OOPM
MiniProject Report
Sem III

Topic: Live Currency Converter.

Group Members:
Gaurav Amarnani, D7A, 67.
Kritika Yadav, D7A, 73.
Eshwar Vazirani, D7B, 73.

Table of content

Topic: Live Currency Converter.

Problem Statement:

We need currency conversion in our day to day lives, majorly for someone in a finance background, sometimes while shopping, sometimes while randomly wanting to know the net worth or captial of a company.

Currency Conversion can be done in a static way using stored values like for example 1 dollar = 70 rupees. But the fact is that these values are ever changing.

Using these static values might not make such a huge difference when the amount in not very high, but when the amount is high it can make huge differences.

Live currency Converter will make the lives of the users easier. We also provide recording conversions of each user so that they can quickly perform the same conversions again and again without having to do much.

Objective:

The main objective of The Live Currency Converter Application is to provide user with ease of converting between currencies without having to go through a lot and get quick accurate results. We will also store conversions done by user so that user can have the option to perform the same conversions quickly.

Proposed System:

In our proposed system we have added the provisions for live currency conversion of the worlds top currencies in a quick and easy manner. Another advantage of our system is that we provide history feature for the logged in users, so that they can perform the similar conversions quickly and easily.

Our proposed system has several advantages

i. User friendly interface.
ii. Fast access to API.
iii. Less error.
iv. Look and Feel Environment.

System Requirements:

Software:
- i. IntelliJ IDEA.
- ii. Java JDK

13. Hardware:
- i. i5 Processor.
- ii. 8gb RAM.

Code:

LoginFrame.java:

```java
/*
 * Copyright 2021 Gaurav Amarnani, Eshwar Vazirani, Kritika Yadav.
 *
 * Licensed under the Apache License, Version 2.0 (the "License");
 * you may not use this file except in compliance with the License.
 * You may obtain a copy of the License at
 *
 *      http://www.apache.org/licenses/LICENSE-2.0
 *
 * Unless required by applicable law or agreed to in writing, software
 * distributed under the License is distributed on an "AS IS" BASIS,
 * WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
 * See the License for the specific language governing permissions and
 * limitations under the License.
 */

package com.byGroup3.ui;

/**
 * This is the GUI Representation Frame for Login Page.
 *
 * @author Gaurav Amarnani, Eshwar Vazirani, Kritika Yadav.
 */

public class LoginFrame extends JFrame {

    private String userName;
    private String password;
    private UserService userService = new UserService();
    private Container container;
    private JLabel headLabel;
    private JLabel userLabel;
    private JLabel passwordLabel;
    private JTextField userTextField;
    private JPasswordField passwordField;
    private JCheckBox showPasswordCheckBox;
    private JButton loginButton;
    private JButton clearButton;
    private JButton signUpButton;
    private JButton skipLoginButton;
    private final Font MY_FONT = new Font("Times New Roman" , Font.BOLD, 25);

    public static void main(String...args) {
        new LoginFrame();
    }

    LoginFrame() {
        setTitle("Login Form!");
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        setSize(600, 600);
        setVisible(true);
        setResizable(false);
        setLocationRelativeTo(null);
        init();
    }

    public void init() {

        //Container:
        container = getContentPane();
        container.setLayout(null);
        container.setBackground(Color.BLACK);

        //Head Label:
        headLabel = new JLabel("LOGIN HERE!");
```

```
headLabel.setForeground(Color.YELLOW);
headLabel.setFont(new Font("Times New Roman", Font.BOLD, 40));
headLabel.setBounds(150, 50, 275, 50);

//User Label:
userLabel = new JLabel("USERNAME :");
userLabel.setForeground(Color.YELLOW);
userLabel.setFont(MY_FONT);
userLabel.setBounds(100, 150, 200, 30);

//Password Label:
passwordLabel = new JLabel("PASSWORD :");
passwordLabel.setForeground(Color.YELLOW);
passwordLabel.setFont(MY_FONT);
passwordLabel.setBounds(100, 220, 200, 30);

//User TextField:
userTextField = new JTextField(30);
userTextField.setFont(MY_FONT);
userTextField.setBounds(300, 150, 200, 40);

//Password Field:
passwordField = new JPasswordField(30);
passwordField.setFont(MY_FONT);
passwordField.setBounds(300, 220, 200, 40);

//Show Password CheckBox:
showPasswordCheckBox = new JCheckBox("Show Password");
showPasswordCheckBox.setFont(new Font("Times New Roman", Font.BOLD, 15));
showPasswordCheckBox.setBackground(Color.BLACK);
showPasswordCheckBox.setForeground(Color.WHITE);
showPasswordCheckBox.setBounds(300, 280, 150, 30);
showPasswordCheckBox.addActionListener(e -> {
    if (showPasswordCheckBox.isSelected())
        passwordField.setEchoChar((char) 0);
    else
        passwordField.setEchoChar('*');
});

//Login Button:
loginButton = new JButton("LOGIN");
loginButton.setFont(MY_FONT);
loginButton.setBounds(60, 350, 200, 40);
loginButton.addActionListener(e -> {
    userName = userTextField.getText();
    password = passwordField.getText();
    String message = userService.validateUsernameAndPassword(userName, password);
    if(message != null)
        JOptionPane.showMessageDialog(rootPane, message);
    else {
        message = userService.checkUsernameAndPassword(userName, password);
        if(message != null)
            JOptionPane.showMessageDialog(rootPane, message);
        else {
            CurrencyConverterFrame currencyConverterFrame = new CurrencyConverterFrame(userName);
            if(currencyConverterFrame.isVisible())
                setVisible(false);
        }
    }
});

//Clear Button:
clearButton = new JButton("CLEAR");
clearButton.setFont(MY_FONT);
clearButton.setBounds(320, 350, 200, 40);
clearButton.addActionListener(e -> {
    userTextField.setText("");
    passwordField.setText("");
});
```

```
        //SignUp Button:
        signUpButton = new JButton("SIGN UP");
        signUpButton.setFont(MY_FONT);
        signUpButton.setBounds(60, 450, 200, 40);
        signUpButton.addActionListener(e -> {
            SignUpFrame signUpFrame = new SignUpFrame();
            if(signUpFrame.isVisible())
                setVisible(false);
        });

        //Skip Login Button:
        skipLoginButton = new JButton("SKIP LOGIN");
        skipLoginButton.setFont(MY_FONT);
        skipLoginButton.setBounds(320, 450, 200, 40);
        skipLoginButton.addActionListener(e -> {
            CurrencyConverterFrame currencyConverterFrame = new CurrencyConverterFrame(null);
            if(currencyConverterFrame.isVisible())
                setVisible(false);
        });

        //Adding All Components :
        container.add(headLabel);
        container.add(userLabel);
        container.add(passwordLabel);
        container.add(userTextField);
        container.add(passwordField);
        container.add(showPasswordCheckBox);
        container.add(loginButton);
        container.add(clearButton);
        container.add(signUpButton);
        container.add(skipLoginButton);
    }
}
```

SignUpFrame.java:

```
package com.byGroup3.ui;

/**
 * This is the GUI Representation Frame for Sign Up Page.
 *
 * @author Gaurav Amarnani, Eshwar Vazirani, Kritika Yadav.
 */

public class SignUpFrame extends JFrame {

    private UserService userService = new UserService();
    private Container container;
    private JLabel headLabel;
    private JLabel nameLabel;
    private JLabel userNameLabel;
    private JLabel emailLabel;
    private JLabel passwordLabel;
    private JTextField nameTextField;
    private JTextField userNameTextField;
    private JTextField emailTextField;
    private JPasswordField passwordField;
    private JCheckBox showPasswordCheckBox;
    private JButton signUpButton;
    private JButton loginButton;
    private Font MY_FONT = new Font("Times New Roman", Font.BOLD, 25);

    public static void main(String...args) {
        new SignUpFrame();
    }

    public SignUpFrame() {
        setTitle("Sign Up Form!");
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        setSize(550, 650);
        setVisible(true);
        setResizable(false);
        setLocationRelativeTo(null);
        init();
    }

    public void init() {

        //Container:
        container = getContentPane();
        container.setLayout(null);
        container.setBackground(Color.BLACK);

        //Head Label:
        headLabel = new JLabel("SIGN UP HERE!");
        headLabel.setForeground(Color.YELLOW);
        headLabel.setFont(new Font("Times New Roman", Font.BOLD, 40));
        headLabel.setBounds(115, 50, 400, 50);

        //Name Label:
        nameLabel = new JLabel("     NAME : ");
        nameLabel.setForeground(Color.YELLOW);
        nameLabel.setFont(MY_FONT);
        nameLabel.setBounds(70, 150, 200, 30);

        //User Name Label:
        userNameLabel = new JLabel("USERNAME : ");
        userNameLabel.setForeground(Color.YELLOW);
        userNameLabel.setFont(MY_FONT);
        userNameLabel.setBounds(70, 230, 200, 30);

        //Email Label:
        emailLabel = new JLabel("EMAIL ID : ");
        emailLabel.setForeground(Color.YELLOW);
        emailLabel.setFont(MY_FONT);
        emailLabel.setBounds(70, 310, 200, 30);

        //Password Label:
        passwordLabel = new JLabel("PASSWORD : ");
        passwordLabel.setForeground(Color.YELLOW);
        passwordLabel.setFont(MY_FONT);
        passwordLabel.setBounds(70, 390, 200, 30);

        //Name TextField:
        nameTextField = new JTextField(30);
        nameTextField.setFont(MY_FONT);
        nameTextField.setBounds(270, 150, 200, 40);

        //User Name TextField:
        userNameTextField = new JTextField(30);
        userNameTextField.setFont(MY_FONT);
        userNameTextField.setBounds(270, 230, 200, 40);

        //Email TextField:
        emailTextField = new JTextField(30);
        emailTextField.setFont(MY_FONT);
        emailTextField.setBounds(270, 310, 200, 40);

        //Password Field:
        passwordField = new JPasswordField(30);
        passwordField.setFont(MY_FONT);
        passwordField.setBounds(270, 390, 200, 40);

        //Show Password CheckBox:
        showPasswordCheckBox = new JCheckBox("Show Password");
        showPasswordCheckBox.setFont(new Font("Times New Roman", Font.BOLD, 15));
        showPasswordCheckBox.setBackground(Color.BLACK);
        showPasswordCheckBox.setForeground(Color.WHITE);
        showPasswordCheckBox.setBounds(270, 440, 150, 30);
        showPasswordCheckBox.addActionListener(e -> {
            if (showPasswordCheckBox.isSelected())
                passwordField.setEchoChar((char) 0);
            else
                passwordField.setEchoChar('*');
        });

        //Sign Up Button:
        signUpButton = new JButton("SIGN UP");
        signUpButton.setFont(MY_FONT);
        signUpButton.setBounds(50, 520, 200, 40);
        signUpButton.addActionListener(new ActionListener(){
            public void actionPerformed(ActionEvent e){
                String message = userService.validateUser(nameTextField.getText(), userNameTextField.getText(),
emailTextField.getText(), passwordField.getText());
                if(message != null)
                    JOptionPane.showMessageDialog(rootPane, message);
                else {
                    UserDTO userDTO = new UserDTO(nameTextField.getText(), userNameTextField.getText(),
emailTextField.getText(), passwordField.getText());
                    if(userService.checkIfUserExists(userNameTextField.getText()))
                        JOptionPane.showMessageDialog(rootPane, userNameTextField.getText() + " already exists.");
                    else {
                        if (userService.saveUser(userDTO)) {
                            CurrencyConverterFrame currencyConverterFrame = new
CurrencyConverterFrame(userNameTextField.getText());
                            if(currencyConverterFrame.isVisible())
                                setVisible(false);
                        }
                        else
                            JOptionPane.showMessageDialog(rootPane, "Could not store into database.");
                    }
                }
            }
        });

        //Login Button:
        loginButton = new JButton("LOGIN!");
        loginButton.setFont(MY_FONT);
        loginButton.setBounds(280, 520, 200, 40);
        loginButton.addActionListener(e -> {
            LoginFrame loginFrame = new LoginFrame();
            if(loginFrame.isVisible())
                setVisible(false);
        });

        //Adding All Components :
        container.add(headLabel);
        container.add(nameLabel);
        container.add(userNameLabel);
        container.add(emailLabel);
        container.add(passwordLabel);
        container.add(nameTextField);
        container.add(userNameTextField);
        container.add(emailTextField);
        container.add(passwordField);
        container.add(showPasswordCheckBox);
        container.add(signUpButton);
        container.add(loginButton);
    }
}
```

CurrencyConverterFrame.java:

```
package com.byGroup3.ui;

/**
 * This is the GUI Representation Frame for Live Currency Converter Application.
 *
 * @author Gaurav Amarnani, Eshwar Vazirani, Kritika Yadav.
 */

public class CurrencyConverterFrame extends JFrame {

    private ArrayList<String> currencyList;
    private String currencyFrom;
    private String currencyTo;
    private Double amount;
    private Double result;
    private CurrencyService currencyService = new CurrencyService();
    private Container container;
    private Image iconImage;
    private JLabel headLabel;
    private JLabel amount1Label;
    private JLabel amount2Label;
    private JTextField inputAmountTextField;
    private JTextField generatedAmountTextField;
    private JComboBox<String> currencyFromComboBox;
    private JComboBox<String> currencyToComboBox;
    private JLabel feedbackLabel;
    private JButton convertCurrencyButton;
    private JButton addConversionButton;
    private final Font MY_FONT = new Font("Times New Roman" , Font.BOLD, 25);

    public static void main(String...args) {
        new CurrencyConverterFrame("Gaurav");
    }

    CurrencyConverterFrame(String userName) {
        setTitle("Live Currency Converter");
        setDefaultCloseOperation(JFrame.DISPOSE_ON_CLOSE);
        if(userName == null)
            setSize(800, 450);
        else
            setSize(800, 500);
        setVisible(true);
        setResizable(false);
        setLocationRelativeTo(null);
        init(userName);
        if(userName != null)
            createUserSpecificUI(userName);
    }

    public void init(String userName) {

        //Setting Icon for JFrame:
        iconImage = new javax.swing.ImageIcon("C:\\Users\\DELL\\Pictures\\College\\OOPM
MP\\logo.png").getImage();
        setIconImage(iconImage);

        //Container:
        container = getContentPane();
        container.setLayout(null);
        container.setBackground(Color.BLACK);

        //Heading Label:
        headLabel = new JLabel("Real Time Currency Converter.");
        headLabel.setForeground(Color.YELLOW);
        headLabel.setFont(new Font("Times New Roman" , Font.BOLD , 40));
        headLabel.setBounds(100,20,680,50);

        //Amount 1 Label:
        amount1Label = new JLabel("Enter Amount:");
        amount1Label.setForeground(Color.YELLOW);
        amount1Label.setFont(MY_FONT);
        amount1Label.setBounds(50,110,200,50);

        //Amount 2 Label:
        amount2Label = new JLabel("Total Amount:");
```

```
amount2Label.setForeground(Color.YELLOW);
amount2Label.setFont(MY_FONT);
amount2Label.setBounds(50,180,200,50);

//Input Amount TextField:
inputAmountTextField = new JTextField(20);
inputAmountTextField.setFont(MY_FONT);
inputAmountTextField.setBounds(250,110,250,50);
inputAmountTextField.addKeyListener(new KeyListener() {

    public void keyPressed(KeyEvent e) {

        //User Should only add number :
        char c = e.getKeyChar();
        if(Character.isLetter(c))
            inputAmountTextField.setEditable(false);
        else
            inputAmountTextField.setEditable(true);

        //User Should only add less than 20 digits :
        if(inputAmountTextField.getText().length() > 19)
            inputAmountTextField.setText("");
    }

    //Empty Implementation :
    public void keyTyped(KeyEvent e) { }
    public void keyReleased(KeyEvent e) { }
});

//Generated Amount TextField:
generatedAmountTextField = new JTextField(30);
generatedAmountTextField.setFont(MY_FONT);
generatedAmountTextField.setBounds(250,180,250,50);
generatedAmountTextField.setEditable(false);

//Populating the ArrayList with the Currency List:
currencyList = currencyService.populateCurrencies();

//Currency From ComboBox:
currencyFromComboBox = new JComboBox<>();
currencyFromComboBox.setFont(MY_FONT);
currencyFromComboBox.setBounds(550,110,200,50);
currencyFromComboBox.setModel(new DefaultComboBoxModel<>(currencyList.toArray(new String[0])));

//Currency From ComboBox:
currencyToComboBox = new JComboBox<>();
currencyToComboBox.setFont(MY_FONT);
currencyToComboBox.setBounds(550,180,200,50);
currencyToComboBox.setModel(new DefaultComboBoxModel<>(currencyList.toArray(new String[0])));

//Feedback Label:
feedbackLabel = new JLabel("");
feedbackLabel.setForeground(Color.YELLOW);
feedbackLabel.setFont(MY_FONT);
feedbackLabel.setBounds(50,250,700,50);
feedbackLabel.setVisible(false);

//Convert Button:
convertCurrencyButton = new JButton("Convert");
convertCurrencyButton.setFont(MY_FONT);
convertCurrencyButton.setBounds(300,320,200,50);
convertCurrencyButton.addActionListener(e -> {
    currencyFrom = (String) currencyFromComboBox.getSelectedItem();
    currencyTo = (String) currencyToComboBox.getSelectedItem();
    amount = Double.valueOf(inputAmountTextField.getText());
    if(currencyFrom.equals(currencyTo))
        generatedAmountTextField.setText(amount + "");
    else {
        CurrencyDTO currencyDTO = new CurrencyDTO(currencyFrom, currencyTo, amount);
        result = currencyService.convertCurrency(currencyDTO);
```

```
                generatedAmountTextField.setText(String.format("%.2f", result));
                currencyDTO.setResult(result);
                if (userName != null)
                    currencyService.updateHistory(userName, currencyDTO);
            }
        });

        //Adding All Components :
        container.add(headLabel);
        container.add(amount1Label);
        container.add(amount2Label);
        container.add(inputAmountTextField);
        container.add(generatedAmountTextField);
        container.add(currencyFromComboBox);
        container.add(currencyToComboBox);
        container.add(feedbackLabel);
        container.add(convertCurrencyButton);
    }

    public void createUserSpecificUI(String userName) {

        //Container:
        container = getContentPane();

        //Add Conversion Button:
        addConversionButton = new JButton("View History");
        addConversionButton.setFont(MY_FONT);
        addConversionButton.setBounds(300,390,200,50);
        addConversionButton.addActionListener(e -> {
            new ConversionHistoryFrame(userName, currencyService.getListOfConversions(userName));
        });

        //Adding Button:
        container.add(addConversionButton);
    }
}
```

ConversionHistoryFrame.java:

```
package com.byGroup3.ui;

/**
 * This is the GUI Representation Frame for showing User's Conversion History.
 *
 * @author Gaurav Amarnani, Eshwar Vazirani, Kritika Yadav.
 */

public class ConversionHistoryFrame extends JFrame {

    private Container container;
    private JTable table;
    private String[] columnNames;
    private TableColumnModel tableColumnModel;
    private JScrollPane scrollPane;
    private DefaultTableModel defaultTableModel;
    private Object[][] data;
    private final Font MY_FONT = new Font("Times New Roman" , Font.BOLD , 22);

    ConversionHistoryFrame(String userName, ArrayList<CurrencyDTO> listOfCurrencyDTO) {
        setTitle("Search History for " + userName + ".");
        setDefaultCloseOperation(JFrame.DISPOSE_ON_CLOSE);
        setSize(1300 , 300);
        setVisible(true);
        setResizable(false);
        setLocationRelativeTo(null);
        init(listOfCurrencyDTO);
    }

    public void init(ArrayList<CurrencyDTO> listOfCurrencyDTO) {
        //Container:
        container = getContentPane();
        container.setLayout(null);

        //Table Info :
        columnNames = new String[]{"Currency From: " , "Currency To : " , "Amount : " , "Result : "};
        data = new Object[listOfCurrencyDTO.size()][4];

        //Inserting data into Table dynamically:
        for(int i = 0; i < listOfCurrencyDTO.size(); i++) {
            CurrencyDTO currencyDTO = listOfCurrencyDTO.get(i);
            data[i][0] = currencyDTO.getFrom();
            data[i][1] = currencyDTO.getTo();
            data[i][2] = currencyDTO.getAmount();
            data[i][3] = currencyDTO.getResult();
        }

        //Table Mode:
        defaultTableModel = new DefaultTableModel(data , columnNames);

        //Table:
        table = new JTable();
        table.setModel(defaultTableModel);
        table.setRowHeight(30);

        //Table Column Model Information:
        tableColumnModel = table.getColumnModel();
        tableColumnModel.getColumn(0).setPreferredWidth(250);
        tableColumnModel.getColumn(1).setPreferredWidth(250);
        tableColumnModel.getColumn(2).setPreferredWidth(250);
        tableColumnModel.getColumn(3).setPreferredWidth(250);
        table.setEnabled(false);
        table.setPreferredScrollableViewportSize(new Dimension(1250,150));
        table.setFillsViewportHeight(true);
        table.setFont(MY_FONT);

        //Scroll Pane:
        scrollPane = new JScrollPane(table);
        scrollPane.setEnabled(false);
        scrollPane.setBounds(20, 50, 1250, 173);

        //Adding Element:
        add(scrollPane);
    }
}
```

CurrencyDTO.java:

```
package com.byGroup3.dto;

public class CurrencyDTO {

    private String from;
    private String to;
    private Double amount;
    private Double result;

    public CurrencyDTO() {   }

    public CurrencyDTO(String from, String to, Double amount) {
        this.from = from;
        this.to = to;
        this.amount = amount;
    }

    public CurrencyDTO(String from, String to, Double amount, Double result) {
        this.from = from;
        this.to = to;
        this.amount = amount;
        this.result = result;
    }

    public String getFrom() {
        return from;
    }

    public void setFrom(String from) {
        this.from = from;
    }

    public String getTo() {
        return to;
    }

    public void setTo(String to) {
        this.to = to;
    }

    public Double getAmount() {
        return amount;
    }

    public void setAmount(Double amount) {
        this.amount = amount;
    }

    public Double getResult() {
        return result;
    }

    public void setResult(Double result) {
        this.result = result;
    }
}
```

UserDTO.java:

```
package com.byGroup3.dto;

/**
 * This is the Data Transfer Object for User Information.
 *
 * @author Gaurav Amarnani, Eshwar Vazirani, Kritika Yadav.
 */

public class UserDTO {

    private String name;
    private String userName;
    private String email;
    private String password;

    public UserDTO() { }

    public UserDTO(String name, String userName, String email, String password) {
        this.name = name;
        this.userName = userName;
        this.email = email;
        this.password = password;
    }

    public String getName() {
        return name;
    }

    public void setName(String name) {
        this.name = name;
    }

    public String getUserName() {
        return userName;
    }

    public void setUserName(String userName) {
        this.userName = userName;
    }

    public String getEmail() {
        return email;
    }

    public void setEmail(String email) {
        this.email = email;
    }

    public String getPassword() {
        return password;
    }

    public void setPassword(String password) {
        this.password = password;
    }

    @Override
    public String toString() {
        return "{ Name = '" + name + "', UserName = '" + userName + "', Email = '" + email + "' }";
    }
}
```

ConnectionService.java:

```
package com.byGroup3.service;

/**
 * This is the Service class which will be used for providing a Connection to the Database.
 *
 * @author Gaurav Amarnani, Eshwar Vazirani, Kritika Yadav.
 */

public class ConnectionService {

    private static String URL;
    private static String USERNAME;
    private static String PASSWORD;

    public ConnectionService() {
        populateCredentials();
    }

    public String populateCredentials() {
        try {
            Properties properties = readPropertiesFile("database.properties");
            URL = properties.getProperty("url");
            USERNAME = properties.getProperty("username");
            PASSWORD = properties.getProperty("password");
            if(URL != null && USERNAME != null && PASSWORD != null)
                return null;
            else
                return "Wrong Database Credentials.";
        } catch (Exception exception) {
            return "Could not connect to Database due to Credentials Issue.";
        }
    }

    public Properties readPropertiesFile(String fileName) throws IOException {
        FileInputStream fileInputStream = null;
        Properties properties = null;
        try {
            fileInputStream = new FileInputStream(fileName);
            properties = new Properties();
            properties.load(fileInputStream);
        } catch(IOException ioException) {
            ioException.printStackTrace();
        } finally {
            if(fileInputStream != null)
                fileInputStream.close();
        }
        return properties;
    }

    public Connection createConnection() throws SQLException {
        if(URL != null && USERNAME != null && PASSWORD != null)
            return DriverManager.getConnection(URL, USERNAME, PASSWORD);
        else
            return null;
    }

    public void dumpConnection(Connection connection) {
        try {
            if(connection != null)
                connection.close();
        } catch(SQLException sqlException) {
            sqlException.printStackTrace();
        }
    }
}
```

CurrencyService.java:

```
package com.byGroup3.service;

/**
 * This is the Service class which will be used by the UI classes to deal with Business Logic
 * and connect with the Live Currency Conversion API.
 *
 * @author Gaurav Amarnani, Eshwar Vazirani, Kritika Yadav.
 */

public class CurrencyService {

    private CurrencyDAOImpl currencyDAOImpl = new CurrencyDAOImpl();

    public void createBackUpFiles() {
        new CurrencyBackupService();
    }

    public ArrayList<String> populateCurrencies() {
        try {
            Properties prop = readPropertiesFile("currencies.properties");
            String currencies = prop.getProperty("currencyList");
            ArrayList<String> currencyList = new ArrayList<>(Arrays.asList(currencies.split(",")));
            return currencyList;
        } catch (Exception exception) {
            return null;
        }
    }

    public Properties readPropertiesFile(String fileName) throws IOException {
        FileInputStream fileInputStream = null;
        Properties properties = null;
        try {
            fileInputStream = new FileInputStream(fileName);
            properties = new Properties();
            properties.load(fileInputStream);
        } catch(IOException ioException) {
            ioException.printStackTrace();
        } finally {
            if(fileInputStream != null)
                fileInputStream.close();
        }
        return properties;
    }

    public Double convertCurrency(CurrencyDTO currencyDTO) {
        String from = currencyDTO.getFrom();
        String to = currencyDTO.getTo();
        Double amount = currencyDTO.getAmount();
        Double result;
        try {
            String apikey = "8597d820-5a04-11ec-9f3a-6b1bc2fb4a16";
            String url = "https://freecurrencyapi.net/api/v2/latest?apikey=" + apikey + "&base_currency=" + from;
            URL urlForGetRequest = new URL(url);
            String readLine = null;
            HttpURLConnection httpURLConnection = (HttpURLConnection) urlForGetRequest.openConnection();
            httpURLConnection.setRequestMethod("GET");
            int responseCode = httpURLConnection.getResponseCode();
            if (responseCode == HttpURLConnection.HTTP_OK) {
                BufferedReader bufferedReader = new BufferedReader(new
InputStreamReader(httpURLConnection.getInputStream()));
                StringBuffer response = new StringBuffer();
                while ((readLine = bufferedReader.readLine()) != null) {
                    response.append(readLine);
                }
                bufferedReader.close();
                JSONObject jsonObject = new JSONObject(response.toString());
                Double exchangeRate = jsonObject.getJSONObject("data").getDouble(to);
                result = exchangeRate * amount;
            } else
                return convertCurrencyUsingPropertiesFile(from, to, amount);
        } catch (Exception ex) {
            return convertCurrencyUsingPropertiesFile(from, to, amount);

        }
        return result;
    }

    public Double convertCurrencyUsingPropertiesFile(String from, String to, Double amount) {
        String path = "C:\\Users\\DELL\\IdeaProjects\\Real_Time_Currency_Converter\\src\\currencies_backup\\";
        try {
            Properties prop = readPropertiesFile(path + from + ".properties");
            Double exchangeRate = Double.valueOf(prop.getProperty(to));
            return exchangeRate * amount;
        } catch(Exception exception) {
            exception.printStackTrace();
            return -1.0;
        }
    }

    public void updateHistory(String userName, CurrencyDTO currencyDTO) {
        currencyDAOImpl.storeConversion(currencyDTO, userName);
    }

    public ArrayList<CurrencyDTO> getListOfConversions(String userName) {
        return currencyDAOImpl.fetchUserHistory(userName);
    }
}
```

UserService.java:

```
package com.byGroup3.service;

/**
 * This is the Service class which will be used by the UI classes to deal with Business Logic and
 * perform database operation without having to access the DAO Object.
 *
 * @author Gaurav Amarnani, Eshwar Vazirani, Kritika Yadav.
 */

public class UserService {

    private UserDAO userDAO;

    public String validateUser(String name, String userName, String email, String password) {
        if(isValid(name)) {
            if(isValid(userName)) {
                if(isValid(email)) {
                    if(isValid(password))
                        return null;
                    return "Password is empty or blank.";
                }
                return "Email is empty or blank.";
            }
            return "Username is empty or blank.";
        }
        return "Name is empty or blank.";
    }

    public String validateUsernameAndPassword(String userName, String password) {
        String message = validateUsername(userName);
        if(message == null)
           return validatePassword(password);
        return message;
    }

    public String validateUsername(String username) {
        if(isValid(username))
            return doesUsernameMeetsRequirements(username);
        return "Username cannot be empty.";
    }

    public String validatePassword(String password) {
        if(isValid(password))
            return doesPasswordMeetsRequirements(password);
        return "Password cannot be empty.";
    }

    public Boolean isValid(String input) {
        if(input != null)
            return input.isBlank();
        return false;
    }

    public String doesUsernameMeetsRequirements(String username) {
        if(username.length() < 7 || username.length() > 20) {
            if(!Pattern.matches("a-zA-Z_", username)) {
                if(Pattern.matches("\\t\\n\\x0B\\f\\r", username))
                    return null;
                return "Username cannot have spaces in between.";
            }
            return "Username can only consist of alphabets and underscore.";
        }
        return "Username length should be between 6 and 20 characters.";
    }

    public String doesPasswordMeetsRequirements(String password) {
        if(password.length() < 7 || password.length() > 20) {
            if(Pattern.matches("\\t\\n\\x0B\\f\\r", password))
                return null;
            return "Password cannot have spaces in between.";
        }
        return "Password length should be between 6 and 20 characters.";
    }

    public String checkUsernameAndPassword(String userName, String password) {
        userDAO = new UserDAOImpl();
        return userDAO.checkUsernameAndPassword(userName, password);
    }

    public Boolean checkIfUserExists(String userName) {
        userDAO = new UserDAOImpl();
        return userDAO.checkIfUserExists(userName);
    }

    public Boolean saveUser(UserDTO userDTO) {
        userDAO = new UserDAOImpl();
        int result = userDAO.storeUserInformation(userDTO);
        return result == 1;
    }
}
```

UserDAO.java:

```
package com.byGroup3.dao;

/**
 * This structure defines the standard operations to be performed on a model object (DTO).
 *
 * @author Gaurav Amarnani, Eshwar Vazirani, Kritika Yadav.
 */

public interface UserDAO {

    String checkUsernameAndPassword(String userName, String password);

    Boolean checkIfUserExists(String userName);

    Integer storeUserInformation(UserDTO userDTO);
}
```

UserDAOImpl.java:

```
package com.byGroup3.dao;

/**
 * This is the Implementation Class which is responsible to get data from the Database.
 *
 * @author Gaurav Amarnani, Eshwar Vazirani, Kritika Yadav.
 */

public class UserDAOImpl implements UserDAO{

    private ConnectionService connectionService;

    @Override
    public String checkUsernameAndPassword(String userName, String password) {
        try {
            connectionService = new ConnectionService();
            PreparedStatement preparedStatement =
connectionService.createConnection().prepareStatement(SQLUtility.checkUsernameAndPassword);
            preparedStatement.setString(1, userName);
            preparedStatement.setString(2, password);
            if(preparedStatement.execute())
                return null;
            return "Wrong ID or Password.";
        } catch (SQLException exception) {
            return exception.getMessage();
        }
    }

    @Override
    public Boolean checkIfUserExists(String userName) {
        try {
            connectionService = new ConnectionService();
            PreparedStatement preparedStatement = 
connectionService.createConnection().prepareStatement(SQLUtility.fetchUserByUsername);
            preparedStatement.setString(1, userName);
            ResultSet resultSet = preparedStatement.executeQuery();
            return resultSet.next();
        } catch(SQLException sqlException) {
            sqlException.printStackTrace();
        }
        return false;
    }

    @Override
    public Integer storeUserInformation(UserDTO userDTO) {
        try {
            connectionService = new ConnectionService();
            PreparedStatement preparedStatement = 
connectionService.createConnection().prepareStatement(SQLUtility.saveUserInformation);
            preparedStatement.setString(1, userDTO.getName());
            preparedStatement.setString(2, userDTO.getUserName());
            preparedStatement.setString(3, userDTO.getEmail());
            preparedStatement.setString(4, userDTO.getPassword());
            return preparedStatement.executeUpdate();
        } catch(SQLException sqlException) {
            sqlException.printStackTrace();
            return -1;
        }
    }
}
```

CurrencyDAO.java:

```
package com.byGroup3.dao;

/**
 * This structure defines the standard operations to be performed on a model object CurrencyDTO.
 *
 * @author Gaurav Amarnani, Eshwar Vazirani, Kritika Yadav.
 */

public interface CurrencyDAO {

    ArrayList<CurrencyDTO> fetchUserHistory(String username);

    Integer storeConversion(CurrencyDTO currencyDTO, String username); }
```

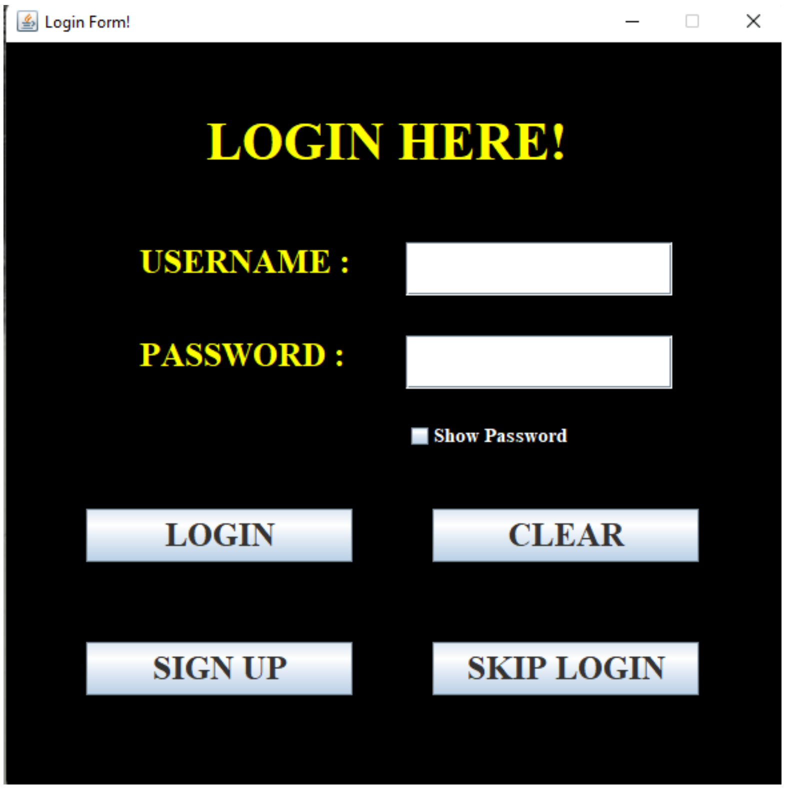
CurrencyDAOImpl.java:

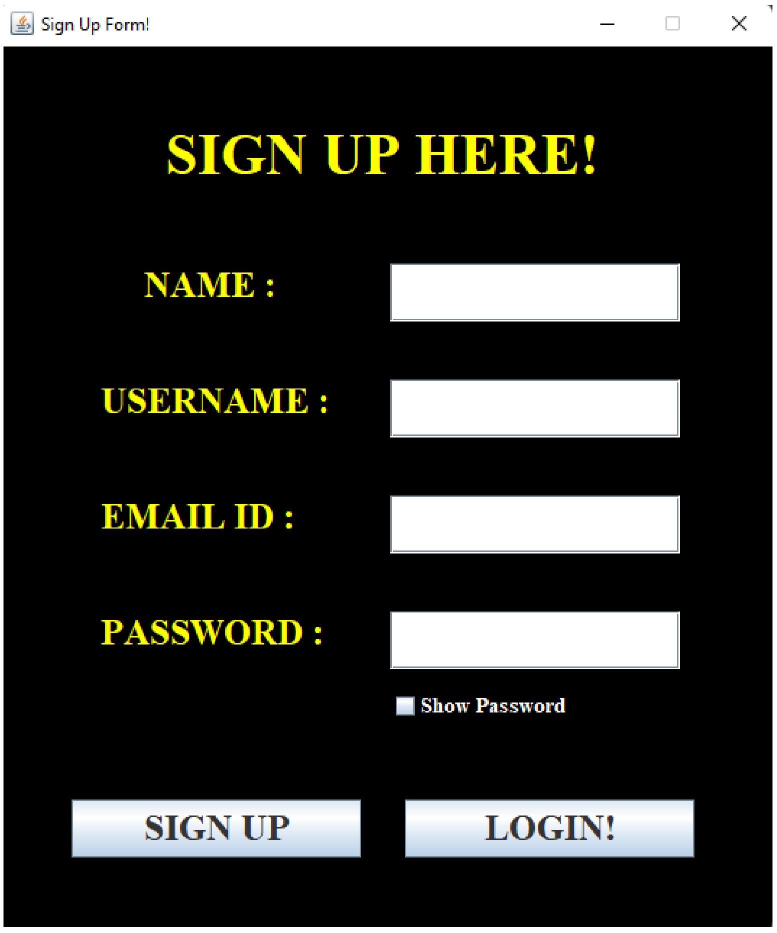
```
package com.byGroup3.dao;

/**
 * This is the Implementation Class which is responsible to get data from the Database.
 *
 * @author Gaurav Amarnani, Eshwar Vazirani, Kritika Yadav.
 */

public class CurrencyDAOImpl implements CurrencyDAO{

    private ConnectionService connectionService;

    @Override
    public ArrayList<CurrencyDTO> fetchUserHistory(String userName) {
        ArrayList<CurrencyDTO> listOfCurrencyDTO = new ArrayList<>();
        try {
            connectionService = new ConnectionService();
            PreparedStatement preparedStatement = 
connectionService.createConnection().prepareStatement(SQLUtility.fetchHistoryByUsername);
            preparedStatement.setString(1, userName);
            ResultSet resultSet = preparedStatement.executeQuery();
            while(resultSet.next())
                listOfCurrencyDTO.add(new CurrencyDTO(resultSet.getString(1), resultSet.getString(2),
resultSet.getDouble(3), resultSet.getDouble(4)));
        } catch(SQLException sqlException) {
            sqlException.printStackTrace();
            return null;
        }
        return listOfCurrencyDTO;
    }

    @Override
    public Integer storeConversion(CurrencyDTO currencyDTO, String userName) {
        try {
            connectionService = new ConnectionService();
            PreparedStatement preparedStatement =
connectionService.createConnection().prepareStatement(SQLUtility.storeConversion);
            preparedStatement.setString(1, userName);
            preparedStatement.setString(2, currencyDTO.getFrom());
            preparedStatement.setString(3, currencyDTO.getTo());
            preparedStatement.setDouble(4, currencyDTO.getAmount());
            preparedStatement.setDouble(5, currencyDTO.getResult());
            return preparedStatement.executeUpdate();
        } catch(SQLException sqlException) {
            sqlException.printStackTrace();
            return -1;
        }
    }
}
```

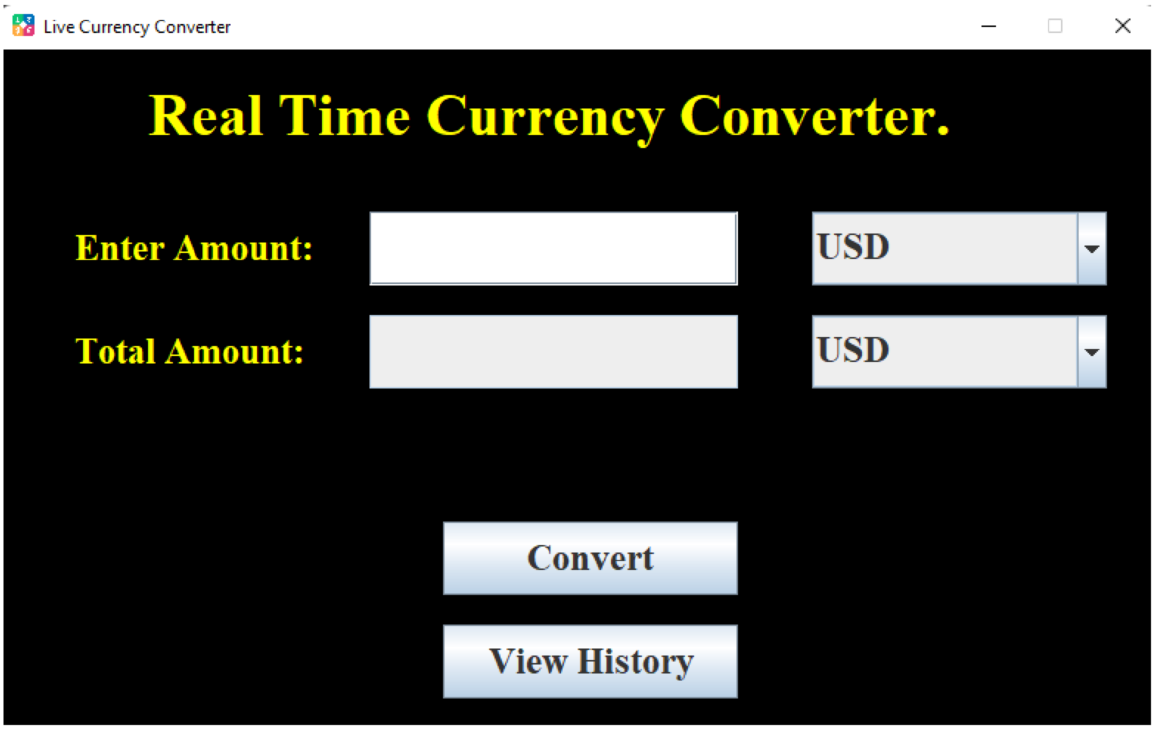
SQLUtility.java:

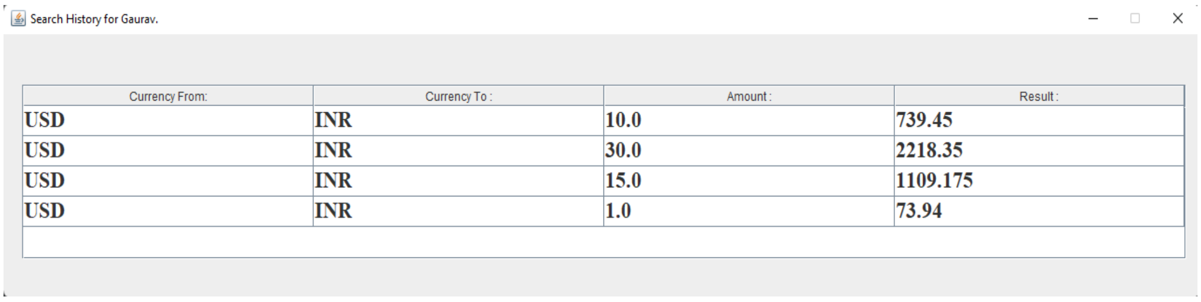
```
package com.byGroup3.utility;

/**
 * This is the utility class which will provide us with the SQL Queries.
 *
 * @author Gaurav Amarnani, Eshwar Vazirani, Kritika Yadav.
 */

public class SQLUtility {

    public static final String fetchUserByUsername = "SELECT * FROM 'rtcc'.'users' WHERE 'username' = ?;";

    public static final String checkUsernameAndPassword = "SELECT * FROM 'rtcc'.'users' WHERE 'username' = ?
AND 'password' = ?;";

    public static final String saveUserInformation = "INSERT INTO 'rtcc'.'users' (`name`, `username`, `email`,
`password`) VALUES (?, ?, ?, ?);";

    public static final String fetchHistoryByUsername = "SELECT * FROM 'rtcc'.'conversions' WHERE 'username' =
?;";

    public static final String storeConversion = "INSERT INTO 'rtcc'.'conversions' ('username', 'from', 'to', 'amount',
'result') VALUES (?, ?, ?, ?, ?);";
}
```

Outputs:


Login Form!
LOGIN HERE!
USERNAME :
PASSWORD :
Show Password
LOGIN
CLEAR
SIGN UP
SKIP LOGIN


Sign Up Form!
SIGN UP HERE!
NAME :
USERNAME :
EMAIL ID :
PASSWORD :
Show Password
SIGN UP
LOGIN!





Currency From:	Currency To :	Amount :	Result :
USD	INR	10.0	739.45
USD	INR	30.0	2218.35
USD	INR	15.0	1109.175
USD	INR	1.0	73.94