ICT3113
Advanced programming in Java

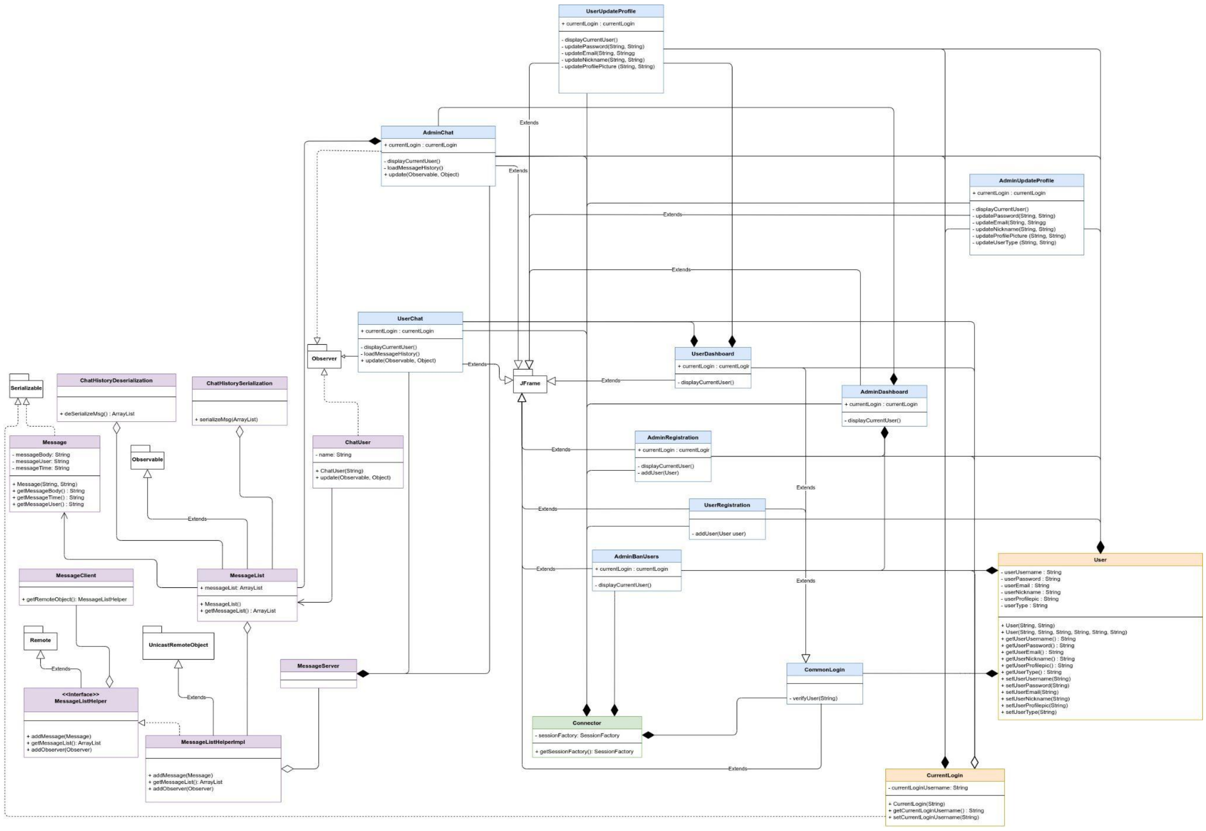
Assignment 01
Multi User Chat Application

Group Members

Registration No:	Name
TG/2018/392	Jayaweera W.M.P.N
TG/2018/421	Weerakoon W.M.W.D
TG/2016/134	Adeesha J.G.N
TG/2016/113	Bandara P.M.C.N

Table of Contents

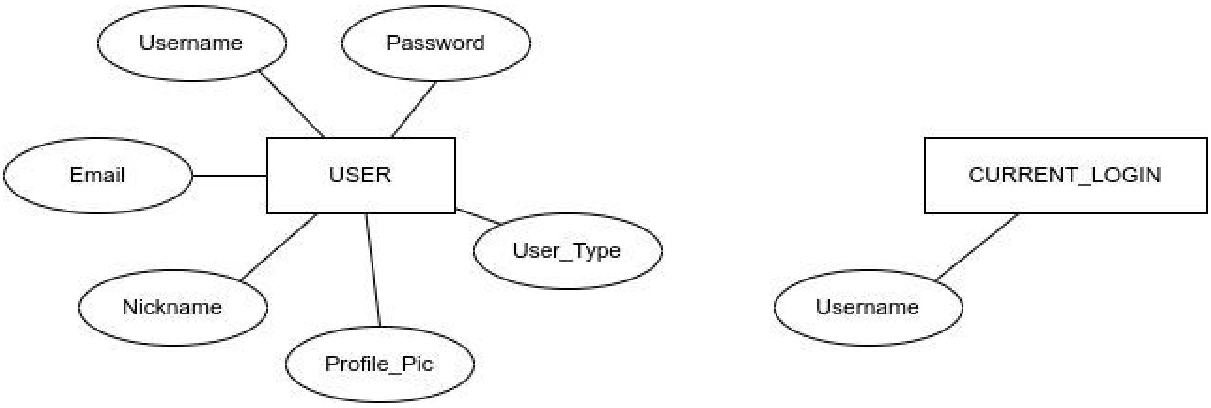
Class Diagrams


UserUpdateProfile
+ currentLogin : currentLogin
- displayCurrentUser()
- updatePassword(String, String)
- updateEmail(String, Stringg
- updateNickname(String, String)
- updateProfilePicture (String, String)
AdminChat
+ currentLogin : currentLogin
- displayCurrentUser()
- loadMessageHistory()
+ update(Observable, Object)
AdminUpdateProfile
+ currentLogin : currentLogin
- displayCurrentUser()
- updatePassword(String, String)
- updateEmail(String, Stringg
- updateNickname(String, String)
- updateProfilePicture (String, String)
- updateUserType (String, String)
UserChat
+ currentLogin : currentLogin
- displayCurrentUser()
- loadMessageHistory()
+ update(Observable, Object)
Observer
JFrame
UserDashboard
+ currentLogin : currentLogir
- displayCurrentUser()
AdminDashboard
+ currentLogin : currentLogin
- displayCurrentUser()
Serializable
ChatHistoryDeserialization
+ deSerializeMsg() : ArrayList
ChatHistorySerialization
+ serializeMsg(ArrayList)
Message
- messageBody: String
- messageUser: String
- messageTime: String
+ Message(String, String)
+ getMessageBody() : String
+ getMessageTime() : String
+ getMessageUser() : String
Observable
ChatUser
- name: String
+ ChatUser(String)
+ update(Observable, Object)
AdminRegistration
+ currentLogin : currentLogir
- displayCurrentUser()
- addUser(User)
UserRegistration
- addUser(User user)
AdminBanUsers
+ currentLogin : currentLogin
- displayCurrentUser()
MessageClient
+ getRemoteObject(): MessageListHelper
MessageList
+ messageList: ArrayList
+ MessageList()
+ getMessageList() : ArrayList
Remote
UnicastRemoteObject
MessageServer
<<Interface>>
MessageListHelper
+ addMessage(Message)
+ getMessageList(): ArrayList
+ addObserver(Observer)
MessageListHelperImpl
+ addMessage(Message)
+ getMessageList(): ArrayList
+ addObserver(Observer)
Connector
- sessionFactory: SessionFactory
+ getSessionFactory(): SessionFactory
CommonLogin
- verifyUser(String)
User
- userUsername : String
- userPassword : String
- userEmail : String
- userNickname : String
- userProfilepic : String
- userType : String
+ User(String, String)
+ User(String, String, String, String, String, String)
+ getUserUsername() : String
+ getUserPassword() : String
+ getUserEmail() : String
+ getUserNickname() : String
+ getUserProfilepic() : String
+ getUserType() : String
+ setUserUsername(String)
+ setUserPassword(String)
+ setUserEmail(String)
+ setUserNickname(String)
+ setUserProfilepic(String)
+ setUserType(String)
CurrentLogin
- currentLoginUsername: String
+ CurrentLogin(String)
+ getCurrentLoginUsername() : String
+ setCurrentLoginUsername(String)
Extends

For a better view

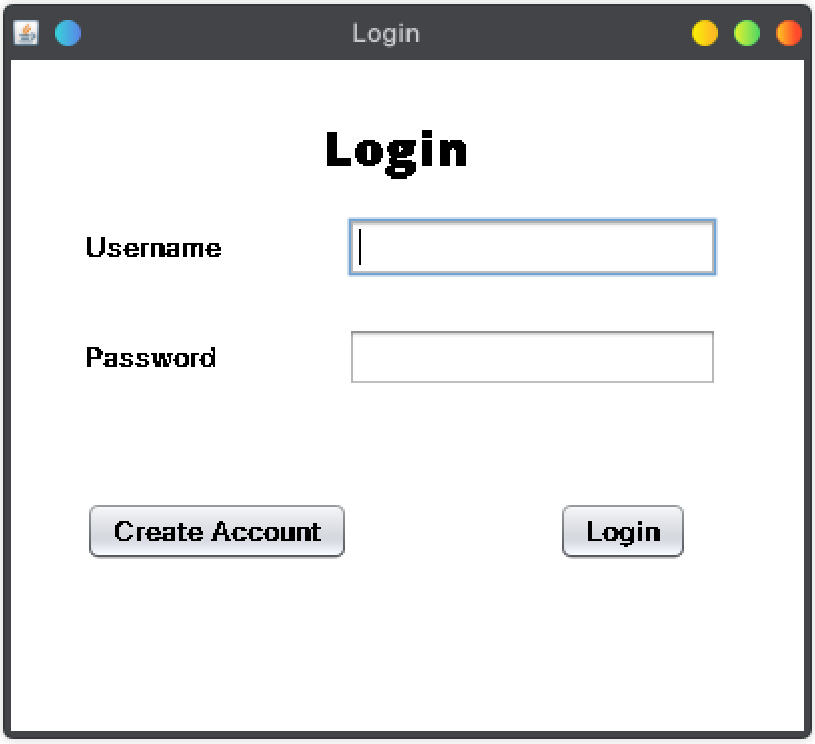
https://bit.ly/3t4NOCB or https://drive.google.com/file/d/1VMqMF8o-I6Vl5zjSoex78PzOkoreqrJG/view?usp=sharing

ER Diagram for MySQL Backend

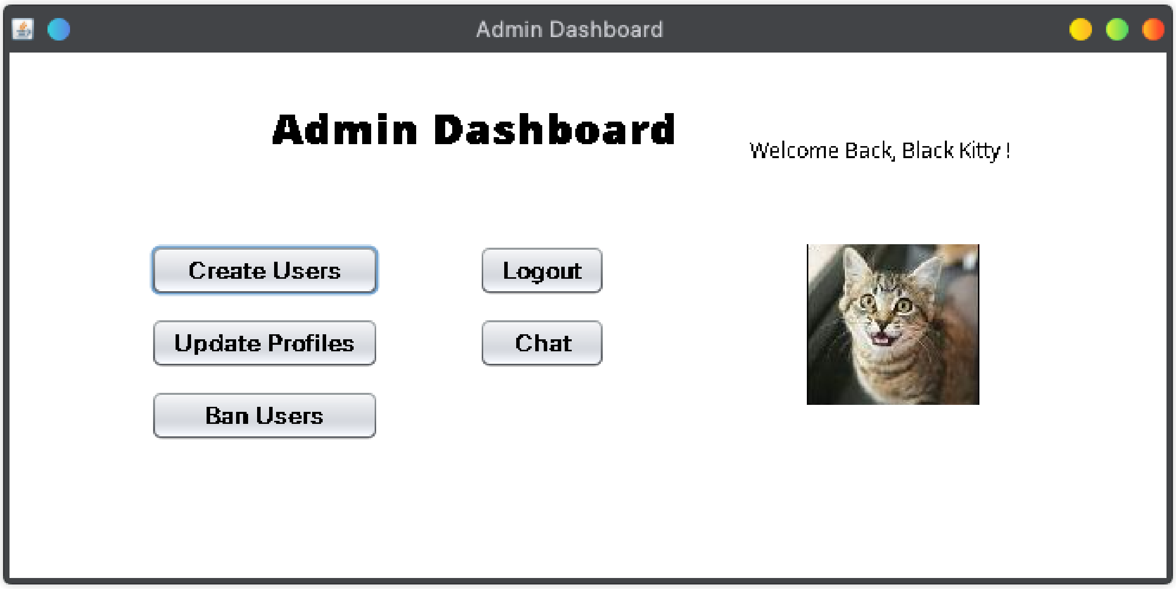


GUI Designs

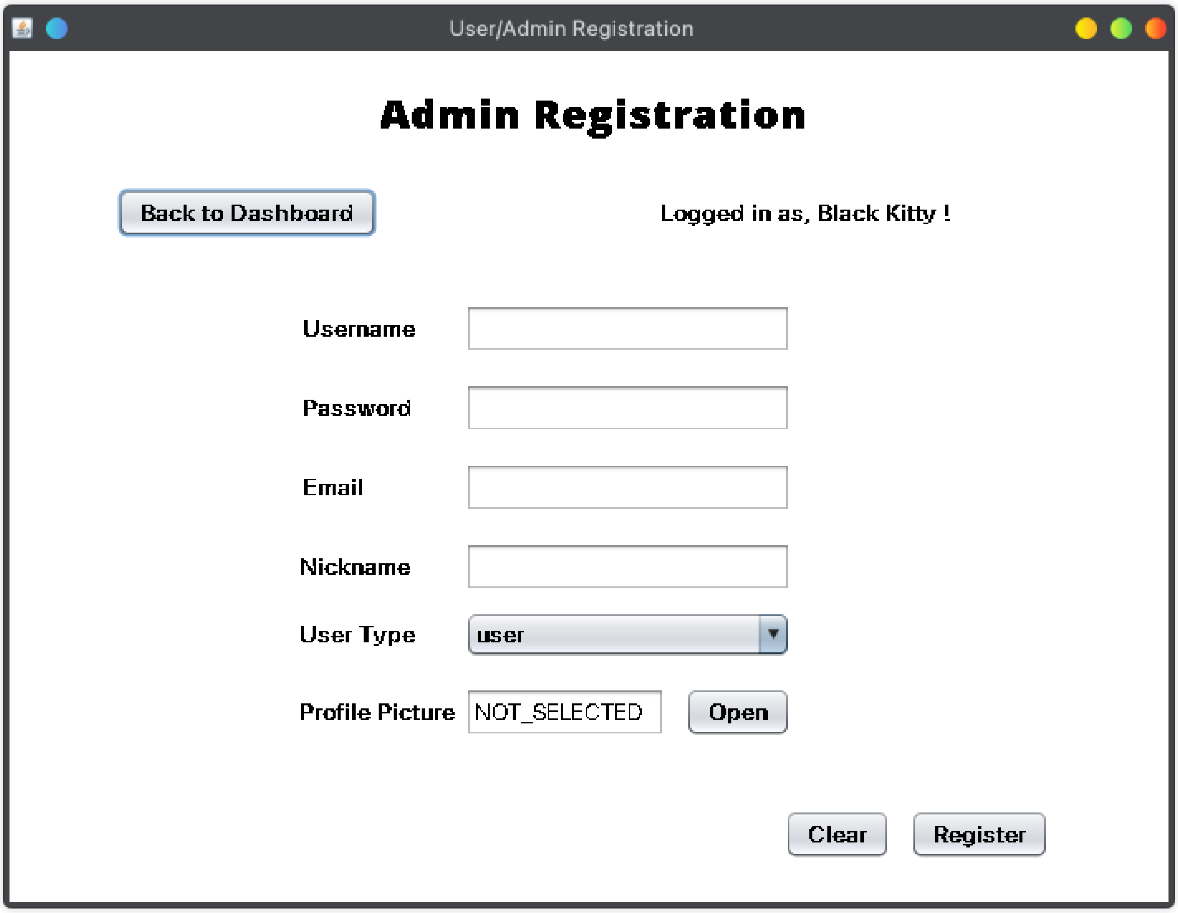
Common UI



Admin UI



Welcome Back, Black Kitty !



User/Admin Registration

Admin Registration

Logged in as, Black Kitty !

Username

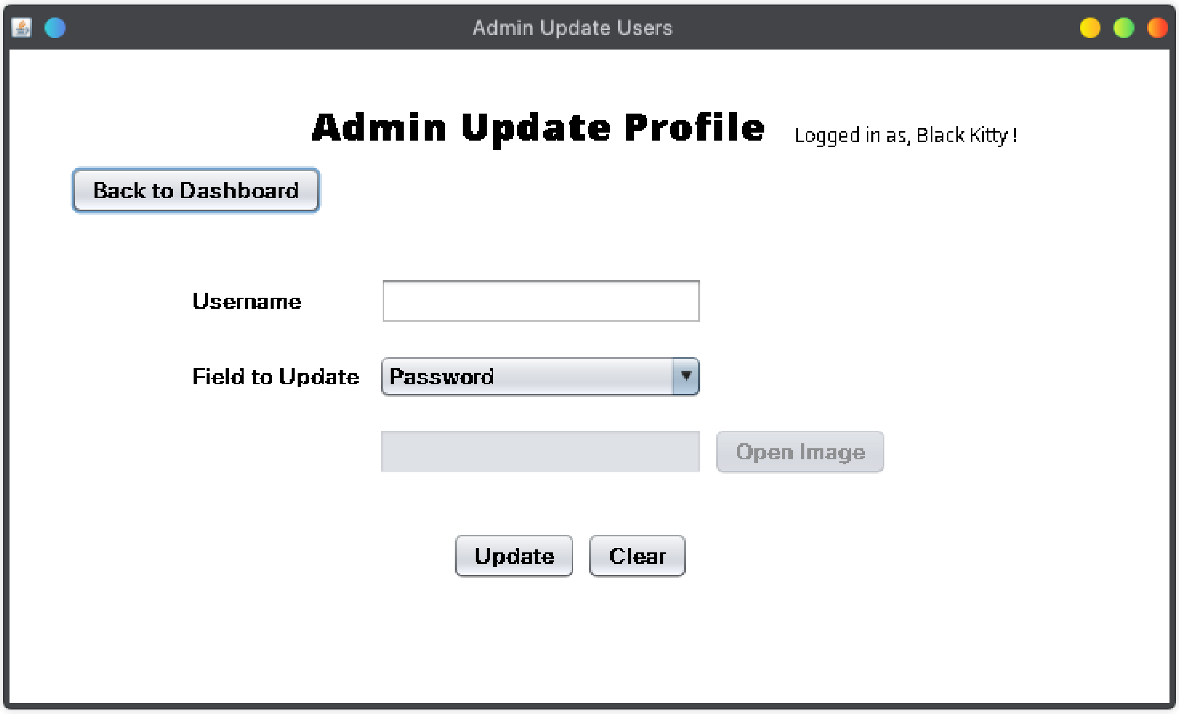
Password

Email

Nickname

User Type user

Profile Picture NOT_SELECTED



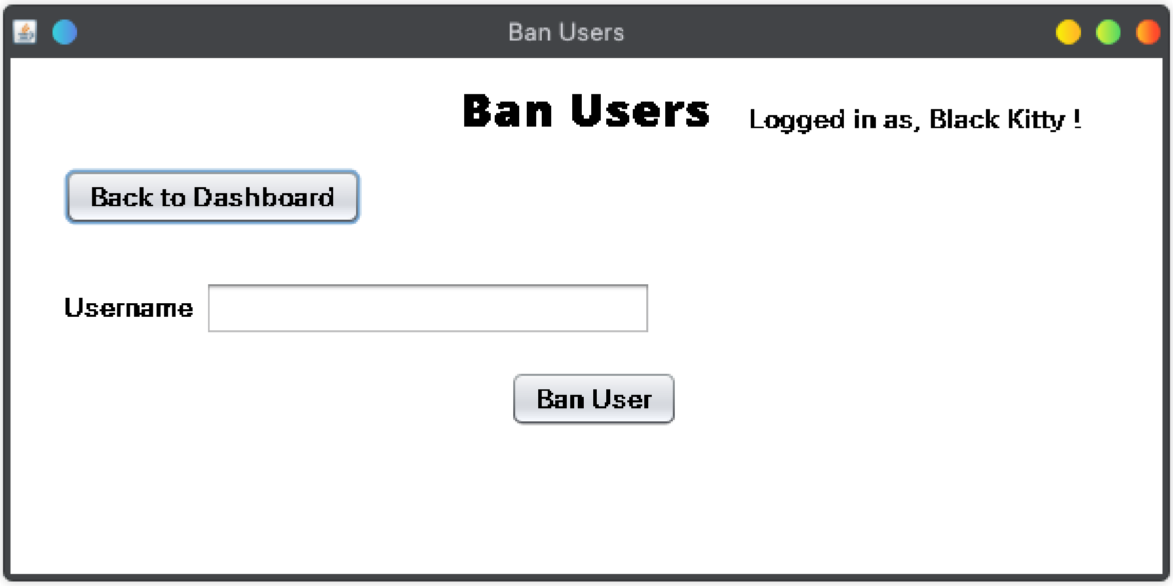
Admin Update Users

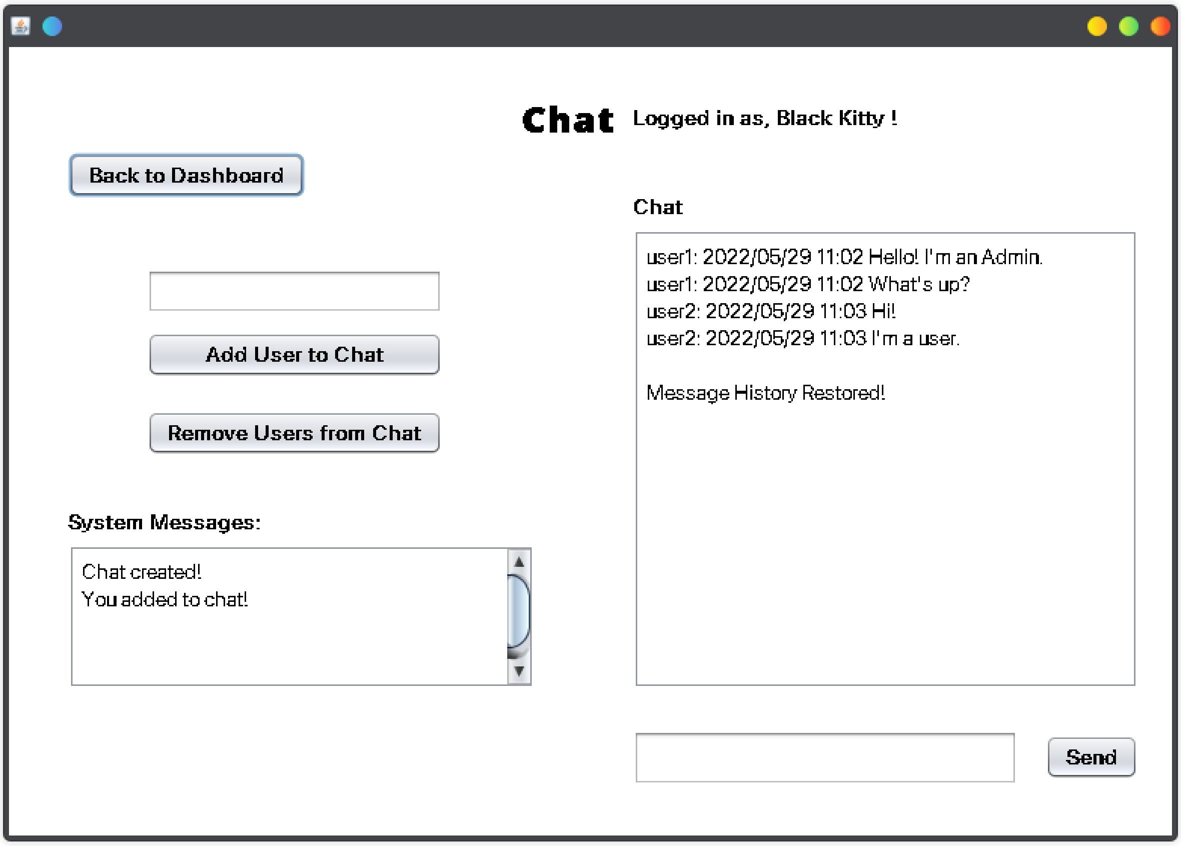
Admin Update Profile

Logged in as, Black Kitty !

Username

Field to Update Password



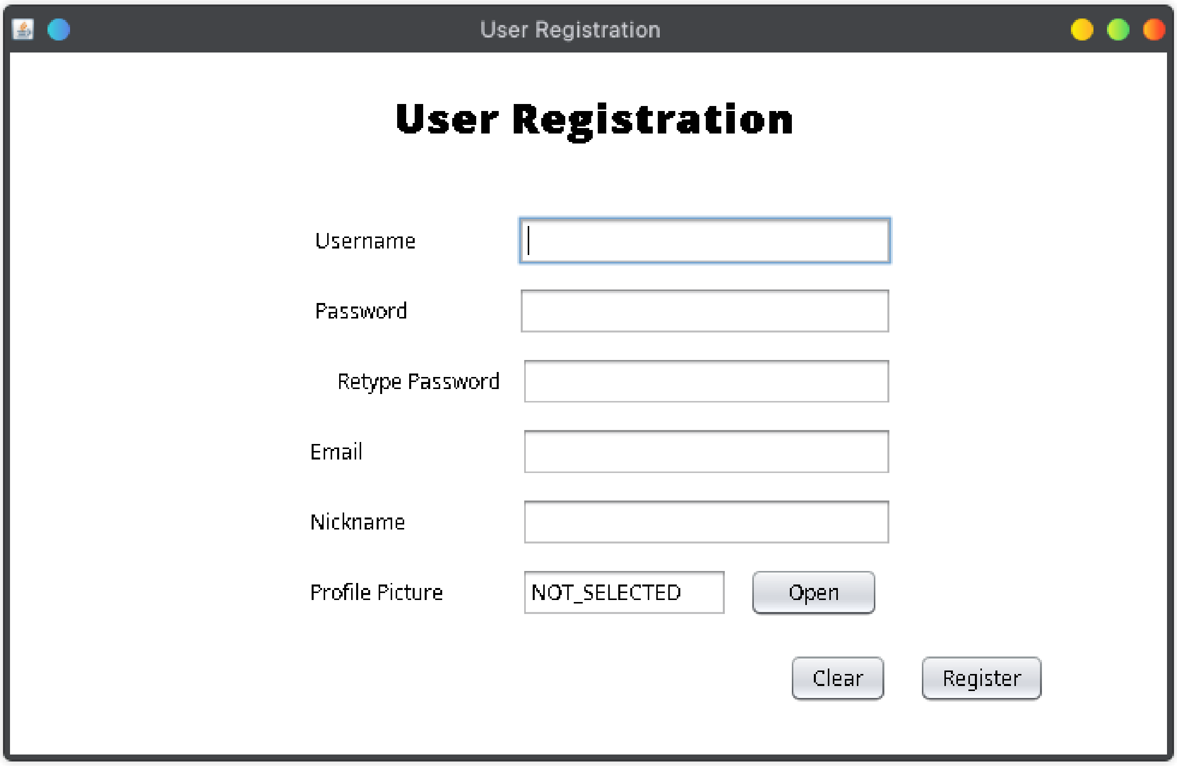


Chat created!
You added to chat!

user1: 2022/05/29 11:02 Hello! I'm an Admin.
user1: 2022/05/29 11:02 What's up?
user2: 2022/05/29 11:03 Hi!
user2: 2022/05/29 11:03 I'm a user.

Message History Restored!

User UI



User Registration

User Registration

Username

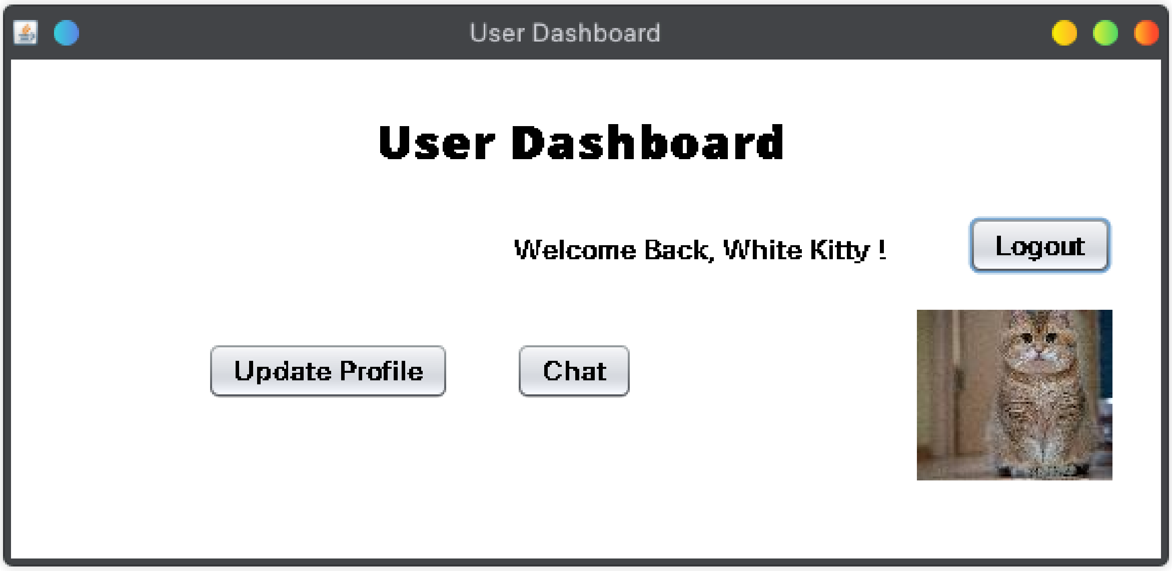
Password

Retype Password

Email

Nickname

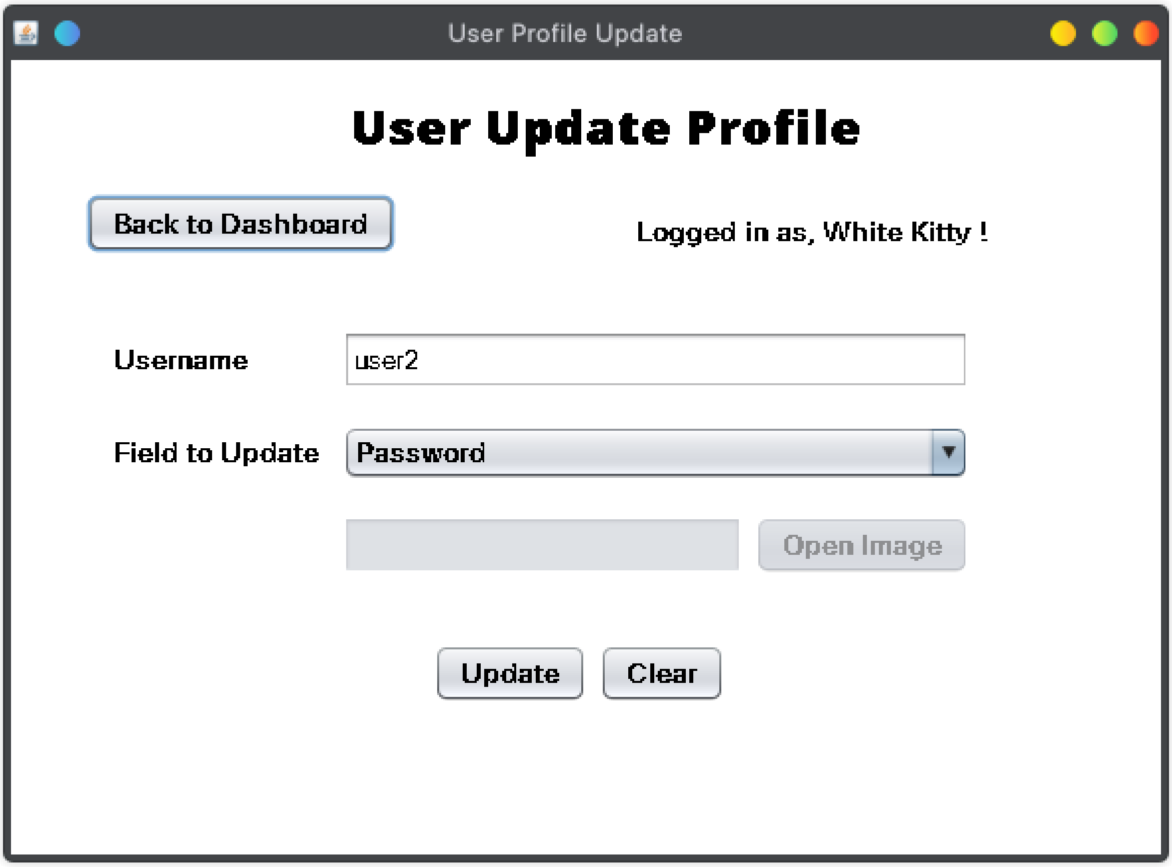
Profile Picture NOT_SELECTED



User Dashboard

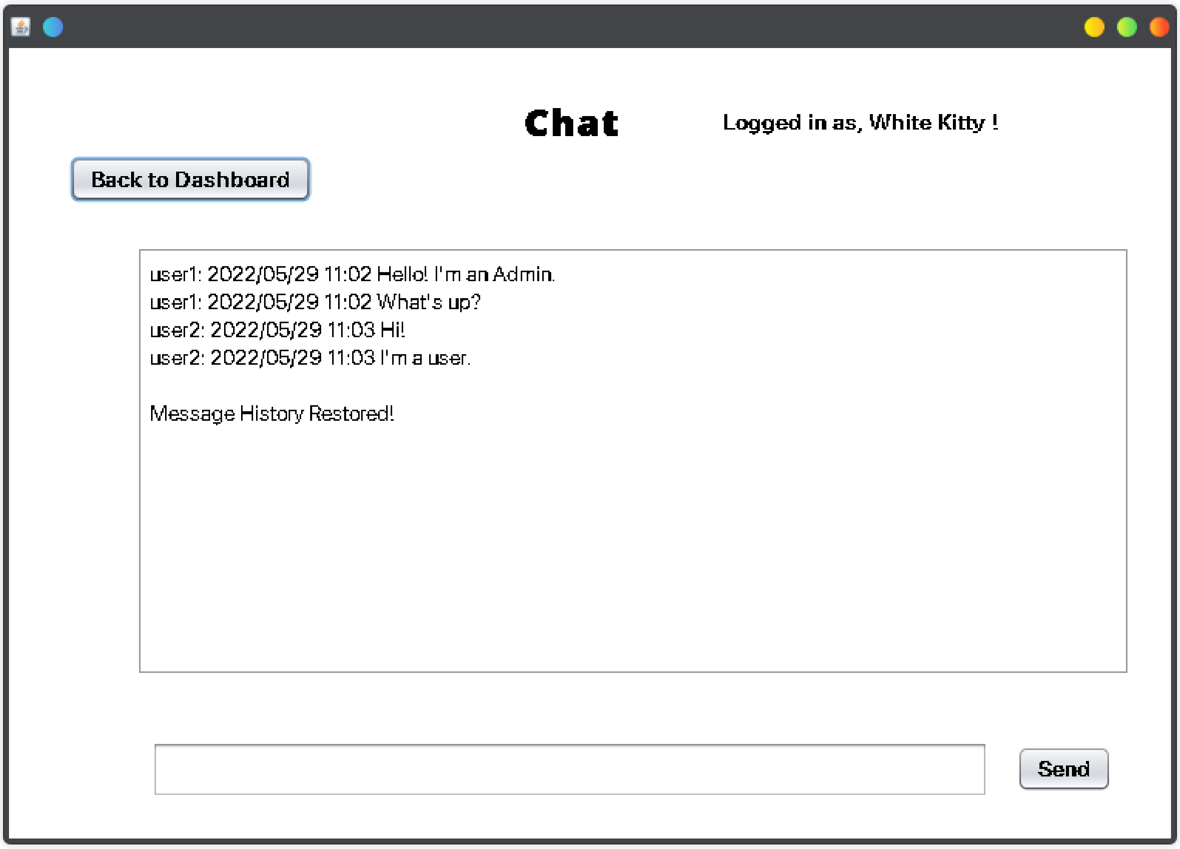
User Dashboard

Welcome Back, White Kitty !



Username: user2

Field to Update: Password



```
user1: 2022/05/29 11:02 Hello! I'm an Admin.
user1: 2022/05/29 11:02 What's up?
user2: 2022/05/29 11:03 Hi!
user2: 2022/05/29 11:03 I'm a user.

Message History Restored!
```

Tools, Technologies, and Concepts

NetBeans IDE 8.2

We have used NetBeans IDE 8.2 as the development environment for this project. At first, we selected Apache NetBeans IDE 13, but it does not have the direct support for the Hibernate Java framework. After trying some workarounds and failures, we have decided to roll back to its earlier deprecated release, which was released by Oracle.

NetBeans IDE 8.2 is a lightweight and feature rich IDE which has direct support for JDK 8. It supports creating interfaces with Java Swing classes very easily. Also, it has good support for plugins like Hibernate, which we needed the most. NetBeans IDE 8.2 has built-in support to connect to MySQL databases and do essential maintenance while the development process is going on. But, it is still an outdated software with lack of modern IDE features, and also it has a very primary interface.

Hibernate 4.3

We have used Hibernate Framework to connect to the MySQL database and do the CRUD operations. Our project has a login system which is supported by MySQL databases, and it does many CRUD operations with the database.

Hibernate is a very easy and convenient framework for building connections to a MySQL database from Java and doing CRUD operations. With the help of the Hibernate plugin for NetBeans, it is easy to create Configurations, Mappings, Reverse engineering files, POJO (Plain Old Java Object) files and connection files.

By using Java Hibernate Framework, the developers can avoid unnecessary coding and easily create apps which have MySQL backends.

MySQL JDBC Driver 5.1

MySQL JDBC Driver 5.1 is the driver we have used to make the connection between Hibernate Framework and MySQL database.

Hibernate Framework supports directly for the MySQL JDBC Driver, and it is the most famous and frequently used driver for Java applications. The driver files are available online to download, and it has several versions according to the host architecture. There is also a platform independent version of it, and it was helpful and easy to configure on any OS platform. We have used Linux as the OS platform and MySQL JDBC Driver 5.1 works well with MySQL (Maria DB) database backend.

Java Swing

Swing is a GUI widget toolkit for Java. We have used it to create user interfaces for the project. Java Swing is a good old toolkit which supports creating GUI interfaces easily and graphically using basic UI principals. Java Swing is better than the old Abstract Window Toolkit (AWT) and creates more attractive GUIs than AWT.

Observer Pattern

Observer pattern is a common software design pattern which has a subject and maintains a list of its dependencies to notify them when the subject has changed.

Java Programming Language has built-in support for observer pattern, and we have used Observer pattern in our project. Our project uses an Observer pattern to notify users when the message list has been changed. The system maintains a list of messages sent by various users, it acts as the observable and the users are considered as observers which can subscribe and unsubscribe from the admin panel.

Observer pattern is the most suitable pattern for that kind of scenario, and implementation is easier with Java built-in support.

Serialization

In computing, serialization or serialization is the process of translating a data structure or object state into a format that can be stored or transmitted and reconstructed later.

Serialization is a technique used to translate data (objects) into a format which can be stored to a file, database or transmit. After, it can translate to its initial format by deserializing the serialized data. In Java, there are high level streams called ObjectInputStream and ObjectOutputStream, and they can be used to serialize and deserialize objects into a file.

We have used serialization techniques to store chat history and later on, users can get their chat history back when log in to the system. This technique is the easiest and the most convenient way to store objects into a file and use them when they are needed.

Java Remote Method Invocation (RMI)

The RMI (Remote Method Invocation) is an API that provides a mechanism to create a distributed application in java. The RMI allows an object to invoke methods on an object running in another JVM.

We have used Java RMI to create a server which broadcasts an array list which contains messages. They can be accessed by remote users to access messages and send messages which can be received by others.

Java RMI methods are accessible by importing UnicastRemoteObject, and It's easy to implement in a chat app. While we have to give access to messages via a LAN, remote method invocation is needed.

Java Swing built-in Threads

Threads in Java are used to run sub-processes parallely. While multiple users are accessing the chat system, it’s needed to implement threads for this project.

Threads are used to run Java Swing interfaces, and multiple Swing interfaces can run simultaneously. When creating Swing interfaces, NetBeans IDE automatically generates the classes as runnable, and they work with threads.

Assumptions and Limitations

Only one chat is available in the system

We have assumed that there is only one chat at a time. Users can access that chat only.

All users are in same LAN

All users must be in the same LAN network because this app is not configured to use over the internet. While we are instructed to run this app on a single PC, we have used localhost as the host.

Users will not use multimedia messages

In this chat application, users are unable to send multimedia messages or attachments as messages.

No encryption (Conversations or Passwords)

Normally, There must be an encryption mechanism implemented to a chat system. We haven't used an encryption mechanism to prevent over complexing the Application.

Conversation limitations

While we are storing messages as a serialized object in a file, that file would be hard to handle if there are more messages and its size will increase. After a finite amount of time, the system will get slow.

No email verification

Normally, a chat system must have a mechanism to verify user emails and verify that email belongs to the user. There is no such an implementation while it’s hard to implement.

Admin can reset passwords and login to user accounts

While Admin has full control of the system, the admin can log into other user’s accounts by resetting their passwords. We assume that the admin will respect the user's privacy.

Using Java Swing is ugly

The Java swing widget tool is an outdated tool. The interfaces which are generated through Java Swing are not very good looking when compared to modern systems.

Individual Contribution

Group member	Contribution
TG/2018/392 - Jayaweera W.M.P.N	ER Diagrams, Database Design and Development, Observer Pattern Implementation
TG/2018/421 - Weerakoon W.M.W.D	Class Diagrams, Hibernate CRUD Functions Creation, Serialization for Saving Message History
TG/2016/134 - Adeesha J.G.N	User Interfaces Design, JAVA RMI Application, Testing Backend
TG/2016/113 - Bandara P.M.C.N	Admin Interfaces Design, JAVA RMI Application, Testing Frontend

References

[1]"GitHub - tg2018392/javachatapp: Java demo chat app with Observer pattern and Hibernate", GitHub, 2022. [Online]. Available: https://github.com/tg2018392/javachatapp. [Accessed: 29- May- 2022].

[2]A. NetBeans, "Welcome to Apache NetBeans", Netbeans.apache.org, 2022. [Online]. Available: https://netbeans.apache.org/. [Accessed: 29- May- 2022].

[3]"Hibernate. Everything data.", Hibernate, 2022. [Online]. Available: https://hibernate.org/. [Accessed: 29- May- 2022].

[4]"MySQL :: Download Connector/J", Dev.mysql.com, 2022. [Online]. Available: https://dev.mysql.com/downloads/connector/j/. [Accessed: 29- May- 2022].