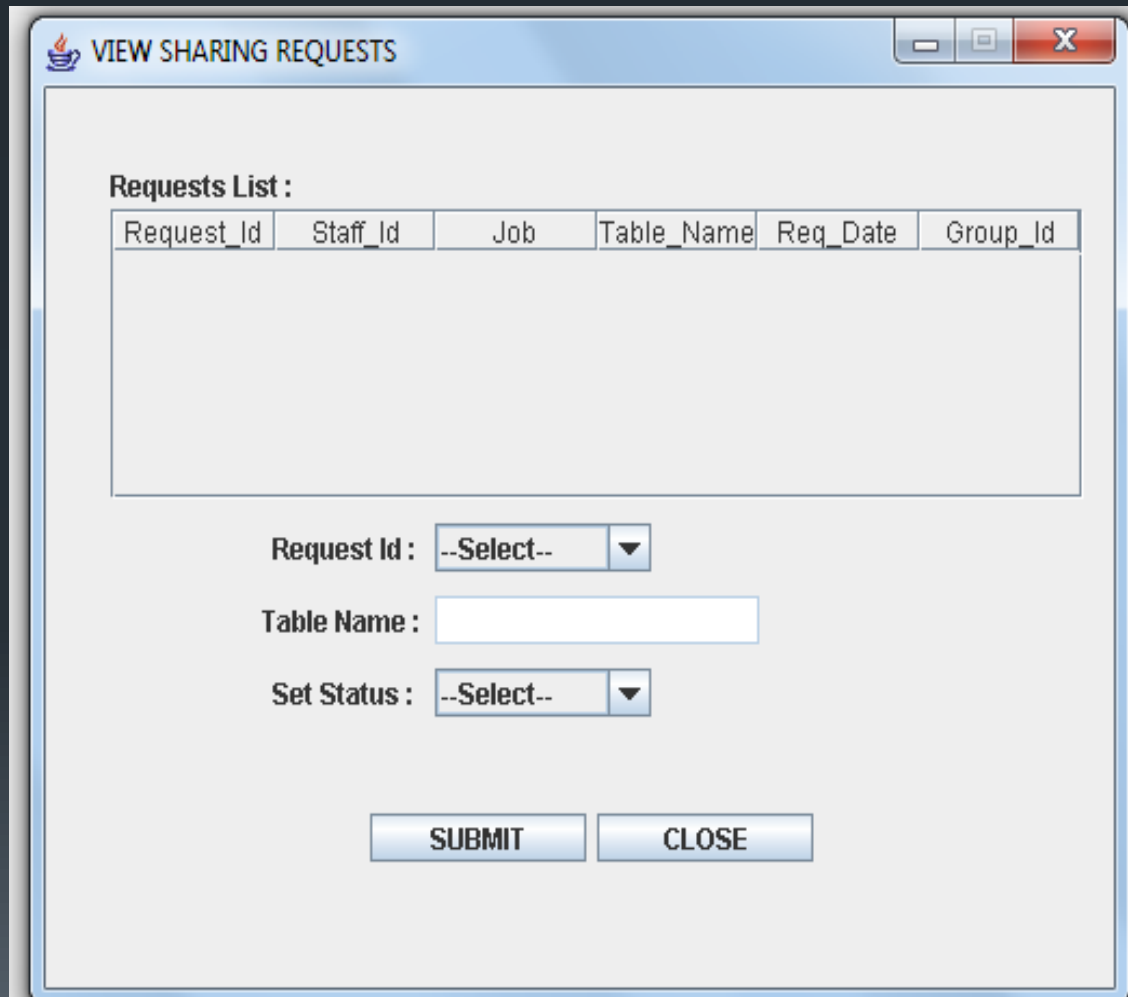
Staff Id : --Select--

Table Name : --Select--

Request Date : 31-Mar-2012

SUBMIT CLOSE

# View Data Sharing Requests



**VIEW SHARING REQUESTS**

**Requests List :**

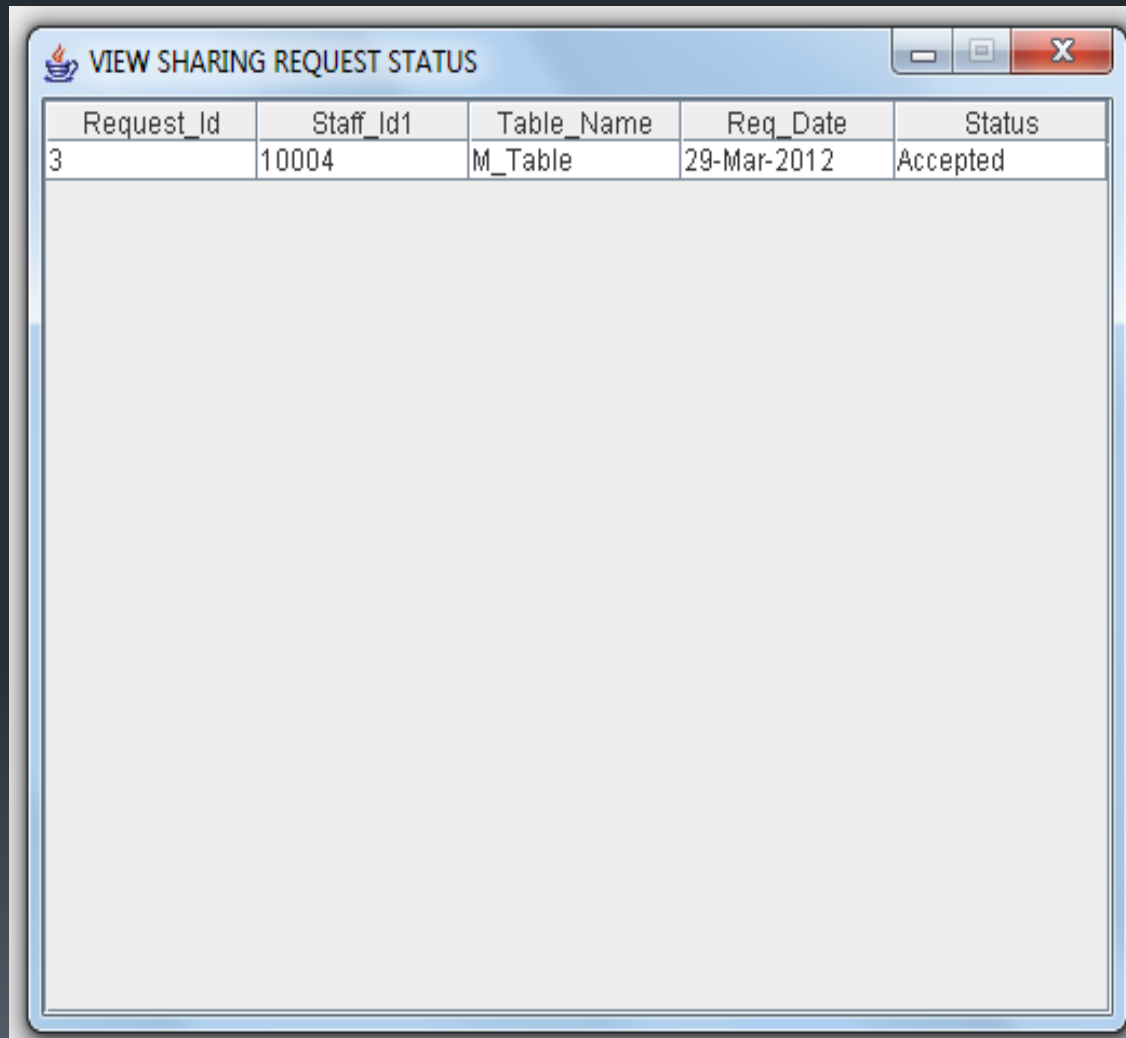
Request_Id	Staff_Id	Job	Table_Name	Req_Date	Group_Id
------------	----------	-----	------------	----------	----------

Request Id :

Table Name :

Set Status :

# View Sharing Request Status



A screenshot of a Java Swing window titled "VIEW SHARING REQUEST STATUS". The window has a standard Mac OS X-style title bar with a red close button, a yellow maximize button, and a green minimize button. Inside the window, there is a table with five columns: "Request\_Id", "Staff\_Id1", "Table\_Name", "Req\_Date", and "Status". The table contains one data row with the following values: "3", "10004", "M\_Table", "29-Mar-2012", and "Accepted". The table is set against a light gray background.

Request_Id	Staff_Id1	Table_Name	Req_Date	Status
3	10004	M_Table	29-Mar-2012	Accepted



# Trace Shared Data


 TRACE SHARED DATA

Table Name :

Shared Path :

Request_Id	Staff_Id	Job	Staff_Id1	Job1	Req_Date	Group_Id	Status
2	10002	Cashier	10001	Clerk	29-Mar-20...	2	Accepted
3	10001	Clerk	10004	Manager	29-Mar-20...	2	Accepted
5	10006	Clerk	10007	Asst. Man...	29-Mar-20...	1	Accepted

Intruder List :

Request_Id	Staff_Id	Job	Staff_Id1	Job1	Req_Date	Group_Id	Status
5	10006	Clerk	10007	Asst. Man...	29-Mar-20...	1	Accepted

# Test Cases



<b>Test Case ID:</b>	<b>1</b>
Purpose:	To login as admin.
Input:	Input the admin's username and password
Expected Result:	User should log in as admin.
Actual Result:	The user has got Administrative privileges by successfully logging in as admin.

<b>Test Case ID:</b>	<b>2</b>
Purpose:	To create a user
Input:	Input a username and his details by logging in as admin.
Expected Result:	When submit button is pressed, new user should be created.
Actual Result:	A new user is created in the database.

**Test Case ID:****3**

Purpose:

To allow sharing of data.

Input:

Request a user of the same group for the data.

Expected Result:

If the user accepts the request then the data is shared and the owner of the data can see the flow of the data.

Actual Result:

After accepting the request the data is shared and the owner of the data can trace the flow of data.

**Test Case ID:****4**

Purpose:

To trace data leakage

Input:

Request a user of different group for unauthorized data sharing.

Expected Result:

Upon acceptance of this unauthorized request, the owner of the data can find the agent guilty of data leakage.

Actual Result:

The owner of the data can see the guilty agent by tracing the flow of data.

# Conclusions

- We have shown that it is possible to assess the likelihood that an agent is responsible for a leak. Our model is relatively simple, but we believe that it captures essential trade-offs.
- Our future work will be to investigate other complex data leakage scenarios such as appropriate model for cases where agents can collude and identify fake tuples.

Thank you!

