```
# Define the root logger with appender file
log = ./
log4j.rootLogger=DEBUG, FILE, stdout

# Define the file appender
log4j.appender.FILE=org.apache.log4j.FileAppender
log4j.appender.FILE.File=${log}/log.out

# Define the layout for file appender
log4j.appender.FILE.layout=org.apache.log4j.PatternLayout
log4j.appender.FILE.layout.ConversionPattern=%d %c - %m%n

# For stdout logging
log4j.appender.stdout=org.apache.log4j.ConsoleAppender
log4j.appender.stdout.layout=org.apache.log4j.PatternLayout
log4j.appender.stdout.layout.ConversionPattern=%d %c - %m%n
```

```java
package Main;

import ViewModels.Source.UserViewModel;
import Views.Source.LoginView;

/**
 * Main class just for starting the main frame
 **/

public class mainSource {
    public static void main(String[] args) {
        LoginView loginFrame = new LoginView(new UserViewModel());
        loginFrame.setVisible(true);
    }
}
```

```java
package Views.Source;

import java.awt.*;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;

import javax.swing.*;

import Services.Source.DBService;
import ViewModels.Source.UserViewModel;
import ViewModels.Source.CalculateInsulinViewModel;
import ViewModels.Source.ViewHistoryViewModel;
import Views.Interface.IUserView;

@SuppressWarnings("serial")
public class UserView extends JFrame implements IUserView {
    // UI elements
    private JLabel fullNameLabel;

    private JButton logout;
    private JButton viewHistoryButton;
    private JButton calculateInsulinButton;

    private DBService db;
    public UserView(String userId) {
        db = new DBService();
        try{ fullNameLabel = new JLabel(db.getName(userId));}
        catch(NullPointerException e) { fullNameLabel = new JLabel(userId); }
        initializeUI();
        commond(userId);

    }

    public void commond(String userId){
        viewHistoryButton.addActionListener(new ActionListener() {
            @Override
            public void actionPerformed(ActionEvent e) {
                ViewHistoryView frame = new ViewHistoryView(new ViewHistoryViewModel(userId));
                frame.setVisible(true);
                setVisible(false);
            }
        });
        calculateInsulinButton.addActionListener(new ActionListener() {
            @Override
            public void actionPerformed(ActionEvent e) {
                CalculateInsulinView frame = new CalculateInsulinView(new CalculateInsulinViewModel(
userId));
                frame.setVisible(true);
                setVisible(false);
            }
        });
        logout.addActionListener(new ActionListener() {
            @Override
            public void actionPerformed(ActionEvent e) {
                LoginView frame = new LoginView(new UserViewModel());
                frame.setVisible(true);
                setVisible(false);
            }
        });
    }

    public void initializeUI() {
        this.setDefaultCloseOperation(EXIT_ON_CLOSE);
        this.setTitle("Insulin Calculator");

        this.logout = new JButton("Logout");
        this.viewHistoryButton = new JButton("View History");
        this.calculateInsulinButton = new JButton("Calculate Insulin");
        JPanel north = new JPanel();
```

```
        north.setLayout(new FlowLayout());
        north.setComponentOrientation(ComponentOrientation.LEFT_TO_RIGHT);
        north.add(fullNameLabel);

        // center Panel
        JPanel center = new JPanel();
        center.setLayout(new FlowLayout());
        center.setComponentOrientation(ComponentOrientation.LEFT_TO_RIGHT);

        //logout.setBackground(Color.RED);
        center.add(logout);
        center.add(viewHistoryButton);
        center.add(calculateInsulinButton);

        // add panels to main frame
        this.setLocationRelativeTo(null);
        this.setBounds(500, 200, 350, 100);
        this.setLayout(new BorderLayout());
        this.add(north, BorderLayout.NORTH);
        this.add(center, BorderLayout.CENTER);
    }
}
```

```java
package Views.Source;

import Models.Source.UserDetails;
import ViewModels.Source.RegisterViewModel;
import ViewModels.Source.UserViewModel;
import Views.Interface.ILoginView;

import javax.swing.*;
import java.awt.*;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;

public class LoginView extends JFrame implements ILoginView {

    // UI elements
    private JPasswordField passField;
    private JTextField userField;
    private JLabel loginLabel;
    private JLabel passLabel;
    private JLabel userLabel;
    private JButton loginButton;
    private JButton registerButton;
    private JButton loginWithGoogleButton;

    public LoginView(UserViewModel userViewModel) {
        initializeUI();
        commond(userViewModel);
    }

    public void commond(UserViewModel userViewModel){

        loginButton.addActionListener(new ActionListener() {
            @Override
            public void actionPerformed(ActionEvent e) {
                String username = userField.getText();
                String password = new String(passField.getPassword());
                UserDetails userDetails = new UserDetails(username, password);
                String doLogin = userViewModel.doLogin(userDetails);
                handleLoginResult(doLogin);
            }
        });

        registerButton.addActionListener(new ActionListener() {
            @Override
            public void actionPerformed(ActionEvent e) {
                handleRegister();
                setVisible(false);
            }
        });

        loginWithGoogleButton.addActionListener(new ActionListener() {
            @Override
            public void actionPerformed(ActionEvent e) {
                 UserDetails doLoginGoogle = userViewModel.doLoginWithGoogle();
                handleLoginWithGoogleResult(doLoginGoogle);
            }
        });
    }

    public void handleLoginWithGoogleResult(UserDetails res) {
        if (res == null) {
            JOptionPane.showMessageDialog(null, "Bad google credentials", "Error", JOptionPane.
ERROR_MESSAGE);
        } else {
            new UserViewModel().doRegister(res);
            UserView userFrame = new UserView(res.getFirstName()+" "+res.getLastName());
            this.dispose();
            userFrame.setVisible(true);
            this.setVisible(false);
```

```
        }
    }

    public void handleRegister() {
        RegisterView registerFrame = new RegisterView(new RegisterViewModel());
        registerFrame.setVisible(true);
        this.setVisible(false);
    }

    public void handleLoginResult(String res) {
        if (res == null) {
            JOptionPane.showMessageDialog(null, "Bad credentials", "Error", JOptionPane.ERROR_MESSAGE);
        } else {
            UserView userFrame = new UserView(res);
            this.dispose();
            userFrame.setVisible(true);
            this.setVisible(false);
        }
    }

    public void initializeUI() {
        this.setDefaultCloseOperation(EXIT_ON_CLOSE);
        this.setTitle("Insulin Calculator - Login");

        // init UI elements
        final Dimension fieldSize = new Dimension(200, 24);
        this.loginLabel = new JLabel("Login");
        this.userLabel = new JLabel("Email:          ");
        this.passLabel = new JLabel("Password: ");
        this.userField = new JTextField();
        this.passField = new JPasswordField();
        this.userField.setPreferredSize(fieldSize);
        this.passField.setPreferredSize(fieldSize);

        loginLabel.setFont(new Font("Serif",Font.PLAIN,20));
        // north Panel
        JPanel north = new JPanel();
        north.setLayout(new FlowLayout());
        north.setComponentOrientation(ComponentOrientation.LEFT_TO_RIGHT);
        north.add(loginLabel);


        // center Panel
        JPanel center = new JPanel();
        center.setLayout(new FlowLayout());
        center.setComponentOrientation(ComponentOrientation.LEFT_TO_RIGHT);
        center.add(userLabel);
        center.add(userField);
        center.add(passLabel);
        center.add(passField);
        // south panel
        JPanel south = new JPanel();
        this.loginButton = new JButton("Login");
        this.registerButton = new JButton("Register");
        this.loginWithGoogleButton = new JButton("Login(Google)");
        south.add(loginButton);
        south.add(registerButton);
        south.add(loginWithGoogleButton);

        // add panels to main frame
        this.setLocationRelativeTo(null);
        this.setBounds(500, 200, 300, 200);
        this.setLayout(new BorderLayout());
        this.add(north, BorderLayout.NORTH);
        this.add(center, BorderLayout.CENTER);
        this.add(south, BorderLayout.SOUTH);
    }
}
```

```java
package Views.Source;

import Exceptions.MyException;
import Models.Source.UserDetails;
import ViewModels.Interface.IRegisterViewModel;
import ViewModels.Source.UserViewModel;
import ViewModels.Source.RegisterViewModel;

import javax.swing.*;
import java.awt.*;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;

public class RegisterView extends JFrame {

    // UI elements
    private JTextField passField;
    private JTextField firstnameField;
    private JTextField lastnameField;
    private JTextField emailField;
    private JTextField phoneField;

    private JLabel passLabel;
    private JLabel firstnameLabel;
    private JLabel lastnameLabel;
    private JLabel emailLabel;
    private JLabel phoneLabel;

    private JButton registerButton;
    private JButton backButton;


    public RegisterView(RegisterViewModel viewModel) {
        initializeUI();
        commond (viewModel);
    }

    public void commond(RegisterViewModel viewModel){
        registerButton.addActionListener(new ActionListener() {
            @Override
            public void actionPerformed(ActionEvent e) {
                UserDetails userDetails = new UserDetails(emailField.getText(),passField.getText(),
firstnameField.getText(),
                        lastnameField.getText(),phoneField.getText());
                try {
                    viewModel.doRegister(userDetails);
                } catch (MyException ex) {
                    ex.printStackTrace();
                }

                new LoginView(new UserViewModel()).handleLoginResult(new UserViewModel().doLogin(
userDetails));
                setVisible(false);

            }
        });

        backButton.addActionListener(new ActionListener() {
            @Override
            public void actionPerformed(ActionEvent e) {
                LoginView frame = new LoginView(new UserViewModel());
                frame.setVisible(true);
                setVisible(false);
            }
        });
    }

    private void handleRegisterResult(boolean res) {
        if (!res) {
```

```
            JOptionPane.showMessageDialog(null, "Failed to register", "Error",
                    JOptionPane.ERROR_MESSAGE);
            this.setVisible(false);
            LoginView login = new LoginView(new UserViewModel());
            login.setVisible(true);
        } else {
            JOptionPane.showMessageDialog(null,
                    "Register success! now you can login.", "Success",
                    JOptionPane.OK_OPTION | JOptionPane.INFORMATION_MESSAGE);
            this.dispose();
            this.setVisible(false);

            UserDetails userDetails = new UserDetails(emailField.getText(),passField.getText());

            UserViewModel userViewModel = new UserViewModel();
            String s =userViewModel.doLogin(userDetails);

            UserView userFrame = new UserView(s);

            this.dispose();

            userFrame.setVisible(true);
        }
    }

    private void initializeUI() {
        this.setTitle("Insulin Calculator - Register");
        // init UI elements
        final Dimension fieldSize = new Dimension(200, 24);
        this.emailLabel = new JLabel("E-mail:        ");
        this.passLabel = new JLabel("Password: ");
        this.firstnameLabel = new JLabel("FirstName: ");
        this.lastnameLabel = new JLabel("LastName: ");
        this.phoneLabel = new JLabel("Phone:       ");

        this.emailField = new JTextField();
        this.passField = new JTextField();
        this.firstnameField = new JTextField();
        this.lastnameField = new JTextField();
        this.phoneField = new JTextField();

        this.emailField.setPreferredSize(fieldSize);
        this.passField.setPreferredSize(fieldSize);
        this.firstnameField.setPreferredSize(fieldSize);
        this.lastnameField.setPreferredSize(fieldSize);
        this.phoneField.setPreferredSize(fieldSize);

        // north Panel
        JPanel panelT = new JPanel();
        panelT.setLayout(new FlowLayout());
        panelT.setComponentOrientation(ComponentOrientation.LEFT_TO_RIGHT);
        panelT.add(emailLabel);
        panelT.add(emailField);
        panelT.add(passLabel);
        panelT.add(passField);
        panelT.add(firstnameLabel);
        panelT.add(firstnameField);
        panelT.add(lastnameLabel);
        panelT.add(lastnameField);
        panelT.add(phoneLabel);
        panelT.add(phoneField);

        JPanel panelR = new JPanel();
        panelR.setLayout(new FlowLayout());
        panelR.setComponentOrientation(ComponentOrientation.LEFT_TO_RIGHT);
        this.registerButton = new JButton("Register");
        this.backButton = new JButton("Back");

        this.registerButton.setBounds(500, 200, 20, 20);
```

```
        this.backButton.setBounds(500,200,20,20);
        panelT.add(backButton);
        panelT.add(registerButton);

        JPanel panel = new JPanel();
        panel.setLayout(new BoxLayout(panel, BoxLayout.Y_AXIS));
        panel.add(panelT);
        // panel.add(panelR);

        // add panels to main frame
        this.setLocationRelativeTo(null);
        this.setLayout(null);
        this.setBounds(500, 200, 300, 300);
        this.setContentPane(panel);
    }
}
```

```java
package Views.Source;

import java.awt.*;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.util.List;

import javax.swing.*;
import javax.swing.table.DefaultTableModel;

import Models.Source.HistoryItem;
import ViewModels.Source.UserViewModel;
import ViewModels.Source.ViewHistoryViewModel;
import Views.Interface.IViewHistoryView;

@SuppressWarnings("serial")
public class ViewHistoryView extends JFrame implements IViewHistoryView {


    // UI elements
    String[] columnNames = { "#", "Time", "Insulin Level", "Carbs Amount", "Result" };
    JTable table;
    List<HistoryItem> history;
    private JButton logoutButton;
    private JButton backButton;
    public ViewHistoryView(ViewHistoryViewModel viewModel) {
        this.history = viewModel.getHistory();
        initializeUI();
        commond(viewModel);

    }
    public void commond(ViewHistoryViewModel viewModel){
        logoutButton.addActionListener(new ActionListener() {
            @Override
            public void actionPerformed(ActionEvent e) {
                LoginView frame = new LoginView(new UserViewModel());
                frame.setVisible(true);
                setVisible(false);
            }
        });
        backButton.addActionListener(new ActionListener() {
            @Override
            public void actionPerformed(ActionEvent e) {
                UserView frame = new UserView(viewModel.getUserId());
                frame.setVisible(true);
                setVisible(false);
            }
        });
    }
    public void initializeUI() {
        this.setTitle("Insulin Calculator - View History");

        this.logoutButton = new JButton("Logout");
        this.backButton = new JButton("Back");
        DefaultTableModel model = new DefaultTableModel(new Object[][] {}, columnNames) {
            @Override
            public boolean isCellEditable(int rowIndex, int columnIndex) {
                return false;
            }
        };
        for (int i = 0; i < this.history.size(); i++) {
            HistoryItem item = this.history.get(i);
            Object[] objects = new Object[] {i+1,item.getDate(),item.getBloodSugar(),item.getCarbsAmount
(),item.getInsulinLevel()};
            model.addRow(objects);
        }

        table = new JTable(model);
        JScrollPane scrollPane = new JScrollPane(table);
```

```
        table.setFillsViewportHeight(true);

        JPanel panelBackLogout = new JPanel();
        panelBackLogout.setLayout(new FlowLayout());
        panelBackLogout.setComponentOrientation(ComponentOrientation.LEFT_TO_RIGHT);
        panelBackLogout.add(backButton);
        panelBackLogout.add(logoutButton);

        JPanel panel = new JPanel();
        panel.setLayout(new BoxLayout(panel, BoxLayout.Y_AXIS));
        panel.add(scrollPane);
        panel.add(panelBackLogout);

        // add panels to main frame
        this.setLocationRelativeTo(null);
        this.setLayout(null);
        this.setBounds(500, 200, 800, 500);
        this.setContentPane(panel);
        //this.setContentPane(backButton);
    }
}
```

```java
package Views.Source;

import java.awt.*;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;

import javax.swing.*;

import Exceptions.MyException;
import ViewModels.Source.UserViewModel;
import ViewModels.Source.CalculateInsulinViewModel;
import Views.Interface.ICalculateInulinView;

@SuppressWarnings("serial")
public class CalculateInsulinView extends JFrame implements ICalculateInulinView {
    // UI elements
    private JTextField bloodSugarField;
    private JTextField carbsAmountField;
    private JLabel bloodSugarLabel;
    private JLabel carbsAmountLabel;
    private JLabel insulinLevelLabel;
    private JButton calcButton;

    private JButton logoutButton;
    private JButton backButton;

    public CalculateInsulinView(CalculateInsulinViewModel viewModel) {
        initializeUI();
        commond(viewModel);
    }

    public void commond (CalculateInsulinViewModel viewModel){
        calcButton.addActionListener(new ActionListener() {
            @Override
            public void actionPerformed(ActionEvent e) {

                try {
                    double bs= Double.parseDouble(bloodSugarField.getText()),
                            ca = Double.parseDouble(carbsAmountField.getText());
                            double resultInsulinLevel = viewModel.calculate(bs,ca);
                    handleCalculateResult(resultInsulinLevel);
                } catch (NumberFormatException |MyException.NegNumNotAllowed w){
                    w.printStackTrace();
                    JOptionPane.showMessageDialog(null, w.getMessage(), "Error", JOptionPane.
ERROR_MESSAGE);
                }
            }
        });
        logoutButton.addActionListener(new ActionListener() {
            @Override
            public void actionPerformed(ActionEvent e) {
                LoginView frame = new LoginView(new UserViewModel());
                frame.setVisible(true);
                setVisible(false);
            }
        });
        backButton.addActionListener(new ActionListener() {
            @Override
            public void actionPerformed(ActionEvent e) {
                UserView frame = new UserView(viewModel.getUserId());
                frame.setVisible(true);
                setVisible(false);
            }
        });
    }

    public void handleCalculateResult(double res) {
        this.insulinLevelLabel.setText("Result: " + String.valueOf(res));
    }
```

```
    public void initializeUI() {
        this.setTitle("Insulin Calculator - Calculate");

        // init UI elements
        final Dimension fieldSize = new Dimension(200, 24);
        this.bloodSugarLabel = new JLabel("Blood Sugar Amount: ");
        this.carbsAmountLabel = new JLabel("Carbs Amount: ");
        this.insulinLevelLabel = new JLabel("Result: ");
         this.logoutButton = new JButton("Logout");
         this.backButton = new JButton("Back");
        this.bloodSugarField = new JTextField();
        this.carbsAmountField = new JTextField();
        this.bloodSugarField.setPreferredSize(fieldSize);
        this.carbsAmountField.setPreferredSize(fieldSize);

        // north Panel
        JPanel north = new JPanel();
        north.setLayout(new FlowLayout());
        north.setComponentOrientation(ComponentOrientation.LEFT_TO_RIGHT);
        north.add(bloodSugarLabel);
        north.add(bloodSugarField);

        // center Panel
         this.calcButton = new JButton("Calculate");
         JPanel center = new JPanel();
         center.setLayout(new FlowLayout());
         center.setComponentOrientation(ComponentOrientation.LEFT_TO_RIGHT);
         center.add(carbsAmountLabel);
         center.add(carbsAmountField);
         center.add(calcButton);
         center.add(insulinLevelLabel);
         // south panel
         JPanel south = new JPanel();
         this.setLayout(new BorderLayout());
        south.add(backButton);
        south.add(logoutButton);

        // add panels to main frame
        this.setLocationRelativeTo(null);
        this.setBounds(500, 200, 350, 180);
        this.setLayout(new BorderLayout());
        this.add(north, BorderLayout.NORTH);
        this.add(center, BorderLayout.CENTER);
        this.add(south, BorderLayout.SOUTH);
    }
}
```

```java
package Views.Interface;

public interface IUserView {
    public void initializeUI();
}
```

```java
package Views.Interface;

import Models.Source.UserDetails;

public interface ILoginView {

    public void handleLoginWithGoogleResult(UserDetails res);

    public void handleRegister();

    public void handleLoginResult(String res);

    public void initializeUI();
}
```

```java
package Views.Interface;

public interface IRegisterView {
}
```

```java
package Views.Interface;

public interface IViewHistoryView {
    public void initializeUI();
}
```

```java
package Views.Interface;

import ViewModels.Source.CalculateInsulinViewModel;
import Views.Source.CalculateInsulinView;

public interface ICalculateInulinView {

    /**
     * on function -
     * has three actions:
     *          1. calcButton -
     *              calls calculate function (of viewModel) and send its 2 params: (one) for amount of
Sugar in the blood and (two) for amount of carbohydrate in a meal
     *              and displays the result (insulin level) to the user
     *          2. logoutButton -
     *              logout of user
     *          3. backButton -
     *              back to the previous screen
     *
     * @param viewModel is connection between View and Model (MVVM)
     */
    public void commond(CalculateInsulinViewModel viewModel);

    /**
     *
     * on function -
     * Creates the screen display for the user
     *
     */
    public void initializeUI();

    /**
     *
     * on function -
     * Print the result of calculation for user
     *
     * @param res is the result (insulin level)
     */
    public void handleCalculateResult(double res);
}
```

```java
package Models.Source;

import Models.Interface.IHistoryItem;

import java.text.DateFormat;
import java.text.SimpleDateFormat;
import java.util.Date;

public class HistoryItem implements IHistoryItem {

    private String date;    // Date of the test (Amount of insulin that needs to be injected)
    private String bloodSugar;  // Amount of sugar in blood
    private String carbsAmount; // Amount of carbohydrate in a meal
    private String insulinLevel;// Insulin level (the result of the calculation)

    public HistoryItem(String bloodSugar, String carbsAmount, String insulinLevel) {
        DateFormat dateFormat = new SimpleDateFormat("dd/MM/yyyy HH:mm");
        Date date = new Date();
        this.date = dateFormat.format(date);
        this.bloodSugar = bloodSugar;
        this.carbsAmount = carbsAmount;
        this.insulinLevel = insulinLevel;
    }

    public String getDate() {
        return date;
    }

    public String getBloodSugar() {
        return bloodSugar;
    }

    public String getCarbsAmount() {
        return carbsAmount;
    }

    public String getInsulinLevel() {
        return insulinLevel;
    }

}
```

```java
package Models.Source;

import Models.Interface.IUserDetails;

public class UserDetails implements IUserDetails {

    private String pass;
    private String firstName;
    private String lastName;
    private String email;
    private String phone;

    public UserDetails(String email, String pass) {
        super();
        this.email = email;
        this.pass = pass;
    }

    public UserDetails(String email, String pass, String firstname,String lastName,String phone) {
        super();
        this.email = email;
        this.pass = pass;
        this.firstName = firstname;
        this.lastName = lastName;
        this.phone = phone;
    }

    public String getPass() { return pass; }

    public void setPass(String pass) { this.pass = pass; }

    public String getFirstName() { return this.firstName; }

    public void setFirstName(String firstName) {
        this.firstName = firstName;
    }

    @Override
    public String getLastName() { return this.lastName; }

    @Override
    public void setLastName(String lastName) { this.lastName = lastName; }

    @Override
    public String getEmail() { return this.email; }

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

    @Override
    public String getPhone() { return this.phone; }

    @Override
    public void setPhone(String phone) { this.phone = phone; }

}
```

```java
package Models.Interface;

public interface IHistoryItem {
    /**
     * @return Date of the test (Amount of insulin that needs to be injected)
     */
    public String getDate();

    /**
     * @return Amount of sugar in blood
     */
    public String getBloodSugar();

    /**
     * @return amount of carbohydrate in a meal
     */
    public String getCarbsAmount();

    /**
     * @return insulin level (the result of the calculation)
     */
    public String getInsulinLevel();
}
```

```java
package Models.Interface;

public interface IUserDetails {

    /**
     * @return password of User
     */
    public String getPass();

    /**
     * @param pass is password, for change the password of User
     */
    public void setPass(String pass);

    /**
     * @return First Name of User
     */
    public String getFirstName();

    /**
     * @param firstName is First Name, for change the First Name of User
     */
    public void setFirstName(String firstName);

    /**
     * @return Last Name of User
     */
    public String getLastName();

    /**
     * @param lastName is Last Name, for change the Last Name of User
     */
    public void setLastName(String lastName);

    /**
     * @return Email of User, is the username for user
     */
    public String getEmail();

    /**
     * @param email is Email, for change the email/username of User
     */
    public void setEmail(String email);

    /**
     * @return Phone of User
     */
    public String getPhone();

    /**
     * @param phone is Phone, for change the Phone of User
     */
    public void setPhone(String phone);

}
```

```
Manifest-Version: 1.0
Main-Class: Main.mainSource
Class-Path: google-api-services-oauth2-v2-rev20190313-1.30.1.jar mongodb
 -driver-3.4.3.jar google-oauth-client-java6-1.30.1.jar checker-compat-q
 ual-2.5.5.jar j2objc-annotations-1.3.jar bson-3.4.3.jar jackson-core-2.
 9.9.jar javax.servlet-3.0.0.v201112011016.jar jetty-continuation-8.2.0.
 v20160908.jar jetty-io-8.2.0.v20160908.jar httpcore-4.4.11.jar google-h
 ttp-client-xml-1.30.1.jar google-api-client-jackson2-1.30.2.jar commons
 -logging-1.2.jar checker-compat-qual-2.5.2.jar log4j-1.2.17.jar jdom-2.
 0.6.jar google-http-client-appengine-1.30.1.jar xpp3-1.1.4c.jar j2objc-
 annotations-1.1.jar httpclient-4.5.10.jar junit-4.12.jar junit.jar guav
 a-28.1-android.jar animal-sniffer-annotations-1.14.jar google-oauth-cli
 ent-1.30.5.jar jackson-core-2.10.2.jar google-api-client-xml-1.30.2.jar
  jdo2-api-2.3-eb.jar google-http-client-1.30.1.jar google-http-client-j
 ackson2-1.34.1.jar opencensus-api-0.24.0.jar google-http-client-gson-1.
 30.1.jar google-api-client-java6-1.30.2.jar animal-sniffer-annotations-
 1.18.jar opencensus-api-0.21.0.jar jetty-util-8.2.0.v20160908.jar grpc-
 context-1.22.1.jar google-api-client-protobuf-1.30.2.jar httpcore-4.4.1
 3.jar google-http-client-1.34.0.jar google-api-client-servlet-1.30.2.ja
 r opencensus-contrib-http-util-0.24.0.jar guava-26.0-android.jar google
 -api-client-1.30.1.jar mongodb-driver-core-3.4.3.jar gson-2.8.5.jar goo
 gle-api-client-1.30.2.jar transaction-api-1.1.jar google-api-client-gso
 n-1.30.2.jar jsr305-3.0.2.jar httpcore-4.4.12.jar error_prone_annotatio
 ns-2.1.3.jar google-oauth-client-servlet-1.30.1.jar httpclient-4.5.8.ja
 r jetty-server-8.2.0.v20160908.jar guava-28.2-android.jar failureaccess
 -1.0.1.jar jetty-http-8.2.0.v20160908.jar google-oauth-client-1.30.1.ja
 r protobuf-java-2.6.1.jar google-api-client-appengine-1.30.2.jar httpcl
 ient-4.5.11.jar google-oauth-client-appengine-1.30.1.jar grpc-context-1
 .19.0.jar error_prone_annotations-2.3.4.jar google-http-client-protobuf
 -1.30.1.jar google-http-client-jackson2-1.30.1.jar google-http-client-1
 .34.1.jar opencensus-contrib-http-util-0.21.0.jar google-http-client-an
 droid-1.30.1.jar httpclient-4.5.9.jar commons-codec-1.11.jar error_pron
 e_annotations-2.3.2.jar google-oauth-client-jetty-1.30.5.jar google-oau
 th-client-java6-1.30.5.jar google-api-client-android-1.30.2.jar
```

```java
package Services.Source;

import static com.mongodb.client.model.Filters.and;
import static com.mongodb.client.model.Filters.eq;


import java.util.LinkedList;
import java.util.List;

import org.bson.Document;

import com.mongodb.MongoClient;
import com.mongodb.MongoClientURI;
import com.mongodb.client.FindIterable;
import com.mongodb.client.MongoCollection;
import com.mongodb.client.MongoDatabase;

import Models.Source.UserDetails;
import Models.Source.HistoryItem;
import org.bson.types.ObjectId;
import Services.Interface.IDBService;

public class DBService implements IDBService {

    private  final String DB_NAME = "InsulinApp"; //Database
    private  final String USERS_COL = "users"; //Table
    private  final String HISTORY_COL = "history";//Table
    private  final MongoClientURI connectionUri = new MongoClientURI(
            "mongodb://root:qwe123qwe@cluster0-shard-00-00-nxrrh.mongodb.net:27017," +
            "cluster0-shard-00-01-nxrrh.mongodb.net:27017," +
                    "cluster0-shard-00-02-nxrrh.mongodb.net:27017" +
                    "/test?ssl=true&replicaSet=Cluster0-shard-0&authSource=admin&retryWrites=true&w=
majority");
    private MongoClient mongoClient;
    private MongoDatabase database;
    private MongoCollection<Document> collectionUser;
    private MongoCollection<Document> collectionHistory;

    private void openDataBase(){
        mongoClient = new MongoClient(connectionUri);
        database = mongoClient.getDatabase(DB_NAME);
        collectionUser = database.getCollection(USERS_COL);
        collectionHistory = database.getCollection(HISTORY_COL);
    }

    private void closeDataBsae(){
        mongoClient.close();
    }

    public String getEmail(UserDetails userDetails) {
        openDataBase();
        FindIterable<Document> docs = collectionUser
                .find(and(eq("email", userDetails.getEmail()), eq("password", userDetails.getPass())));
        Document user = docs.first();
        closeDataBsae();
        //return user.getObjectId("email").toString();
        return userDetails.getEmail();
    }

    public List<HistoryItem> getUserHistory(String userId) {
        openDataBase();
        FindIterable<Document> docs = collectionHistory.find(eq("userId", userId));
        List<HistoryItem> history = new LinkedList<HistoryItem>();
        for (Document doc : docs) {
            history.add(new HistoryItem(doc.getString("BloodSugar"), doc.getString("CarbsAmount"),
                    doc.getString("InsulineLevel")));
        }
        closeDataBsae();
        return history;
```

```
    }

    public void addHistoryItem(String userId, HistoryItem item) {
        openDataBase();
        collectionHistory.insertOne(new Document("date", item.getDate()).append("BloodSugar", item.
getBloodSugar())
                .append("CarbsAmount", item.getCarbsAmount()).append("InsulineLevel", item.
getInsulinLevel())
                .append("userId", userId));
        closeDataBsae();
    }

    public boolean isUserExist(String username) {
        openDataBase();
        long count = collectionHistory.count(eq("userId", username));
        closeDataBsae();
        return count > 0;
    }

    public boolean isUserExist(UserDetails userDetails) {
        openDataBase();
        long count = collectionUser
                .count(and(eq("email", userDetails.getEmail()), eq("password", userDetails.getPass())));
        closeDataBsae();
        return count == 1;
    }

    public boolean addUser(UserDetails userDetails) {
        if (isUserExist(userDetails)) {
            return false;
        }
        openDataBase();
        collectionUser.insertOne(new Document("email", userDetails.getEmail()).append("password",
userDetails.getPass()).
                append("firstname",userDetails.getFirstName()).append("lastname",userDetails.getLastName
()).
                append("phone",userDetails.getPhone()));
        closeDataBsae();
        return true;
    }

    public String getName(String userId) {
        openDataBase();
        FindIterable<Document> docs = collectionUser.find(eq("email", userId));
        Document user = docs.first();
        closeDataBsae();
        return user.getString("firstname").toString()+" "+user.getString("lastname").toString();
    }
}
```

```java
package Services.Source;

import java.io.File;
import java.io.IOException;
import java.io.StringReader;
import java.util.Arrays;
import java.util.List;

import com.google.api.client.auth.oauth2.Credential;
import com.google.api.client.auth.oauth2.StoredCredential;
import com.google.api.client.extensions.java6.auth.oauth2.AuthorizationCodeInstalledApp;
import com.google.api.client.extensions.jetty.auth.oauth2.LocalServerReceiver;
import com.google.api.client.googleapis.auth.oauth2.GoogleAuthorizationCodeFlow;
import com.google.api.client.googleapis.auth.oauth2.GoogleClientSecrets;
import com.google.api.client.googleapis.javanet.GoogleNetHttpTransport;
import com.google.api.client.http.HttpTransport;
import com.google.api.client.json.JsonFactory;
import com.google.api.client.json.jackson2.JacksonFactory;
import com.google.api.client.util.store.FileDataStoreFactory;
import com.google.api.services.oauth2.Oauth2;
import com.google.api.services.oauth2.model.Tokeninfo;
import Models.Source.UserDetails;
import Services.Interface.IGoogleAuthenticator;

public class GoogleAuthenticator implements IGoogleAuthenticator {
    private final String APPLICATION_NAME = "InsulinApp";

    /** Directory to store user credentials. */
    private File DATA_STORE_DIR = new File(System.getProperty("/"), ".store/oauth2_sample");
    private final JsonFactory JSON_FACTORY = JacksonFactory.getDefaultInstance();
    private FileDataStoreFactory dataStoreFactory;
    private HttpTransport httpTransport;

    private final List<String> SCOPES = Arrays.asList("https://www.googleapis.com/auth/userinfo.profile",
            "https://www.googleapis.com/auth/userinfo.email");

    private Oauth2 oauth2;
    private GoogleClientSecrets clientSecrets;
    private String fname;
    private String lastName;
    private String email;
    private String phone;


    public UserDetails authenticate() {
        try {

            httpTransport = GoogleNetHttpTransport.newTrustedTransport();
            dataStoreFactory = new FileDataStoreFactory(DATA_STORE_DIR);
            // authorization
            Credential credential = authorize();
            // set up global Oauth2 instance
            oauth2 = new Oauth2.Builder(httpTransport, JSON_FACTORY, credential).setApplicationName(
APPLICATION_NAME).build();
            // run commands
            fname = oauth2.userinfo().get().execute().getGivenName();
            lastName = oauth2.userinfo().get().execute().getFamilyName();
            email = oauth2.userinfo().get().execute().getEmail();
            String pass = oauth2.userinfo().get().execute().getLocale();

            /**
             * delete te file to save the connection with google and App
             * Why?
             *      because We always want the user to choose which google account they want to connect
to
             */
            File file = new File(DATA_STORE_DIR, StoredCredential.DEFAULT_DATA_STORE_ID);
            if (file.exists()) {
                java.nio.file.Files.delete(file.toPath());
```

```
            }

            return new UserDetails(email,null,fname,lastName,phone);
            // success!

        } catch (IOException e) {
            return null;
        } catch (Throwable t) {
            return null;
        }
    }

    /** Authorizes the installed application to access user's protected data. */
     public Credential authorize() throws Exception {
        // load client secrets
         String json = "{\r\n" +
                               "  \"installed\": {\r\n" +
                                "    \"client_id\": \"497713811114-c5ac3i1tsar9f3aslmpkirje0ucsq4dg.apps.
googleusercontent.com\"," +
                                "\r\n" +
                                "    \"client_secret\": \"0KrqEjONEzgQYy1fqMF-yibw\"\r\n" +
                                   "  }\r\n" +
                         "}";
         clientSecrets = GoogleClientSecrets.load(JSON_FACTORY, new StringReader(json));
         if (clientSecrets.getDetails().getClientId().startsWith("Enter")
                 || clientSecrets.getDetails().getClientSecret().startsWith("Enter ")) {
             System.out.println("Enter Client ID and Secret from https://code.google.com/apis/console/ "
                     + "into oauth2-cmdline-sample/src/main/resources/client_secrets.json");
             System.exit(1);
         }
         // set up authorization code flow
         GoogleAuthorizationCodeFlow flow = new GoogleAuthorizationCodeFlow.Builder(httpTransport,
JSON_FACTORY,
                 clientSecrets, SCOPES).setDataStoreFactory(dataStoreFactory).build();
         // authorize
         return new AuthorizationCodeInstalledApp(flow, new LocalServerReceiver()).authorize("user");
    }

}
```

```java
package Services.Interface;

import Models.Source.UserDetails;
import Models.Source.HistoryItem;

import java.util.List;

public interface IDBService {

    /**
     * on this function -
     * call to function openDataBase for open insulin database.
     * find the user on database with email and password same email and password for this user (find the
user if is exits on database).
     * call to function closeDataBase for close insylin database.
     *
     * @param userDetails is the user (mean all details of user).
     * @return email for this user from database(MongoDB) if user is exits.
     */
    public String getEmail(UserDetails userDetails);

    /**
     * on this function -
     * call to function openDataBase for open insulin database.
     * find all list history for this user (user with param userId) and save all in list with type
HistoryItem.
     * call to function closeDataBase for close insylin database.
     *
     * @param userId is email/username of user.
     * @return list all history of tests for calculating the amount of insulin injection.
     */
    public List<HistoryItem> getUserHistory(String userId);

    /**
     * on this function -
     * call to function openDataBase for open insulin database.
     * SAVE Or ADD the Details with specific calculation on database.
     * call to function closeDataBase for close insylin database.
     *
     * @param userId is email/username of user.
     * @param item is Details with specific calculation.
     */
    public void addHistoryItem(String userId, HistoryItem item);

    /**
     * on this function -
     * call to function openDataBase for open insulin database.
     * this function counts how much user with username have in history collection
     * call to function closeDataBase for close insylin database.
     *
     * @param username of user
     * @return True if this user have histories on database else false
     */
    public boolean isUserExist(String username);

    /**
     * on this function -
     * call to function openDataBase for open insulin database.
     * it finds the user have username/email same on database.
     * it Chicks if this user exists on database
     * call to function closeDataBase for close insylin database.
     *
     * @param userDetails is the user (mean all details of user).
     * @return True if this user exits on database else false
     */
    public boolean isUserExist(UserDetails userDetails);

    /**
     * on this function -
```

```
     * call to function openDataBase for open insulin database.
     * SAVE Or ADD user on database.
     * call to function closeDataBase for close insylin database.
     *
     * @param userDetails is the user (all Details of user).
     * @return True if the user is added to a database or False if not (for most, the user with same
email/username exists).
     */
    public boolean addUser(UserDetails userDetails);

    /**
     * on this function -
     * call to function openDataBase for open insulin database.
     * find user with userId from a database if exits and return her name (firstName with Lastname)
     * call to function closeDataBase for close insylin database.
     *
     * @param userId is email/ username of user
     * @return fist Name + " " + last Name
     */
    public String getName(String userId);
}
```

```java
package Services.Interface;

import com.google.api.client.auth.oauth2.Credential;
import Models.Source.UserDetails;

public interface IGoogleAuthenticator {

    /**
     * on function -
     * Connection to google account
     *
     * @return all Details of user from google account
     */
    public UserDetails authenticate();

    /**
     * on function -
     * Authorizes the installed application to access user's protected data.
     *
     * @return Credential for connection with google
     * @throws Exception if have exception when connection with google
     */
    public Credential authorize() throws Exception;
}
```

```java
package Exceptions;

public class MyException extends Exception{
    public MyException() {
        super("Invalid Input");
    }
    public static class justNumbers extends NumberFormatException{
        public justNumbers(){
            super("You must only to request digits");
        }
    }
    public static class NegNumNotAllowed extends Exception {
        public NegNumNotAllowed(){
            super("must be number >=0");
        }
    }
}
```

```java
package ViewModels.Source;

import Models.Source.UserDetails;
import Services.Source.DBService;
import Services.Source.GoogleAuthenticator;
import ViewModels.Interface.IUserViewModel;
import Views.Source.UserView;

import org.apache.log4j.Logger;

public class UserViewModel implements IUserViewModel {

    private DBService dbService;
    final static Logger logger = Logger.getLogger(UserView.class);

    public UserViewModel() {
        this.dbService = new DBService();
    }

    public String doLogin(UserDetails userDetails) {
        logger.info("Trying to login");
        if (!dbService.isUserExist(userDetails)) {
            logger.error("User does not exist");
            return null;
        }

        String emailuser= dbService.getEmail(userDetails);
        logger.info("success! email=" + emailuser);
        return dbService.getName(emailuser);
    }

    public boolean doRegister(UserDetails userDetails) {
        logger.info("Registering with Email=" + userDetails.getEmail()
                + " Password=" + userDetails.getPass());
        return dbService.addUser(userDetails);
    }

    public UserDetails doLoginWithGoogle() {
        logger.info("Trying to login with google credentials");
        UserDetails  user = new GoogleAuthenticator().authenticate();

        return user;
    }
}
```

```java
package ViewModels.Source;

import Exceptions.MyException;
import Models.Source.UserDetails;
import Services.Source.DBService;
import ViewModels.Interface.IRegisterViewModel;
import org.apache.log4j.Logger;

public class RegisterViewModel implements IRegisterViewModel {
    private DBService dbService;
    final static Logger logger = Logger.getLogger(RegisterViewModel.class);

    public RegisterViewModel() {
        this.dbService = new DBService();
    }

    public boolean doRegister(UserDetails userDetails) throws MyException {
        logger.info("Registering with Email=" + userDetails.getEmail()
                + " Password=" + userDetails.getPass());
        boolean fNB = userDetails.getFirstName().isEmpty();
        boolean lNB = userDetails.getLastName().isEmpty();
        boolean passB = userDetails.getPass().isEmpty();
        boolean emailB = userDetails.getEmail().isEmpty();

        if (fNB || lNB || passB || emailB) {
            throw new MyException();
        }
        return dbService.addUser(userDetails);
    }

}
```

```java
package ViewModels.Source;

import java.util.List;

import Models.Source.HistoryItem;
import Services.Source.DBService;
import ViewModels.Interface.IViewHistoryViewModel;


public class ViewHistoryViewModel implements IViewHistoryViewModel {
    private DBService dbService;
    private String userId;

    public ViewHistoryViewModel(String userId) {
        this.dbService = new DBService();
        this.userId = userId;
    }
    public String getUserId(){return this.userId;}
    public List<HistoryItem> getHistory() {
        return this.dbService.getUserHistory(userId);
    }
}
```

```java
package ViewModels.Source;

import Exceptions.MyException;
import Models.Source.HistoryItem;
import Services.Source.DBService;
import ViewModels.Interface.ICalculateInulinViewModel;
import com.google.api.client.util.StringUtils;
import org.apache.log4j.Logger;

public class CalculateInsulinViewModel implements ICalculateInulinViewModel {
    private DBService dbService;
    private String userId;
    final static Logger logger = Logger.getLogger(CalculateInsulinViewModel.class);

    public CalculateInsulinViewModel(String userId) {
        this.dbService = new DBService();
        this.userId = userId;
    }


    public double calculate(double bloodSugar, double carbsAmount) throws MyException.NegNumNotAllowed {
        if (bloodSugar <= 0 || carbsAmount < 0) {
            throw new MyException.NegNumNotAllowed();
        }
            double res = (bloodSugar - 100) / 80 + carbsAmount / 15;
            logger.info("Adding history item to userId=" + userId);
            HistoryItem item = new HistoryItem(String.valueOf(bloodSugar), String.valueOf(carbsAmount),
String.valueOf(res));
            dbService.addHistoryItem(userId,item);
            return res;


    }

    @Override
    public String getUserId() {
        return this.userId;
    }

}
```

```java
package ViewModels.Interface;

import Models.Source.UserDetails;

public interface IUserViewModel {
    /**
     * on function -
     * first - checks if user with userDetails exists
     *          if not return null
     * second - if the user exists so save the id of user on value and return it
     *
     * @param userDetails is all Details of user
     * @return the id of the user in database if he exists or null if not
     */
    public String doLogin(UserDetails userDetails);

    /**
     *
     *
     * @return the details of user from her account in google
     */
    public UserDetails doLoginWithGoogle();
}
```

```java
package ViewModels.Interface;

import Exceptions.MyException;
import Models.Source.UserDetails;

public interface IRegisterViewModel {

    /**
     * on function -
     * First - it checks if there are any felds that need to be filled
     *      if the required fields are full so moves on all user to Database (mongoDB)
     *              return like a return addUser( in DBService class)
     *      else throws an error
     *
     * @param userDetails is user (all Details of user)
     * @return   True if the user is added to a database or False if not (for most, the user with same email/username exists).
     * @throws MyException if one or more details is Empty
     */
    public boolean doRegister(UserDetails userDetails) throws MyException;

}
```

```java
package ViewModels.Interface;

import Models.Source.HistoryItem;
import java.util.List;

public interface IViewHistoryViewModel {
    /**
     *
     * @return the userId of user
     */
    public String getUserId();

    /**
     *
     * @return return the list of history of user
     */
    public List<HistoryItem> getHistory();
}
```

```java
package ViewModels.Interface;

import Exceptions.MyException;

public interface ICalculateInulinViewModel {

    /**
     * on function -
     * it calculates how much insulin should be injected according to a fixed formula
     * a fixed formula is :
     *      ((bloodSugar - 100) / 80) + (carbsAmount / 15)
     * and save all this (bloodSugar,carbsAmount, resualt) and Date
     *
     * @param bloodSugar is amount of sugar in blood
     * @param carbsAmount is amount of carbohydrate in a meal
     * @return insulin level, the result from calculation
     * @throws MyException if the Sugar in blood <= 0 or carbsAmount < 0
     */
    public double calculate(double bloodSugar, double carbsAmount) throws MyException.NegNumNotAllowed,
Exception;

    /**
     *
     * @return
     */
    public String getUserId();
}
```