



**ADA UNIVERSITY
SCHOOL OF IT & ENGINEERING**

Course: Object-Oriented Analysis & Design

HOMEWORK ASSIGNMENT

Project title: ***“Human Resources Management Systems”***

Students: Aytan Babayeva
Nijat Mursali
Ismayil Shahaliyev
Mehdi Hajiyeu

| Team member | Contribution to the homework |
|--------------------|------------------------------|
| Aytan Babayeva | Classes and Methods Design |
| Nijat Mursali | User Interface Layer |
| Ismayil Shahaliyev | Data Management Layer |
| Mehdi Hajiyeu | Physical Architecture Layer |

Instructor: **Dr. Abzatdin Adamov**

Baku 2019

Table of Contents

Contents

HOMework ASSIGNMENT 1

 Baku 2019..... 1

CLASSES AND METHODS DESIGN3

DATA MANAGEMENT LAYER.....6

USER INTERFACE LAYER.....6

PHYSICAL ARCHITECTURE LAYER.....9

1. Classes and Methods Design

```
public class Database {
    private int empID;
    private String empName;
    private int empSalary;
    Employee emp;

    public Database() {
        this.emp = new Employee();
        this.empID = emp.empID;
        this.empName = emp.empName;
        empSalary = emp.getRewards();
    }

    public void checkEmp(int empID) {
        System.out.println("Given ID attached to the employee of " + emp.users.get(empID));
    }

    public int checkDetails(String empName) {
        return emp.users.indexOf(empName);
    }

    public void modifyChanges() {
        System.out.println("Changes are modified\n");
    }

    public void editEmp() {
        emp.Modify(empName);
    }

    public void searchEmp(String empName) {
        System.out.println("Employee found in our DB with details of " + emp.empAddress + emp.uEmail + emp.uRole + "\n");
    }

    public void assignTask(String newTask) {
        System.out.println("Employee " + empName + "is assigned with the task of " + newTask + "\n");
    }

    public void identifyPriv() {
        System.out.println("Privileges are identifying...\n");
    }

    public int getEmpID() {
        return empID;
    }

    public int getEmpID() {
        return empID;
    }

    public void setEmpID(int empID) {
        this.empID = empID;
    }

    public String getEmpName() {
        return empName;
    }

    public void setEmpName(String empName) {
        this.empName = empName;
    }

    public int getEmpSalary() {
        return empSalary;
    }

    public void setEmpSalary(int empSalary) {
        this.empSalary = empSalary;
    }
}
```

Fig 1. Database Class

```
import java.util.ArrayList;

public class Department {
    ArrayList<String> departments;
    int depID;
    String depType;
    String depName;
    int empID;
    Employee emp;

    public Department(String depName, String depType) {
        departments = new ArrayList<String>();
        emp = new Employee();
        this.empID = emp.empID;
        this.depType = depType;
        this.depName = depName;
    }

    public void addEmployee() {
        emp.Register();
    }

    public void addDepartment() {
        departments.add(depName);
        this.depID = departments.indexOf(depName);
        System.out.println("Department: " + depName + " is added to the System\n");
    }

    public void editDepartment() {
        System.out.println("Department" + depName + "is edited\n");
    }

    public void searchDepartment(int depID) {
        System.out.println("Given ID attached to the department of " + departments.get(depID) + "\n");
    }
}
```

Fig 2. Department Class

```

public class Employee extends User {
    String empName;
    int empID;
    String uRole;
    String empAddress;
    String empCard;
    String empUsername;
    String empPass;
    int amount;
    Salary salary;

    public Employee() {
        salary = new Salary();
        amount = salary.getAmount();
        this.empAddress = super.uAddress;
        this.uRole = super.uRole;
        this.empName = super.uName;
        this.empID = super.uId;

        // TODO Auto-generated constructor stub
    }

    public void Register() {
        super.addUser();
    }

    @Override
    public void deleteUser() {
        int ind = users.indexOf(empName);
        users.remove(ind);
        System.out.println("Employee: " + uName + " is fired from the job of " + uRole + "\n");
    }

    public void Login() {
        System.out.println("Employee" + empName + " is logged in\n");
    }

    public void Modify(String newUsername) {
        System.out.println("Username" + empUsername + " is modified to" + newUsername + "\n");
        empUsername = newUsername;
    }

    public void checkProgress() {
        System.out.println("Employee" + empName + " checked progress\n");
    }

    public int getRewards() {

```

Fig 3. Department Class

```

public class Salary {

    String type;
    public int getAmount() {
        return amount;
    }

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

    String description;
    private int amount;
    int empID;

    public Salary() {
        amount=5000;
        type="high-salary";

        empID = 6677;

        description = "This is the salary employee gets";
    }

    public int addSalary() {
        return amount;
    }

    public int editSalary(int newSalary) {
        this.amount = newSalary;
        return amount;
    }

}

```

Fig 4. Salary Class

```

public class System {
    Employee user;
    int uId;
    String uRole;
    String uName;
    String uEmail;
    String uAddress;
    String vacancyName;
    String vacancyDescript;
    String reports;

    public System() {
        this.user = new Employee();
        this.uAddress = user.uAddress;
        this.uId = user.uId;
        this.uEmail = user.uEmail;
        this.uRole = user.uRole;
        this.uName = user.uName;
    }

    public void giveSalary() {
        user.getRewards();
    }

    public void addVacancy(String vacancyName, String vacancyDescript) {
        this.vacancyName = vacancyName;
        this.vacancyDescript = vacancyDescript;
        System.out.println("Vacant place for the job of" + this.vacancyName + this.vacancyDescript + "\n");
    }

    public void giveReports(String reports) {
        this.reports = reports;
    }

    public void checkTasks() {
        for (int i = 0; i < user.users.size(); i++)
            System.out.println("Tasks of" + user.users.get(i) + "are checken\n");
    }

    public void addUser() {
        user.addUser();
    }

    public void editUser() {
        user.editUser();
    }

    public void deleteUser() {
        user.deleteUser();
    }
}

```

Fig 5. System Class

```

import java.util.ArrayList;

public class User {
    ArrayList<String> users;
    int uId;
    String uRole;
    String uName;
    String uEmail;
    String uAddress;

    public User() {
        users = new ArrayList<String>();
        this.uName = "Aytan";
        this.uAddress = "Baku, Azerbaijan";
        this.uEmail = "ababayeva2019@gmail.com";
        this.uRole = "Software Engineer";
        this.uId = 5566;
    }

    public void addUser() {
        users.add(uName);
        System.out.println("User: " + uName + " is added to the System\n");
    }

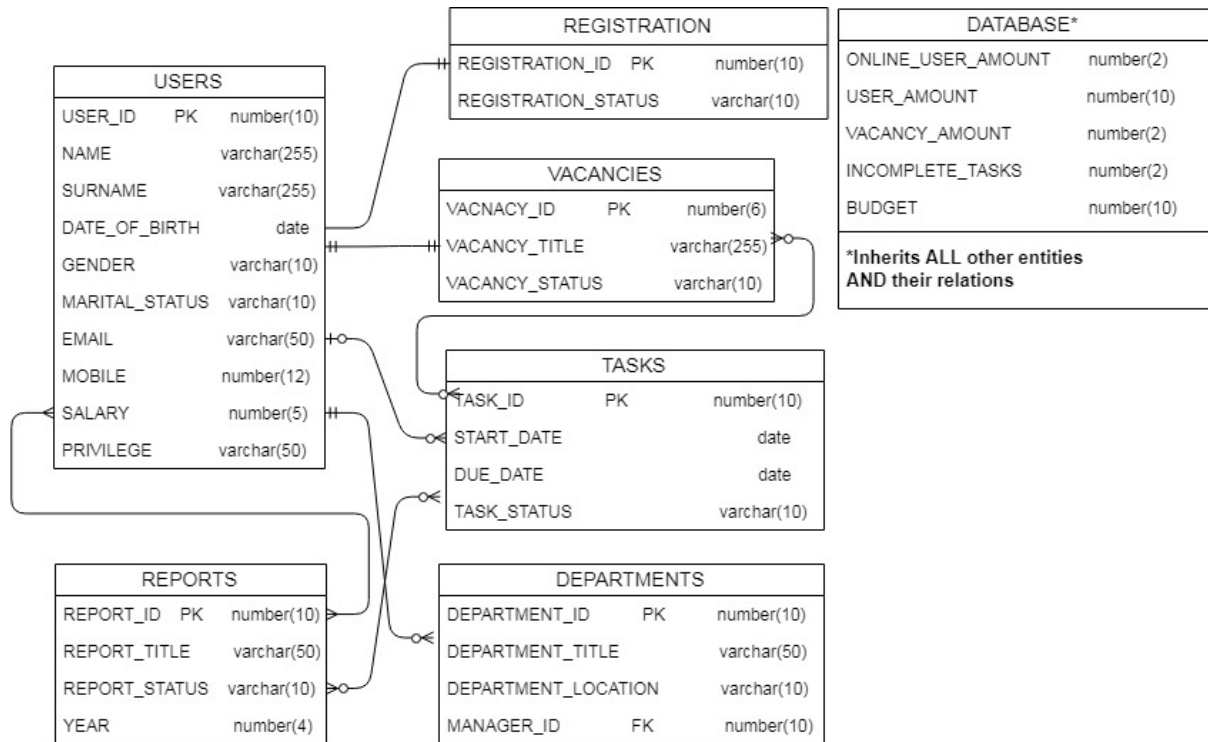
    public void editUser() {
        System.out.println("User" + uName + "is edited\n");
    }

    public void deleteUser() {
        int ind = users.indexOf(uName);
        users.remove(ind);
        System.out.println("User: " + uName + " is deleted from the System\n");
    }
}

```

Fig 6. User Class

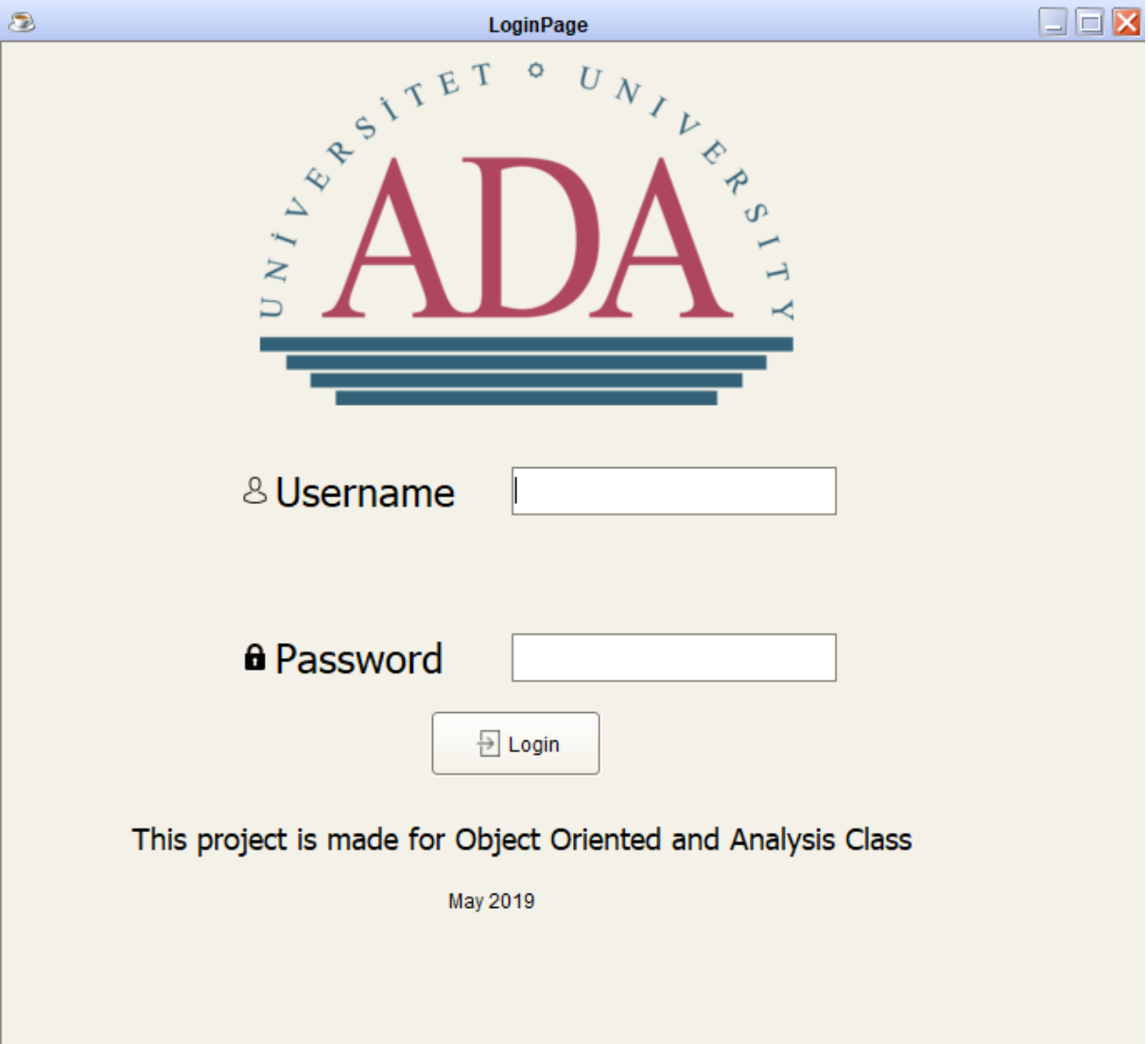
2. Data Management Layer



3. User Interface Layer

The CEO (stands for a chief executive officer) can login into the system in the first page for our system which is called **LoginPage**. Login page includes two text inputs in itself: the first one is the username and the second one is password field for CEO's information that has added previously. If the CEO adds the information correctly, he/she is able to enter the system. The next step is **LoadingPage** which makes the user wait for about 5 seconds in order to enter to the system and see the desired result. For this case, we have added ADA's logo and animated gif in the footer. After several seconds, user makes it and goes to **HomePage** which shows all the required information about all the users in the system. The information about the employee contains several fields such as employee id, name, surname, age, gender, department of employee, salary, shift and division. We have also added the login page for every user and when they enter to the system they can have different privileges such as read only or read/write. If the user is general user, he/she can only read the information about himself/herself or other users, but not able to modify the data that was set by CEO. After being an authenticated user, all the information about any employee can be shown in the system.

The following figure shows the user-interface for the **LoginPage** which contains several labels and two important user inputs which are username and password. We have used SQLITE database for creating the backend of our application which contained the information about the CEO who was the creator of the company. If the user enters the data correctly, he/she is directed to the **LoadingPage**, but if enters the information incorrectly, he/she asked for the correct data by our system.



The screenshot shows a window titled "LoginPage". At the top center is the logo of ADA University, featuring the word "ADA" in large red letters, with "UNIVERSITET" and "UNIVERSITY" in smaller blue letters above it, and several horizontal blue bars below. Below the logo are two input fields: "Username" with a person icon and "Password" with a lock icon. A "Login" button with a right-pointing arrow is positioned below the password field. At the bottom, a message reads "This project is made for Object Oriented and Analysis Class" followed by "May 2019".

Figure. Login Page into the System

If the CEO enters the data correctly, he/she is being directed to the **LoadingPage** which again contains several labels, progress bars and one loading GIF. Progress bar has been implemented using coding in Java which took two arguments: maximum and current values. If

the current value is less than maximum value, we added one to the progress bar until it got maximum. If it gets the maximum, another page which is called **HomePage** is loaded and user sees different types of fields which we will describe in the following paragraphs.

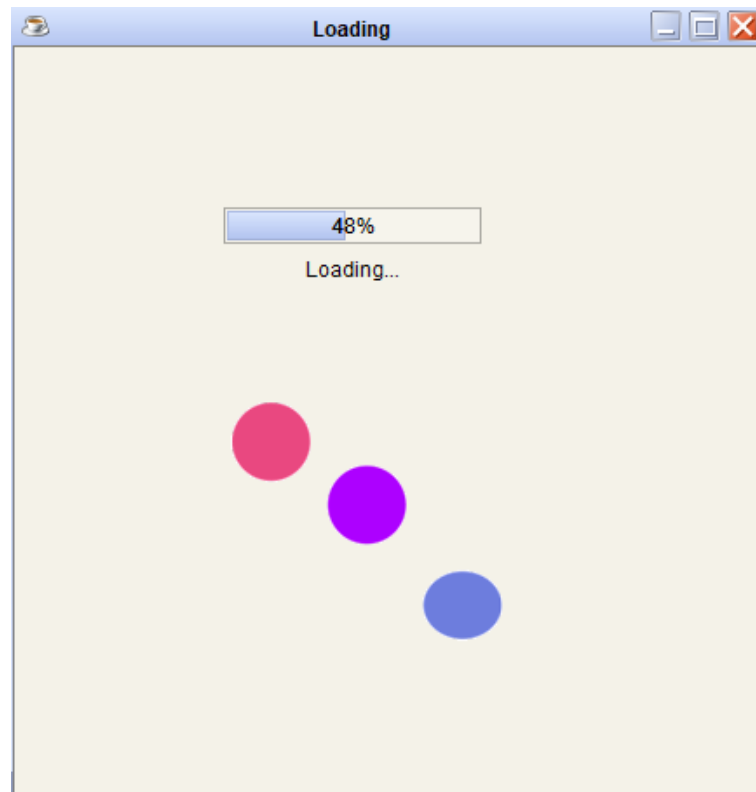


Figure. Loading Page of the System

The most important part of our system is **HomePage** which includes all the important functionalities and information about the system and employees. As shown in the Figure. the home page contains several labels, buttons and tables which we will describe more in following paragraphs.

In the top left corner, we have added the ADA's logo and below that we have added several labels for employee id, name, surname, age, gender and other functionalities. User text inputs (which our CEO does) for the following labels can be added into the system and with the help of **Save()** button (we can also call it method because we have implemented it) CEO will be able to add users into the database. **Delete()** deletes the selected employee from the database and system with the help of methods we have implemented. **Update()** function updates the user's information in the system. With the help of getting information from the database, we

were able to apply **Search()** function as well. The usage of search function is to add the name of any user from the table into the text fields and the system automatically derives the information about the user in the text input fields in the left corner. We have also added the **Report()** and **Print()** functions which creates the PDF file with the help of Java libraries and prints the files with printer. In the right corner, we have added the table and filled it dynamically with the help of SQLITE. As mentioned above, it works dynamically, so if CEO deletes any user in the end of deleting command that user will be vanished from the table.

The screenshot shows a web application interface for ADA University. The header includes the university logo and a title. The main area contains a search bar, a 'Logout' button, and a 'Total employee' counter. On the left, there are input fields for Employee ID, Name, Surname, Age, Gender, Department, Shift, Salary, and Division. In the center, there is an 'Operations' panel with buttons for Save, Delete, Update, and Clear. On the right, there is a table displaying employee data and a 'Show' button.

| empid | name | age | gender | departme | shift | division |
|-------|---------|-----|--------|------------|-------|----------|
| 1 | Nijat | 22 | Male | Compute... | 1 | CEO |
| 2 | Arzu | 20 | | IT | 2 | Manager |
| 3 | Nijatas | 22 | Male | Science | 1 | Test |

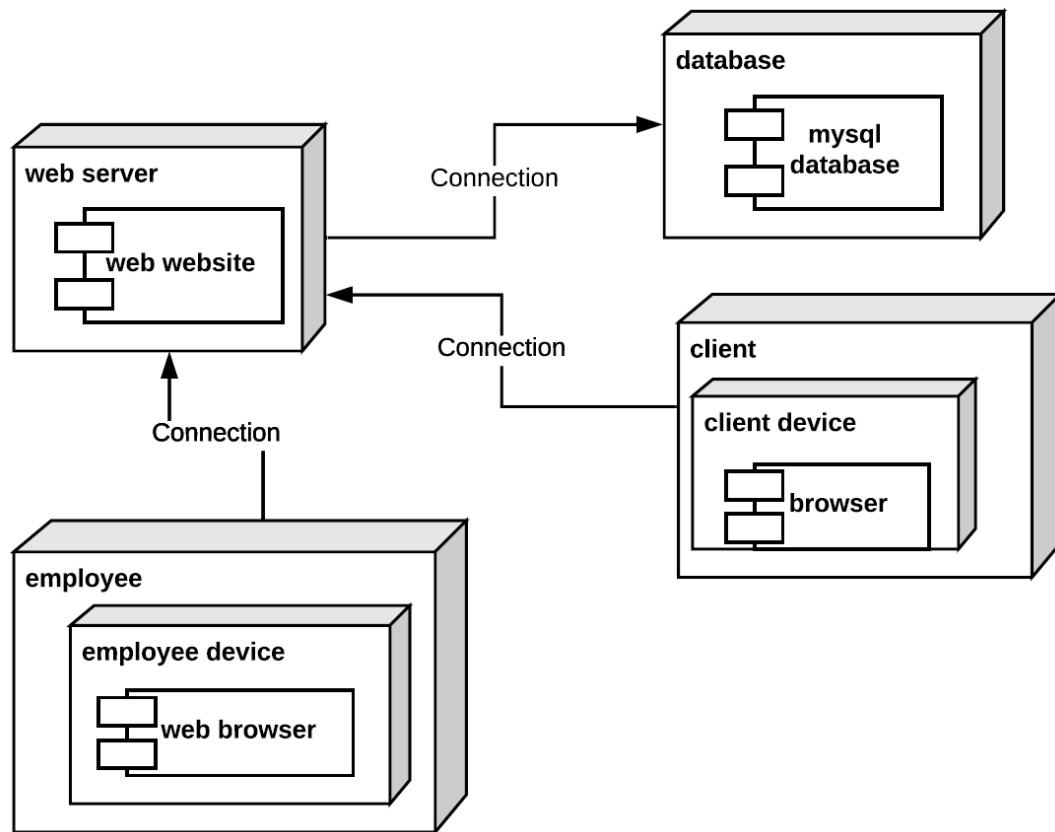
Figure. Home Page of the System

We have also implemented several other functions such as **Logout()**, **ShowingnumberofEmployees()**, **Exit()** and others. In addition, when CEO adds new data, he/she is also able to add the employee picture as seen from the Figure.

4. Physical Architecture Layer

The system will be client-server-based system which brings out 3-tier architecture. In our application the clients will be the employees and while there will be employees the HR manager should be responsible for adding them into the server. Our application will work on desktop with the help of Java environment. Due to having lack of time, we were not able to design the application for the mobile; however, for the upcoming

months we are planning to do it for android as well. All the operating system which have Java environment installed can run our application.



References

Human resource management system

- https://en.wikipedia.org/wiki/Human_resource_management_system

Human Resources Management System (HRMS)

- <https://www.bamboohr.com/hr-glossary/human-resources-management-system-hrms/>