University Management System
Design Document

Authors:
Sefa Ağardan

Group: 0

The purpose of this document is to provide you with a guideline for writing the software design document for your project.

Points to remember:

- *Content is important, not the volume.* ***Another team should be able to develop this system from only this document.***
- *Pay attention to details.*
- *Completeness and consistency will be rewarded*
- *Readability is important.*

This page intentionally left blank.

Document Revision History

Date	Version	Description	Author
28/12/2023	0.0	Initial draft	Prof. Dr. Taner Çevik
28/12/2023	0.1	Field 1 is done.	Sefa Ağardan
29/12/2023	0.2	Fields 2 to 4.3 are done.	Sefa Ağardan
30/12/2023	0.3	Fields to 4 are done.	Sefa Ağardan
31/12/2023	1	Report is completed.	Sefa Ağardan

This page intentionally left blank.

Contents

1 Introduction

1.1 Purpose

This project has been created to make university management easier for everybody including students, instructors/professors, and the IT staff which are admins in the program.

1.2 System Overview

Its target users are students, instructors, and the university management.

Students can see their courses and grades and edit their personal information like phone number and email address. Instructors can see their students and edit their grades, also they too can edit their personal information. Admin can create, read, update, and delete all users including students and instructors.

Although it was created as a desktop app, this project can be redesigned as a website for all students and academic staff to use it easily.

1.3 Design Objectives

The University Management System is a project designed to provide effective communication and data management among students, instructors, and administrators. The system offers various functionalities for each role.

Student Capabilities:

Students can read their personal information and update some of them like phone number or email address. Also, they can see their academic information like faculty, department, or grades. Lastly, they can see the clubs they joined and their short descriptions.

Instructor Capabilities:

Like students, instructors too can see and update their personal information. Additionally, they can see students' courses and edit their grades.

Administrator Capabilities:

Administrators can create, read, update, and delete students and instructors.

1.4 References

1.5 Definitions, Acronyms, and Abbreviations

Club Desc: Club descriptions, which are brief descriptions for clubs for users to understand their purpose better. It is shortened in the program for a better design.

2 Design Overview

2.1 Introduction

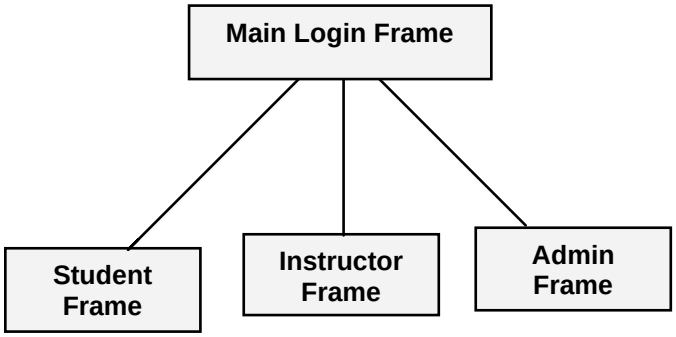
The system is programmed fully object-oriented. It has Person, Student, Instructor, Admin classes and graphical user interfaces which will be mentioned below. Java Swing and Eclipse Window Builder are used to create all GUI's. Software Ideas Modeler is used to create UML diagrams.

2.2 Environment Overview

The system will run as a desktop application. The user will use the given ".jar" file to execute the program.

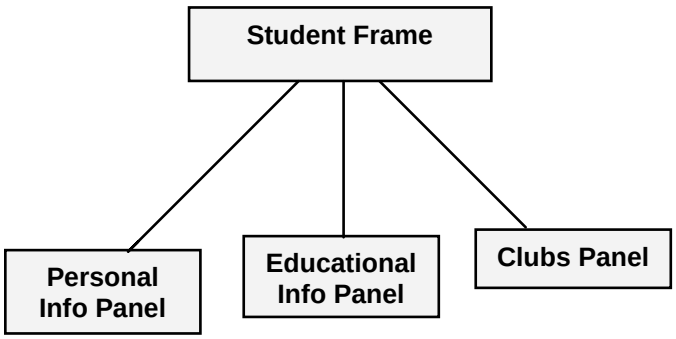
2.3 System Architecture

2.3.1 Top-level system structure of University Management System



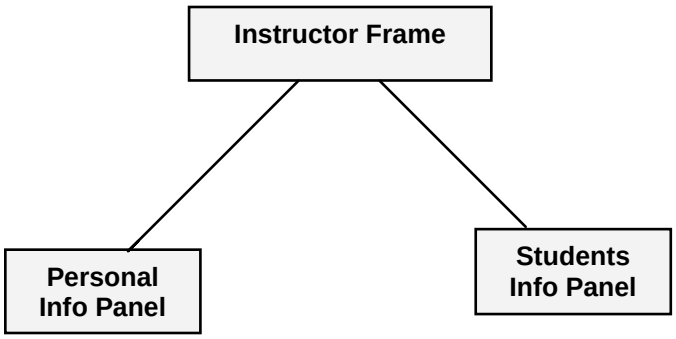
The system consists of 3 main components: Student Frame, Instructor Frame and Admin Frame. These components are reached from Main Login Frame depending on the user type.

2.3.2 Student Frame Sub-system



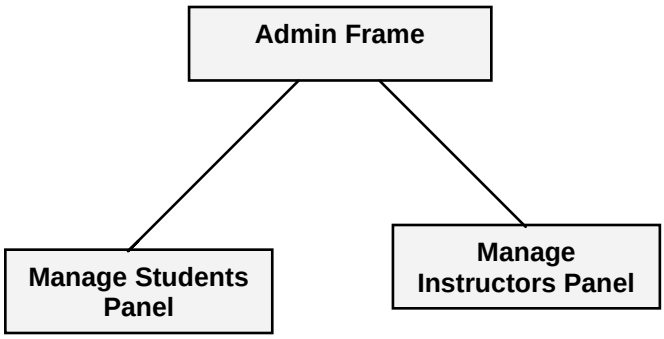
In this sub-system students can see personal, academical, and club-related information about themselves, also they can update some of their personal information.

2.3.3 Instructor Frame Sub-system



In this sub-system instructors can see and update their personal information. Also they see their students' courses and grades, and edit their grades.

2.3.4 Admin Frame Sub-system



In this sub-system admin can create, read, update, and delete students and instructors.

2.4 Constraints and Assumptions

System forces the admin about instructors' usernames. Since they are using them to log in, usernames must be unique.

Another constraint is about students' ID's and student numbers. They must be unique too because of the same reason, so IDs are declared by the system automatically and uniquely; and student numbers are created according to the ID of the student.

3 Interfaces and Data Stores

3.1 System Interfaces

3.1.1 Manageable Interface

Manageable is the only interface of the system, it is created to make user management easier. It is used in Student and Instructor classes. It includes three abstract methods: add(), delete(int id), addExistingUsersToTheMaps()

add(): For admin to add newly created user-student or instructor- to their hashmaps.
delete(): For admin to delete a user by id.
addExistingUsersToTheMaps(): It is called when the app started. Gets the data from database and puts it into the hashmaps.

3.2 Data Stores

Data is stored as objects in ".ser" files. Each user type has their ser files like "admin.ser", "instructor.ser", and "student.ser". While instructor and student files contain 1 hashmap of type <Integer, UserType>, admin file contains only one admin.

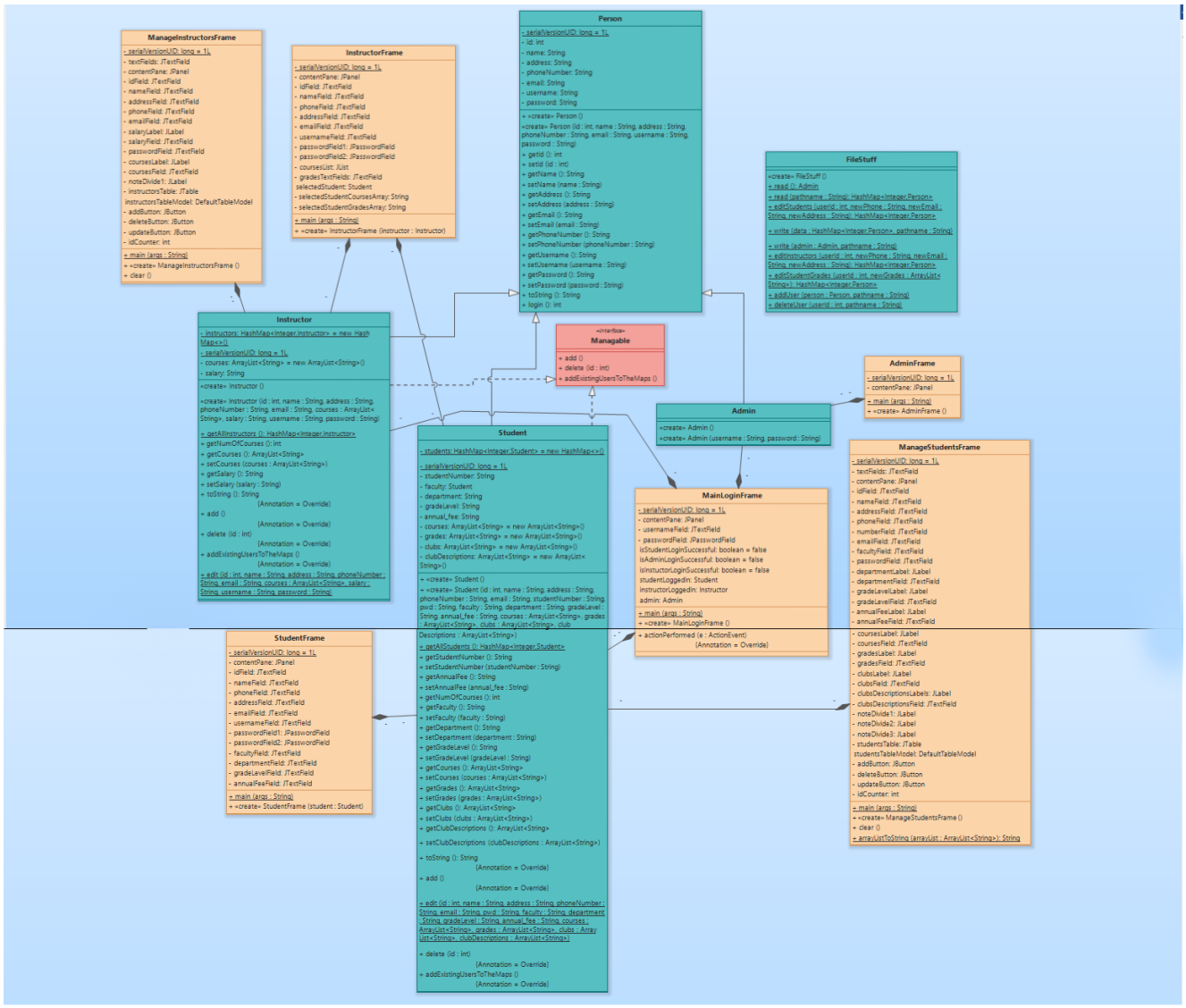
4 Structural Design

4.1 Design Explanation and Rationale

The system is OOP based, which makes it easier to handle and edit. It also makes it more readable.

Data is being created, read, updated and deleted in other related class by the methods which is already in FileStuff class. While this class contains all methods related with the file management, other classes use its methods in theirselves when needed.

4.2 Class Diagram



*Red: Interface: Manageable
*Yellow: Window classes which extends JFrame: MainLoginFrame, StudentFrame, InstructorFrame, AdminFrame, ManageStudentsFrame, ManageInstructorsFrame.
*Blue: Other classes: Person, Admin, Instructor, Student, FileStuff.

4.3 Class Descriptions

4.3.1 Class: Person

- Purpose: To model the relevant aspects of a person
- Constraints: None

4.3.1.1 Attribute Descriptions

1. Attribute: id
   Type: int
   Description: Stores the id of a user
   Constraints: must be an integer

2. Attribute: name
   Type: String
   Description: Stores the name of a user
   Constraints: None

3. Attribute: address
   Type: String
   Description: Stores the address of a user
   Constraints: None

4. Attribute: phoneNumber
   Type: String
   Description: Stores the phone number of a user
   Constraints: None

5. Attribute: email
   Type: String
   Description: Stores the email of a user
   Constraints: None

6. Attribute: username
   Type: String
   Description: Stores the username of a user
   Constraints: Must be unique

7. Attribute: password
   Type: String
   Description: Stores the password of a user
   Constraints: None

4.3.1.2 Method Descriptions

Does not have a method other than getters, setters and toString.

4.3.2 Class: Admin (extends Person)

- Purpose: To model the relevant aspects of a admin
- Constraints: None

4.3.2.1 Attribute Descriptions

Does not have any attributes.

4.3.2.2 Method Descriptions

It has two constructor methods: Admin(), Admin(String username, String password)

4.3.3 Class: Instructor (extends Person implements Manageable)

- Purpose: To model the relevant aspects of an instructor
- Constraints: None

4.3.3.1 Attribute Descriptions

8. Attribute: instructors
   Type: HashMap<Integer, Instructor>
   Description: Stores all instructors
   Constraints: None

9. Attribute: courses
   Type: ArrayList<String>
   Description: Stores the courses of the instructor
   Constraints: None

10. Attribute: salary
    Type: String
    Description: Stores the salary of the instructor
    Constraints: None

4.3.3.2 Method Descriptions

1. Method: add()
   Return Type: void
   Purpose: Adding instructor to the hashmap

2. Method: delete(int id)
   Return Type: void
   Purpose: Deleting instructor from the hashmap

3. Method: addExistingUsersToTheMaps()
   Return Type: void
   Purpose: Getting instructors from the database and putting them into the hashmap when the program is started

4. Method: edit(int id,String name,String address,String phoneNumber,String email,ArrayList<String> courses,String salary, String username, String password)
   Return Type: void
   Purpose: Updating the chosen instructor

It also has two constructor methods:
Instructor(),
Instructor(int id,String name,String address,String phoneNumber,String email,ArrayList<String> courses,String salary, String username, String password)

4.3.4 Class: Student (extends Person implements Manageable)

- Purpose: To model the relevant aspects of a student
- Constraints: None

4.3.4.1 Attribute Descriptions

11. Attribute: students
Type: HashMap<Integer, Student>
Description: Stores all student
Constraints: None

12. Attribute: studentNumber
Type: String
Description: Stores the student number of the student
Constraints: Must be unique and same with the username

13. Attribute: faculty
Type: String
Description: Stores the faculty of the student
Constraints: None

14. Attribute: department
Type: String
Description: Stores the department of the student
Constraints: None

15. Attribute: gradeLevel
Type: String
Description: Stores the grade level of the student
Constraints: None

16. Attribute: annualFee
Type: String
Description: Stores the annual fee of the student
Constraints: None

17. Attribute: courses
Type: ArrayList<String>
Description: Stores the courses of the student
Constraints: None

18. Attribute: grades
Type: ArrayList<String>
Description: Stores the grades of the student
Constraints: None

19. Attribute: clubs
Type: ArrayList<String>
Description: Stores the clubs of the student
Constraints: None

20. Attribute: club descriptions
Type: ArrayList<String>
Description: Stores the club descriptions of the student
Constraints: None

4.3.4.2 Method Descriptions

5. Method: add()
   Return Type: void
   Purpose: Adding student to the hashmap

6. Method: delete(int id)
   Return Type: void
   Purpose: Deleting student from the hashmap

7. Method: addExistingUsersToTheMaps()
   Return Type: void
   Purpose: Getting student from the database and putting them into the hashmap when the program is started

8. Method: edit(int id, String name,String address,String phoneNumber,String email,String pwd, String faculty, String department, String gradeLevel, String annual_fee,ArrayList<String> courses, ArrayList<String> grades, ArrayList<String> clubs, ArrayList<String> clubDescriptions)
   Return Type: void
   Purpose: Updating the chosen student

It also has two constructor methods:
Student(),
Student(int id,String name,String address,String phoneNumber,String email,String studentNumber,String pwd, String faculty, String department, String gradeLevel, String annual_fee,ArrayList<String> courses, ArrayList<String> grades, ArrayList<String> clubs, ArrayList<String> clubDescriptions)

4.3.5 Class: FileStuff

- Purpose: Performing file-related operations
- Constraints: None

4.3.5.1 Attribute Descriptions

No attribute included.

4.3.5.2 Method Descriptions

9. Method: read()
   Return Type: Admin
   Purpose: Reading admin.ser and returning the admin inside of the file

10. Method: read(String pathname)
    Return Type: HashMap<Integer,Person>
    Purpose: Reading “instructor.ser” and “student.ser” and returning the hashmap which is inside of the given “pathname”

11. Method: editStudents(int userId, String newPhone, String newEmail, String newAddress)
    Return Type: HashMap<Integer,Person>
    Purpose: Editing students hashmap and writing updated version to the students.ser file.

12. Method: write(HashMap<Integer,Person> data,String pathname)
    Return Type: void
    Purpose: Writing the given “data” to the given “pathname”.

13. Method: write(Admin admin,String pathname)

Return Type: void
Purpose: Writing the given “admin” to the given “pathname”.

14. Method: editInstructors (int userId, String newPhone, String newEmail, String newAddress)
Return Type: HashMap<Integer,Person>
Purpose: Editing the given instructor and writing updated hashmap to the instructors.ser file.

15. Method: editStudentGrades(int userId,ArrayList<String> newGrades)
Return Type: HashMap<Integer,Person>
Purpose: Editing the given students’ grades and writing updated hashmap to the students.ser file.

16. Method: addUser(Person person,String pathname)
Return Type: void
Purpose: Adding the given person to the given pathname.

17. Method: deleteUser(int userId,String pathname)
Return Type: void
Purpose: Deleting the given person from the given pathname.

4.3.6 Other Classes: Frame Classes (extends JFrame)

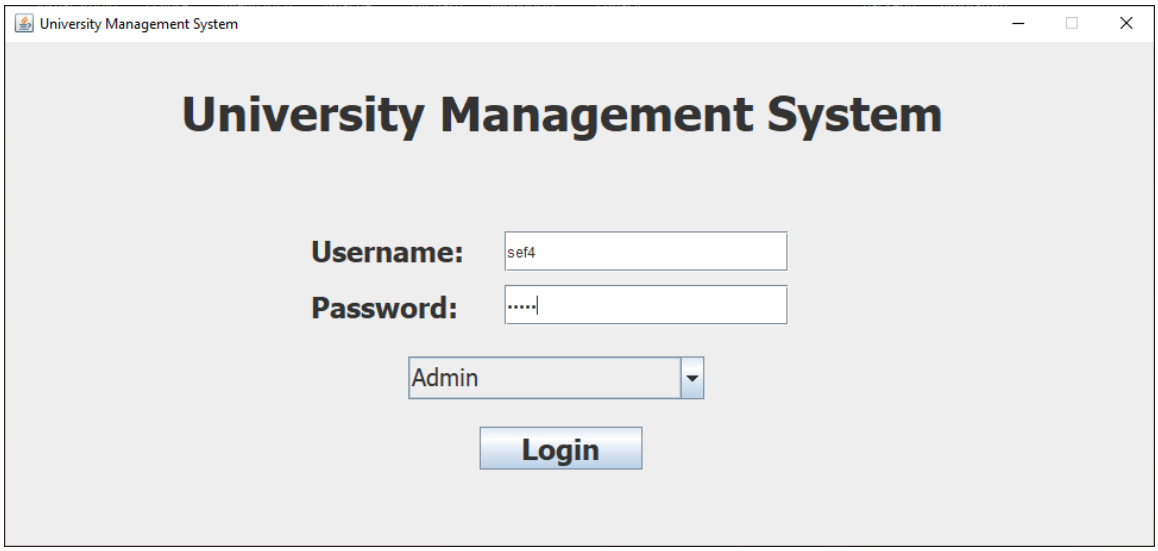
- Purpose: Providing GUI
- Constraints: None
- They include a lot of buttons, fields, labels etc., and they use the methods inside of the FileStuff class to perform operations

5 Dynamic Model

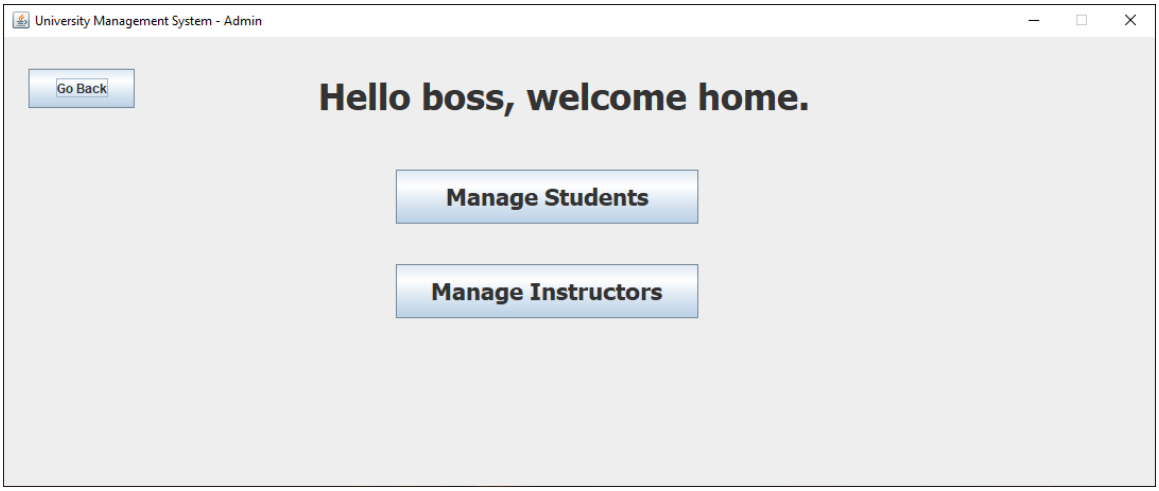
Since the main diagram is given in the part “4.2 Class Diagram”, no diagram is provided in this section.

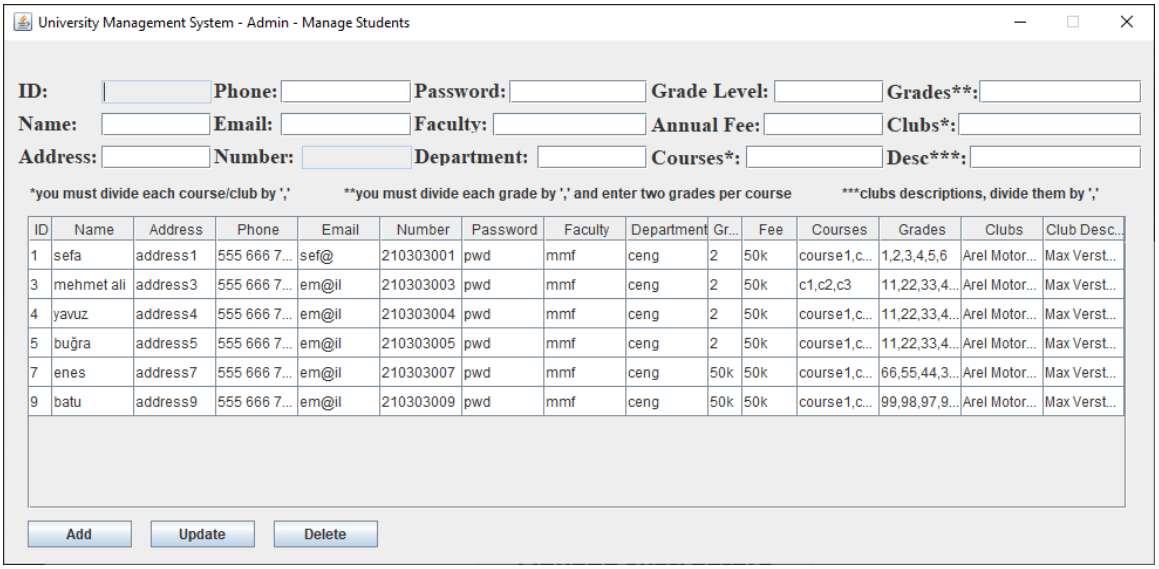
5.1 Scenario 1: You are the admin.

- *Scenario Description: Firstly, you have to login as admin using MainLoginFrame by checking the “Admin” option in the checkbox.*

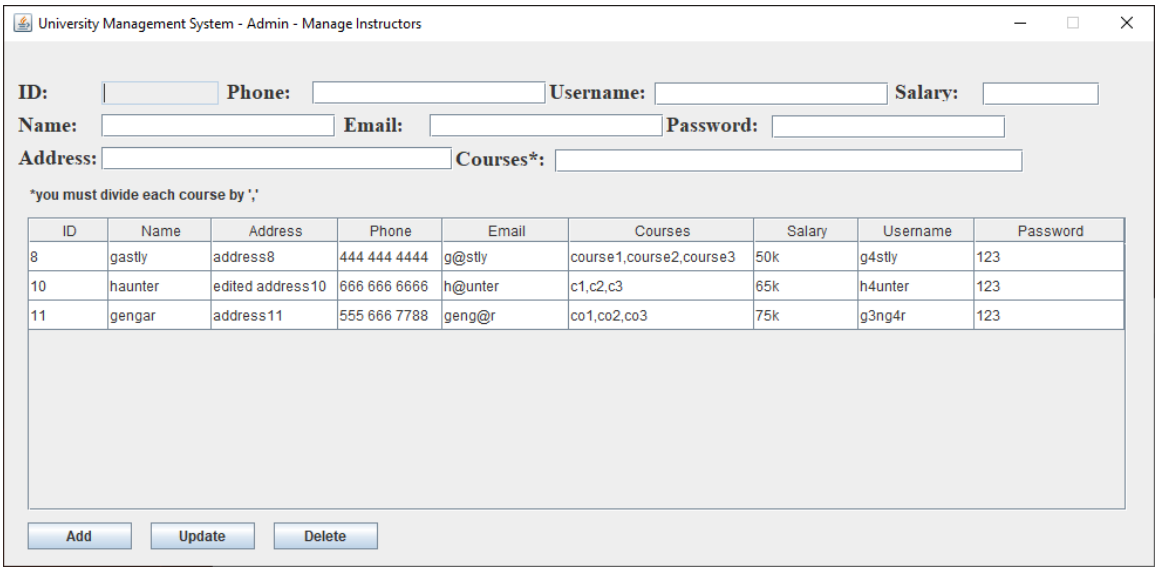


Once you filled in the inputs correctly, you will see two buttons as “Manage Students” and “Manage Instructors”. You can perform CRUD operations on the user type you want to perform, by simply clicking one of the buttons.





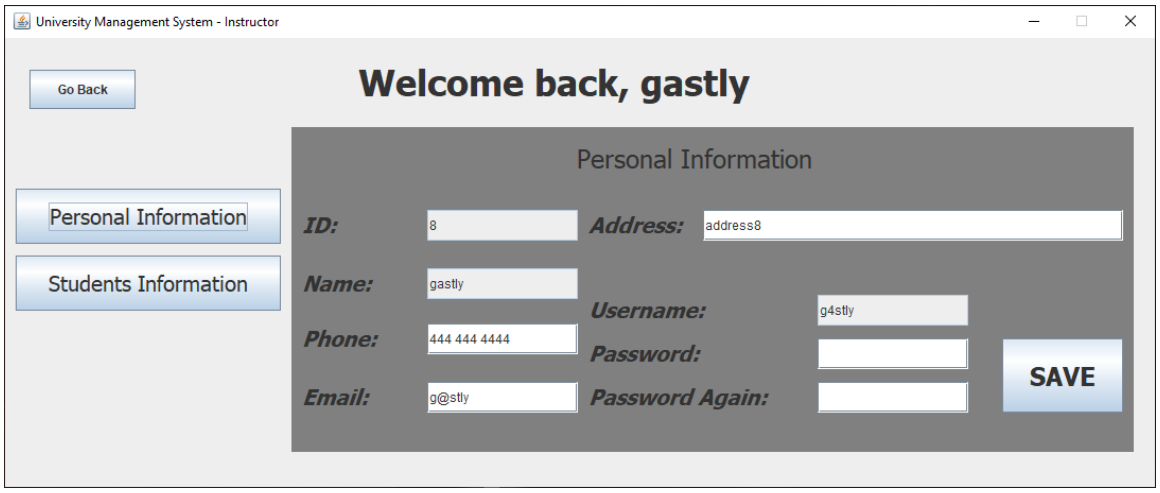
ID	Name	Address	Phone	Email	Number	Password	Faculty	Department	Gr...	Fee	Courses	Grades	Clubs	Club Desc...
1	sefa	address1	555 666 7...	sef@	210303001	pwd	mmf	ceng	2	50k	course1,c...	1,2,3,4,5,6	Arel Motor...	Max Verst...
3	mehmet ali	address3	555 666 7...	em@il	210303003	pwd	mmf	ceng	2	50k	c1,c2,c3	11,22,33,4...	Arel Motor...	Max Verst...
4	yavuz	address4	555 666 7...	em@il	210303004	pwd	mmf	ceng	2	50k	course1,c...	11,22,33,4...	Arel Motor...	Max Verst...
5	buğra	address5	555 666 7...	em@il	210303005	pwd	mmf	ceng	2	50k	course1,c...	11,22,33,4...	Arel Motor...	Max Verst...
7	enes	address7	555 666 7...	em@il	210303007	pwd	mmf	ceng	50k	50k	course1,c...	66,55,44,3...	Arel Motor...	Max Verst...
9	batu	address9	555 666 7...	em@il	210303009	pwd	mmf	ceng	50k	50k	course1,c...	99,98,97,9...	Arel Motor...	Max Verst...

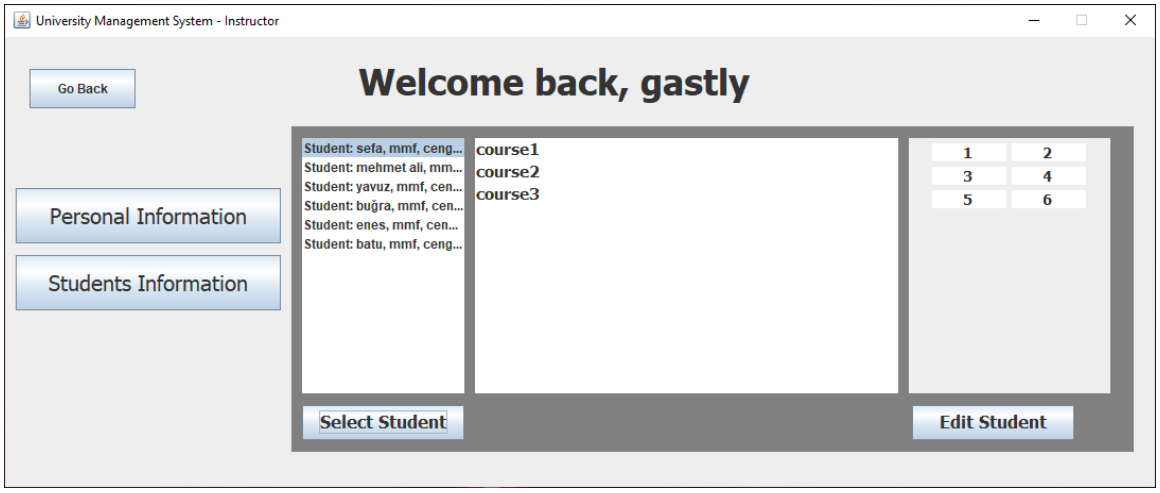


5.2 Scenario 2: You are an instructor.

- *Scenario Description: Firstly, you have to login as instructor using MainLoginFrame by checking the "Instructor" option in the checkbox. Once you filled in the inputs correctly, you will see two buttons as "Personal Information and "Students Information".*

You can see and edit your personal information from the panel which will be opened after clicking on "Personal Information" button. From the panel which will be opened after clicking on "Students Information" button, you can see students' courses and edit their grades.

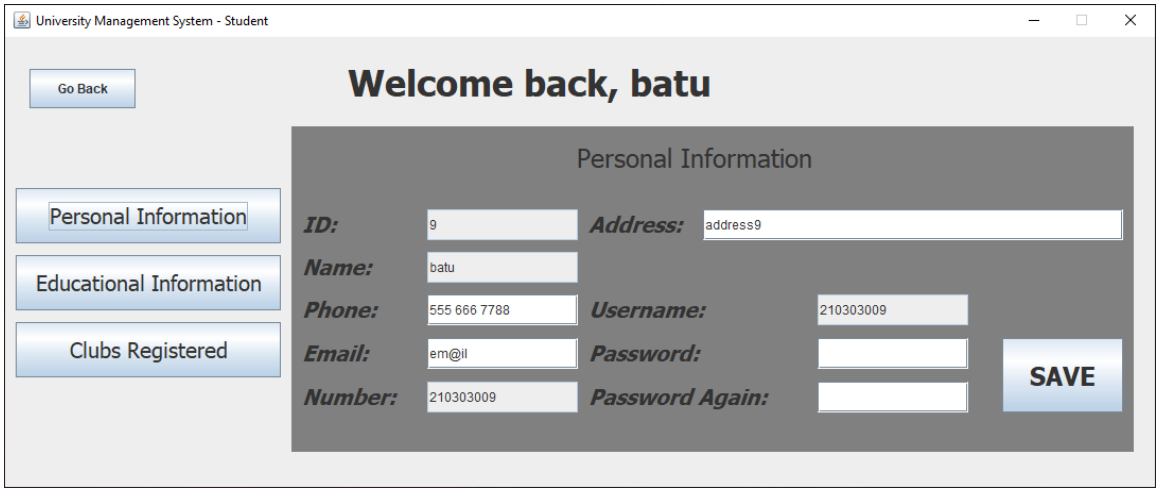


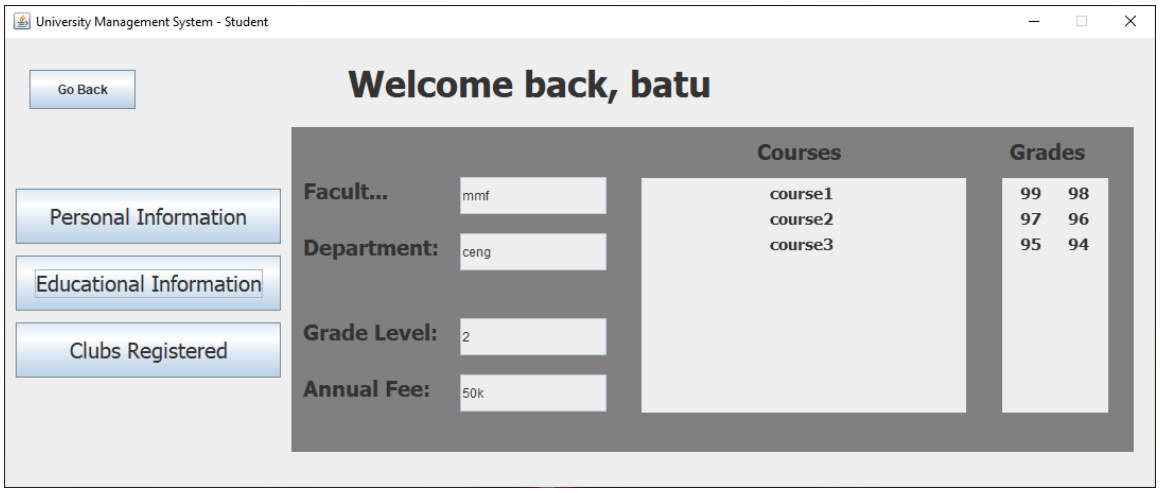


5.3 Scenario 3: You are a student.

- *Scenario Description: Firstly, you have to login as student using MainLoginFrame by checking the "Student" option in the checkbox. Once you filled in the inputs correctly, you will see three buttons: "Personal Information", "Educational Information", "Clubs Registered".*

By clicking the first button, you can see and edit your personal information. From the second one, you can see your academical information like your faculty, courses etc. By clicking the last button, you can see the clubs you registered and their short descriptions.




University Management System - Student
Go Back
Welcome back, batu
Personal Information
Educational Information
Clubs Registered
Facult...
mmf
Department:
ceng
Grade Level:
2
Annual Fee:
50k
Courses
course1
course2
course3
Grades
99 98
97 96
95 94


University Management System - Student
Go Back
Welcome back, batu
Personal Information
Educational Information
Clubs Registered
Arel Motorsports Club
-Max Verstappen has fresher faster tyres -2021 Abu Dhabi
club2
-club desc 2
club3
- club desc 3