שמות מגישים: איילון כהן 312534266
פאדי עותמאנה 206793275

Icontroller.java

```
package Controller;

import Model.*;

public interface IController {
   public User signUp(String id, String name, String password, double budget);

   public User getUserDetails(String id, String password);

   public void getInvoices(User s);

   public void insertInvoice(String userID, double amount, String description, String date);

   public void deleteInvoice(int id);
}
```

controller.java

```
package Controller;

import Model.*;
import java.net.URI;
import java.net.http.HttpClient;
import java.net.http.HttpRequest;
import java.net.http.HttpResponse;
import java.util.*;
import java.text.SimpleDateFormat;

import mjson.Json;

public class Controller implements IController {

    /**
     * Function creates a new user and saves it in the database
     *
     * @param id       the id of the user to be created
     * @param name     the name of the user to be created
     * @param password the password of the user to be created
     * @param budget   the budget of the user to be created
     * @return User / null
     */
    @Override
    public User signUp(String id, String name, String password, double
budget) {
        User s = null;
        HttpClient client;
        HttpRequest request;
```

```
        HttpResponse<String> response;
        try {
            final Json x = Json.object().set("id", id).set("password",
password).set("name", name).set("budget",
                    budget);

            client = HttpClient.newHttpClient();
            request = HttpRequest.newBuilder()

.uri(URI.create("https://invoicing-java-backend.herokuapp.com/api/signu
p"))
                    .header("Content-Type",
"application/json").POST(HttpRequest.BodyPublishers.ofString(x.toStrin
g()))
                    .build();
            response = client.send(request,
HttpResponse.BodyHandlers.ofString());
            if (response.body().equals("Inserted Successfully")) {
                s = new User(id, name, password, budget);
            }

        } catch (final Exception e) {
            System.err.println(e.toString());
        } finally {
            return s;
        }
    }

    /**
     * Function retrieves a users details and verifies him otherwise returns
null
     *
     * @param id       The id of the user to retriev
     * @param password The password of the user
     * @return User or null
```

```
     */
    @Override
    public User getUserDetails(final String id, final String password) {
        User s = null;
        HttpClient client;
        HttpRequest request;
        HttpResponse<String> response;
        try {
            final Json x = Json.object().set("id", id).set("password",
password);

            client = HttpClient.newHttpClient();
            request =
HttpRequest.newBuilder().uri(URI.create("https://invoicing-java-backend.
herokuapp.com/api/login"))
                    .header("Content-Type",
"application/json").POST(HttpRequest.BodyPublishers.ofString(x.toStrin
g()))
                    .build();
            response = client.send(request,
HttpResponse.BodyHandlers.ofString());
            if (!response.body().equals("User doesnt exist")) {
                final Json u = Json.read(response.body());
                s = new User(u.at("id").asString(), u.at("name").asString(),
u.at("password").asString(),
                        u.at("budget").asDouble());
            }

        } catch (final Exception e) {
            System.err.println(e.toString());
        } finally {
            return s;
        }
    }
```

```
    /**
     * Function retrieves all invoices that belong to a certain user
     *
     * @param s The user you want to retrieve his invoices
     */
    @Override
    public void getInvoices(final User s) {
        HttpClient client;
        HttpRequest request;
        HttpResponse<String> response;

        try {
            s.dumpList();
            client = HttpClient.newHttpClient();

            request = HttpRequest.newBuilder()

.uri(URI.create("https://invoicing-java-backend.herokuapp.com/invoice/all
/" + s.getID()))
                    .header("Accept", "application/json").build();

            response = client.send(request,
HttpResponse.BodyHandlers.ofString());

            if (response.body().length() != 0 && !response.body().equals("[]"))
{
                final Json y = Json.read(response.body());
                final List<Json> i = y.asJsonList();
                for (final Json obj : i) {
                    s.insertInvoice(new Invoice(obj.at("id").asInteger(),
obj.at("amount").asDouble(),
                            obj.at("description").asString(),
obj.at("date").asString()));
                }
            }
```

```
        } catch (final Exception e) {
            System.err.println(e.toString());
        }
    }

    /**
     * Function inserts a new invoice to a given user
     *
     * @param userID      The users id to insert invoice to
     * @param amount      The amount in the invoice
     * @param description The description of the invoice
     * @param date        A string represnting the date in format
YYYY-MM-DD
     */
    @Override
    public void insertInvoice(final String userID, final double amount, final
String description, final String date) {
        HttpClient client;
        HttpRequest request;
        HttpResponse<String> response;

        try {
            final Json x = Json.object().set("amount",
amount).set("description", description).set("UserID", userID)
                    .set("date", date);

            client = HttpClient.newHttpClient();
            request = HttpRequest.newBuilder()

.uri(URI.create("https://invoicing-java-backend.herokuapp.com/invoice/cr
eate/"))
                    .header("Content-Type",
"application/json").POST(HttpRequest.BodyPublishers.ofString(x.toStrin
g())).build();
 response = client.send(request,HttpResponse.BodyHandlers.ofString());
```

```
        } catch (final Exception e) {
            System.err.println(e.toString());
        }
    }

    /**
     * Function removes an invoice given its id
     *
     * @param id The id of the invoice that needs to be deleted
     */
    @Override
    public void deleteInvoice(final int id) {
        HttpClient client;
        HttpRequest request;
        HttpResponse<String> response;
        final SimpleDateFormat f = new
SimpleDateFormat("YYYY-MM-DD");

        try {
            client = HttpClient.newHttpClient();
            request = HttpRequest.newBuilder()

.uri(URI.create("https://invoicing-java-backend.herokuapp.com/invoice/d
elete/" + id))
                    .header("Content-Type",
"application/json").POST(HttpRequest.BodyPublishers.noBody()).build();
            response = client.send(request,
HttpResponse.BodyHandlers.ofString());
        } catch (Exception e) {
            System.err.println(e.toString());
        }
    }
}
```

Imodle.java

```
package Model;

public interface IModel{

}
```

Invoice.java

```
package Model;

public class Invoice implements IModel {
    private int _id;
    private double _amount;
    private String _description;
    private String _date;


    /**
     * @param _id
     * @param _amount
     * @param _description
     * @param _date
     * @return
     */
    public Invoice(final int _id, final double _amount, final String _description, final String
_date) {
        this._id = _id;
        this._amount = _amount;
        this._description = _description;
        this._date = _date;
    }


    /**
     * @return int
     */
    public int getID() {
        return _id;
    }
```

```
/**
 * @return double
 */
public double getAmount() {
    return _amount;
}


/**
 * @return String
 */
public String getDescription() {
    return _description;
}


/**
 * @return String
 */
public String getDate() {
    return _date;
}


/**
 * @param id
 */
public void setID(final int id) {
    this._id = id;
}


/**
 * @param amount
 */
public void setAmount(final double amount) {
    this._amount = amount;
}


/**
 * @param description
 */
public void setDescription(final String description) {
    this._description = description;
}
```

```
    /**
     * @param date
     */
    public void setDate(final String date) {
        this._date = date;
    }

}
```

user.java

```
package Model;

import java.util.ArrayList;
import java.util.List;

public class User implements IModel {
    private final String _id;
    private String _name;
    private String _password;
    private double _budget = 0;
    private final List<Invoice> _invoices;


    /**
     * @param id
     * @param name
     * @param password
     * @param budget
     * @return
     */
    public User(final String id, final String name, final String password, final double budget) {
        this._id = id;
        this._name = name;
        this._password = password;
        this._budget = budget;
        this._invoices = new ArrayList<>();
    }


    /**
     * @return String
     */
```

```
public String getName() {
    return this._name;
}


/**
 * @return String
 */
public String getID() {
    return this._id;
}


/**
 * @param name
 */
public void setName(final String name) {
    this._name = name;
}


/**
 * @return String
 */
public String getPassword() {
    return this._password;
}


/**
 * @param password
 */
public void setPassword(final String password) {
    this._password = password;
}


/**
 * @return double
 */
public double getBudget() {
    return this._budget;
}


/**
```

```
     * @param budget
     */
    public void setBudget(final double budget) {
        this._budget = budget;
    }

    public void dumpList() {
        this._invoices.clear();
    }


    /**
     * @param v
     */
    public void insertInvoice(final Invoice v) {
        this._invoices.add(v);
    }


    /**
     * @return List<Invoice>
     */
    public List<Invoice> getInvoices() {
        return this._invoices;
    }


    /**
     * @return double
     */
    public double getCurrent() {
        double sum = 0;

        for (final Invoice v : this._invoices) {
            sum += v.getAmount();
        }
        return sum;
    }
}
```

IView.java

```
package View;

public interface IView {
    public void loginMenu();

    public void mainMenu();

    public void signUpWindow();
}
```

View.java

```
package View;

import Controller.*;
import Model.*;
import javax.swing.*;
import java.awt.*;
import java.awt.event.*;
import java.util.List;
import javax.swing.table.DefaultTableModel;
import java.util.Calendar;
import java.util.logging.*;

public class View implements IView {

    private JFrame menuFrame;
    private IModel data;
    private final IController c;
    private JTable table;
    private JTextField textAmount;
    private JTextField textDescription;
    private JLabel idHolder, current;
    private Logger logger;


    /**
     * @return
     */
    public View() {
        this.c = new Controller();
```

```
    }

    /**
     * Function for showing the main menu of the programm
     */
    @Override
    public void mainMenu()
    {

        logger = Logger.getLogger("");
        this.menuFrame = new JFrame("Invoice Manager");
        this.table = new JTable();

        // Change A JTable Background Color, Font Size, Font Color, Row Height
        this.table.setBackground(Color.LIGHT_GRAY);
        this.table.setForeground(Color.black);
        final Font font = new Font("", 1, 22);
        this.table.setFont(font);
        this.table.setRowHeight(30);

        // create JTextFields
        final JLabel idLabel = new JLabel("id:");
        idHolder = new JLabel("Choose A Row Please!");
        final JLabel amountLabel = new JLabel("Amount:");
        final JLabel descriptionLabel = new JLabel("Description:");

        this.textAmount = new JTextField();
        this.textDescription = new JTextField();

        // create labels

        this.current = new JLabel("Current: " + (((User) this.data).getBudget() - ((User)
this.data).getCurrent()));
        this.current.setBounds(650, 200, 100, 25);

        final JLabel budget = new JLabel("Budget: " + ((User) this.data).getBudget());
        budget.setBounds(750, 200, 100, 25);

        final JLabel userNameLabel = new JLabel("Welcome " + ((User) this.data).getName());
        userNameLabel.setBounds(10, 200, 100, 25);

        this.loadList();

        // create JButtons
        final JButton btnAdd = new JButton("Add");
        final JButton btnDelete = new JButton("Delete");
```

```
        final JButton btnUpdate = new JButton("Clear");

        // set sizes to the textfields
        idLabel.setBounds(305, 200, 100, 25);
        idHolder.setBounds(385, 200, 100, 25);
        amountLabel.setBounds(305, 240, 100, 25);
        textAmount.setBounds(385, 240, 100, 25);
        descriptionLabel.setBounds(305, 280, 100, 25);
        textDescription.setBounds(385, 280, 100, 25);
        btnAdd.setBounds(255, 320, 100, 25);
        btnUpdate.setBounds(375, 320, 100, 25);
        btnDelete.setBounds(495, 320, 100, 25);

        // create JScrollPane
        final JScrollPane pane = new JScrollPane(this.table);
        pane.setBounds(0, 0, 880, 200);
        menuFrame.setLayout(null);
        menuFrame.add(pane);

        // add JTextFields to the jframe
        menuFrame.add(idLabel);
        menuFrame.add(amountLabel);
        menuFrame.add(descriptionLabel);

        menuFrame.add(this.idHolder);
        menuFrame.add(this.textAmount);
        menuFrame.add(this.textDescription);

        buttonActions(btnAdd, btnUpdate, btnDelete);

        this.table.addMouseListener(new MouseAdapter() { // get selected row data From
this.table to textfields
            @Override
            public void mouseClicked(final MouseEvent e) {
                final int i = table.getSelectedRow();
                idHolder.setBounds(420, 200, 100, 25);
                idHolder.setText(table.getModel().getValueAt(i, 0).toString());
                textAmount.setText(table.getModel().getValueAt(i, 1).toString());
                textDescription.setText(table.getModel().getValueAt(i, 2).toString());
            }
        });

        // add JButtons to the jframe
        menuFrame.add(btnAdd);
        menuFrame.add(btnDelete);
        menuFrame.add(btnUpdate);
```

```
        menuFrame.add(this.current);
        menuFrame.add(budget);
        menuFrame.add(userNameLabel);

        menuFrame.setSize(890, 400);
        menuFrame.setLocationRelativeTo(null);
        menuFrame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        menuFrame.setVisible(true);
        logger.info("Login of " + ((User) this.data).getName() + " Succeeded. " + " \n" + " His
current expend is: " + (((User) this.data).getBudget() - ((User) this.data).getCurrent())+ "$" +
        "\n " + "His budget is: " + ((User) this.data).getBudget()+"$");
    }

    /**
     * Function that clears all data from the table
     */
    private void cleanTable() {
        final DefaultTableModel dm = (DefaultTableModel) this.table.getModel();
        dm.getDataVector().removeAllElements();
        this.table.revalidate();
    }

    private void cleanSelection() {
        idHolder.setBounds(385, 200, 100, 25);
        idHolder.setText("Choose A Row Please!");
        textAmount.setText("");
        textDescription.setText("");
    }

    /**
     * Function that handles showing the invoice list into the table
     */
    private void loadList() {
        this.cleanTable();
        c.getInvoices((User) this.data);
        this.current.setText("Budget: " + (((User) this.data).getBudget() - ((User)
this.data).getCurrent()));
        final List<Invoice> l = ((User) this.data).getInvoices();
        final Object[] row = new Object[4];
        final DefaultTableModel model = new DefaultTableModel();
        final Object[] columns = { "ID", "Amount", "Description", "Date" };
        model.setColumnIdentifiers(columns);

        for (final Invoice v : l) {
            row[0] = v.getID();
```

```
            row[1] = v.getAmount();
            row[2] = v.getDescription();
            row[3] = v.getDate().split("T")[0];

            model.addRow(row);
        }

        this.table.setModel(model);
    }

    /**
     * Function that handles all button Action Events initilization
     *
     * @param btnAdd    Button for adding a new invoice
     * @param btnClear  Button for updating a new invoice
     * @param btnDelete Button for deleting a new invoice
     */
    private void buttonActions(final JButton btnAdd, final JButton btnClear, final JButton
btnDelete) {
        logger = Logger.getLogger("");
        btnAdd.addActionListener(e -> {
            if (!textAmount.getText().equals("") && !textDescription.getText().equals("")) {
                try {
                    Double.parseDouble(textAmount.getText());
                    new Thread() {
                        @Override
                        public void run() {
                            final String s = String.format("%s-%s-%s",
Calendar.getInstance().get(Calendar.YEAR),
                                    Calendar.getInstance().get(Calendar.MONTH) + 1,
                                    Calendar.getInstance().get(Calendar.DAY_OF_MONTH));

                            c.insertInvoice(((User) data).getID(),
Double.parseDouble(textAmount.getText()),
                                    textDescription.getText(), s);
                            cleanSelection();
                            loadList();
                            logger.info("add invoice Succeeded. " + " \n" + " His current expend is: " +
(((User) data).getBudget() - ((User) data).getCurrent())+ "$" +
                            "\n " + "His budget is: " + ((User) data).getBudget()+"$");

                        }
                    }.start();
                } catch (NumberFormatException ex) {
                    System.err.println(ex);
                }
```

```
            }
            else
            {
                logger.info("Add ivoice faild");
            }
        });

        btnDelete.addActionListener(e -> {
            if (!idHolder.getText().equals("Choose A Row Please!"))
                new Thread() {
                    @Override
                    public void run() {
                        c.deleteInvoice(Integer.parseInt(idHolder.getText()));
                        cleanSelection();
                        loadList();
                        logger.info("delete invoice Succeeded. " + " \n" + " His current expend is: " +
((((User) data).getBudget() - ((User) data).getCurrent()))+ "$" +
                                "\n " + "His budget is: " + ((User) data).getBudget()+"$");
                    }
                }.start();
            else
            {
                logger.info("Delete invoice faild");
            }
        });
        btnClear.addActionListener(e -> cleanSelection());
    }

    /**
     * Draws the initial login menu
     */
    @Override
    public void loginMenu() {
        logger = Logger.getLogger("");
        final JFrame frame = new JFrame("Invoice Manager - Login");
        frame.setSize(350, 200);
        final JPanel panel = new JPanel();
        panel.setLayout(null);
        final JLabel userLabel = new JLabel("User");
        userLabel.setBounds(10, 20, 80, 25);
        panel.add(userLabel);
        final JTextField userText = new JTextField(20);
        userText.setBounds(100, 20, 165, 25);
        panel.add(userText);
        final JLabel passwordLabel = new JLabel("Password");
        passwordLabel.setBounds(10, 50, 80, 25);
```

```
    panel.add(passwordLabel);
    final JTextField passwordText = new JTextField(20);
    passwordText.setBounds(100, 50, 165, 25);
    panel.add(passwordText);
    final JButton loginButton = new JButton("login");
    loginButton.setBounds(10, 80, 80, 25);
    panel.add(loginButton);
    final JButton signUpButton = new JButton("Sign Up");
    signUpButton.setBounds(100, 80, 100, 25);
    panel.add(signUpButton);
    loginButton.addActionListener(e -> {
        if (!userText.getText().equals("") && !passwordText.getText().equals("")) {
            frame.setVisible(false);
            login(userText.getText(), passwordText.getText());
        }
        else{
            logger.info("Login faild");
        }
    });
    signUpButton.addActionListener(e -> {
        frame.setVisible(false);
    });
    signUpButton.addActionListener(e -> signUpWindow());
    frame.add(panel);
    frame.setVisible(true);
}

/**
 * Function to verify login data
 *
 * @param id
 * @param password
 */
public void login(final String id, final String password) {
    this.data = this.c.getUserDetails(id, password);
    if (this.data != null) {
        mainMenu();
    }
}

/**
 * Function to draw the signup window
 */
@Override
public void signUpWindow() {
    logger = Logger.getLogger("");
```

```
        final JFrame frame = new JFrame("Invoice Manager - Sign Up");
        frame.setSize(300, 220);
        final JPanel panel = new JPanel();
        panel.setLayout(null);
        final JLabel userLabel = new JLabel("User: ");
        userLabel.setBounds(10, 20, 80, 25);
        panel.add(userLabel);
        final JTextField userText = new JTextField(20);
        userText.setBounds(100, 20, 165, 25);
        panel.add(userText);

        final JLabel passwordLabel = new JLabel("Password: ");
        passwordLabel.setBounds(10, 50, 80, 25);
        panel.add(passwordLabel);
        final JTextField passwordText = new JTextField(20);
        passwordText.setBounds(100, 50, 165, 25);
        panel.add(passwordText);

        final JLabel nameLabel = new JLabel("Name: ");
        nameLabel.setBounds(10, 80, 80, 25);
        panel.add(nameLabel);
        final JTextField nameText = new JTextField(20);
        nameText.setBounds(100, 80, 165, 25);
        panel.add(nameText);

        final JLabel budgetLabel = new JLabel("Budget: ");
        budgetLabel.setBounds(10, 110, 80, 25);
        panel.add(budgetLabel);
        final JTextField budgetText = new JTextField(20);
        budgetText.setBounds(100, 110, 165, 25);
        panel.add(budgetText);

        final JButton submitButton = new JButton("submit");
        submitButton.setBounds(10, 150, 100, 25);
        panel.add(submitButton);
        submitButton.addActionListener(e -> {
            if (!userText.getText().equals("") && !passwordText.getText().equals("") &&
!nameText.getText().equals("")
                    && !budgetText.getText().equals("")) {
                try {
                    Double.parseDouble(budgetText.getText());
                    new Thread() {
                        @Override
                        public void run() {
                            data = c.signUp(userText.getText(), nameText.getText(),
passwordText.getText(),
```

```
                        Double.parseDouble(budgetText.getText()));
                    frame.setVisible(false);
                    mainMenu();
                }
            }.start();
        } catch (NumberFormatException ex) {
            System.err.println(ex);
        }
      }
      else{
        logger.info("Sign Up faild");
      }
    });
    frame.add(panel);
    frame.setVisible(true);
  }


  /**
   * @param args[]
   */
  public static void main(final String args[]) {
    final View simpleGUI = new View();
    simpleGUI.loginMenu();
  }

}
```