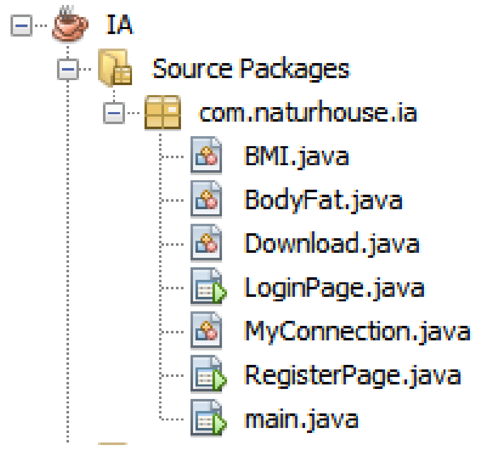
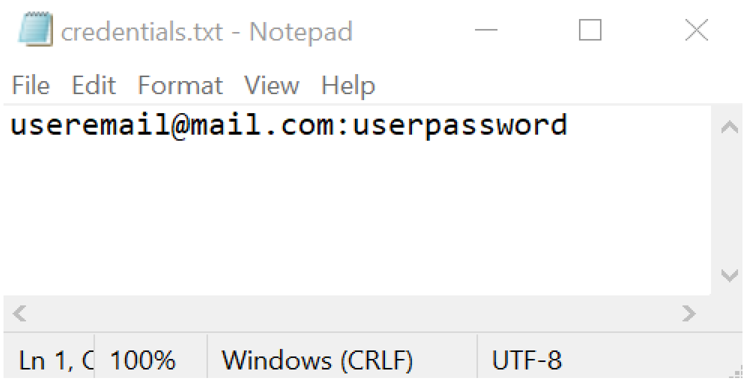
Criterion C: Development

All classes:


```
IA
  Source Packages
    com.naturhouse.ia
      BMI.java
      BodyFat.java
      Download.java
      LoginPage.java
      MyConnection.java
      RegisterPage.java
      main.java
```

Random Access Files:


credentials.txt - Notepad
File Edit Format View Help
Ln 1, C 100% Windows (CRLF) UTF-8

```
useremail@mail.com:userpassword
```

Functionality of each class and techniques used:

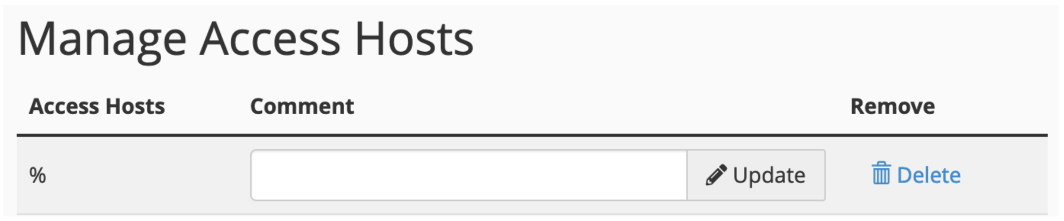
- LoginPage, RegisterPage and main use the inbuilt NetBeans GUI tool and contain the specific functions like Login, Register, CheckLogged or Update, which have files handling, loops, SQL queries, prepared statements, scanners, switch cases etc.
- BMI – contains all the formulas used to calculate the Body Mass Index.
- BodyFat – contains all the formulas used to calculate the Body Fat of the user.
- Download – contains algorithms to suggest the diet for cut or bulk from the database.
- MyConnection – contains the function to connect to the MySQL database and the password hashing SHA-256 function.

Program file:

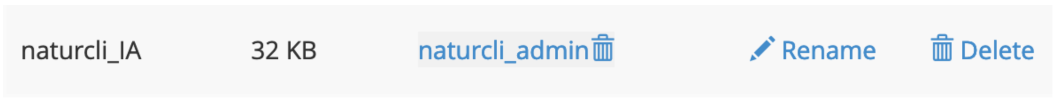
IA.jar
Type: Executable Jar File Size: 77.6 KB

The program was tested on Windows 10, Kali Linux and macOS Catalina. The program performed in all of the operating systems as expected. The MySQL database was hosted on a webserver running cPanel.

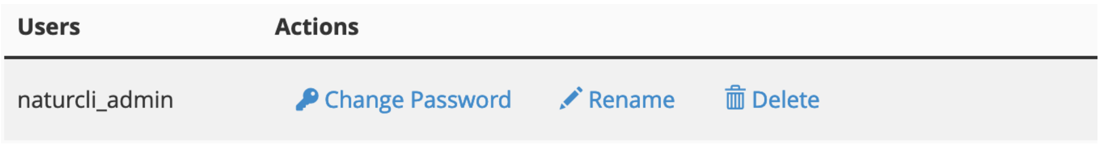
Connection to the server side:


Manage Access Hosts

Access Hosts	Comment	Remove
%	Update	Delete

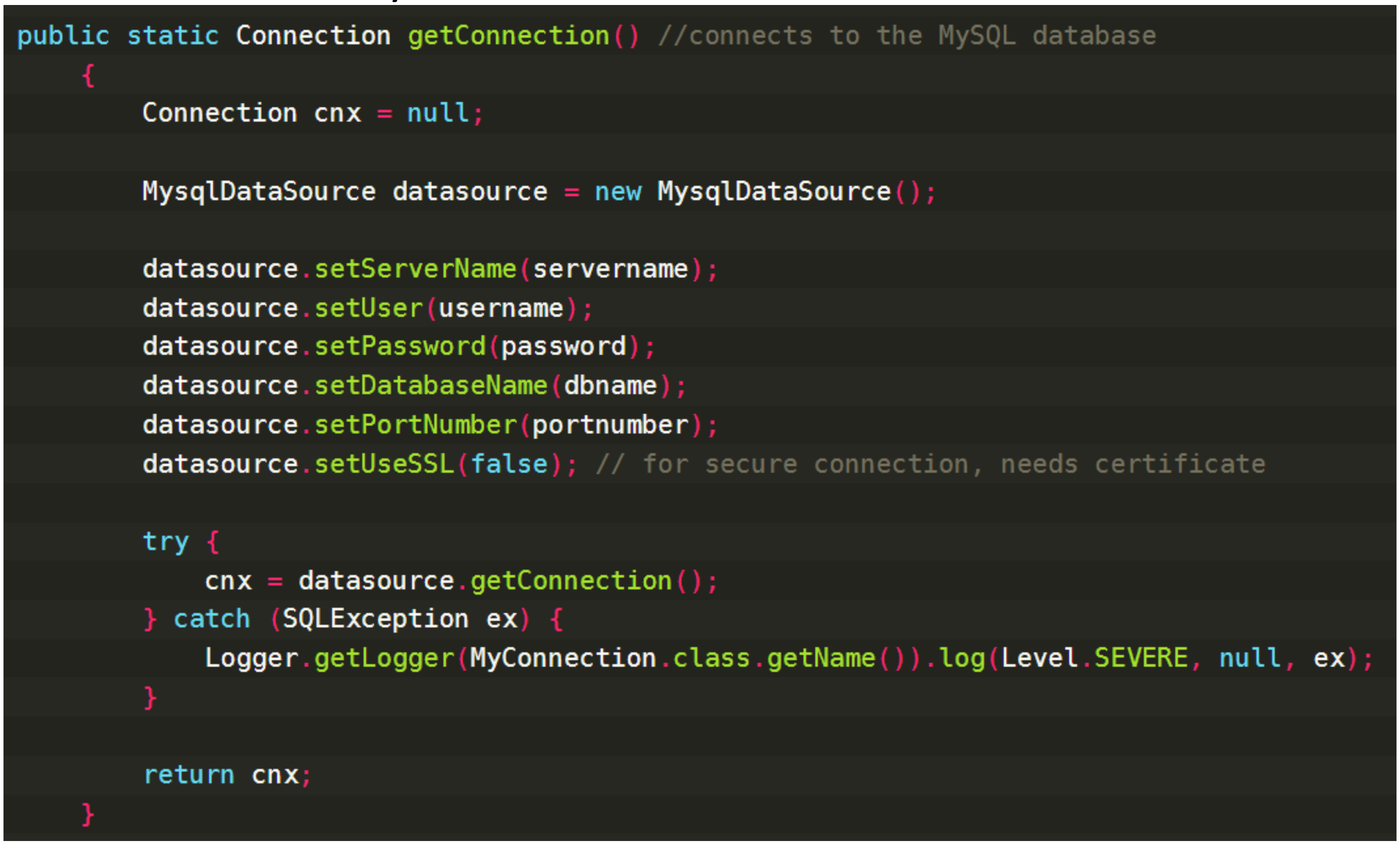


naturcli_IA	32 KB	naturcli_admin	Rename	Delete



Users	Actions
naturcli_admin	Change Password Rename Delete

Function to connect to the MySQL database:



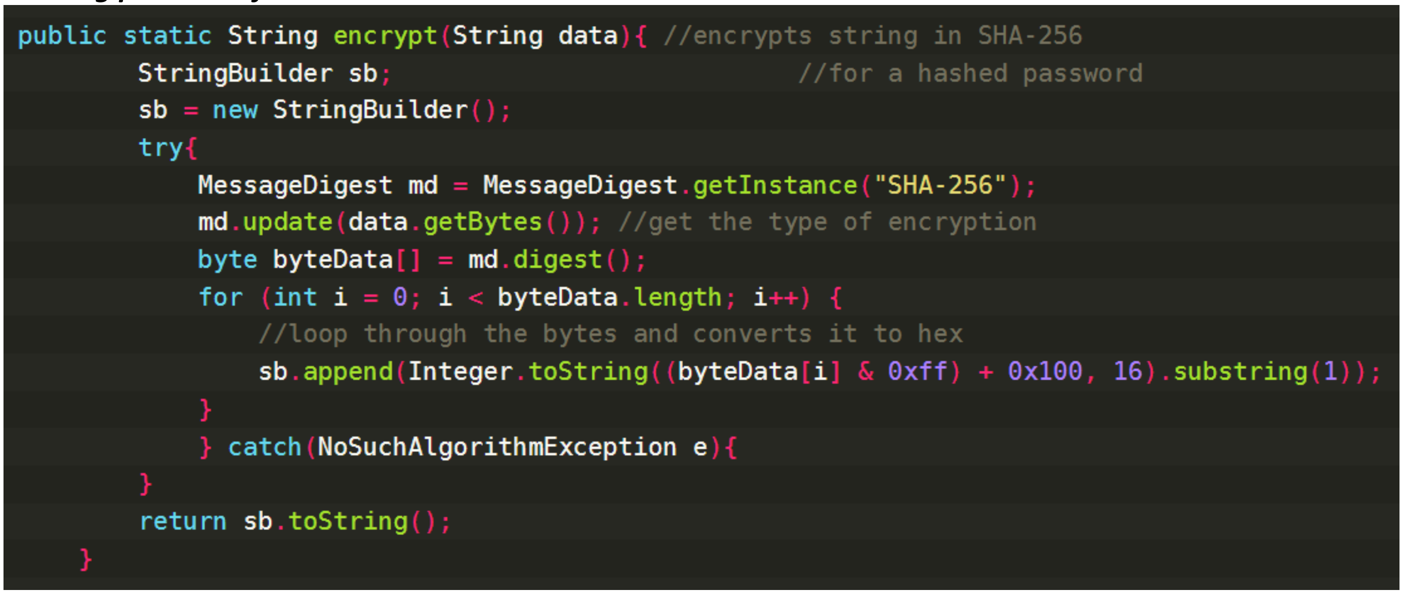
```java
public static Connection getConnection() //connects to the MySQL database
    {
        Connection cnx = null;

        MysqlDataSource datasource = new MysqlDataSource();

        datasource.setServerName(servername);
        datasource.setUser(username);
        datasource.setPassword(password);
        datasource.setDatabaseName(dbname);
        datasource.setPortNumber(portnumber);
        datasource.setUseSSL(false); // for secure connection, needs certificate

        try {
            cnx = datasource.getConnection();
        } catch (SQLException ex) {
            Logger.getLogger(MyConnection.class.getName()).log(Level.SEVERE, null, ex);
        }

        return cnx;
    }
```

I created a MySQLDataSource with the details used for login to the cPanel server phpMyAdmin (servername, username, password, dbname, portnumber), so I don't have to write this on every function where I connect to the database, I will just use getConnection().

All the SQL Exceptions are logged for easier debugging. The function has the ability to use an SSL certificate for encrypted communication between the client and server (datasource.setUseSSL(false)).

The technique implemented for the communication should make the program theoretically immune to the SQL injection as everything is saved as a string and then sent rather than having one query in the end, where it can be closed using an apostrophe.

Hashing password function:



```java
public static String encrypt(String data){ //encrypts string in SHA-256
        StringBuilder sb;                                //for a hashed password
        sb = new StringBuilder();
        try{
            MessageDigest md = MessageDigest.getInstance("SHA-256");
            md.update(data.getBytes()); //get the type of encryption
            byte byteData[] = md.digest();
            for (int i = 0; i < byteData.length; i++) {
                //loop through the bytes and converts it to hex
                sb.append(Integer.toString((byteData[i] & 0xff) + 0x100, 16).substring(1));
            }
            } catch(NoSuchAlgorithmException e){
        }
        return sb.toString();
    }
```

I am hashing password for security reasons. If the packets are interfered using a program like WireShark, then the attacker cannot understand the password.

MySQL database structure:

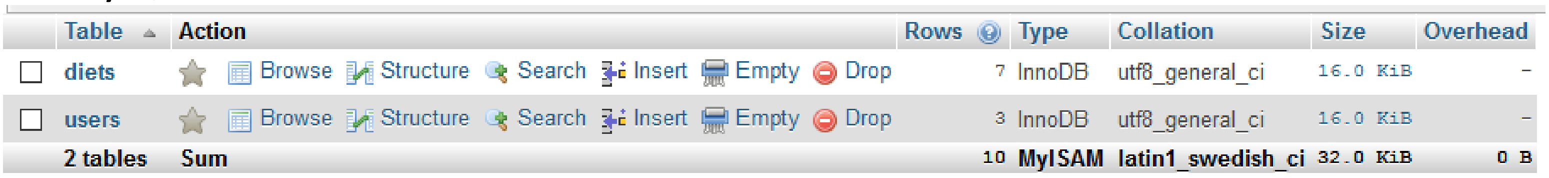
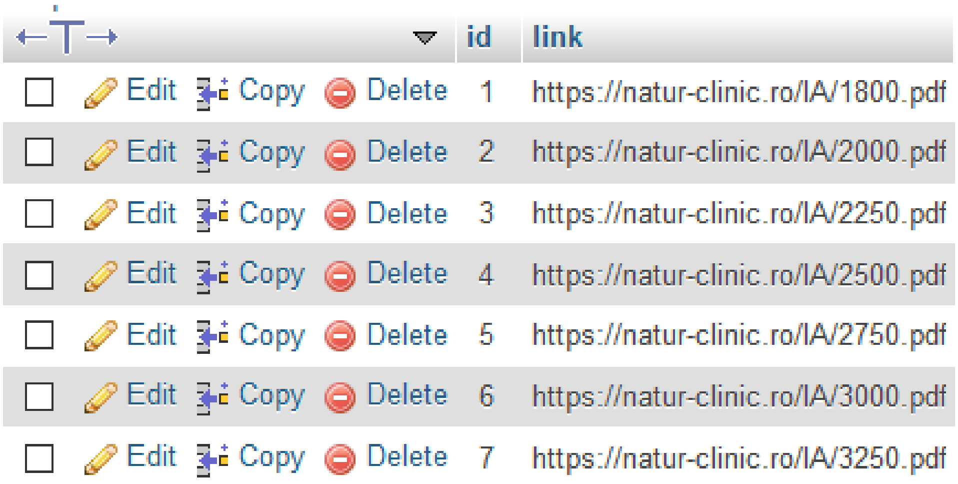


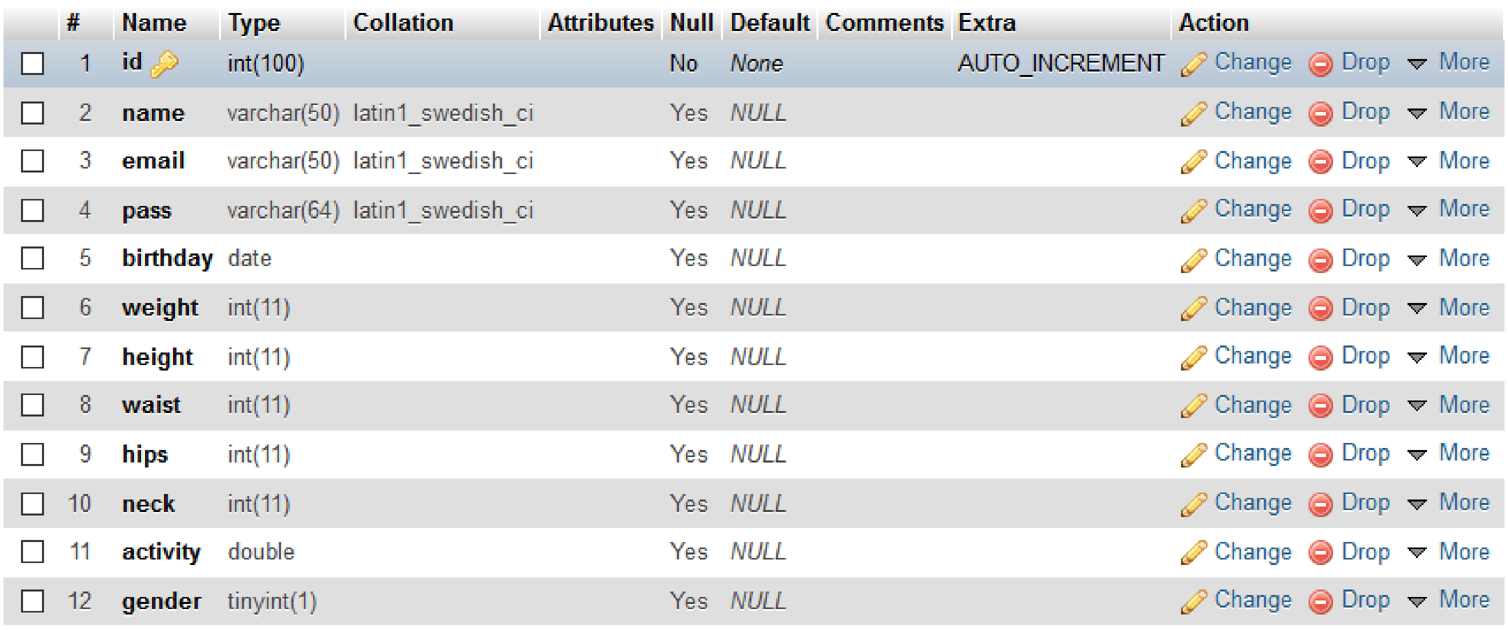
	Table	Action	Rows	Type	Collation	Size	Overhead
☐	diets	Browse Structure Search Insert Empty Drop	7	InnoDB	utf8_general_ci	16.0 KiB	-
☐	users	Browse Structure Search Insert Empty Drop	3	InnoDB	utf8_general_ci	16.0 KiB	-
	2 tables	Sum	10	MyISAM	latin1_swedish_ci	32.0 KiB	0 B

diets table:



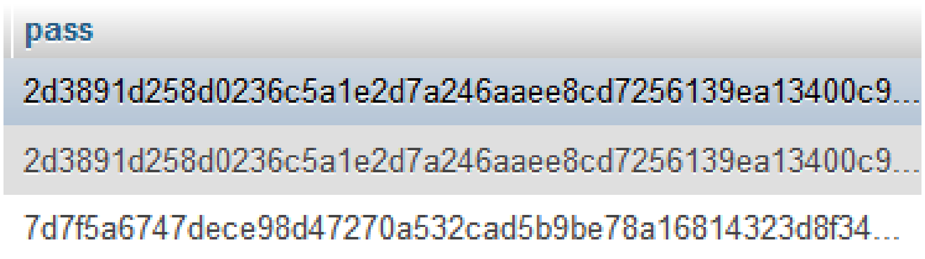
		id	link
☐	Edit Copy Delete	1	https://natur-clinic.ro/IA/1800.pdf
☐	Edit Copy Delete	2	https://natur-clinic.ro/IA/2000.pdf
☐	Edit Copy Delete	3	https://natur-clinic.ro/IA/2250.pdf
☐	Edit Copy Delete	4	https://natur-clinic.ro/IA/2500.pdf
☐	Edit Copy Delete	5	https://natur-clinic.ro/IA/2750.pdf
☐	Edit Copy Delete	6	https://natur-clinic.ro/IA/3000.pdf
☐	Edit Copy Delete	7	https://natur-clinic.ro/IA/3250.pdf

users table:



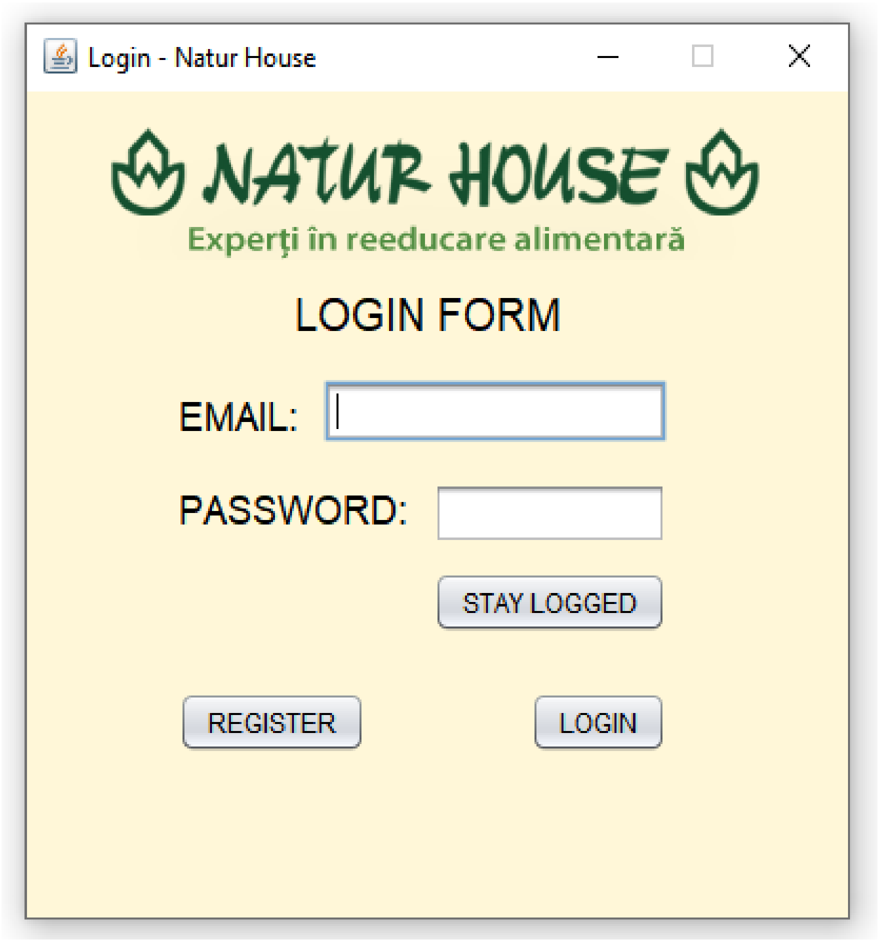
	#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra	Action
☐	1	**id**	int(100)			No	*None*		AUTO_INCREMENT	Change Drop More
☐	2	**name**	varchar(50)	latin1_swedish_ci		Yes	*NULL*			Change Drop More
☐	3	**email**	varchar(50)	latin1_swedish_ci		Yes	*NULL*			Change Drop More
☐	4	**pass**	varchar(64)	latin1_swedish_ci		Yes	*NULL*			Change Drop More
☐	5	**birthday**	date			Yes	*NULL*			Change Drop More
☐	6	**weight**	int(11)			Yes	*NULL*			Change Drop More
☐	7	**height**	int(11)			Yes	*NULL*			Change Drop More
☐	8	**waist**	int(11)			Yes	*NULL*			Change Drop More
☐	9	**hips**	int(11)			Yes	*NULL*			Change Drop More
☐	10	**neck**	int(11)			Yes	*NULL*			Change Drop More
☐	11	**activity**	double			Yes	*NULL*			Change Drop More
☐	12	**gender**	tinyint(1)			Yes	*NULL*			Change Drop More

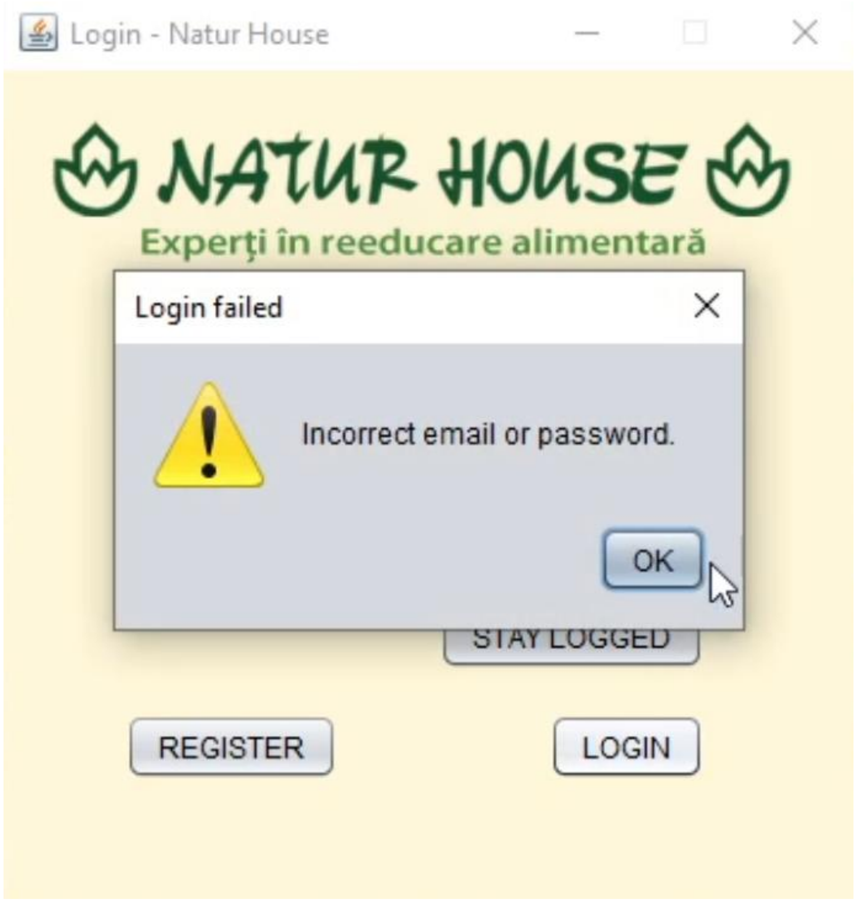
Passwords are encrypted in SHA-256:



pass
2d3891d258d0236c5a1e2d7a246aaee8cd7256139ea13400c9...
2d3891d258d0236c5a1e2d7a246aaee8cd7256139ea13400c9...
7d7f5a6747dece98d47270a532cad5b9be78a16814323d8f34...

LoginPage window:





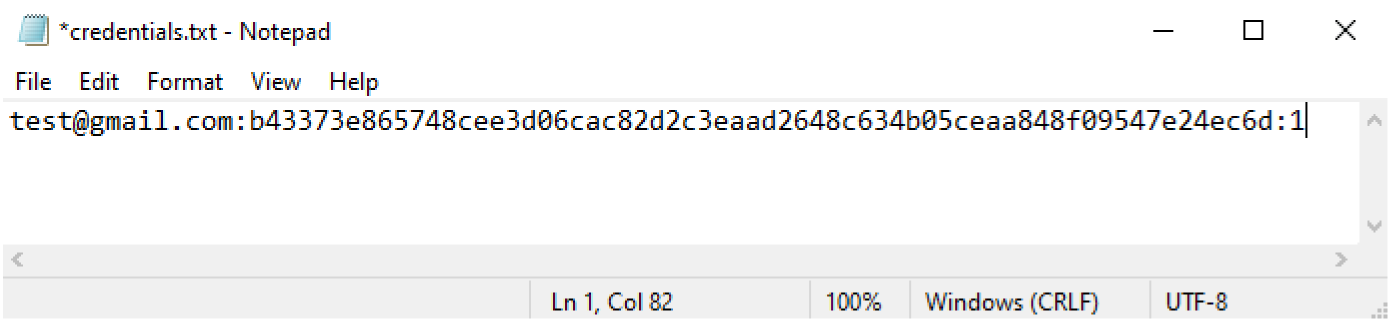
Incorrect email or password.

Login button function:

```java
private void LoginButtonActionPerformed(java.awt.event.ActionEvent evt) {
        // function to login to MySQL database
        PreparedStatement ps;
        ResultSet rs;
        String email = EmailLogTextField.getText();
        String pass = String.valueOf(PasswordLogTextField.getPassword());

        String query = "SELECT * FROM `users` WHERE `email` =? AND `pass` =?";

        try {
            pass = MyConnection.encrypt(pass);
            ps = MyConnection.getConnection().prepareStatement(query);

            ps.setString(1, email);
            ps.setString(2, pass);

            rs = ps.executeQuery();

            if(rs.next())
            {
                try (Writer writer = new BufferedWriter(new OutputStreamWriter(
                        new FileOutputStream("credentials.txt"), "utf-8"))) {
                    if (LoggedButton.isSelected()==true){ //if stay logged box
                                                          //is pressed
                        writer.write(email + ":" + pass +":1");
                    } else {
                        writer.write(email + ":" + pass + ":0");
                    }
                    Login();
                }catch (IOException ex) {
                    Logger.getLogger(LoginPage.class.getName()).log(Level.SEVERE, null, ex);
                }
            }
            else{
                JOptionPane.showMessageDialog(null, "Incorrect email or password.", "Login failed", 2);
            }

        } catch (SQLException ex) {
            Logger.getLogger(LoginPage.class.getName()).log(Level.SEVERE, null, ex);
        }


    }
```

After clicking the “LOGIN” button, the LoginButtonActionPerformed is performed. This starts by collecting the inputted information from the TextFields Email and Password into string, which are then added to the SQL query, which is send to the MySQL database for a response, if it comes back, then it is true. If the “STAY LOGGED” button is selected, then the credentials.txt file saves the email, password and 1, which shows that user wishes to save his credentials for the next time automatically login.

If it is not selected, then the saved 0 shows to the closing app button to remove all the saved credentials (although the information is secured anyway because of the hashed password). It uses Scanner for reading the file, saving the variable using the delimiter ";".

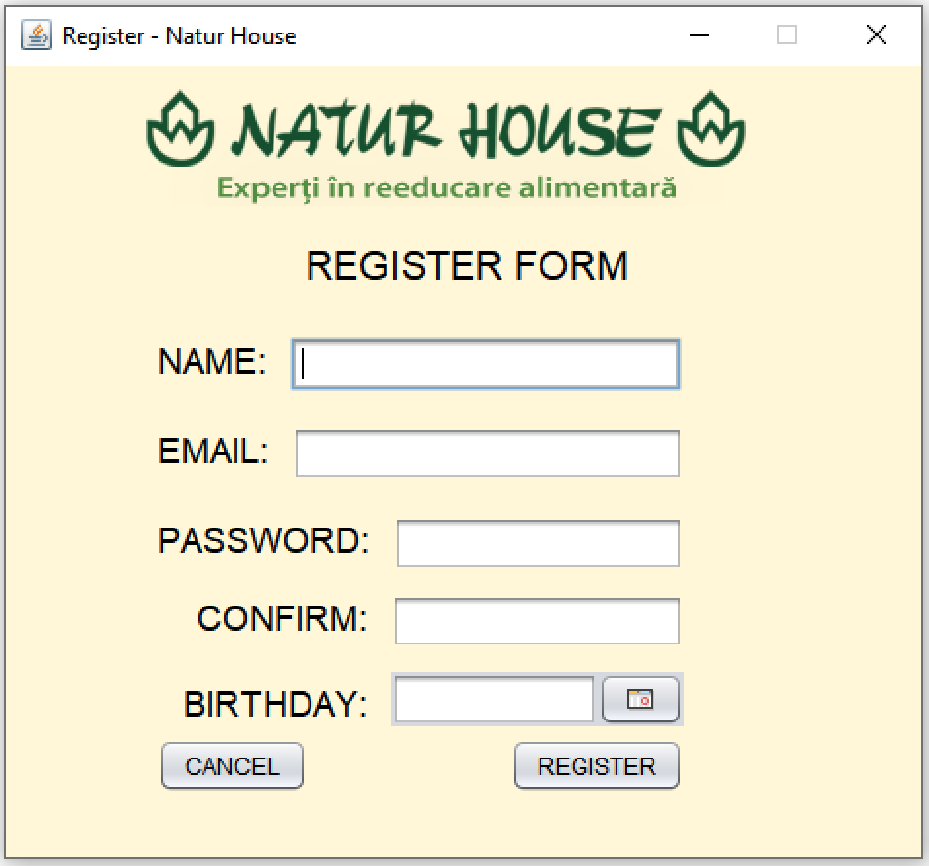


```
test@gmail.com:b43373e865748cee3d06cac82d2c3eaad2648c634b05ceaa848f09547e24ec6d:1
```

Check if the user chose "STAY LOGGED" function:

```java
public void checkLogged(){ // function to check if user chose to stay logged
        PreparedStatement ps;
        ResultSet rs;
        String Logged = null;
        String email = null;
        String pass = null;
        File f = new File("credentials.txt"); //open the file
        try {
            Scanner read = new Scanner(f);
            read.useDelimiter(":"); //save the data between : into below variable
            while(read.hasNext()){
               email = read.next();
               pass = read.next();
               Logged = read.next();
           }
        } catch (FileNotFoundException ex) {
            Logger.getLogger(main.class.getName()).log(Level.SEVERE, null, ex);
        }
        if ("1".equals(Logged)){ //if it is then login using the credentials
            try {                              //saved in the credentials.txt
                String query = "SELECT * FROM `users` WHERE `email` =? AND `pass` =?";
                ps = MyConnection.getConnection().prepareStatement(query);
                ps.setString(1, email);
                ps.setString(2, pass);
                rs = ps.executeQuery();
                if(rs.next())
                {
                    Login();
                }
            } catch (SQLException ex) {
                Logger.getLogger(LoginPage.class.getName()).log(Level.SEVERE, null, ex);
            }

        }
}
```

RegisterPage window:



Register - Natur House

NATUR HOUSE

Experţi în reeducare alimentară

REGISTER FORM

NAME:

EMAIL:

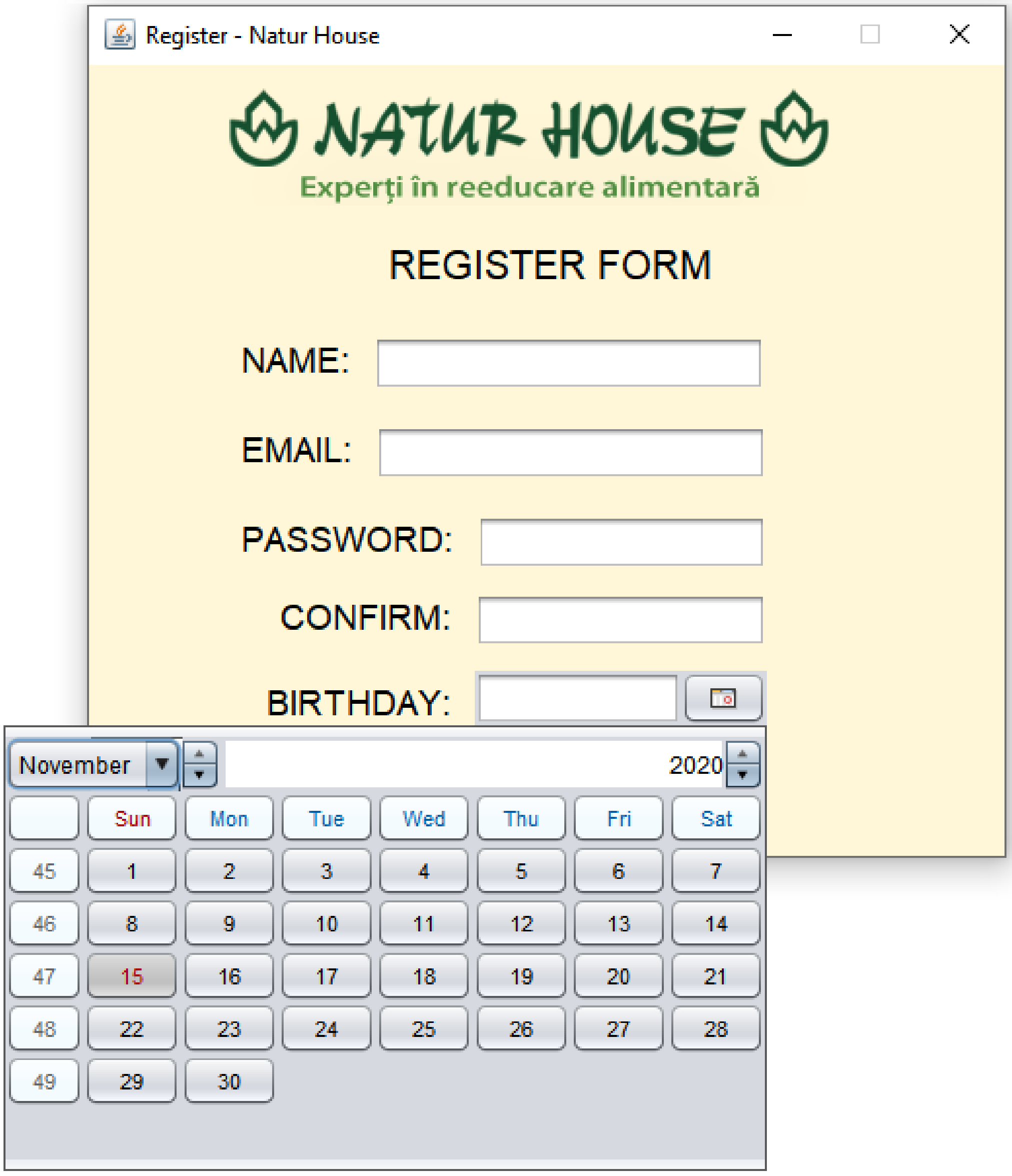
PASSWORD:

CONFIRM:

BIRTHDAY:

CANCEL

REGISTER



Register - Natur House

NATUR HOUSE

Experţi în reeducare alimentară

REGISTER FORM

NAME:

EMAIL:

PASSWORD:

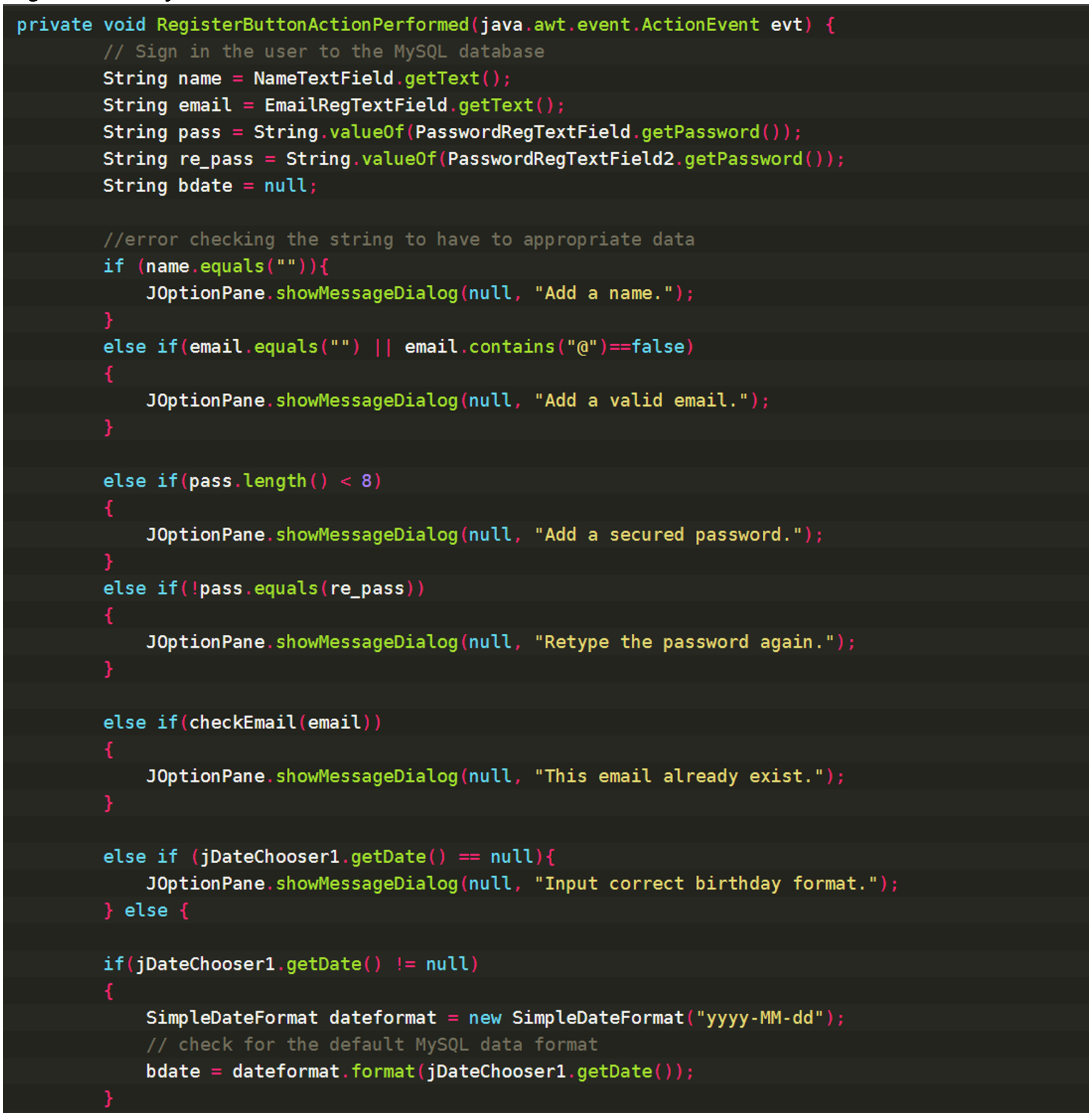
CONFIRM:

BIRTHDAY:

November 2020

	Sun	Mon	Tue	Wed	Thu	Fri	Sat
45	1	2	3	4	5	6	7
46	8	9	10	11	12	13	14
47	15	16	17	18	19	20	21
48	22	23	24	25	26	27	28
49	29	30					

Register button function:



```java
private void RegisterButtonActionPerformed(java.awt.event.ActionEvent evt) {
        // Sign in the user to the MySQL database
        String name = NameTextField.getText();
        String email = EmailRegTextField.getText();
        String pass = String.valueOf(PasswordRegTextField.getPassword());
        String re_pass = String.valueOf(PasswordRegTextField2.getPassword());
        String bdate = null;

        //error checking the string to have to appropriate data
        if (name.equals("")){
            JOptionPane.showMessageDialog(null, "Add a name.");
        }
        else if(email.equals("") || email.contains("@")==false)
        {
            JOptionPane.showMessageDialog(null, "Add a valid email.");
        }

        else if(pass.length() < 8)
        {
            JOptionPane.showMessageDialog(null, "Add a secured password.");
        }
        else if(!pass.equals(re_pass))
        {
            JOptionPane.showMessageDialog(null, "Retype the password again.");
        }

        else if(checkEmail(email))
        {
            JOptionPane.showMessageDialog(null, "This email already exist.");
        }

        else if (jDateChooser1.getDate() == null){
            JOptionPane.showMessageDialog(null, "Input correct birthday format.");
        } else {

        if(jDateChooser1.getDate() != null)
        {
            SimpleDateFormat dateformat = new SimpleDateFormat("yyyy-MM-dd");
            // check for the default MySQL data format
            bdate = dateformat.format(jDateChooser1.getDate());
        }
```

It saves as strings the data inputted from TextFields. Then checks that all the TextFields contain the appropriate information:

- For name to be inputted.
- For email checks that the string contains "@" and it is not associated with another account using the checkEmail function represented below.
- For password to be at least 8 characters, and that the password and its confirmation are equal.
- Birthday is saved in the correct format (if the user wishes to type it instead of using jDataChooser to graphically select it).

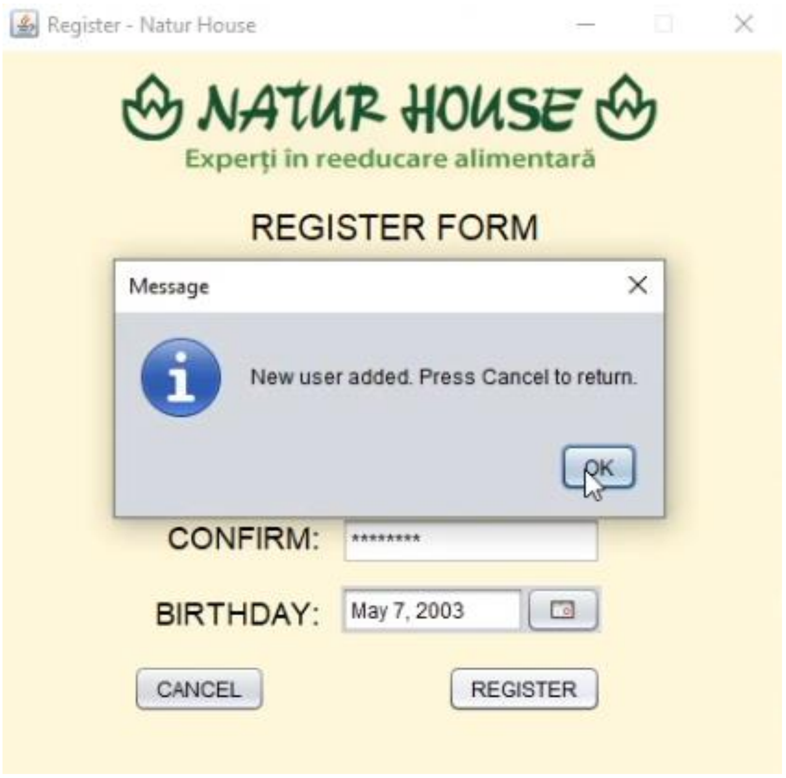
Register button function (continued):

```
        PreparedStatement ps;
        String query = "INSERT INTO `users`(`name`, `email`, `pass`, `birthday`) VALUES (?,?,?,?)";

        try {
            pass = MyConnection.encrypt(pass); //has password
            ps = MyConnection.getConnection().prepareStatement(query); //connect to db
            ps.setString(1, name);
            ps.setString(2, email);
            ps.setString(3, pass);

            if(bdate != null)
            {
                ps.setString(4, bdate);
            }else{
                ps.setNull(4, 0);
            }

            if(ps.executeUpdate() > 0)
            {
                JOptionPane.showMessageDialog(null, "New user added. Press Cancel to return.");
            }

        } catch (SQLException ex) {
            Logger.getLogger(RegisterPage.class.getName()).log(Level.SEVERE, null, ex);
        }
        }
    }
```

This connects to the MySQL database server, sending the variables: name, email, pass, birthday.

Password is encrypted beforehand using the encrypt() function. The program always encrypts when there a password input, so there is only one way to keep it secured.
The variables stored in local variables are sent using the Prepared Statement, so there is no need to send a separately string query for each variable or create a very long one. I consider this method more elegant.

If everything was successfully executed, it shows the message dialog “New user added”. At the end there is a catch for SQL Exception in case there is a problem. The user can later send the logs with the errors for support.


Register - Natur House

NATUR HOUSE

Experți în reeducare alimentară

REGISTER FORM

Message

New user added. Press Cancel to return.

OK

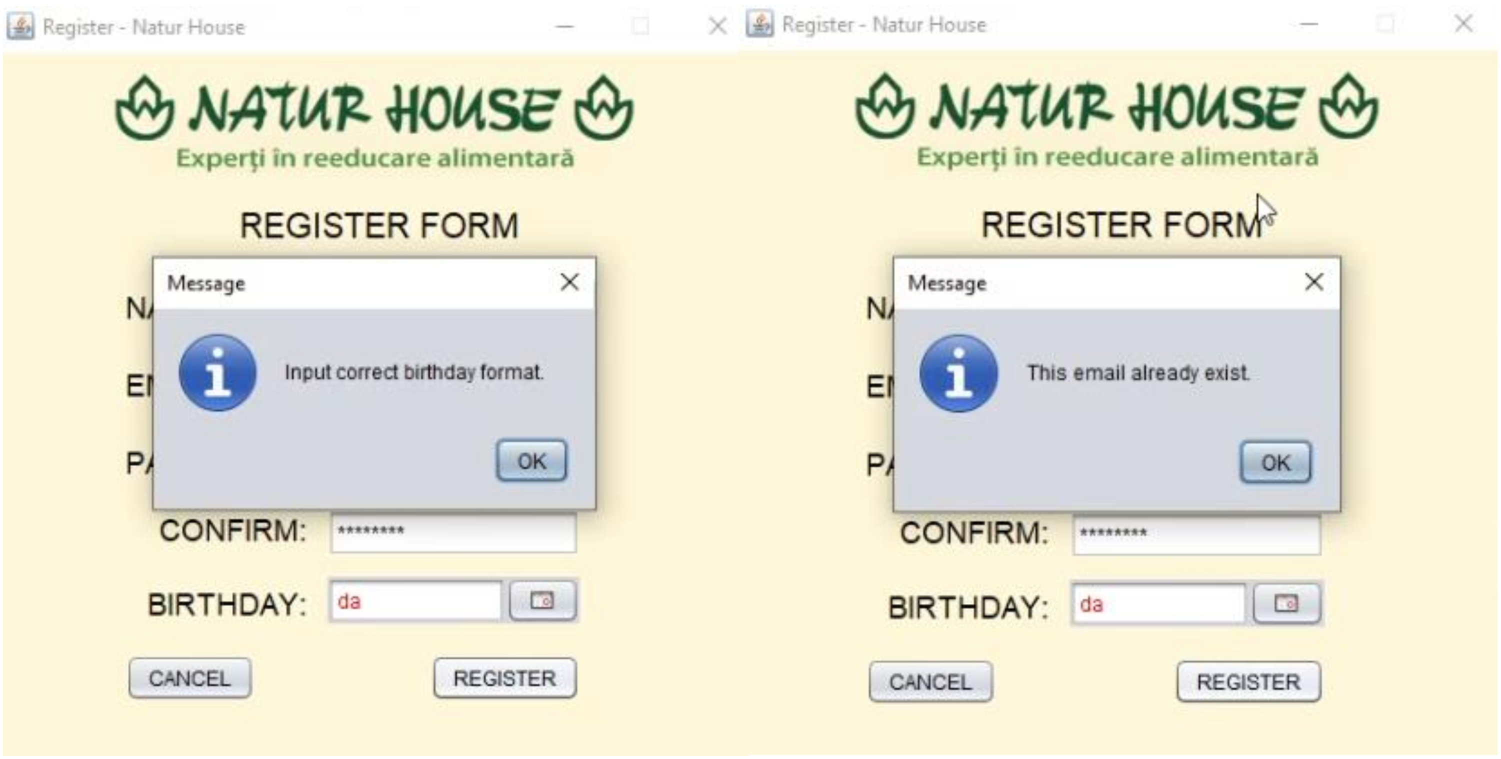
CONFIRM: ********

BIRTHDAY: May 7, 2003

CANCEL

REGISTER

All of these return alerts if they are not correct:


Input correct birthday format.

This email already exist.

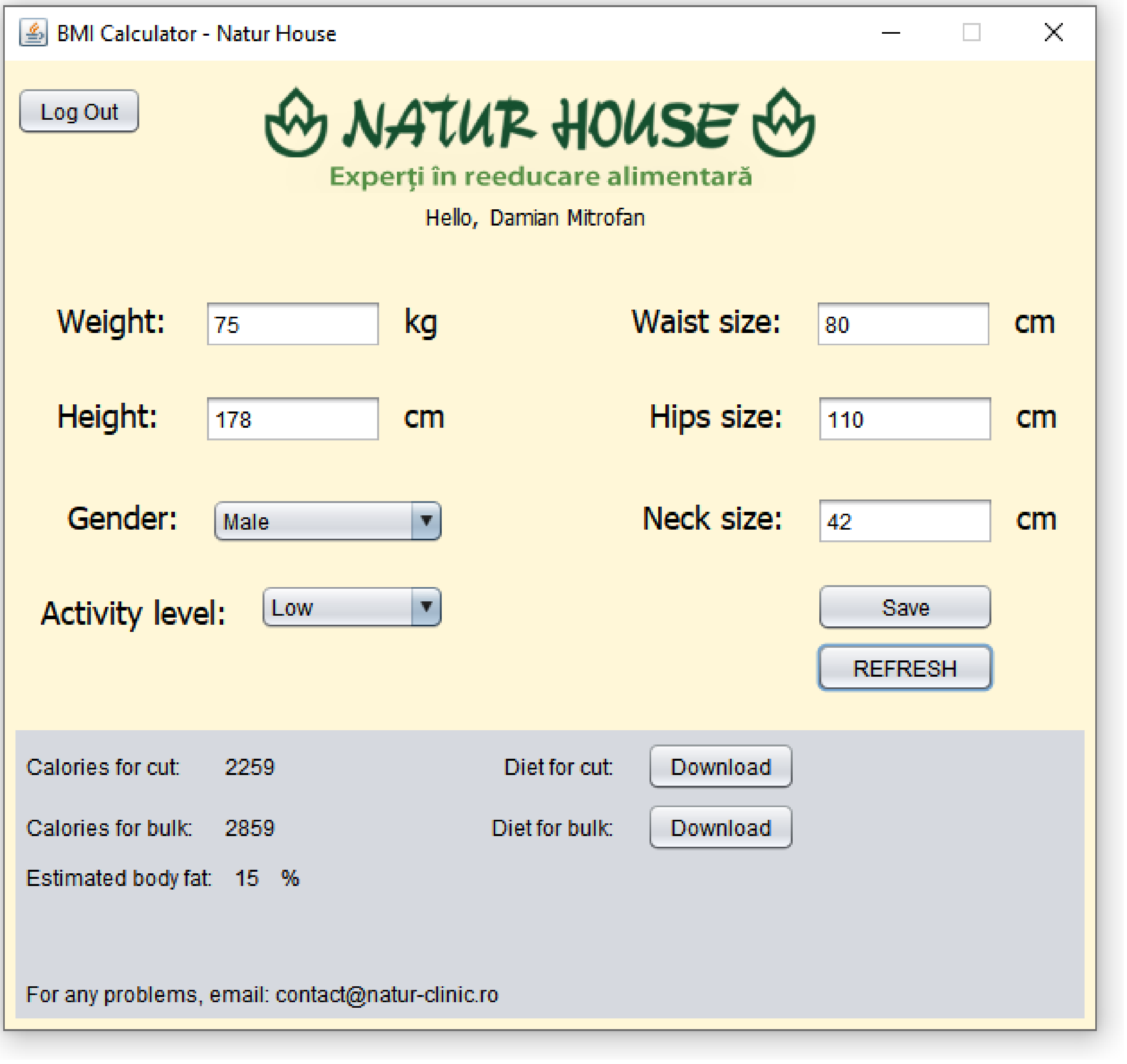
The logs text with catches of exceptions can be found in the same location as the Jar (app executable).

Check if email is already in used function:

```
public boolean checkEmail(String email) // function to check if the email is already in used.
    {
        PreparedStatement ps;
        ResultSet rs;
        boolean checkUser = false;
        String query = "SELECT * FROM `users` WHERE `email` =?";

        try {
            ps = MyConnection.getConnection().prepareStatement(query);
            ps.setString(1, email);

            rs = ps.executeQuery();

            if(rs.next()) // db returns only if there is something saved
            {
                checkUser = true; //therefore, already exist
            }
        } catch (SQLException ex) {
            Logger.getLogger(RegisterPage.class.getName()).log(Level.SEVERE, null, ex);
        }
         return checkUser;
    }
```

It checks it by looking for the email in the database, and if something comes up, then it returns true, otherwise does not enter the if and returns the pre-set checkUser = false.

main window:


BMI Calculator - Natur House

Log Out

NATUR HOUSE
Experți în reeducare alimentară

Hello, Damian Mitrofan

Weight: 75 kg

Height: 178 cm

Gender: Male

Activity level: Low

Waist size: 80 cm

Hips size: 110 cm

Neck size: 42 cm

Save

REFRESH

Calories for cut: 2259

Calories for bulk: 2859

Estimated body fat: 15 %

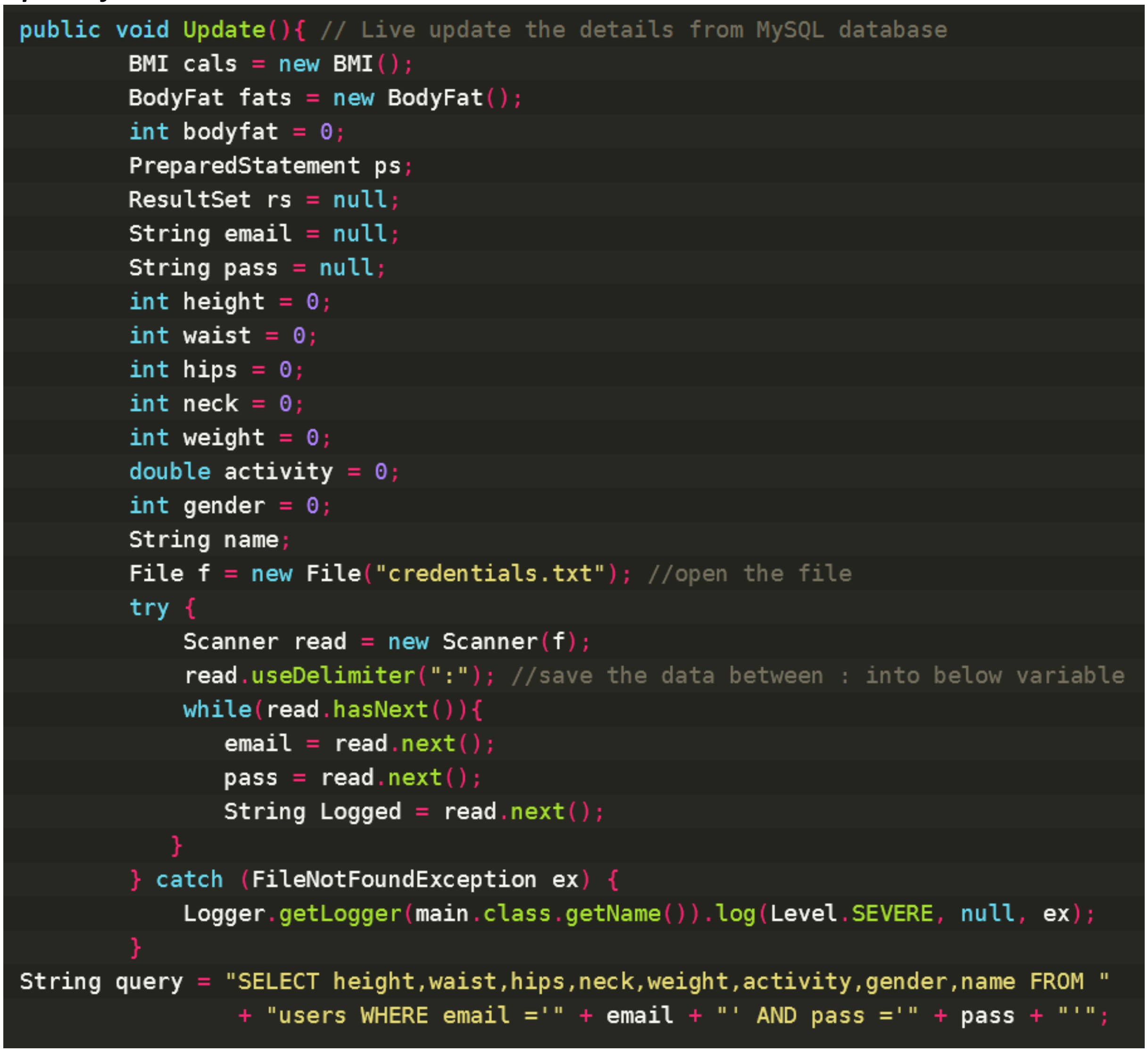
Diet for cut: Download

Diet for bulk: Download

For any problems, email: contact@natur-clinic.ro

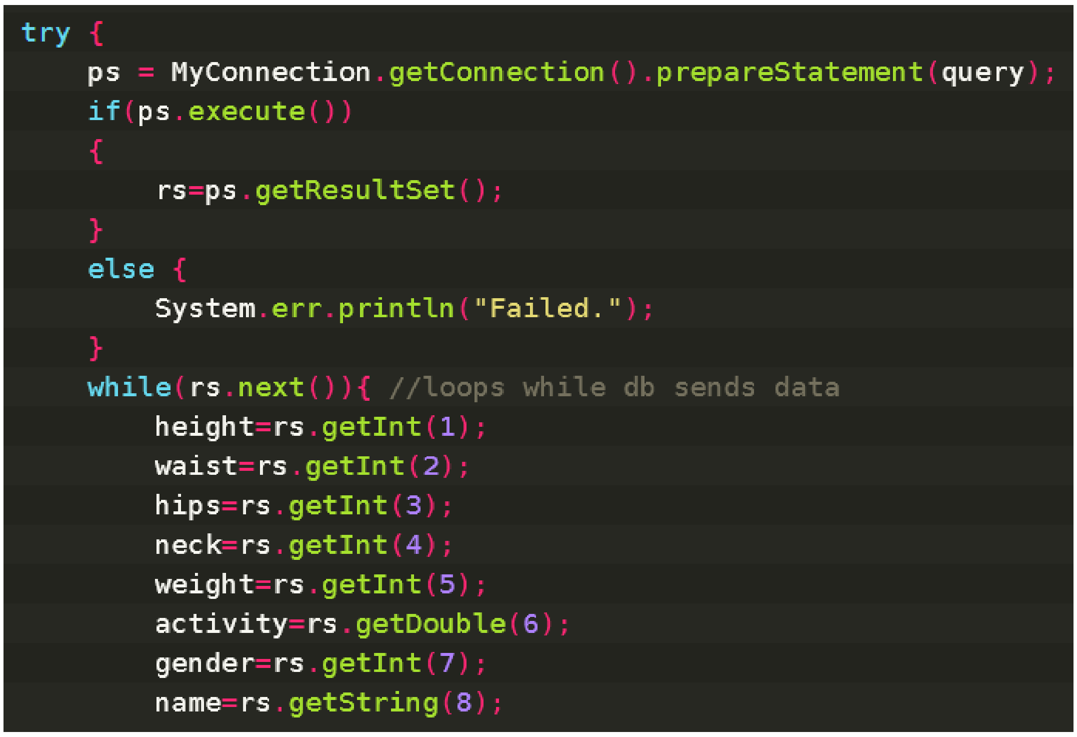
All the text, combo boxes and buttons are updated with the data from database using the below Update function. Every combo box is associated with a value that is used in BMI and BodyFat classes. The return is then output in the grey jPanelBox with calories and estimated bodyfat (the functions for these are explained at the end of this document). The “Download” buttons link to a function which opens the accordingly diet from the database. The administrator has the ability to add, change, edit or delete more diets from the MySQL database. As everything is updated live to the users, there is no delay or cache saved which might indicate previous data that is not relevant anymore. It also has error checking for all variables, so either there is a wrong input such as a String instead of an integer or detects inputs like 1kg body mass that does not make sense and the formula does not work properly.

Update function:



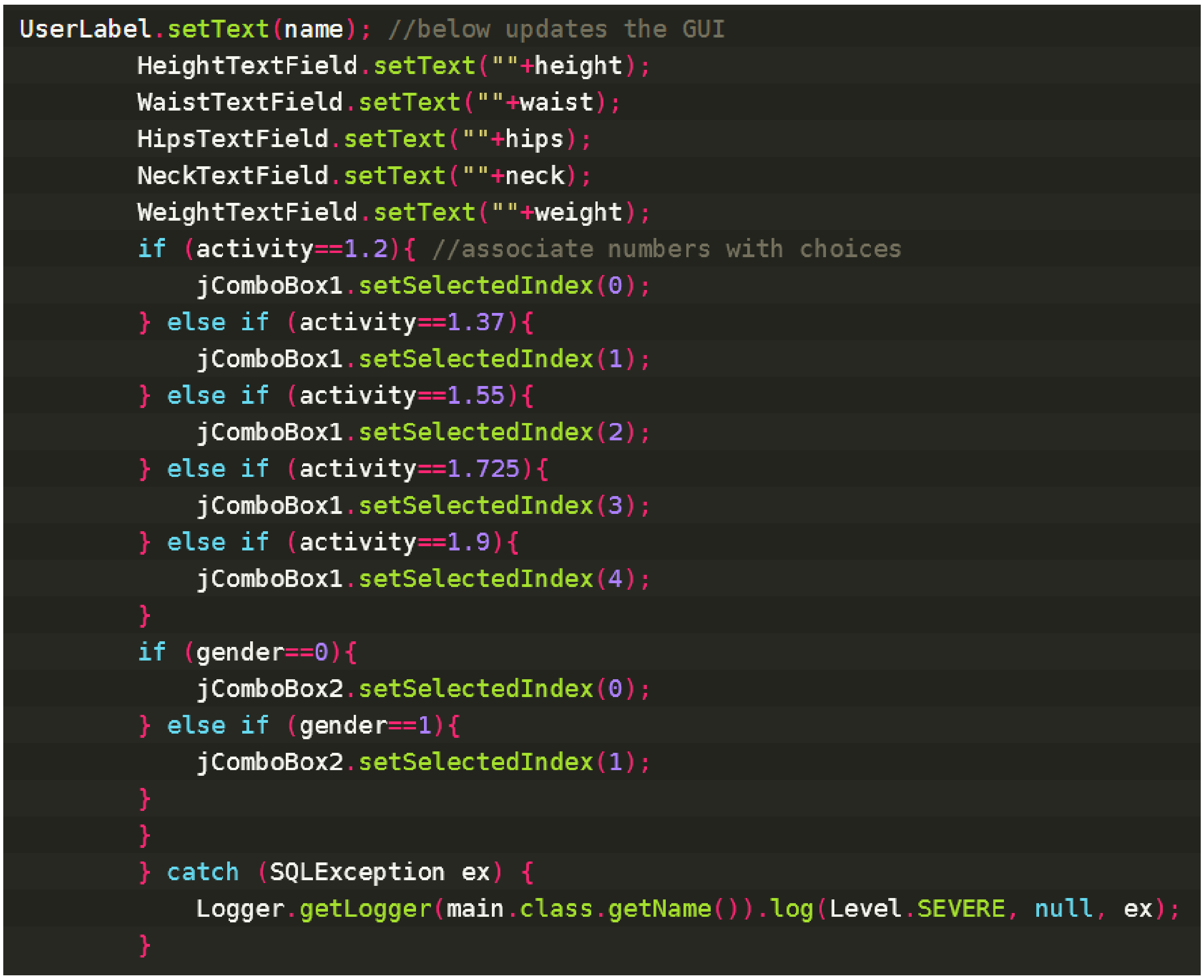
```java
public void Update(){ // Live update the details from MySQL database
        BMI cals = new BMI();
        BodyFat fats = new BodyFat();
        int bodyfat = 0;
        PreparedStatement ps;
        ResultSet rs = null;
        String email = null;
        String pass = null;
        int height = 0;
        int waist = 0;
        int hips = 0;
        int neck = 0;
        int weight = 0;
        double activity = 0;
        int gender = 0;
        String name;
        File f = new File("credentials.txt"); //open the file
        try {
            Scanner read = new Scanner(f);
            read.useDelimiter(":"); //save the data between : into below variable
            while(read.hasNext()){
               email = read.next();
               pass = read.next();
               String Logged = read.next();
           }
        } catch (FileNotFoundException ex) {
            Logger.getLogger(main.class.getName()).log(Level.SEVERE, null, ex);
        }
String query = "SELECT height,waist,hips,neck,weight,activity,gender,name FROM "
                + "users WHERE email ='" + email + "' AND pass ='" + pass + "'";
```

This reads the made file “credentials.txt” and continues to fetch from the database all the required data for the formulas, but only if the saved email and password combination is correct. I decided to this instead of cookies because besides that is more secure as none can just change the memory pattern to access the database, it also gives the flexibility to the user to move his account around from one computer to another.



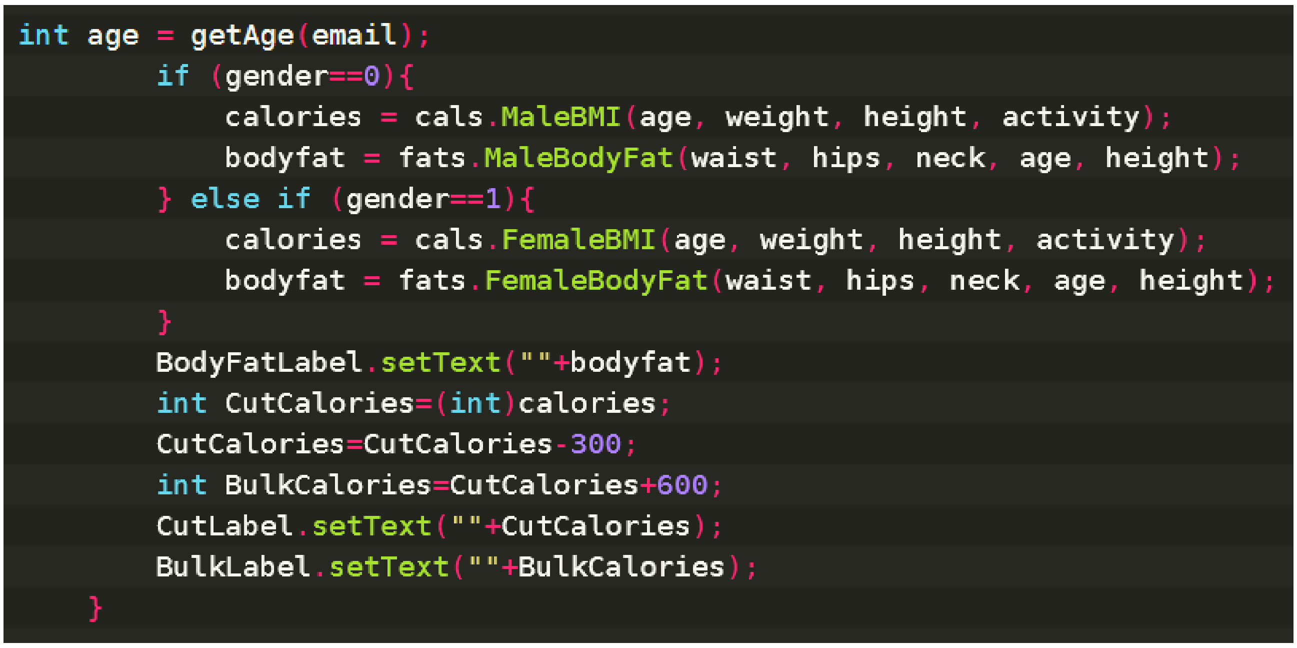
```java
try {
    ps = MyConnection.getConnection().prepareStatement(query);
    if(ps.execute())
    {
        rs=ps.getResultSet();
    }
    else {
        System.err.println("Failed.");
    }
    while(rs.next()){ //loops while db sends data
        height=rs.getInt(1);
        waist=rs.getInt(2);
        hips=rs.getInt(3);
        neck=rs.getInt(4);
        weight=rs.getInt(5);
        activity=rs.getDouble(6);
        gender=rs.getInt(7);
        name=rs.getString(8);
```

Connects to the database using the previous stated query, then it saves everything received into local variables, using the ResultSet Get method from MySQL-Connector-Java-5.1.49 library.



```java
UserLabel.setText(name); //below updates the GUI
        HeightTextField.setText(""+height);
        WaistTextField.setText(""+waist);
        HipsTextField.setText(""+hips);
        NeckTextField.setText(""+neck);
        WeightTextField.setText(""+weight);
        if (activity==1.2){ //associate numbers with choices
            jComboBox1.setSelectedIndex(0);
        } else if (activity==1.37){
            jComboBox1.setSelectedIndex(1);
        } else if (activity==1.55){
            jComboBox1.setSelectedIndex(2);
        } else if (activity==1.725){
            jComboBox1.setSelectedIndex(3);
        } else if (activity==1.9){
            jComboBox1.setSelectedIndex(4);
        }
        if (gender==0){
            jComboBox2.setSelectedIndex(0);
        } else if (gender==1){
            jComboBox2.setSelectedIndex(1);
        }
        }
        } catch (SQLException ex) {
            Logger.getLogger(main.class.getName()).log(Level.SEVERE, null, ex);
        }
```

Updates the GUI accordingly with the variable values from the database, for an easy way for the user to track the progress. It either update by using the “REFRESH” button or when the user logs in / changes any variable save, because otherwise the program might be too heavy on the CPU’s tasks priority.



```java
int age = getAge(email);
        if (gender==0){
            calories = cals.MaleBMI(age, weight, height, activity);
            bodyfat = fats.MaleBodyFat(waist, hips, neck, age, height);
        } else if (gender==1){
            calories = cals.FemaleBMI(age, weight, height, activity);
            bodyfat = fats.FemaleBodyFat(waist, hips, neck, age, height);
        }
        BodyFatLabel.setText(""+bodyfat);
        int CutCalories=(int)calories;
        CutCalories=CutCalories-300;
        int BulkCalories=CutCalories+600;
        CutLabel.setText(""+CutCalories);
        BulkLabel.setText(""+BulkCalories);
    }
```

Function to get the age:

```java
public int getAge(String email){ // Calculate the user's age from his birthday
        PreparedStatement ps;
        ResultSet rs = null;
        int age = 0;
        try {
        String query = "SELECT DATE_FORMAT(FROM_DAYS(DATEDIFF(CURDATE(),birthday)),"
                            + "'%Y')+0 AS age FROM users WHERE email='"+ email +"'";
                        // change date to days and then make the difference
                        // ending with converting it to years
        ps = MyConnection.getConnection().prepareStatement(query);
            if(ps.execute())
            {
                rs=ps.getResultSet();
            }
            else {
                System.err.println("Failed.");
            }
            while(rs.next()){
            age=rs.getInt(1);
            }
            } catch (SQLException ex) {
                Logger.getLogger(main.class.getName()).log(Level.SEVERE, null, ex);
            }
            return age;
    }
```

It uses SQL to get the date from database and then coverts it to days, for the difference from today's date also in days. Days are next converted in years, saved as integer to ignore the decimals as they are irrelevant for the formulas we are using.

Save button function:

```java
private void SaveButtonActionPerformed(java.awt.event.ActionEvent evt) {
        // Save details to MySQL database
        // get details from TextFields and save to local variables
        String weight = WeightTextField.getText();
        String height = HeightTextField.getText();
        String waist = WaistTextField.getText();
        String hips = HipsTextField.getText();
        String neck = NeckTextField.getText();
        String gender = jComboBox2.getSelectedItem().toString();
        String email = null;
        String pass = null;
        String activity = jComboBox1.getSelectedItem().toString();

        File f = new File("credentials.txt"); //open the file
        try {
            Scanner read = new Scanner(f);
            read.useDelimiter(":");//save the data between :
            while(read.hasNext()){     //into below variable
               email = read.next();
               pass = read.next();
               String Logged = read.next();
           }
        } catch (FileNotFoundException ex) {
            Logger.getLogger(main.class.getName()).log(Level.SEVERE, null, ex);
        }
```

```java
// switch case to associate strings with numbers
                // for the db
        if (null != activity)switch (activity) {
            case "None":
                activity = "1.2";
                break;
            case "Low":
                activity = "1.37";
                break;
            case "Medium":
                activity = "1.55";
                break;
            case "High":
                activity = "1.725";
                break;
            case "Performance":
                activity = "1.9";
                break;
            default:
                break;
        }

        if ("Male".equals(gender)){
            gender = "0";
        } else if ("Female".equals(gender)){
            gender = "1";
        }
```

Associate numbers with the keys from combo box, for the storage in database and later use for formulas.

```java
PreparedStatement ps;
        String query = "UPDATE users SET weight='"+ weight +"' , height='"
                + height +"' , waist='"+ waist +"' , hips='"+ hips +"' , neck='"
                + neck +"' , gender='"+ gender +"' , activity='" + activity
                + "' WHERE email='" + email +"' AND pass='" + pass + "'";
        try {
            ps = MyConnection.getConnection().prepareStatement(query);
            ps.executeUpdate();
            JOptionPane.showMessageDialog(null, "Details updated.");
        } catch (SQLException ex) {
            Logger.getLogger(main.class.getName()).log(Level.SEVERE, null, ex);
            JOptionPane.showMessageDialog(null, "Please check your input.");
            // error checking if the inputs are not accordingly
            // for example: inputing string into int
        }
            Update(); //refresh GUI
    }
```

Update the new data to the database, with error checking if the input is wrong. As the MySQL database is expecting to receive an integer, if the user inputs accidently a string, then it will throw an SQL exception, not updating the database and letting the user know that the input is wrong with a dialog alert.

Functions for calculating the calories for male and female:

```
public class BMI {
    public int MaleBMI(int Age, int Weight, int Height, double Activity){
        double calories= (66 + (13.7*Weight) + (5*Height) - (6.8*Age))*Activity;
        int Calories=(int)calories; //calculate calories for male
        return Calories;
    }
    public int FemaleBMI(int Age, int Weight, int Height, double Activity){
        double calories=(655 + (9.6*Weight) + (1.8*Height) - (4.7*Age))*Activity;
        int Calories=(int)calories; //calculate calories for male
        return Calories;
    }
}
```

The calories are calculated in double first, as the formula requires, but it is later converted it to integer as .x calories are not relevant.

Function to fetch the correct diet from calories inputted:

```
public void DietDownloadCut(int x){ //function to open the url
           try {   //according to the inputed calories, Cut version.
                    if (x < 1800){ //range of calories
                        x = 1;
                    } else if (x > 1800 | x < 2000){
                        x = 2;
                    } else if (x > 2000 | x < 2250){
                        x = 3;
                    } else if (x > 2250 | x < 2500){
                        x = 4;
                    } else if (x > 2500 | x < 2750){
                        x = 5;
                    } else if (x > 2750){
                        x = 6;
                    }
                String CutLink = null;
                PreparedStatement ps;
                ResultSet rs = null;
                String s=String.valueOf(x);
                String query = "SELECT link FROM diets WHERE id = '" + s + "'";
                ps = MyConnection.getConnection().prepareStatement(query);
                if(ps.execute())
                {
                    rs=ps.getResultSet(); //send the query to db
                }
                while(rs.next()){
                    CutLink = rs.getString(1); //receive from db once
                }
                OpenLink(CutLink); //pop the link in browser
            } catch (SQLException ex) {
                Logger.getLogger(Download.class.getName()).log(Level.SEVERE, null, ex);
            }
        }
```

The inputted calories are compared to pre-set calories ranges, then is associated with an ID, that has a website link. The link is then saved, and it uses the OpenLink function to open it to the OS's default browser.

```
public void DietDownloadBulk(int x){ //function to open the url
            try {  //according to the inputed calories, Bulk version.
                    if (x < 1800){ //range of calories
                        x = 1;
                    } else if (x > 1800 | x < 2000){
                        x = 2;
                    } else if (x > 2000 | x < 2250){
                        x = 3;
                    } else if (x > 2250 | x < 2500){
                        x = 4;
                    } else if (x > 2500 | x < 2750){
                        x = 5;
                    } else if (x > 2750){
                        x = 6;
                    }
                String BulkLink = null;
                PreparedStatement ps;
                ResultSet rs = null;
                x = x+1;
                String s=String.valueOf(x);
                String query = "SELECT link FROM diets WHERE id = '" + s + "'";
                ps = MyConnection.getConnection().prepareStatement(query);
                if(ps.execute())
                {
                    rs=ps.getResultSet(); //send the query to db
                }
                while(rs.next()){
                    BulkLink = rs.getString(1); //receive from db once
                }
                OpenLink(BulkLink); //pop the link in browser
            } catch (SQLException ex) {
                Logger.getLogger(Download.class.getName()).log(Level.SEVERE, null, ex);
            }
    }
```

Open the website link on browser:

```
public void OpenLink(String link){
    try { //open the link in the OS's default browser
                Desktop.getDesktop().browse(new URL(link).toURI());
                } catch (IOException | URISyntaxException e) {
            }
        }
```

The library used for this function:

- java.net.URL
- import java.awt.Desktop
- import java.net.URISyntaxException

Functions to calculate the body fat:

```
public class BodyFat {
    public int MaleBodyFat(int Waist, int Hips, int Neck, int Age, int Height){
        double bodyfat=(495/(1.0324-0.19077*(log(Waist-Neck))+0.15456*(log(Height))))-450;
        //calculate body fat percentage for male
        int BodyFat=(int)bodyfat; //convert double to int
        return BodyFat;
    }
    public int FemaleBodyFat(int Waist, int Hips, int Neck, int Age, int Height){
        double bodyfat=(495/(1.29579-0.35004*(log(Waist+Hips-Neck))+0.22100*(log(Height))))-450;
        //calculate body fat percentage for female
        int BodyFat=(int)bodyfat; //convert double to int
        return BodyFat;
    }
}
```

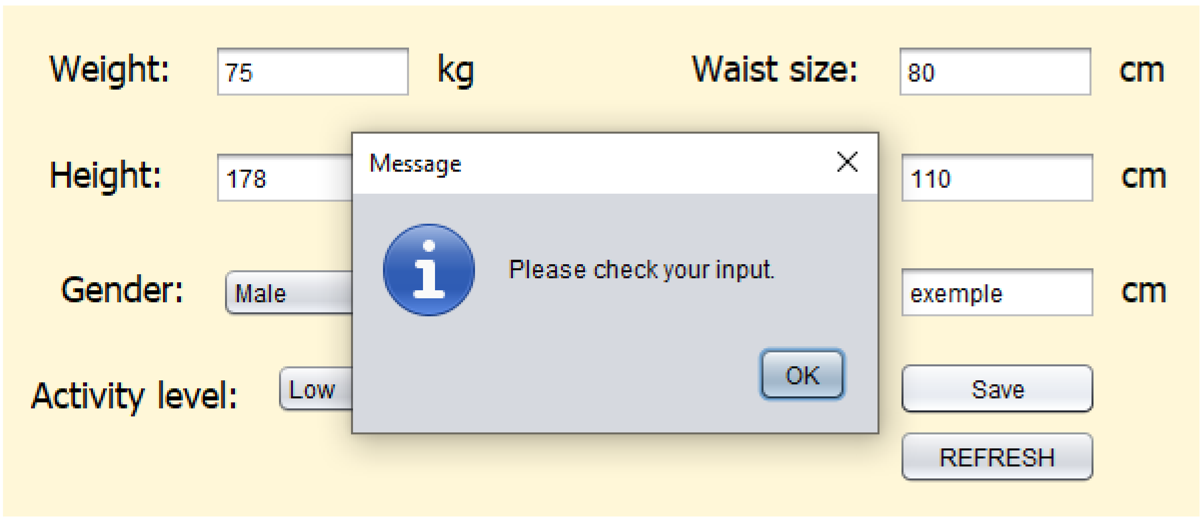
The formula gives the result as a double, but we need an integer. Mrs. Sochirca said that we should ignore the minus before the result. Therefore, the function does the absolute of the number to remove it.

Pages are changed with this:

```
private void RegisterPageButtonMouseClicked(java.awt.event.MouseEvent evt) {
        // open the register page while closing the login one
        RegisterPage rgf = new RegisterPage();
        rgf.setVisible(true);
        rgf.pack();
        rgf.setLocationRelativeTo(null);
        rgf.setDefaultCloseOperation(LoginPage.EXIT_ON_CLOSE);
        this.dispose();
    }
```

It sets the new page visible, in the same location as the one on its location, then it closes the previous page. By doing this, it processes the next page in advance, offering the user almost instantly loading times.

There is error checking for the main page where user is supposed to enter his measurements. For example, if you input a string but the expect is an integer, then SQL will timeout giving the alert to recheck your inputs.

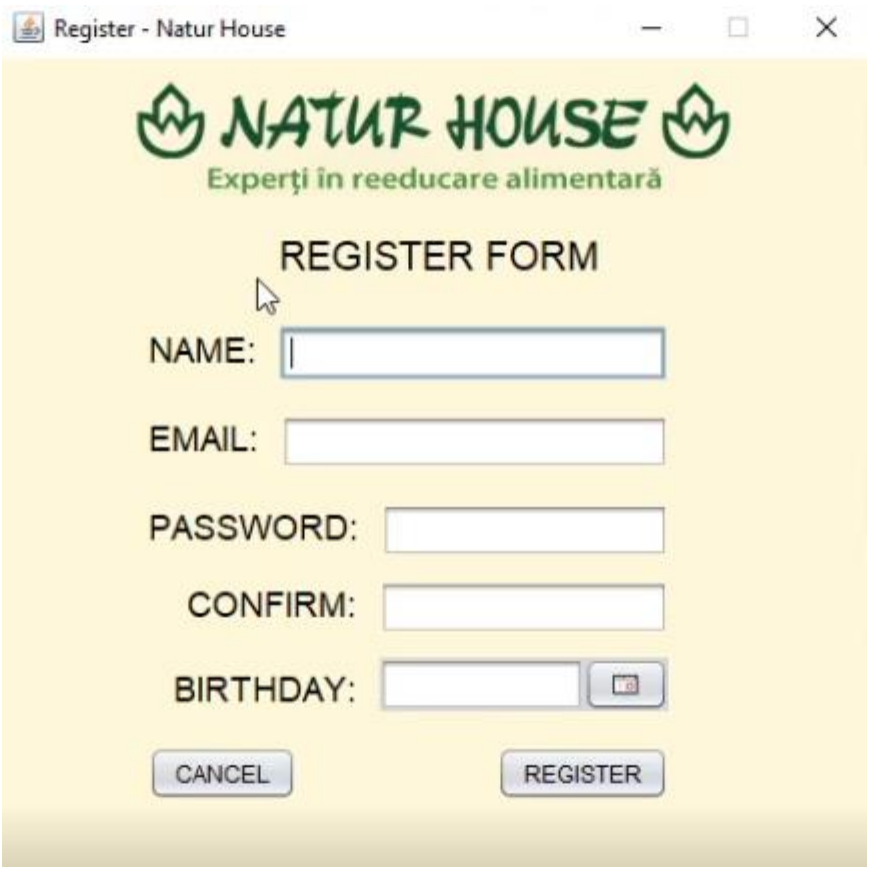


Weight: 75 kg
Waist size: 80 cm
Height: 178
110 cm
Gender: Male
exemple cm
Activity level: Low

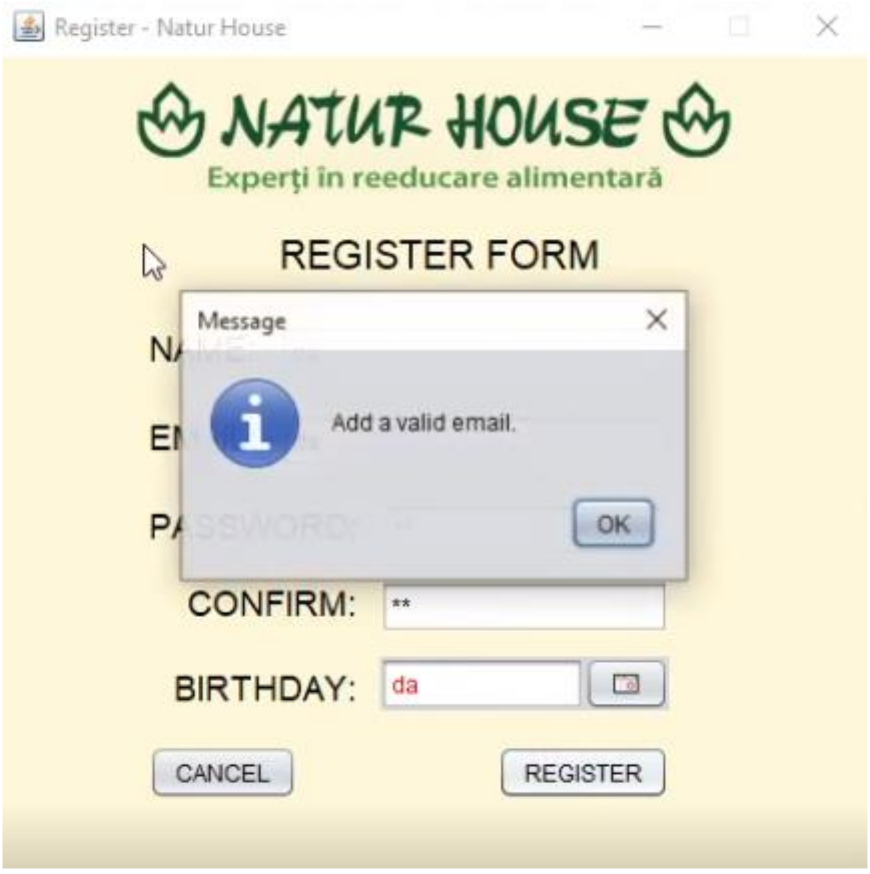
Message

Please check your input.

The walkthrough of the program:



Register form page

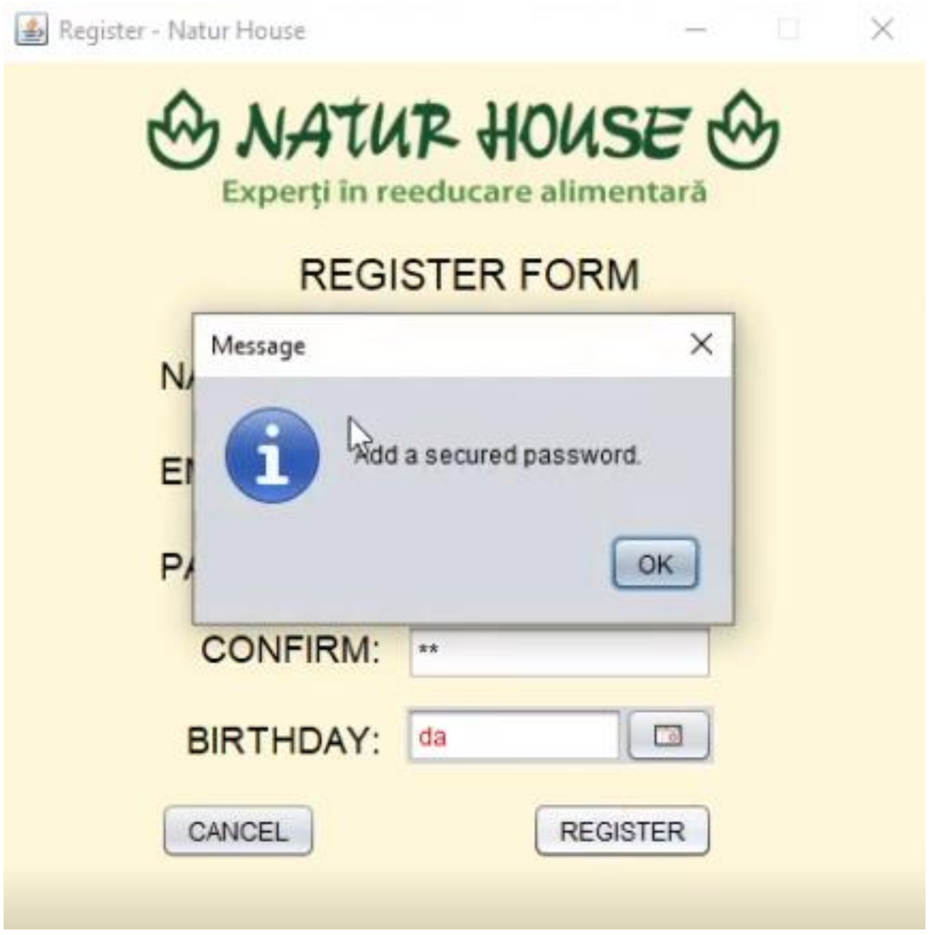


Add a valid email.

CONFIRM: **

BIRTHDAY: da

Register form page – valid email checking.


Register - Natur House

REGISTER FORM

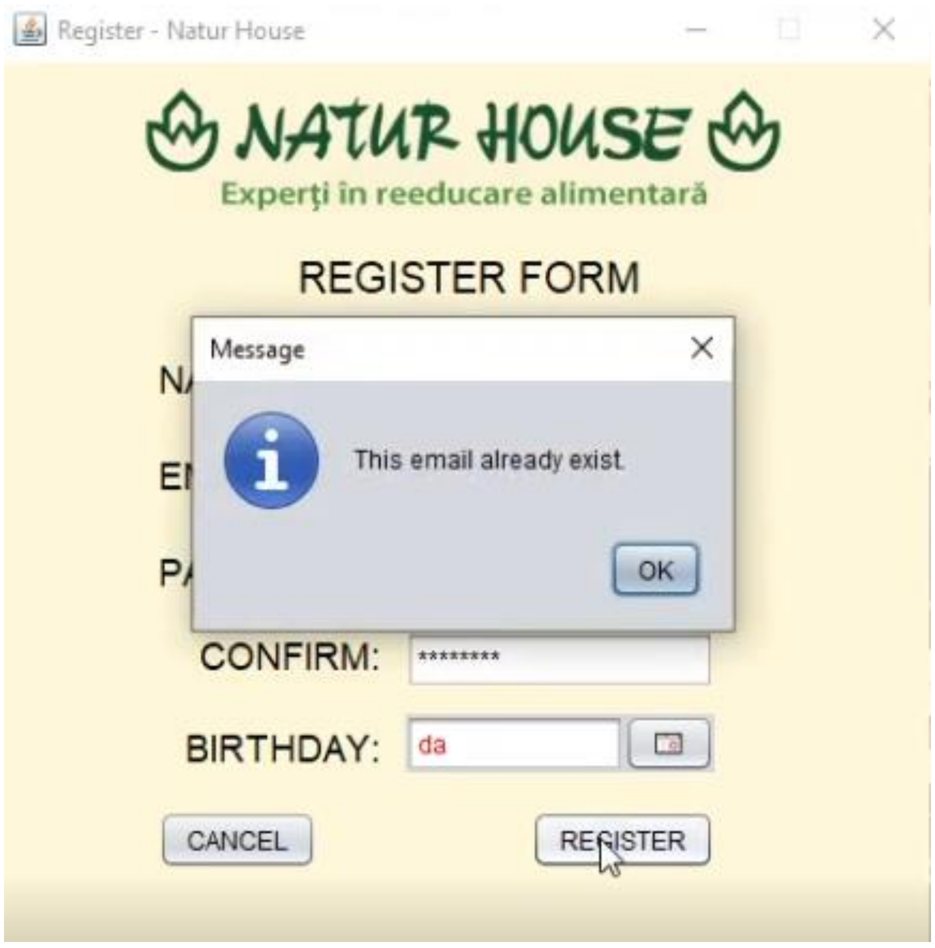
Message

Add a secured password.

CONFIRM: **

BIRTHDAY: da

Register form page – secured password checking.


Register - Natur House

REGISTER FORM

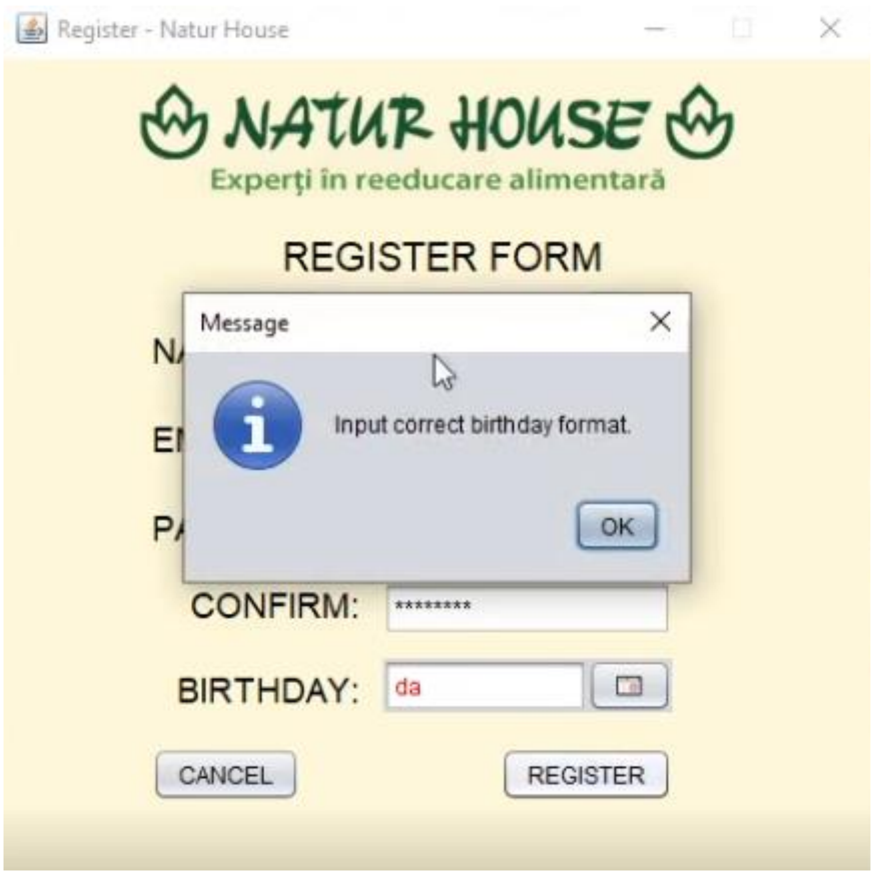
Message

This email already exist.

CONFIRM: ********

BIRTHDAY: da

Register form page – email already exist checking.


Register - Natur House

NATUR HOUSE
Experţi în reeducare alimentară

REGISTER FORM

Message

Input correct birthday format.

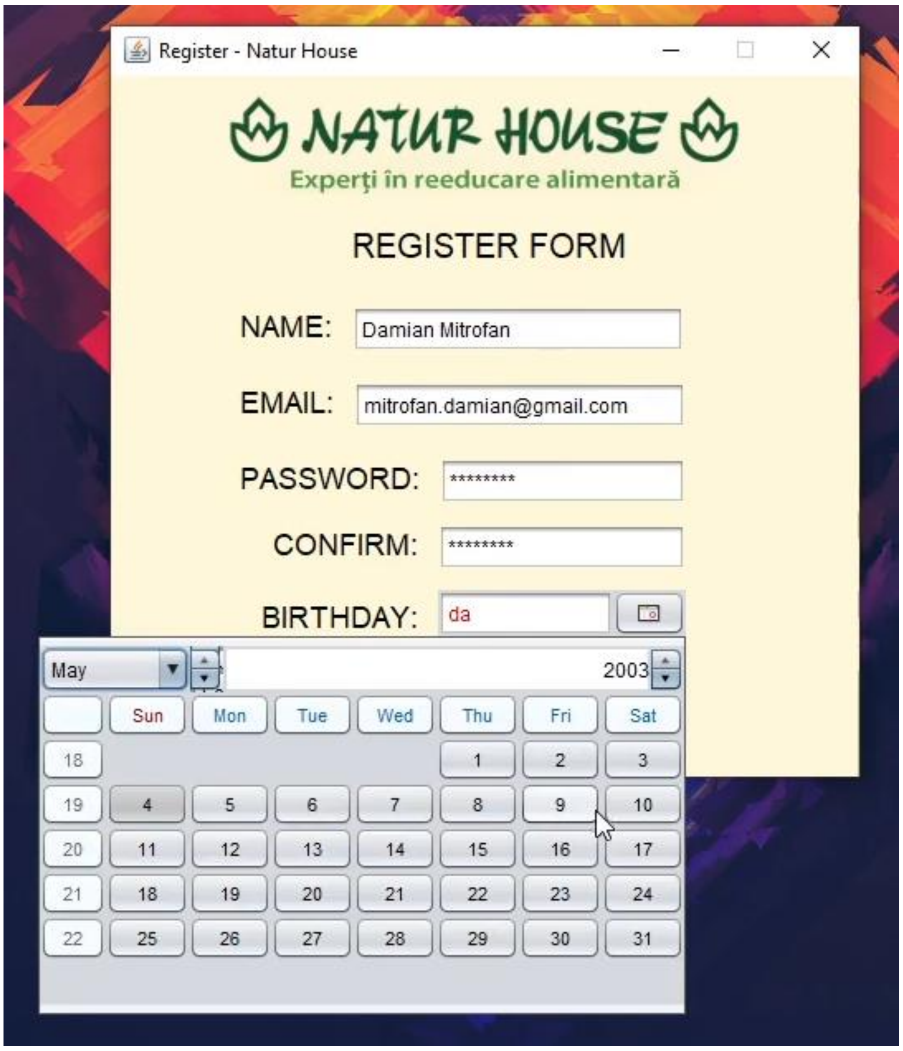
OK

CONFIRM: ********

BIRTHDAY: da

CANCEL REGISTER

Register form page – birthday format checking.


Register - Natur House

NATUR HOUSE
Experţi în reeducare alimentară

REGISTER FORM

NAME: Damian Mitrofan

EMAIL: mitrofan.damian@gmail.com

PASSWORD: ********

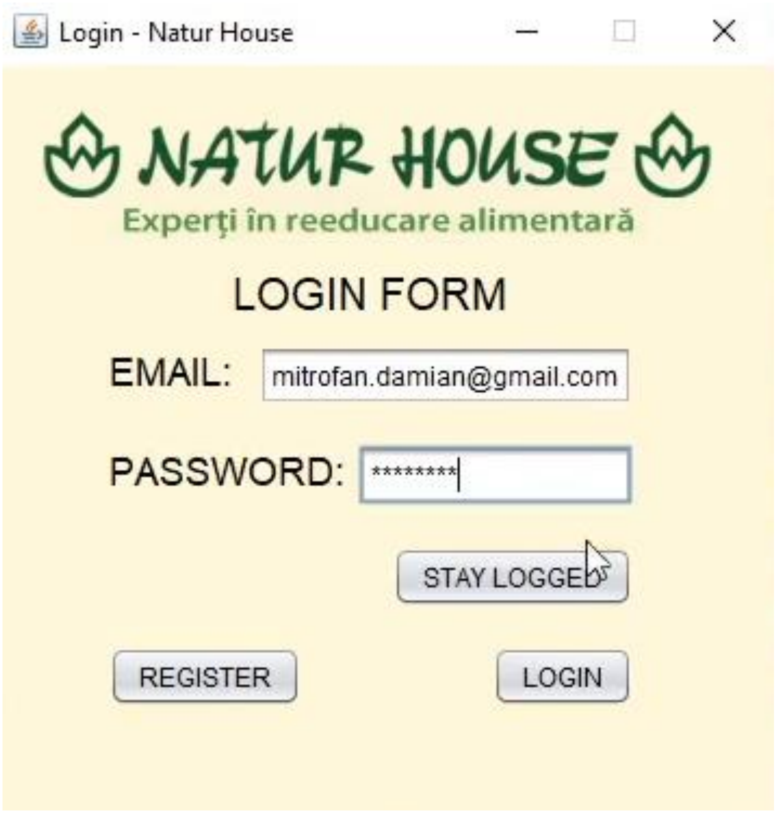
CONFIRM: ********

BIRTHDAY: da

May 2003

	Sun	Mon	Tue	Wed	Thu	Fri	Sat
18					1	2	3
19	4	5	6	7	8	9	10
20	11	12	13	14	15	16	17
21	18	19	20	21	22	23	24
22	25	26	27	28	29	30	31

Register form page – birthday choosing table.



Login - Natur House

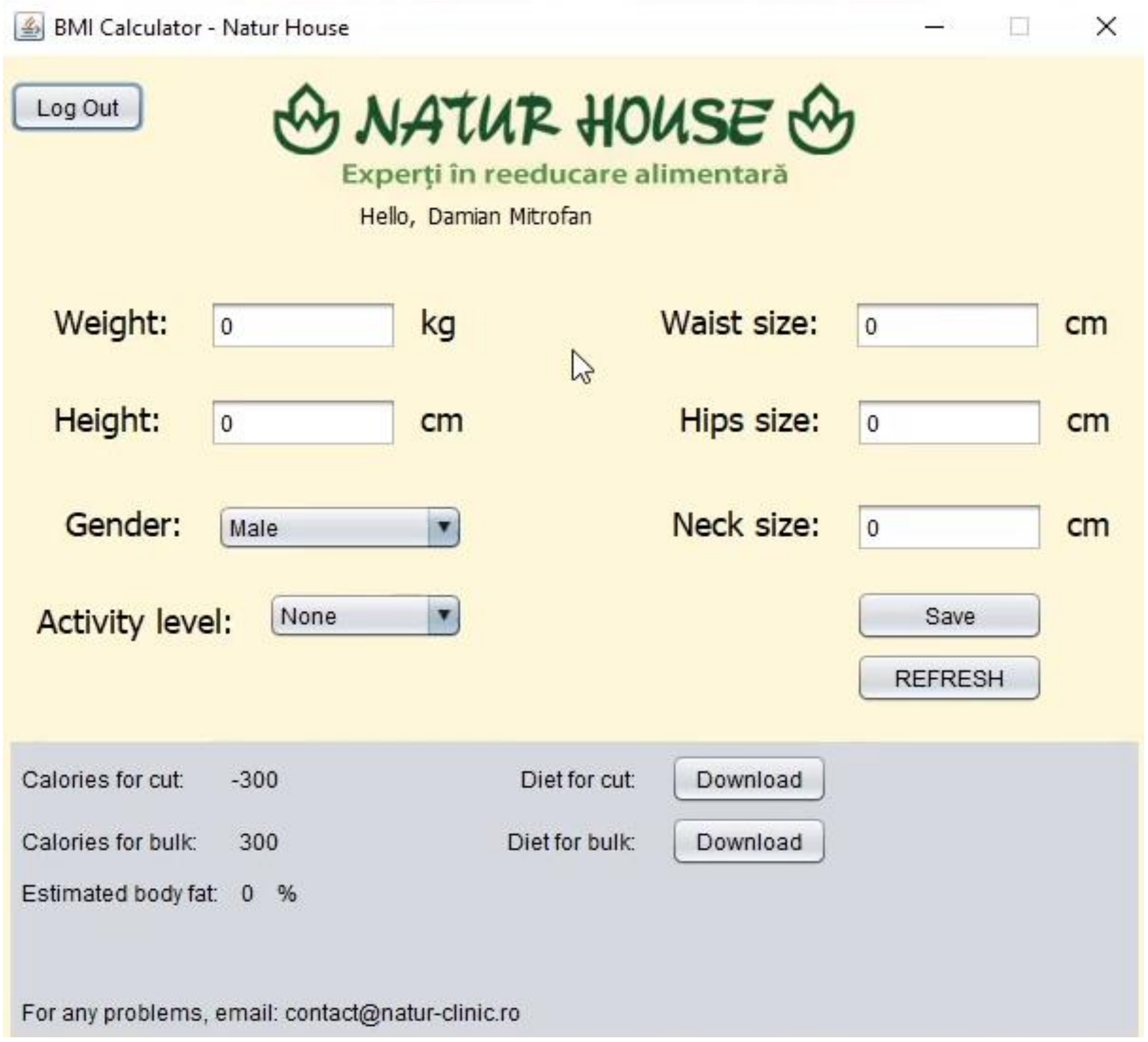
Experţi în reeducare alimentară

LOGIN FORM

EMAIL: mitrofan.damian@gmail.com

PASSWORD: *********

Login form page – with the stay logged function.



BMI Calculator - Natur House

Experţi în reeducare alimentară

Hello, Damian Mitrofan

Weight:	0	kg	Waist size:	0	cm
Height:	0	cm	Hips size:	0	cm
Gender:	Male		Neck size:	0	cm
Activity level:	None				

Calories for cut: -300

Calories for bulk: 300

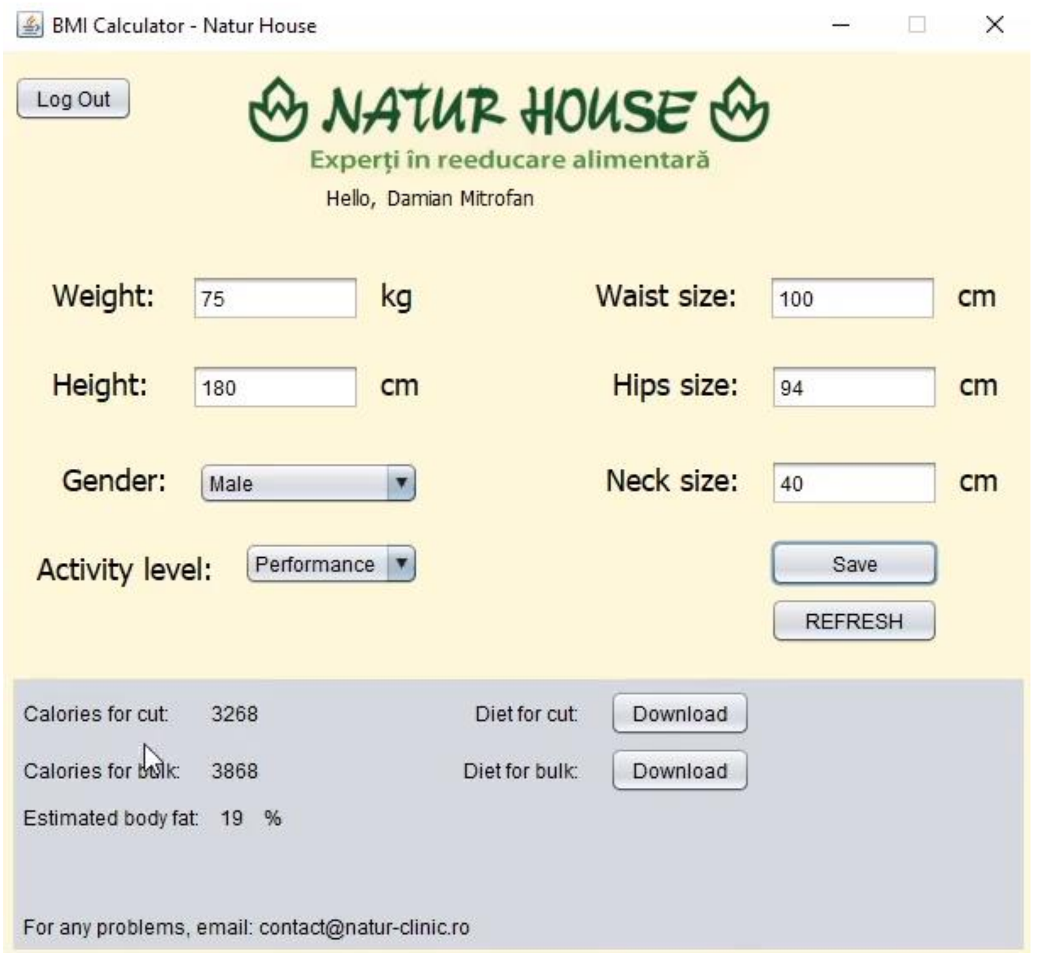
Estimated body fat: 0 %

Diet for cut:

Diet for bulk:

For any problems, email: contact@natur-clinic.ro

Main form page – blank details.



BMI Calculator - Natur House

Log Out

NATUR HOUSE

Experţi în reeducare alimentară

Hello, Damian Mitrofan

Weight:	75	kg	Waist size:	100	cm
Height:	180	cm	Hips size:	94	cm
Gender:	Male		Neck size:	40	cm
Activity level:	Performance				

Save

REFRESH

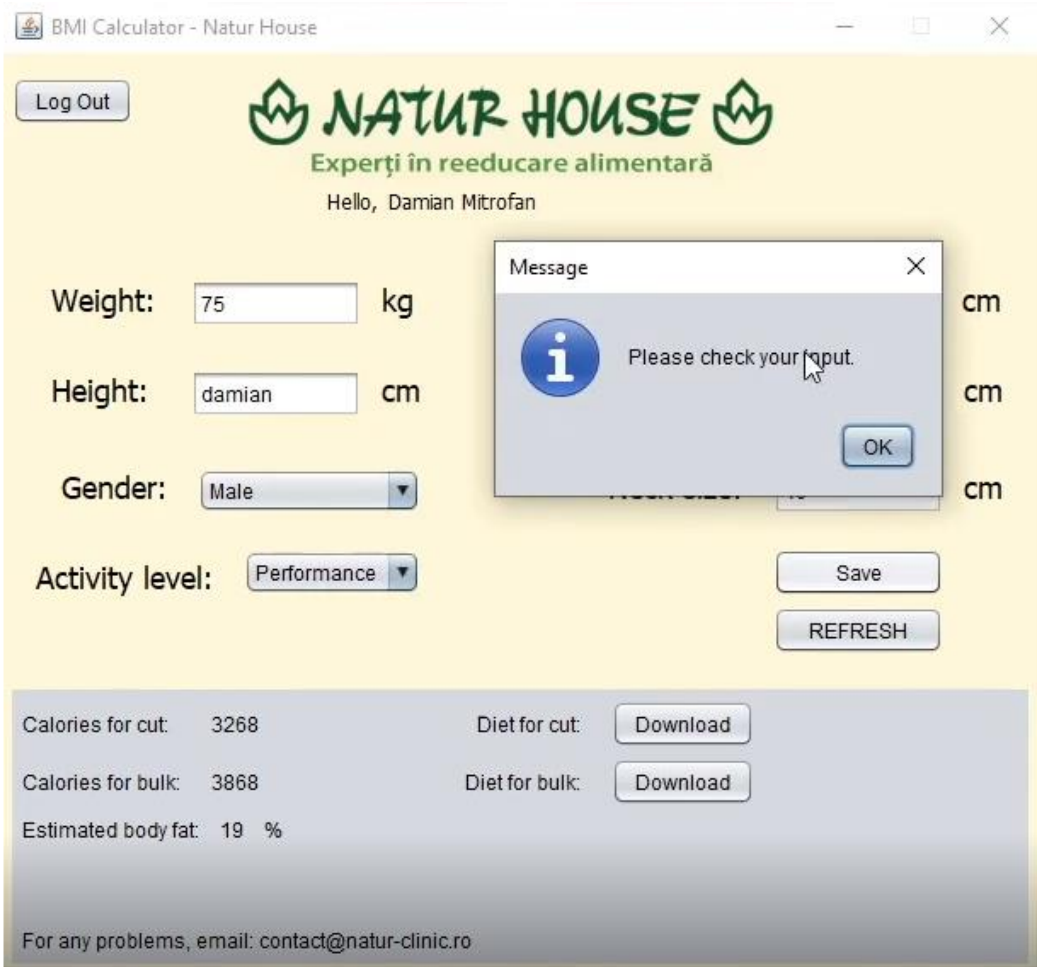
Calories for cut: 3268 Diet for cut: Download

Calories for bulk: 3868 Diet for bulk: Download

Estimated body fat: 19 %

For any problems, email: contact@natur-clinic.ro

Main form page – completed example details.



BMI Calculator - Natur House

Log Out

NATUR HOUSE

Experţi în reeducare alimentară

Hello, Damian Mitrofan

Weight:	75	kg
Height:	damian	cm
Gender:	Male	
Activity level:	Performance	

Message

Please check your input.

OK

Save

REFRESH

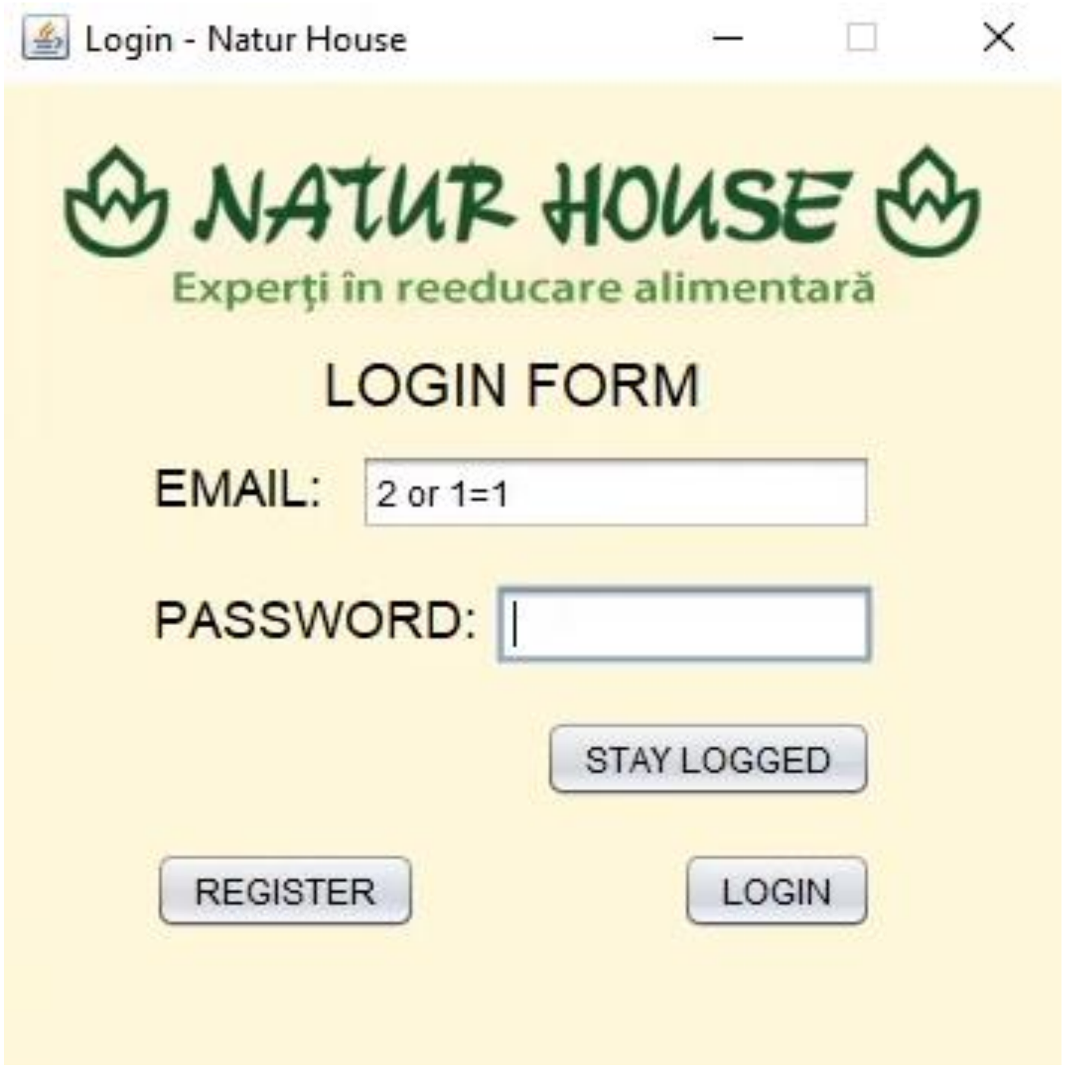
Calories for cut: 3268 Diet for cut: Download

Calories for bulk: 3868 Diet for bulk: Download

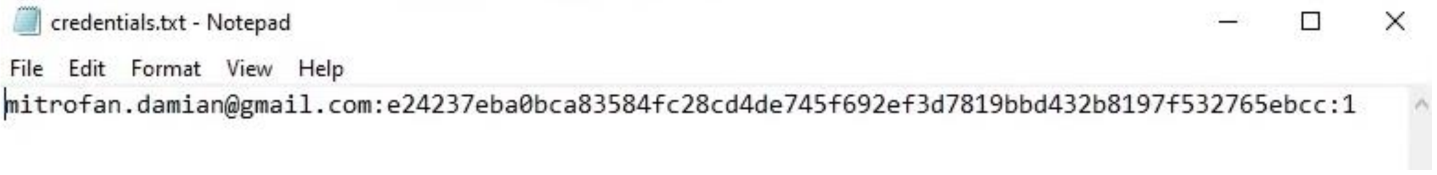
Estimated body fat: 19 %

For any problems, email: contact@natur-clinic.ro

Main form page – input checking.



Login form page – SQL injection verification (not vulnerable).



```
mitrofan.damian@gmail.com:e24237eba0bca83584fc28cd4de745f692ef3d7819bbd432b8197f532765ebcc:1
```

Saved files when a user logs in, 1 shows if the user selected to stay logged in or not. With this, next time the user open the program, will skip the login form page directly to the main form.