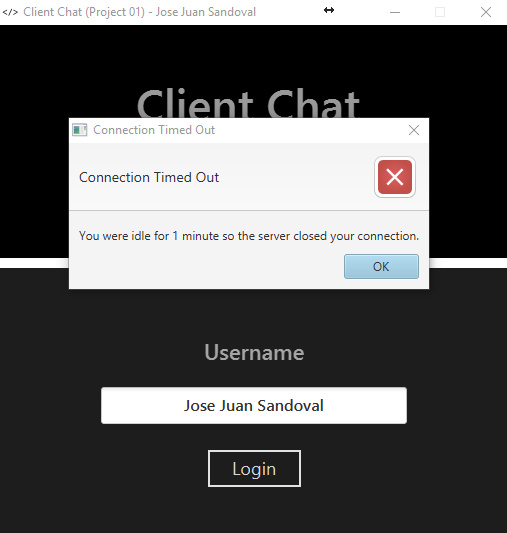
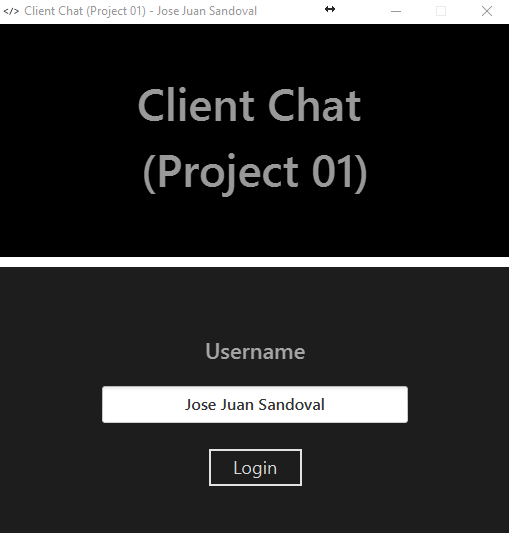
**Project 01**

**Name:** Jose Juan Sandoval

**Link to Project:** <https://github.com/Juanchiselo/CS380/tree/master/Projects/Project%2001>

**Screen Shots**







**Java Code**

ChatClient.java

package Project01;  
  
import javafx.application.Application;  
import javafx.application.Platform;  
import javafx.fxml.FXMLLoader;  
import javafx.scene.Scene;  
import javafx.scene.image.Image;  
import javafx.stage.Stage;  
import java.io.\*;  
import java.net.Socket;  
import java.net.UnknownHostException;  
  
public class ChatClient extends Application  
{  
 private static Socket *socket*;  
 public static Controller *controller*;  
 public static Scene *chatWindowScene*;  
 public static Scene *chatLoginScene*;  
 public static Stage *stage*;  
  
 */\*\*  
 \* The overridden start() method belonging to the  
 \* Application class.  
 \** ***@param*** *primaryStage  
 \** ***@throws*** *Exception  
 \*/* @Override  
 public void start(Stage primaryStage) throws Exception  
 {  
 // Loads the FXML for the Login Scene and creates the Scene.  
 FXMLLoader loader = new FXMLLoader(getClass().getResource("Layouts/ChatLoginLayout.fxml"));  
 *chatLoginScene* = new Scene(loader.load(), 500, 500);  
  
 // Loads the FXML for the Chat Scene and creates the Scene.  
 loader = new FXMLLoader(getClass().getResource("Layouts/ChatWindowLayout.fxml"));  
 *chatWindowScene* = new Scene(loader.load(), 1280, 720);  
  
 // Saves a reference of the Controller object so  
 // the Listener thread can access it.  
 *controller* = loader.getController();  
  
 // Saves a reference of the Stage object so  
 // the Controller class can access it.  
 // It also sets the stage.  
 *stage* = primaryStage;  
 *stage*.getIcons().add(new Image(ChatClient.class.getResourceAsStream("Drawable/Icon.png")));  
 *stage*.setTitle("Client Chat (Project 01) - Jose Juan Sandoval");  
 *stage*.setResizable(false);  
 *stage*.setScene(*chatLoginScene*);  
 *stage*.show();  
 }  
  
 public static void main(String[] args)  
 {  
 *launch*(args);  
 ListenerThread.*endThread* = true;  
 }  
  
 */\*\*  
 \* Sends a message to the server.  
 \** ***@param*** *message - The message to be sent.  
 \*/* public static void sendMessage(String message)  
 {  
 try  
 {  
 OutputStream outputStream = *socket*.getOutputStream();  
 PrintStream out = new PrintStream(outputStream, false, "UTF-8");  
 out.println(message);  
 }  
 catch (Exception e)  
 {  
 System.*err*.println(e.getMessage());  
  
 // Updates the statusLabel GUI object with the  
 // error message.  
 Platform.*runLater*(() ->  
 *controller*.setStatus(e.getMessage()));  
 }  
 }  
  
 */\*\*  
 \* Connects the client to the server and  
 \* creates a Listener thread.  
 \*/* public static void connectToServer()  
 {  
 String hostName = "codebank.xyz";  
 int portNumber = 38001;  
  
 try  
 {  
 *socket* = new Socket(hostName, portNumber);  
 ListenerThread.*endThread* = false;  
  
 // Creates and starts a new thread to listen for messages.  
 new ListenerThread(*socket*).start();  
 }  
 catch (UnknownHostException e)  
 {  
 System.*err*.println("ERROR: Unknown host " + hostName + ".");  
 }  
 catch (Exception e)  
 {  
 System.*err*.println("ERROR: Could not connect to " + hostName + ".");  
 }  
 }  
  
 */\*\*  
 \* Disconnects the client from the server.  
 \*/* public static void disconnectFromServer()  
 {  
 try  
 {  
 *socket*.close();  
 }  
 catch (IOException e)  
 {  
 System.*err*.println(e.getMessage());  
 }  
 }  
}

ListenerThread.java

package Project01;  
  
import javafx.application.Platform;  
import java.io.\*;  
import java.net.Socket;  
  
public class ListenerThread extends Thread  
{  
 public volatile static boolean *endThread* = false;  
 private Socket socket = null;  
  
 public ListenerThread(Socket socket)  
 {  
 super("Chat Listener Thread");  
 this.socket = socket;  
 }  
  
 */\*\*  
 \* The overridden run() function belonging to the Thread class.  
 \* This is what handles the communication between the server and the client.  
 \*/* public void run()  
 {  
 try  
 {  
 // Objects needed for receiving and reading the server's messages.  
 String receivedMessage;  
 InputStream inputStream = socket.getInputStream();  
 InputStreamReader inputStreamReader = new InputStreamReader(inputStream, "UTF-8");  
 BufferedReader in = new BufferedReader(inputStreamReader);  
  
 // The main loop of execution.  
 // This executes when the servers sends a message  
 // and the thread has not received a flag to terminate.  
 while((receivedMessage = in.readLine()) != null  
 && !*endThread*)  
 {  
 // NOTE: The variable sent to the  
 // Application GUI thread has to be final.  
 final String message = receivedMessage;  
  
 // Displays the received messages.  
 Platform.*runLater*(() ->  
 ChatClient.*controller*.setMessages(message));  
  
 // Catches the Unavailable Username error thrown by the server.  
 if(message.equals("Name in use."))  
 {  
 Platform.*runLater*(() ->  
 ChatClient.*controller*.displayError("Unavailable Username",  
 "The username you entered is already being used."));  
 *endThread* = true;  
 }  
  
 // Catches the Inactivity error thrown by the server.  
 if(message.equals("Connection idle for 1 minute, closing connection."))  
 {  
 Thread.*sleep*(5000);  
 Platform.*runLater*(() ->  
 ChatClient.*controller*.displayError("Connection Timed Out",  
 "You were idle for 1 minute "  
 + "so the server closed your connection."));  
 *endThread* = true;  
 }  
 }  
 ChatClient.*disconnectFromServer*();  
 }  
 catch (IOException e)  
 {  
 System.*err*.println("ERROR: Connection lost with server.");  
 }  
 catch (Exception e)  
 {  
 System.*err*.println(e.getMessage());  
 }  
 }  
}

Controller.java

package Project01;  
  
import javafx.collections.FXCollections;  
import javafx.collections.ObservableList;  
import javafx.fxml.FXML;  
import javafx.scene.control.\*;  
  
public class Controller  
{  
 @FXML private TextField messageTextField;  
 @FXML private ListView<String> chatWindowListView;  
 @FXML private Label statusLabel;  
 @FXML private TextField usernameTextField;  
 private ObservableList<String> messages = FXCollections.*observableArrayList*();  
  
 */\*\*  
 \* Switches between the Login and Chat windows.  
 \*/* public void switchWindows()  
 {  
 if(ChatClient.*stage*.getScene().equals(ChatClient.*chatLoginScene*))  
 {  
 ChatClient.*stage*.setScene(ChatClient.*chatWindowScene*);  
 ChatClient.*stage*.setResizable(true);  
 }  
 else  
 {  
 ChatClient.*stage*.setScene(ChatClient.*chatLoginScene*);  
 ChatClient.*stage*.setResizable(false);  
 }  
 ChatClient.*stage*.centerOnScreen();  
 }  
  
 */\*\*  
 \* Logs in the user by connecting him/her to the server  
 \* and sends the username to the server.  
 \*/* public void login()  
 {  
 ChatClient.*connectToServer*();  
 ChatClient.*sendMessage*(usernameTextField.getText());  
 switchWindows();  
 }  
  
 */\*\*  
 \* Logs out the user by disconnecting from the server  
 \* and returns him/her to the login screen.  
 \* It also notifies the Listener thread to terminate.  
 \*/* public void logout()  
 {  
 switchWindows();  
 messages.clear();  
 ListenerThread.*endThread* = true;  
 ChatClient.*disconnectFromServer*();  
 }  
  
 */\*\*  
 \* Displays error messages in the form of alerts.  
 \** ***@param*** *title - The title of the alert.  
 \** ***@param*** *errorMessage - The error message.  
 \*/* public void displayError(String title, String errorMessage)  
 {  
 logout();  
 Alert alert = new Alert(Alert.AlertType.*ERROR*);  
 alert.setTitle(title);  
 alert.setHeaderText(title);  
 alert.setContentText(errorMessage);  
 alert.showAndWait();  
 }  
  
 */\*\*  
 \* Sends a message to the server when the user  
 \* either clicks on the Send button or presses  
 \* the Enter key while the Message TextField is focused.  
 \*/* public void sendMessage()  
 {  
 String message = messageTextField.getText();  
  
 // Trimming the message and checking if the  
 // String is not empty after trimming it  
 // prevents blank messages from being sent.  
 if(!message.trim().isEmpty())  
 {  
 ChatClient.*sendMessage*(message);  
 messageTextField.clear();  
 }  
 }  
  
 */\*\*  
 \* Displays the messages received from the server.  
 \** ***@param*** *message - The last received message.  
 \*/* public void setMessages(String message)  
 {  
 messages.add(message);  
 chatWindowListView.setItems(messages);  
 }  
  
 */\*\*  
 \* Displays the status messages located in the status bar.  
 \** ***@param*** *status - The status message.  
 \*/* public void setStatus(String status)  
 {  
 statusLabel.setText("Status: " + status + ".");  
 }  
}

ChatLoginLayout.fxml

<?xml version="1.0" encoding="UTF-8"?>  
  
<?import javafx.scene.control.Button?>  
<?import javafx.scene.control.Label?>  
<?import javafx.scene.control.TextField?>  
<?import javafx.scene.layout.BorderPane?>  
<?import javafx.scene.layout.ColumnConstraints?>  
<?import javafx.scene.layout.GridPane?>  
<?import javafx.scene.layout.Pane?>  
<?import javafx.scene.layout.RowConstraints?>  
  
<BorderPane fx:id="borderPaneMain" maxHeight="-Infinity" maxWidth="-Infinity" minHeight="-Infinity" minWidth="-Infinity" prefHeight="500.0" prefWidth="500.0" styleClass="background" stylesheets="@../Styles/DarkTheme.css" xmlns="http://javafx.com/javafx/8.0.111" xmlns:fx="http://javafx.com/fxml/1" fx:controller="Project01.Controller">  
  
  
 <center>  
 <GridPane prefHeight="579.0" prefWidth="500.0">  
 <columnConstraints>  
 <ColumnConstraints hgrow="SOMETIMES" maxWidth="1233.0" minWidth="10.0" percentWidth="100.0" prefWidth="1100.0" />  
 </columnConstraints>  
 <rowConstraints>  
 <RowConstraints maxHeight="320.6666564941406" minHeight="10.0" percentHeight="48.0" prefHeight="89.33333587646484" vgrow="SOMETIMES" />  
 <RowConstraints maxHeight="320.6666564941406" minHeight="10.0" percentHeight="2.0" prefHeight="89.33333587646484" vgrow="SOMETIMES" />  
 <RowConstraints maxHeight="320.6666564941406" minHeight="10.0" percentHeight="10.0" prefHeight="89.33333587646484" vgrow="SOMETIMES" />  
 <RowConstraints maxHeight="320.6666564941406" minHeight="10.0" percentHeight="10.0" prefHeight="89.33333587646484" vgrow="SOMETIMES" />  
 <RowConstraints maxHeight="320.6666564941406" minHeight="0.0" percentHeight="15.0" prefHeight="89.33333587646484" vgrow="SOMETIMES" />  
 <RowConstraints maxHeight="320.6666564941406" minHeight="10.0" percentHeight="10.0" prefHeight="89.33333587646484" vgrow="SOMETIMES" />  
 <RowConstraints maxHeight="320.6666564941406" minHeight="10.0" percentHeight="10.0" prefHeight="89.33333587646484" vgrow="SOMETIMES" />  
 </rowConstraints>  
 <children>  
 <GridPane alignment="CENTER" GridPane.rowIndex="4">  
 <columnConstraints>  
 <ColumnConstraints halignment="CENTER" hgrow="ALWAYS" maxWidth="797.6666870117188" minWidth="10.0" percentWidth="60.0" prefWidth="730.0" />  
 </columnConstraints>  
 <rowConstraints>  
 <RowConstraints minHeight="10.0" prefHeight="30.0" vgrow="SOMETIMES" />  
 </rowConstraints>  
 <children>  
 <TextField fx:id="usernameTextField" alignment="CENTER" onAction="#login" prefHeight="37.0" prefWidth="278.0" GridPane.halignment="CENTER" GridPane.valignment="CENTER" />  
 </children>  
 </GridPane>  
 <Button fx:id="loginButton" mnemonicParsing="false" onAction="#login" style="-fx-font-size: 18;" text="Login" GridPane.halignment="CENTER" GridPane.rowIndex="5" GridPane.valignment="CENTER" />  
 <Label style="-fx-font-size: 22;" text="Username" GridPane.halignment="CENTER" GridPane.rowIndex="3" GridPane.valignment="BOTTOM" />  
 <GridPane style="-fx-background-color: black;">  
 <columnConstraints>  
 <ColumnConstraints hgrow="SOMETIMES" minWidth="10.0" prefWidth="100.0" />  
 </columnConstraints>  
 <rowConstraints>  
 <RowConstraints minHeight="10.0" prefHeight="30.0" vgrow="SOMETIMES" />  
 </rowConstraints>  
 <children>  
 <Label id="titleLabel" alignment="CENTER" prefHeight="151.0" prefWidth="309.0" style="-fx-font-size: 45;" stylesheets="@../Styles/DarkTheme.css" text="Client Chat (Project 01)" textAlignment="CENTER" wrapText="true" GridPane.halignment="CENTER" GridPane.valignment="CENTER" />  
 </children>  
 </GridPane>  
 <Pane prefHeight="200.0" prefWidth="200.0" style="-fx-background-color: white;" GridPane.rowIndex="1" />  
 </children>  
 </GridPane>  
 </center>  
</BorderPane>

ChatWindowLayout.fxml

<?xml version="1.0" encoding="UTF-8"?>  
  
<?import javafx.geometry.Insets?>  
<?import javafx.scene.control.Button?>  
<?import javafx.scene.control.Label?>  
<?import javafx.scene.control.ListView?>  
<?import javafx.scene.control.TextField?>  
<?import javafx.scene.layout.BorderPane?>  
<?import javafx.scene.layout.ColumnConstraints?>  
<?import javafx.scene.layout.GridPane?>  
<?import javafx.scene.layout.HBox?>  
<?import javafx.scene.layout.Pane?>  
<?import javafx.scene.layout.RowConstraints?>  
  
<BorderPane fx:id="borderPaneMain" maxHeight="-Infinity" maxWidth="-Infinity" prefHeight="720.0" prefWidth="1280.0" styleClass="background" stylesheets="@../Styles/DarkTheme.css" xmlns="http://javafx.com/javafx/8.0.111" xmlns:fx="http://javafx.com/fxml/1" fx:controller="Project01.Controller">  
  
  
 <bottom>  
 <HBox alignment="CENTER\_LEFT" style="-fx-background-color: black;" BorderPane.alignment="CENTER">  
 <Label fx:id="statusLabel" text="Status:">  
 <padding>  
 <Insets bottom="5.0" left="5.0" right="5.0" top="5.0" />  
 </padding>  
 </Label>  
 </HBox>  
 </bottom>  
  
  
 <center>  
 <GridPane>  
 <columnConstraints>  
 <ColumnConstraints hgrow="SOMETIMES" maxWidth="1233.0" minWidth="10.0" percentWidth="100.0" prefWidth="1100.0" />  
 </columnConstraints>  
 <rowConstraints>  
 <RowConstraints maxHeight="320.6666564941406" minHeight="10.0" percentHeight="80.0" prefHeight="89.33333587646484" vgrow="SOMETIMES" />  
 <RowConstraints maxHeight="320.6666564941406" minHeight="10.0" percentHeight="2.0" prefHeight="89.33333587646484" vgrow="SOMETIMES" />  
 <RowConstraints maxHeight="320.6666564941406" minHeight="0.0" percentHeight="20.0" prefHeight="89.33333587646484" vgrow="SOMETIMES" />  
 </rowConstraints>  
 <GridPane alignment="CENTER" GridPane.rowIndex="2">  
 <columnConstraints>  
 <ColumnConstraints halignment="CENTER" hgrow="ALWAYS" maxWidth="797.6666870117188" minWidth="10.0" percentWidth="5.0" prefWidth="730.0" />  
 <ColumnConstraints halignment="CENTER" hgrow="ALWAYS" maxWidth="797.6666870117188" minWidth="10.0" percentWidth="90.0" prefWidth="730.0" />  
 <ColumnConstraints halignment="CENTER" hgrow="ALWAYS" maxWidth="901.3333129882812" minWidth="10.0" percentWidth="15.0" prefWidth="192.0" />  
 </columnConstraints>  
 <rowConstraints>  
 <RowConstraints minHeight="10.0" prefHeight="30.0" vgrow="SOMETIMES" />  
 </rowConstraints>  
 <TextField fx:id="messageTextField" onAction="#sendMessage" promptText="Type a message here..." GridPane.columnIndex="1" GridPane.halignment="CENTER" GridPane.valignment="CENTER" />  
 <Button fx:id="sendButton" mnemonicParsing="false" onAction="#sendMessage" text="Send" GridPane.columnIndex="2" GridPane.halignment="CENTER" GridPane.valignment="CENTER" />  
 </GridPane>  
 <ListView fx:id="chatWindowListView" editable="true" prefHeight="200.0" prefWidth="200.0" style="-fx-background-color: #3f3f3f;" />  
 <Pane prefHeight="200.0" prefWidth="200.0" style="-fx-background-color: black;" GridPane.rowIndex="1" />  
 </GridPane>  
 </center>  
 <top>  
 <Label onMouseClicked="#logout" style="-fx-font-size: 20;" text="Logout" textAlignment="CENTER" BorderPane.alignment="CENTER\_RIGHT">  
 <padding>  
 <Insets right="20.0" />  
 </padding>  
 </Label>  
 </top>  
</BorderPane>

DarkTheme.css

.background {  
 -fx-background-color: #1d1d1d;  
}  
  
.label {  
 -fx-font-size: 11pt;  
 -fx-font-family: "Segoe UI Semibold";  
 -fx-text-fill: white;  
 -fx-opacity: 0.6;  
}  
  
.label-bright {  
 -fx-font-size: 11pt;  
 -fx-font-family: "Segoe UI Semibold";  
 -fx-text-fill: white;  
 -fx-opacity: 1;  
}  
  
.label-header {  
 -fx-font-size: 32pt;  
 -fx-font-family: "Segoe UI Light";  
 -fx-text-fill: white;  
 -fx-opacity: 1;  
}  
  
.table-view {  
 -fx-base: #1d1d1d;  
 -fx-control-inner-background: #1d1d1d;  
 -fx-background-color: #1d1d1d;  
 -fx-table-cell-border-color: transparent;  
 -fx-table-header-border-color: transparent;  
 -fx-padding: 5;  
}  
  
.table-view .column-header-background {  
 -fx-background-color: transparent;  
}  
  
.table-view .column-header, .table-view .filler {  
 -fx-size: 35;  
 -fx-border-width: 0 0 1 0;  
 -fx-border-color:  
 transparent  
 transparent  
 derive(-fx-base, 80%)  
 transparent;  
 -fx-border-insets: 0 10 1 0;  
}  
  
.table-view .column-header .label {  
 -fx-font-size: 20pt;  
 -fx-font-family: "Segoe UI Light";  
 -fx-text-fill: white;  
 -fx-alignment: center-left;  
 -fx-opacity: 1;  
}  
  
.table-view:focused .table-row-cell:filled:focused:selected {  
 -fx-background-color: -fx-focus-color;  
}  
  
.split-pane:horizontal > \* > .split-pane-divider {  
 -fx-border-color: transparent #1d1d1d transparent #1d1d1d;  
 -fx-background-color: transparent, derive(#1d1d1d,20%);  
}  
  
.split-pane {  
 -fx-padding: 1 0 0 0;  
}  
  
.menu {  
 -fx-text-fill: white;  
}  
  
.menu-bar {  
 -fx-background-color: #1d1d1d;  
 -fx-selection-bar: #1d1d1d;  
}  
  
.menu-bar .label {  
 -fx-font-size: 12pt;  
 -fx-font-family: "Segoe UI Light";  
 -fx-text-fill: white;  
 -fx-opacity: 0.9;  
}  
  
.text-field {  
 -fx-font-size: 12pt;  
 -fx-font-family: "Segoe UI Semibold";  
}  
  
  
  
/\*  
 \* Metro style Push Button  
 \* Author: Pedro Duque Vieira  
 \* http://pixelduke.wordpress.com/2012/10/23/jmetro-windows-8-controls-on-java/  
 \*/  
.button {  
 -fx-padding: 5 22 5 22;  
 -fx-border-color: #e2e2e2;  
 -fx-border-width: 2;  
 -fx-background-radius: 0;  
 -fx-background-color: #1d1d1d;  
 -fx-font-family: "Segoe UI", Helvetica, Arial, sans-serif;  
 -fx-font-size: 11pt;  
 -fx-text-fill: #d8d8d8;  
 -fx-background-insets: 0 0 0 0, 0, 1, 2;  
}  
  
.button:hover {  
 -fx-background-color: #3a3a3a;  
}  
  
.button:pressed, .button:default:hover:pressed {  
 -fx-background-color: white;  
 -fx-text-fill: #1d1d1d;  
}  
  
.button:focused {  
 -fx-border-color: white, white;  
 -fx-border-width: 1, 1;  
 -fx-border-style: solid, segments(1, 1);  
 -fx-border-radius: 0, 0;  
 -fx-border-insets: 1 1 1 1, 0;  
}  
  
.button:disabled, .button:default:disabled {  
 -fx-opacity: 0.4;  
 -fx-background-color: #1d1d1d;  
 -fx-text-fill: white;  
}  
  
.button:default {  
 -fx-background-color: -fx-focus-color;  
 -fx-text-fill: #ffffff;  
}  
  
.button:default:hover {  
 -fx-background-color: derive(-fx-focus-color,30%);  
}  
  
.scroll-pane > .viewport {  
 -fx-background-color: transparent;  
}