

Communications App

By: Ayoub Mekkaoui, Ben Levy, Akbar Hashimi, Van Nguyen, Ibraheem Fawal

Design

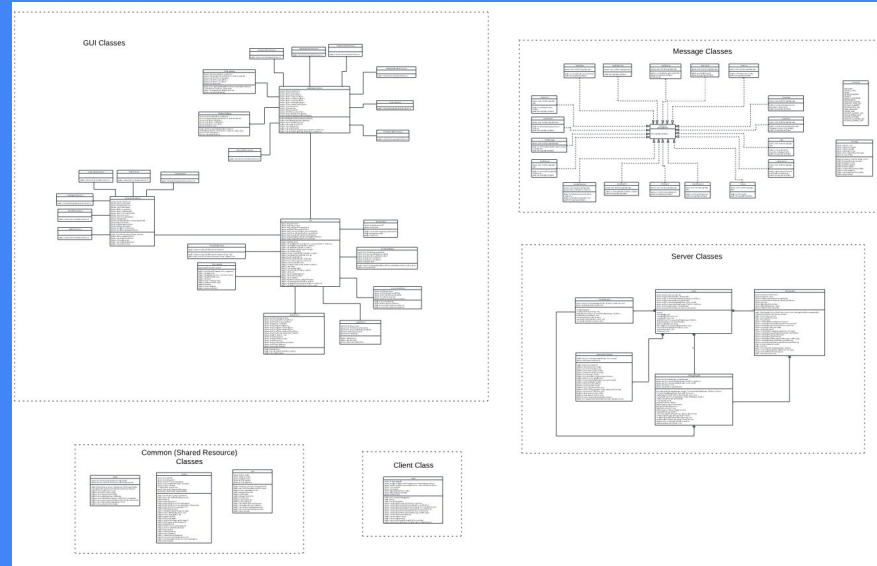
Client-Server

Client Responsibilities

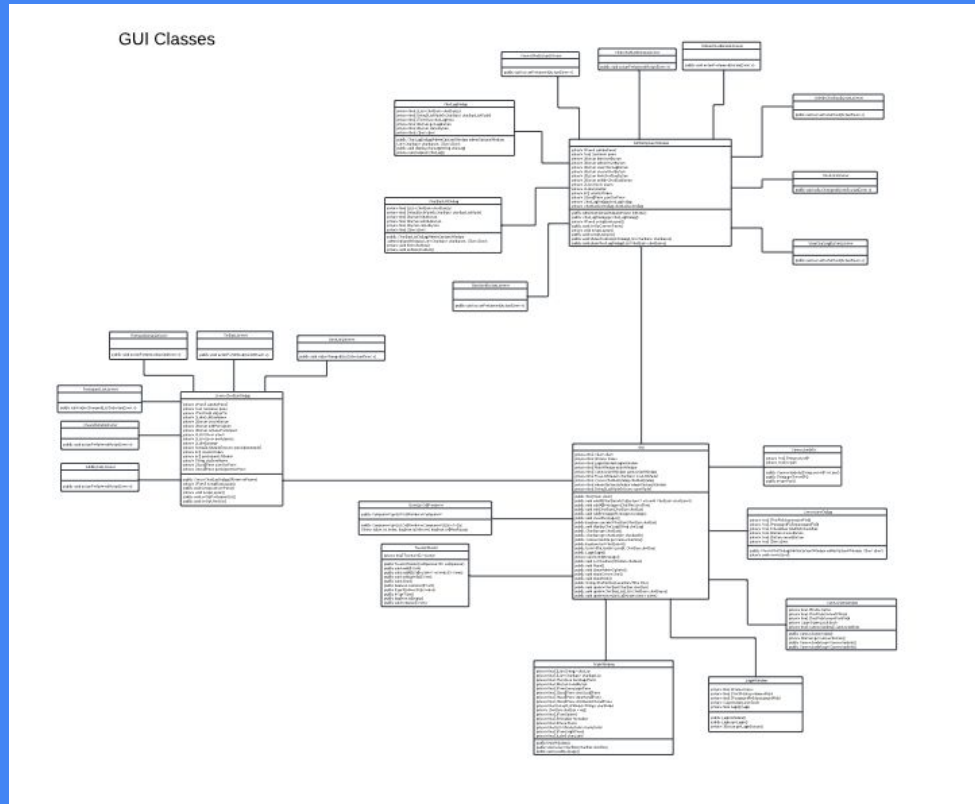
- GUI component
- Sending/receiving messages
- Local state updates

Server Responsibilities

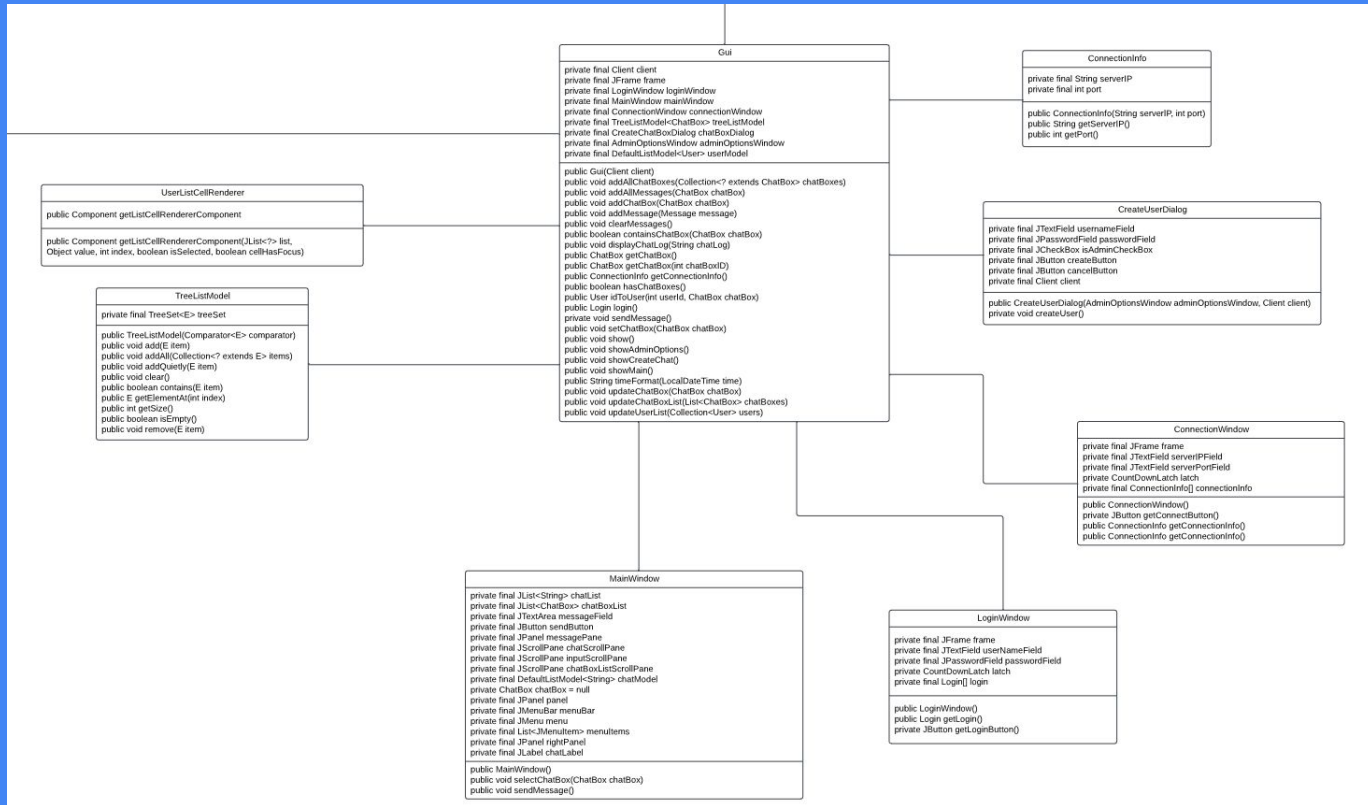
- User Authentication
- Handle multiple clients
- Data storage



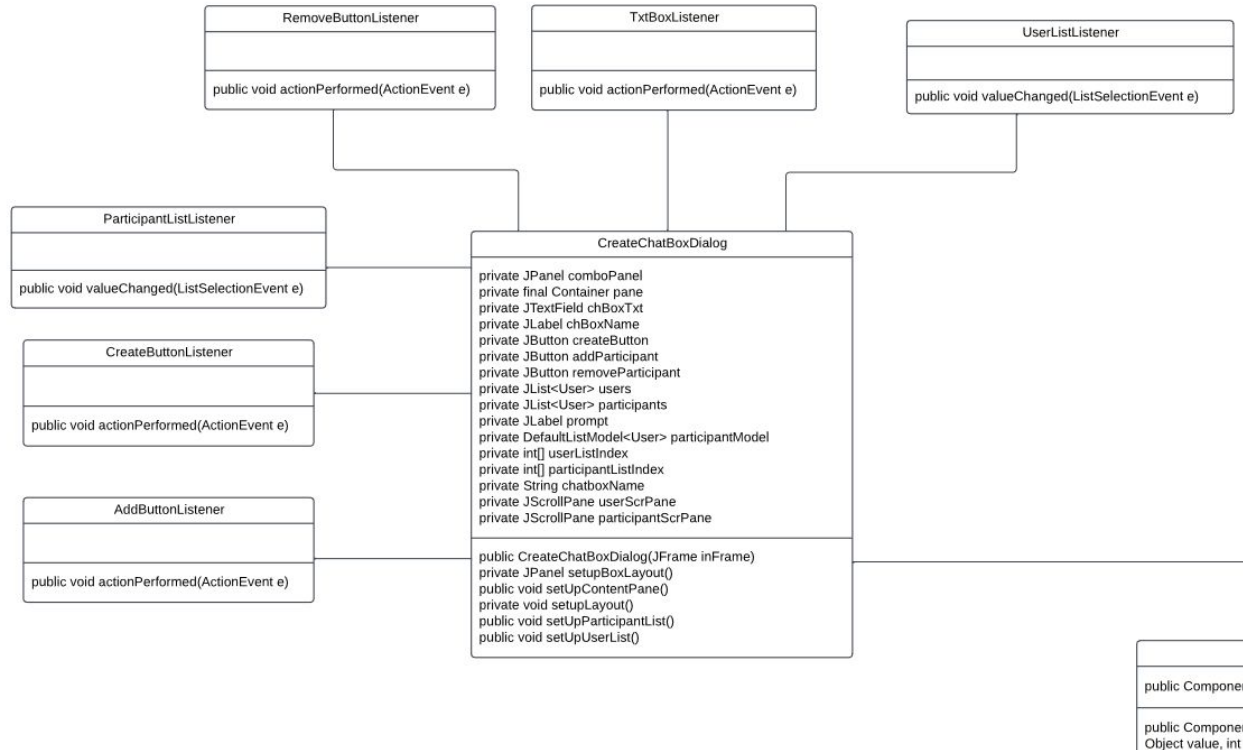
GUI (Class Diagram)



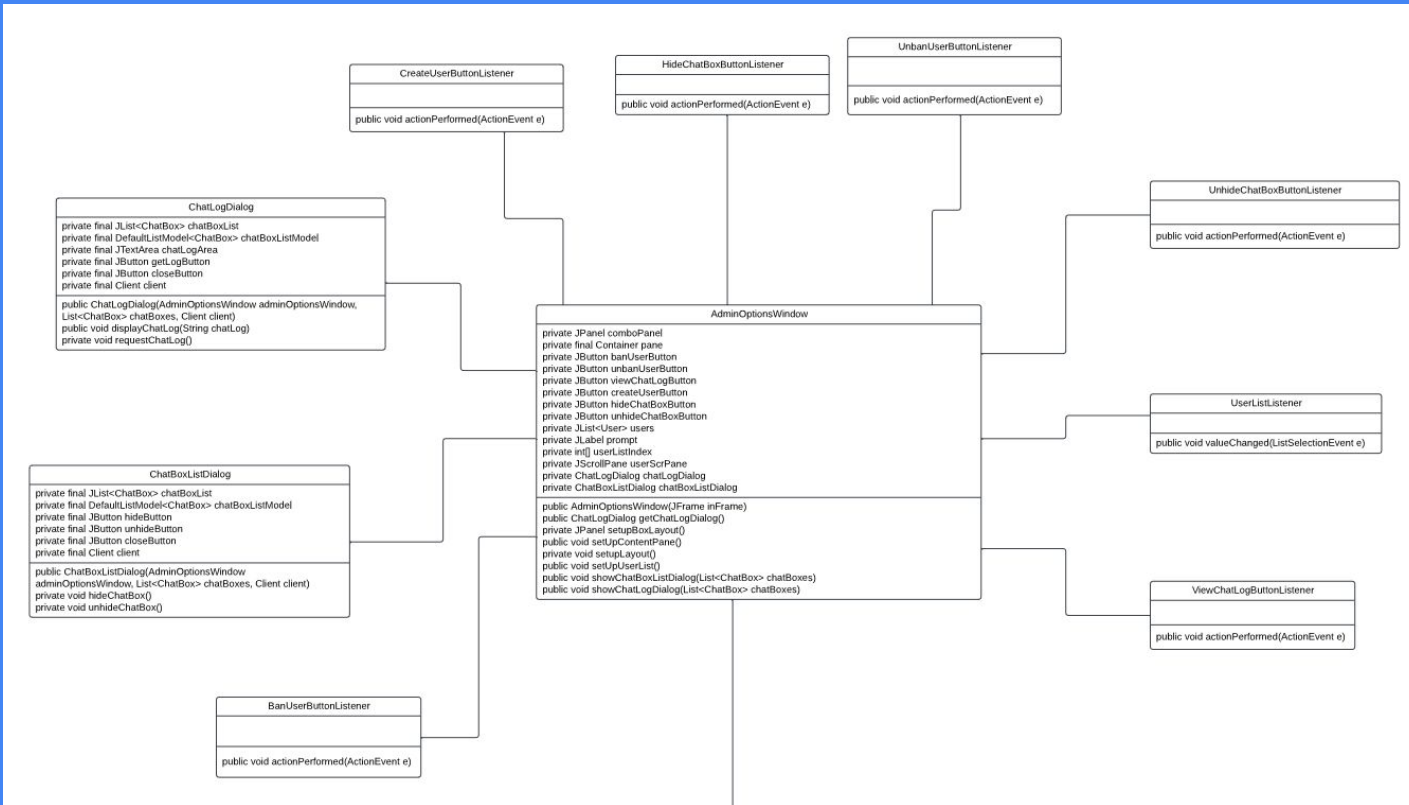
GUI (Class Diagram) - cont



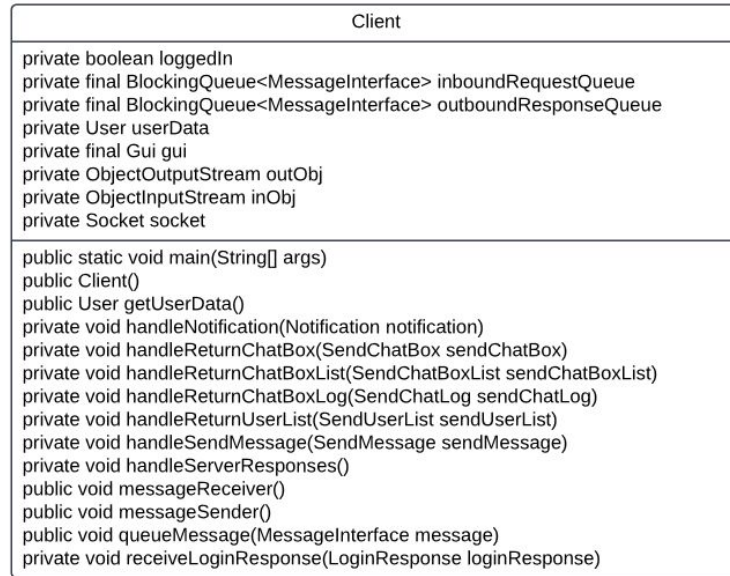
GUI (Class Diagram) - cont



GUI (Class Diagram) - cont



Client (Class Diagrams)



Common (Class Diagrams)

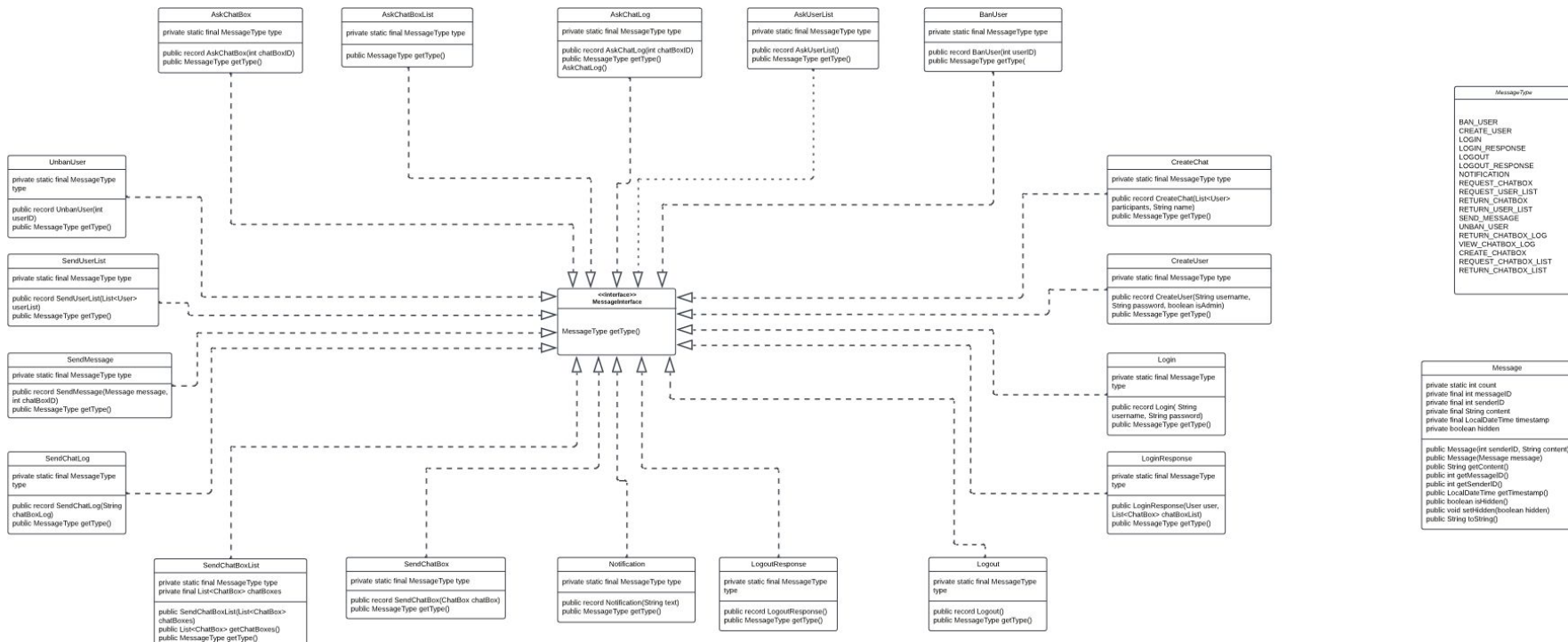
Admin
<pre>private transient MessageHandler messageHandler private transient AuthenticationSystem authenticationSystem</pre>
<pre>public Admin(String username, String password, MessageHandler messageHandler, AuthenticationSystem authenticationSystem) public boolean addUser(User user) public boolean banUser(int userID) public boolean deleteUser(int userID) public boolean hideChatBox(int chatBoxID) public boolean hideChatMessage(int chatBoxID, int messageID) public boolean resetUserPassword(int userID, String newPassword) public void sendSystemMessage(String content) public boolean unbanUser(int userID)</pre>

ChatBox
<pre>private int chatBoxID; private final String name; private Collection<User> participants; private final SortedSet<Message> messages; private boolean isHidden; LocalDateTime creationTime private static final Comparator<Message> MESSAGE_TIMESTAMP_COMPARATOR</pre>
<pre>public static ChatBox getSystemChatBox() public static void resetChatBoxIDGenerator() public ChatBox() private ChatBox(boolean t) public ChatBox(Collection<User> participants) public ChatBox(Collection<User> participants, String name) public ChatBox(List<User> participantsList) public ChatBox(String name) public void addMessage(Message message) public boolean addParticipant(User user) public boolean equals(Object obj) public int getChatBoxID() public ChatBox getEmpty() public SortedSet<Message> getMessages() public List<Message> getMessagesList() public String getName() public Collection<User> getParticipants() public List<User> getParticipantsList() public int hashCode() public void hideChatBox() public boolean isHidden() public LocalDateTime lastUpdated() public boolean removeParticipant(User user) public void setParticipants(Collection<User> participants) public String toString()</pre>

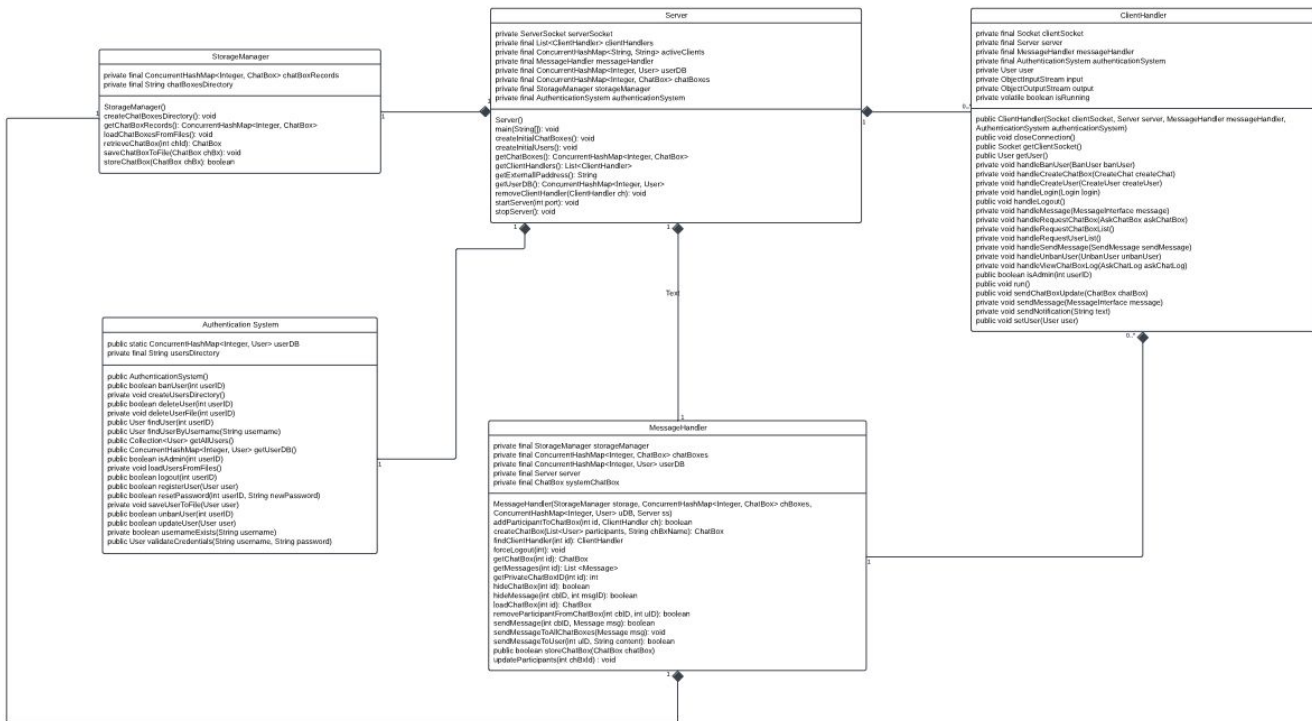
User
<pre>private final int userID private String username private String password private boolean isOnline private boolean isBanned</pre>
<pre>public User(String username, String password) public static void setUserIDGenerator(int value) public boolean equals(Object user) public String getPassword() public int getUserID() public String getUsername() public int hashCode() public boolean isBanned() public boolean isOnline() public void setBanned(boolean banned) public void setOnline(boolean isOnline) public void setPassword(String password) public void setUsername(String username) public String toString()</pre>

Message (Class Diagrams)

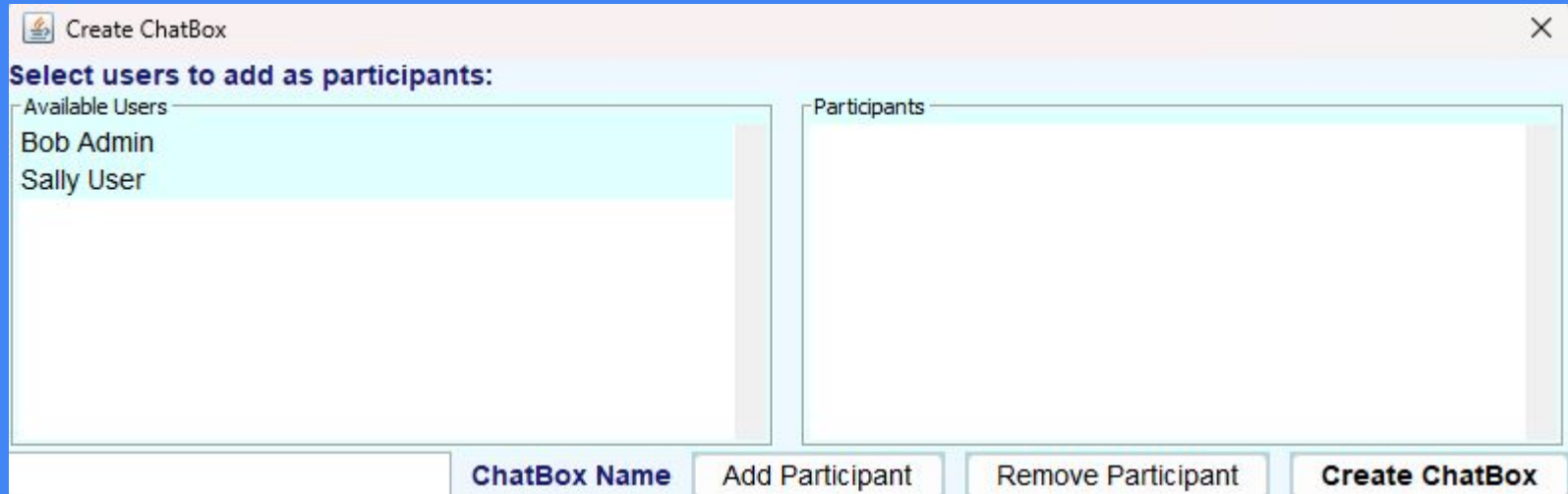
Message Classes



Server (Class Diagrams)



Implementation - GUI



The screenshot shows a window titled "Create ChatBox" with a close button (X) in the top right corner. The window contains the following elements:

- Select users to add as participants:** A label above two list boxes.
- Available Users:** A list box containing "Bob Admin" and "Sally User".
- Participants:** An empty list box.
- ChatBox Name:** A text input field.
- Add Participant:** A button.
- Remove Participant:** A button.
- Create ChatBox:** A button.

Implementation - GUI

```
public class CreateChatBoxDialog extends JDialog {  
  
    private JPanel comboPanel;  
    private final Container pane; // content pane of dialog  
    private JTextField chBoxTxt;  
    private JLabel chBoxName;  
    private JButton createButton;  
    private JButton addParticipant;  
    private JButton removeParticipant;  
    private JList<User> users;  
    private JList<User> participants;  
    private JLabel prompt;  
    private DefaultListModel<User> participantModel;  
    private int[] userListIndex;  
    private int[] participantListIndex;  
    private String chatboxName;  
    private JScrollPane userScrPane;  
    private JScrollPane participantScrPane;
```

```
public CreateChatBoxDialog(JFrame inFrame) {}  
  
// Precondition: pane must be the content pane of the JDialog  
// Postcondition: add all contents to content pane in proper layout  
public void setUpContentPane() {}  
  
// Precondition: None  
// Post: sets up the box layout portion of the content pane  
private JPanel setupBoxLayout() {}  
  
// Precondition: pane must be the content pane of the JDialog  
// Postcondition: finished up on BorderLayout and brings it all together  
private void setupLayout() {}  
  
public void setUpUserList() {}  
  
public void setUpParticipantList() {}  
  
// Handles when user is clicking on the list of users  
public class UserListListener implements ListSelectionListener {}  
  
// handles when user is clicking on the list of participants  
public class ParticipantListListener implements ListSelectionListener {}  
  
// Handles when user is clicking on the AddParticipants button  
public class AddButtonListener implements ActionListener {}  
  
// Handles when user is clicking on the RemoveParticipants button  
public class RemoveButtonListener implements ActionListener {}  
  
// ChatBox Name Listener  
// Upon hitting "enter" when typing in the textfield, the label will update with chatbox name  
public class TextBoxListener implements ActionListener {}  
  
// Create ChatBox button  
public class CreateButtonListener implements ActionListener {}
```

Testing

What we tested:

- Authentication (login, registration, banned users)
- Chat Boxes (creating, retrieving, managing participants)
- Message Handling (sending out messages, retrieving logs)
- Storage management (saving data, retrieving data)
- Admin
- Message
- User

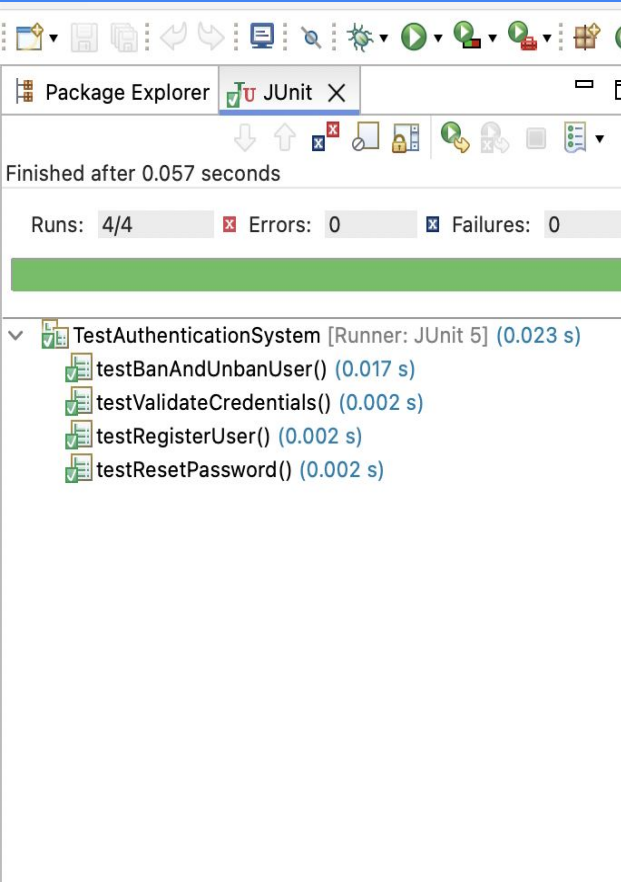
```
AllTests.java  TestStorageManager.java  TestMessageHandler.java X
17
18 class TestMessageHandler {
19     private MessageHandler msgHandler;
20     private StorageManager storageManager;
21     private ChatBox chatbox;
22     private ConcurrentHashMap<Integer, ChatBox> chatBoxes;
23     private ConcurrentHashMap<Integer, User> userDB;
24     private Server server;
25
26 @BeforeEach
27     public void setUpMessageHandler() {
28         storageManager = new StorageManager();
29         chatbox = new ChatBox();
30         chatBoxes = new ConcurrentHashMap<>();
31         userDB = new ConcurrentHashMap<>();
32         server = new Server();
33         msgHandler = new MessageHandler(storageManager, chatBoxes, userDB, server);
34     }
35 @Test
36     void testCreateChatBox() {
37         User user1 = new User("user1", "pass");
38         User user2 = new User("user2", "pass");
39         chatbox.addParticipant(user1);
40         chatbox.addParticipant(user2);
41         List<User> participants = chatbox.getParticipantsList();
42         assertNotNull(msgHandler.createChatBox(participants, "chat1"));
43     }
44 @Test
45     void testGetChatBox() {
46         User user1 = new User("user1", "pass");
47         User user2 = new User("user2", "pass");
48         ChatBox chatbox = new ChatBox();
49         chatbox.addParticipant(user1);
50         chatbox.addParticipant(user2);
51         List<User> participants = chatbox.getParticipantsList();
52         ChatBox createdChatBox = msgHandler.createChatBox(participants, "chat1");
53         int boxId = createdChatBox.getChatBoxID();
54         assertEquals(createdChatBox, msgHandler.getChatBox(boxId));
55     }
56 }
57 @Test
58     void testSendMessage() {
59         User user1 = new User("user1", "pass");
60         User user2 = new User("user2", "pass");
61         ChatBox chatbox = new ChatBox();
62         chatbox.addParticipant(user1);
63         chatbox.addParticipant(user2);
64         List<User> participants = chatbox.getParticipantsList();
65         ChatBox createdChatBox = msgHandler.createChatBox(participants, "chat1");
66         int boxId = createdChatBox.getChatBoxID();
67         Message msg = new Message(1, "This is test msg");
68         assertTrue(msgHandler.sendMessage(boxId, msg));
69     }
}
```

Testing - cont

Test Suite

```
AllTests.java X
1  package Testing;
2
3
4  import org.junit.runner.RunWith;
6  //import org.junit.runners.Suite.SuiteClasses;
7
8  @RunWith(Suite.class)
9
10 @Suite.SuiteClasses({
11     TestAdmin.class,
12     TestChatBox.class,
13     TestUser.class,
14     TestMessage.class,
15     TestServer.class,
16     TestAuthenticationSystem.class,
17     TestMessageHandler.class,
18     TestStorageManager.class
19 })
20 public class AllTests {
21 }
22
```

Test Results

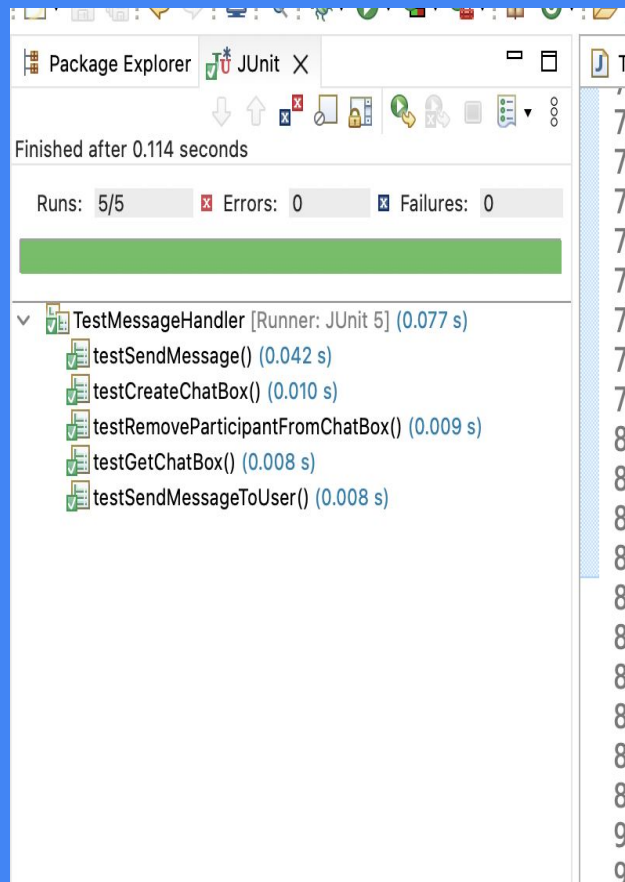


Package Explorer JUnit X

Finished after 0.057 seconds

Runs: 4/4 Errors: 0 Failures: 0

- TestAuthenticationSystem [Runner: JUnit 5] (0.023 s)
 - testBanAndUnbanUser() (0.017 s)
 - testValidateCredentials() (0.002 s)
 - testRegisterUser() (0.002 s)
 - testResetPassword() (0.002 s)

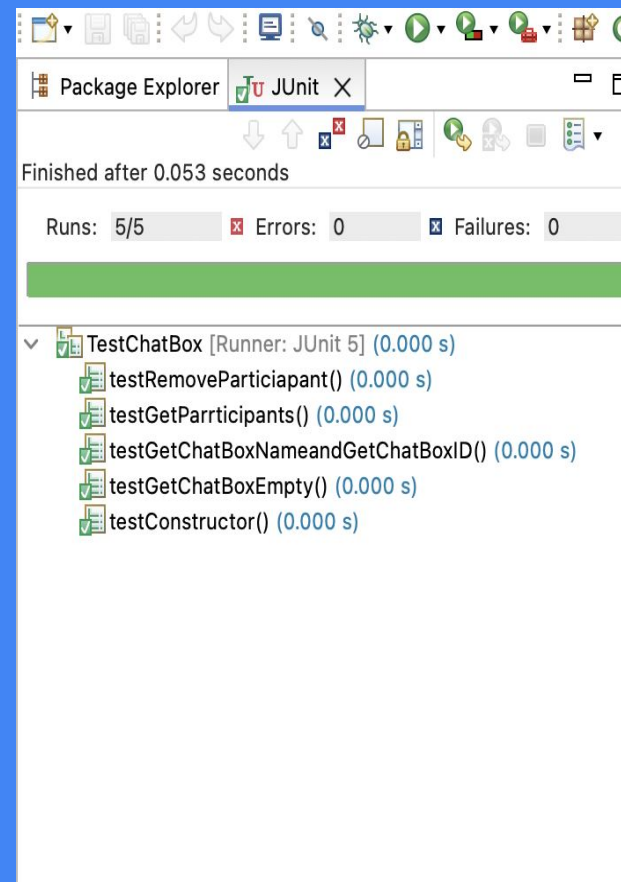


Package Explorer JUnit X

Finished after 0.114 seconds

Runs: 5/5 Errors: 0 Failures: 0

- TestMessageHandler [Runner: JUnit 5] (0.077 s)
 - testSendMessage() (0.042 s)
 - testCreateChatBox() (0.010 s)
 - testRemoveParticipantFromChatBox() (0.009 s)
 - testGetChatBox() (0.008 s)
 - testSendMessageToUser() (0.008 s)



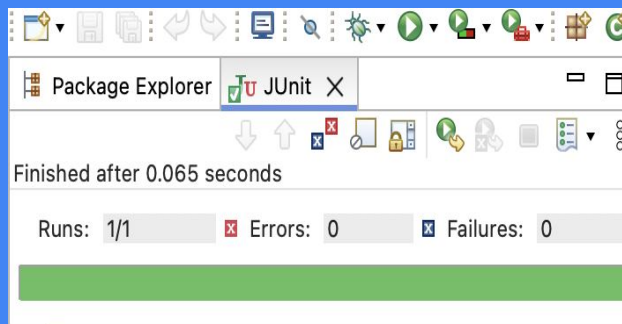
Package Explorer JUnit X

Finished after 0.053 seconds

Runs: 5/5 Errors: 0 Failures: 0

- TestChatBox [Runner: JUnit 5] (0.000 s)
 - testRemoveParticiapant() (0.000 s)
 - testGetParricipants() (0.000 s)
 - testGetChatBoxNameandGetChatBoxID() (0.000 s)
 - testGetChatBoxEmpty() (0.000 s)
 - testConstructor() (0.000 s)

Test Results



Package Explorer JUnit X

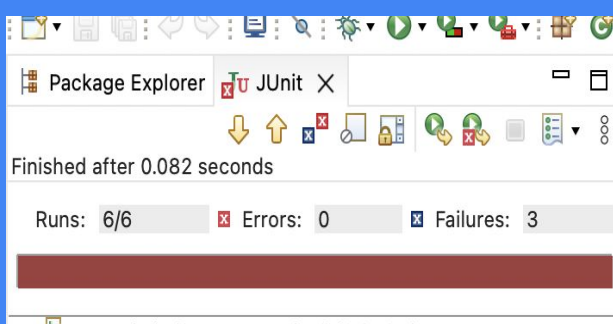
Finished after 0.065 seconds

Runs: 1/1 Errors: 0 Failures: 0

A green progress bar indicates 100% success.

TestStorageManager [Runner: JUnit 5] (0.030 s)

- testStoreAndRetrieveChatBox() (0.030 s)



Package Explorer JUnit X

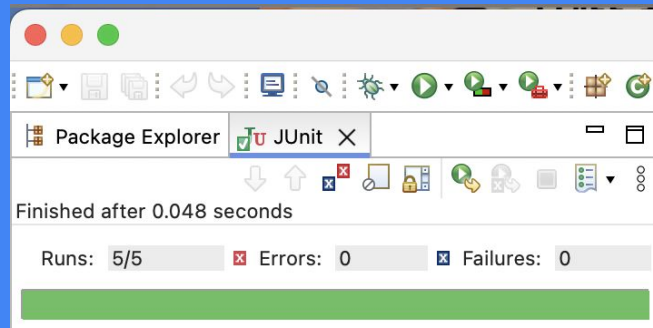
Finished after 0.082 seconds

Runs: 6/6 Errors: 0 Failures: 3

A red progress bar indicates 50% success (3 failures).

TestAdmin [Runner: JUnit 5] (0.049 s)

- testAddUser() (0.026 s)
- testBanUser() (0.005 s)
- testUnBanUser() (0.005 s)
- testToSendSystemMessage() (0.003 s)
- testResetPasswordforUser() (0.003 s)
- testDeleteUser() (0.003 s)



Package Explorer JUnit X

Finished after 0.048 seconds

Runs: 5/5 Errors: 0 Failures: 0

A green progress bar indicates 100% success.

TestMessage [Runner: JUnit 5] (0.017 s)

- testGetSenderId() (0.013 s)
- testMessageConstructorwithObjectMessage() (0.001 s)
- testGetMessageID() (0.000 s)
- testMessageConstructorSenderIdDandtext() (0.000 s)
- testGetConTent() (0.001 s)