KAZAKH KBTU BRITISH
TECHNICAL
UNIVERSITY

School of Information Technologies and Engineering

CSCI2106: Object-Oriented Programming and Design

Project Overview

University Management System

Authors: Zhanserik Kalmukhambet,

Ernat Manapaly,

Tileukhan Alibai,

Abdullina Alina

Project Overview

Project Objectives

Create a university system Java application that provides basic functionality for different types of users (students, teachers, etc.) and facilitates interactions between them.

Project Team

Zhanserik Kalmukhambet – leaded the team, organize all process and bring all codes together;

Ernat Manapaly – implemented manager and teacher functionality, also realize Attendance class;

Alina Abdullina – implemented library functionality, proposed design pattern usage (decorator, mediator, factory)

Tileukhan Alibai – implemented admin functionality, construct Schedule model;

Phases of work and Technologies Used

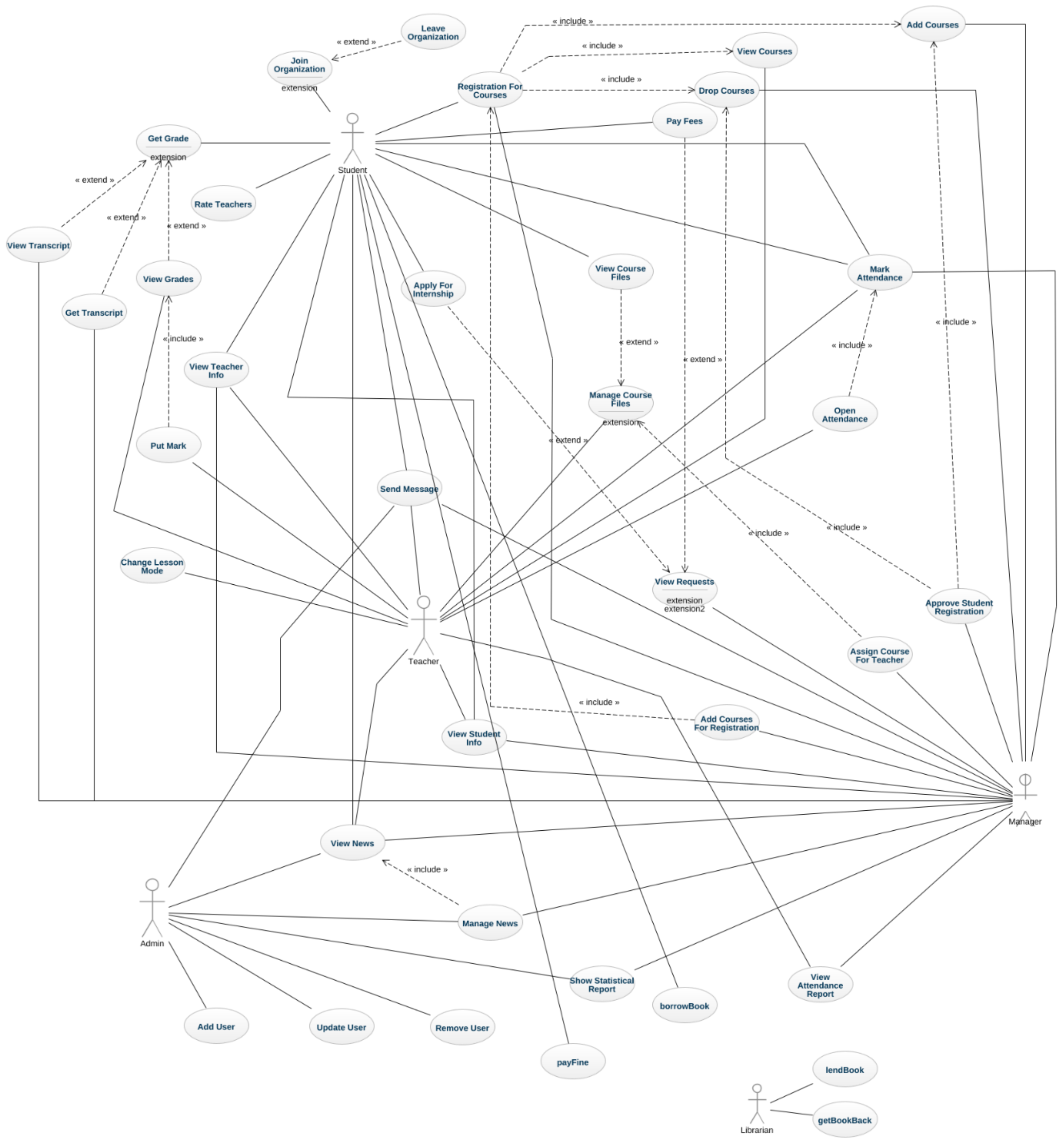
For our project we followed the Waterfall Software Development model.

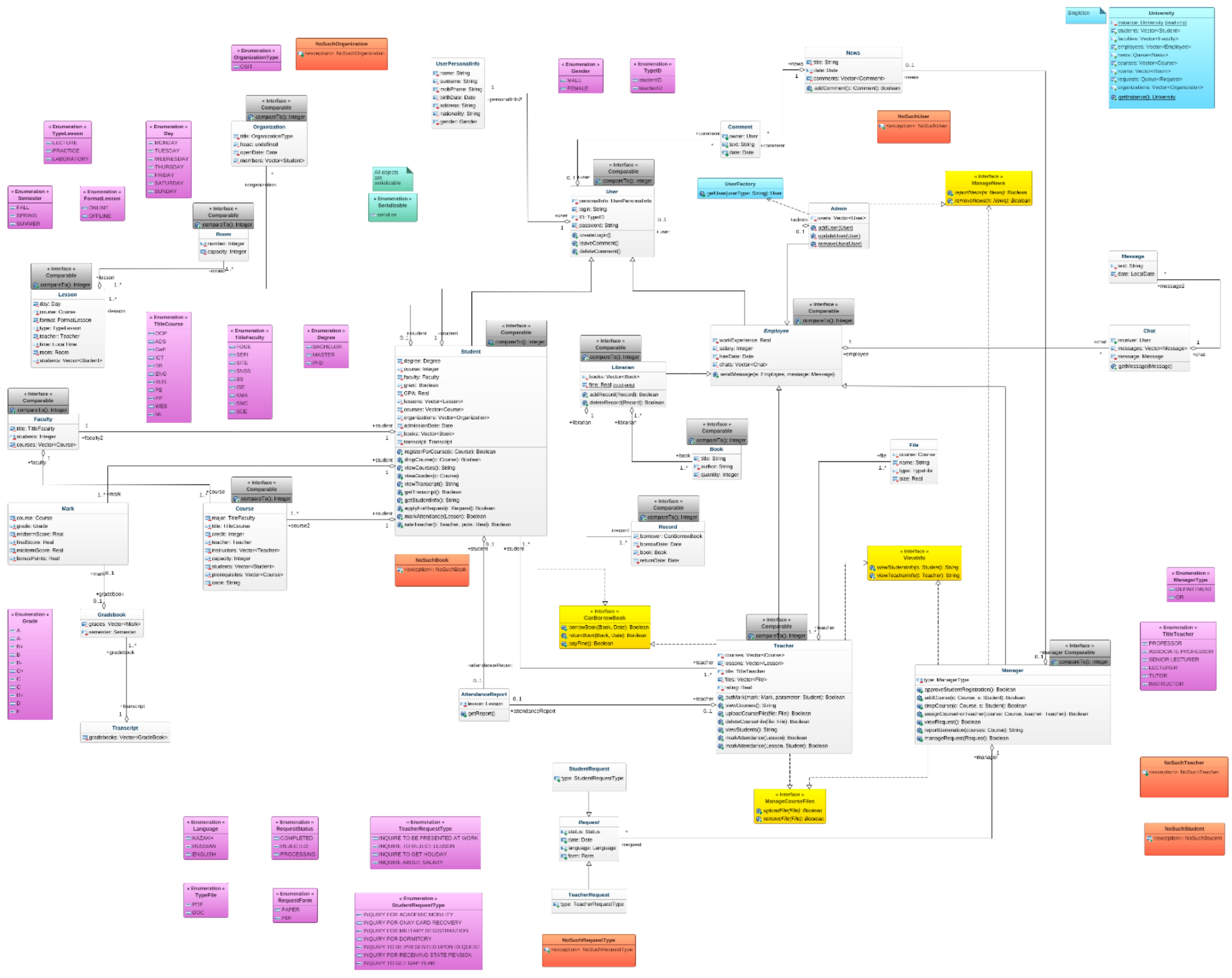
1. Requirements Gathering

First of all, we carefully analyzed the project description provided to outline key requirements.

2. Use Case and Class Diagrams – TopCoder, GenMyModel

We created use case diagrams to visualize classes and functionalities using TopCoder. However, due to corporate access issues we switched to GenMyModel.


« include »
Add Courses
Leave Organization
« include »
View Courses
« extend »
Join Organization
extension
« include »
Registration For Courses
Drop Courses
Pay Fees
Get Grade
extension
Student
« extend »
Rate Teachers
« extend »
« extend »
View Transcript
View Grades
View Course Files
Mark Attendance
Get Transcript
Apply For Internship
« include »
« extend »
« include »
« include »
« extend »
View Teacher Info
Open Attendance
Manage Course Files
extension
« extend »
Put Mark
Send Message
« include »
« include »
Change Lesson Mode
View Requests
extension
extension2
Approve Student Registration
Teacher
Assign Course For Teacher
« include »
Add Courses For Registration
View Student Info
Manager
View News
« include »
Admin
Manage News
Show Statistical Report
View Attendance Report
borrowBook
Add User
Update User
Remove User
payFine
lendBook
Librarian
getBookBack

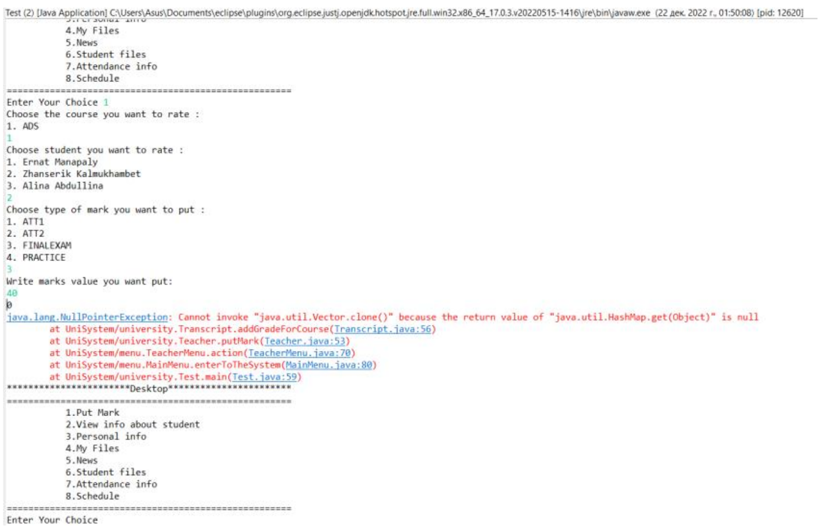


3. Coding – Eclipse

We made and re-made design decisions and built the coding solutions based on them. The Class Diagrams were altered in accordance to changes made on this stage.

4. Testing

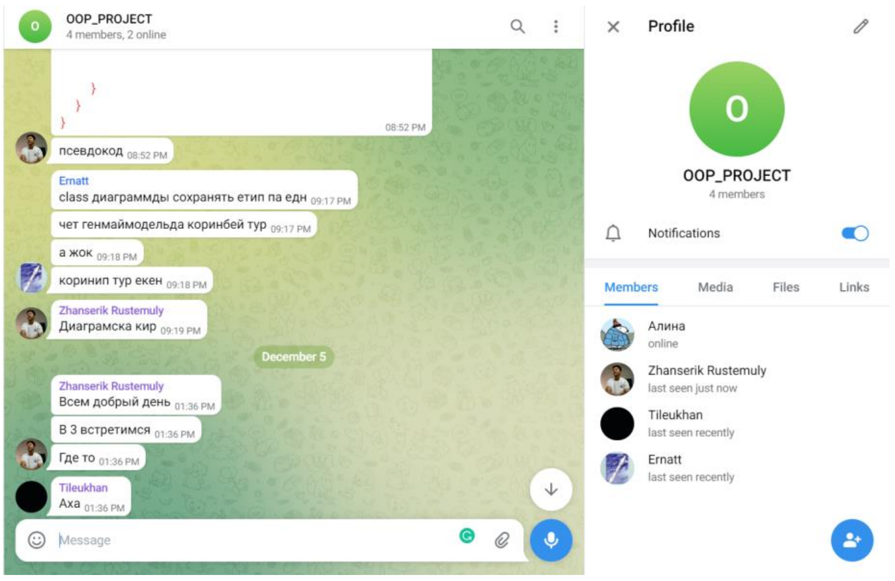
We tested each application users' functionality by separate test classes, fixing arising issues along the way.



```
Test (2) [Java Application] C:\Users\Asus\Documents\eclipse\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win32.x86_64_17.0.3.v20220515-1416\jre\bin\javaw.exe  (22 дек. 2022 г., 01:50:08) [pid: 12620]
           4.My Files
           5.News
           6.Student files
           7.Attendance info
           8.Schedule
=====================================================
Enter Your Choice 1
Choose the course you want to rate :
1. ADS
1
Choose student you want to rate :
1. Ernat Manapaly
2. Zhanserik Kalmukhambet
3. Alina Abdullina
2
Choose type of mark you want to put :
1. ATT1
2. ATT2
3. FINALEXAM
4. PRACTICE
3
Write marks value you want put:
40
0
java.lang.NullPointerException: Cannot invoke "java.util.Vector.clone()" because the return value of "java.util.HashMap.get(Object)" is null
        at UniSystem/university.Transcript.addGradeForCourse(Transcript.java:56)
        at UniSystem/university.Teacher.putMark(Teacher.java:53)
        at UniSystem/menu.TeacherMenu.action(TeacherMenu.java:70)
        at UniSystem/menu.MainMenu.enterToTheSystem(MainMenu.java:80)
        at UniSystem/university.Test.main(Test.java:59)
***********************Desktop***********************
=====================================================
           1.Put Mark
           2.View info about student
           3.Personal info
           4.My Files
           5.News
           6.Student files
           7.Attendance info
           8.Schedule
=====================================================
Enter Your Choice
```

Project Management

We used Telegram for communication – we shared code there and arranged offline meetings. Major part of meetings has been held offline at the university.


OOP_PROJECT
4 members, 2 online

```
      }
   }
}
```
08:52 PM

псевдокод 08:52 PM

Ernatt
class диаграммды сохранять етип па едн 09:17 PM

чет генмаймодельда коринбей тур 09:17 PM

а жок 09:18 PM

коринип тур екен 09:18 PM

Zhanserik Rustemuly
Диаграмска кир 09:19 PM

December 5

Zhanserik Rustemuly
Всем добрый день 01:36 PM

В 3 встретимся 01:36 PM

Где то 01:36 PM

Tileukhan
Аха 01:36 PM

Profile

OOP_PROJECT
4 members

Notifications

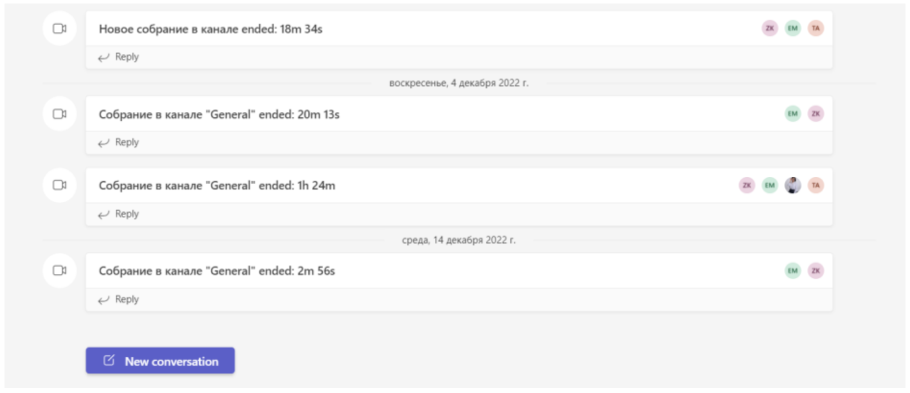
Алина
online

Zhanserik Rustemuly
last seen just now

Tileukhan
last seen recently

Ernatt
last seen recently

Telegram chat


Новое собрание в канале ended: 18m 34s

воскресенье, 4 декабря 2022 г.

Собрание в канале "General" ended: 20m 13s

Собрание в канале "General" ended: 1h 24m

среда, 14 декабря 2022 г.

Собрание в канале "General" ended: 2m 56s

Team in Microsoft Teams

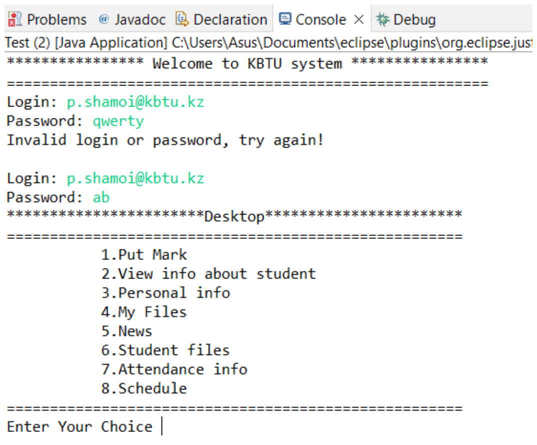
Results

1. System Initialization

Before logging in, our system first adds a user and assigns them login and password. Login generation is similar to that of the KBTU. Available functionality is to be provided depending on the user's class.

2. Log In

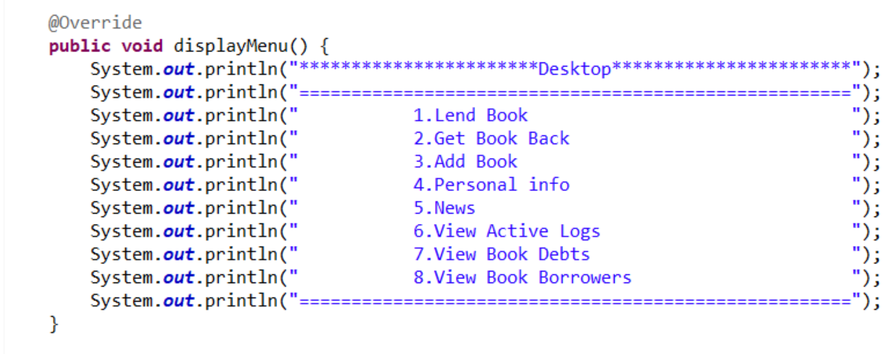
The system checks login and password for validity, after entering the menu with available actions is shown.



```
Test (2) [Java Application] C:\Users\Asus\Documents\eclipse\plugins\org.eclipse.jus
*************** Welcome to KBTU system ***************
========================================================
Login: p.shamoi@kbtu.kz
Password: qwerty
Invalid login or password, try again!

Login: p.shamoi@kbtu.kz
Password: ab
*********************Desktop*********************
=====================================================
           1.Put Mark
           2.View info about student
           3.Personal info
           4.My Files
           5.News
           6.Student files
           7.Attendance info
           8.Schedule
=====================================================
Enter Your Choice |
```

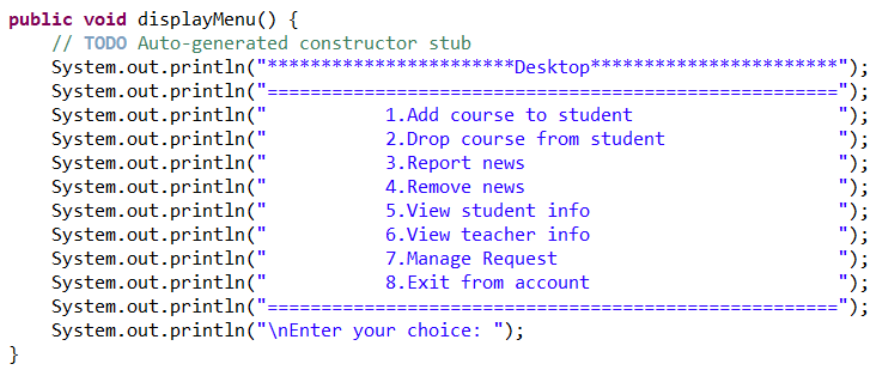
3. Commands

The system dynamically decides which menu to show – Teacher's, Student's, Manager's, Librarian's, or Admin's. The commands have to be chosen manually.



```java
@Override
public void displayMenu() {
    System.out.println("**********************Desktop**********************");
    System.out.println("===================================================");
    System.out.println("          1.Lend Book                              ");
    System.out.println("          2.Get Book Back                          ");
    System.out.println("          3.Add Book                               ");
    System.out.println("          4.Personal info                          ");
    System.out.println("          5.News                                   ");
    System.out.println("          6.View Active Logs                       ");
    System.out.println("          7.View Book Debts                        ");
    System.out.println("          8.View Book Borrowers                    ");
    System.out.println("===================================================");
}
```

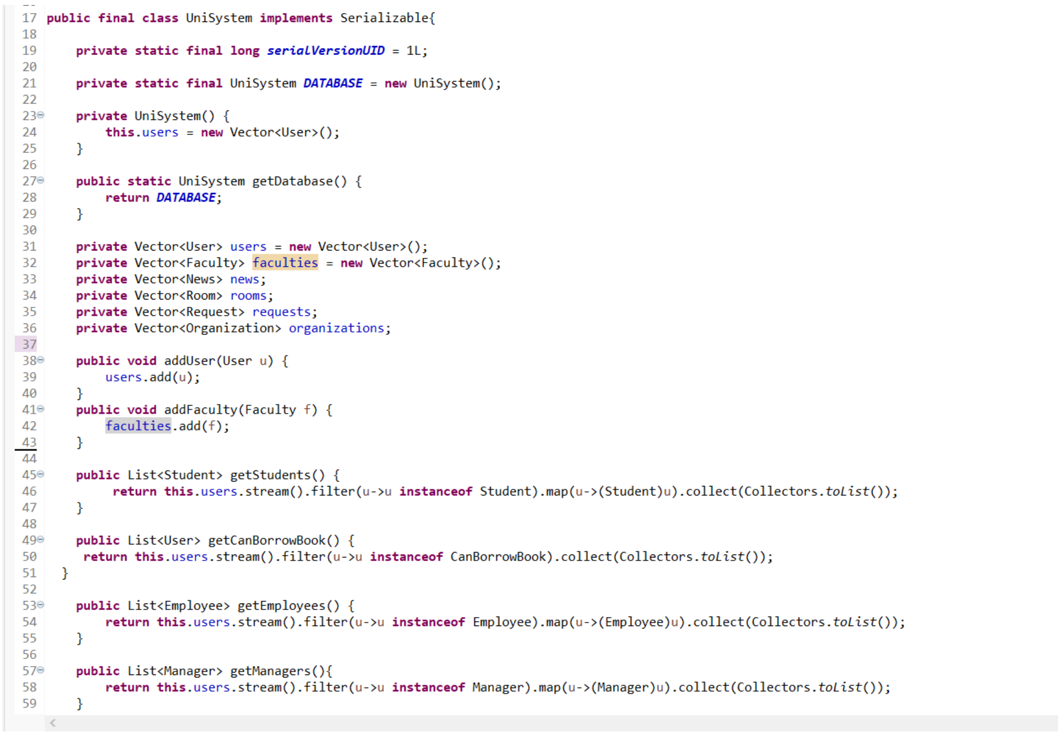
Example: Librarian Menu



```java
public void displayMenu() {
    // TODO Auto-generated constructor stub
    System.out.println("**********************Desktop**********************");
    System.out.println("=====================================================");
    System.out.println("          1.Add course to student                    ");
    System.out.println("          2.Drop course from student                 ");
    System.out.println("          3.Report news                              ");
    System.out.println("          4.Remove news                              ");
    System.out.println("          5.View student info                        ");
    System.out.println("          6.View teacher info                        ");
    System.out.println("          7.Manage Request                           ");
    System.out.println("          8.Exit from account                        ");
    System.out.println("=====================================================");
    System.out.println("\nEnter your choice: ");
}
```

Example: Manager Menu

4. Database



```java
public final class UniSystem implements Serializable{

    private static final long serialVersionUID = 1L;

    private static final UniSystem DATABASE = new UniSystem();

    private UniSystem() {
        this.users = new Vector<User>();
    }

    public static UniSystem getDatabase() {
        return DATABASE;
    }

    private Vector<User> users = new Vector<User>();
    private Vector<Faculty> faculties = new Vector<Faculty>();
    private Vector<News> news;
    private Vector<Room> rooms;
    private Vector<Request> requests;
    private Vector<Organization> organizations;

    public void addUser(User u) {
        users.add(u);
    }
    public void addFaculty(Faculty f) {
        faculties.add(f);
    }

    public List<Student> getStudents() {
         return this.users.stream().filter(u->u instanceof Student).map(u->(Student)u).collect(Collectors.toList());
    }

    public List<User> getCanBorrowBook() {
     return this.users.stream().filter(u->u instanceof CanBorrowBook).collect(Collectors.toList());
  }

    public List<Employee> getEmployees() {
        return this.users.stream().filter(u->u instanceof Employee).map(u->(Employee)u).collect(Collectors.toList());
    }

    public List<Manager> getManagers(){
        return this.users.stream().filter(u->u instanceof Manager).map(u->(Manager)u).collect(Collectors.toList());
    }
```

Example: UniSystem class storing Database

We have a serializable singleton class UniSystem where the database is stored. There is only one container for all the system users, but we effectively fetch different user types via stream.

5. Menu Sample Code

Below you can see an example of a working menu made for Manager.

```
	private void addCourseToStudent() throws Exception {
		System.out.println("Choose course that you want add to student : ");
		int ind = 1;
		for(Course c : UniSystem.getDatabase().getCourses()) {
			System.out.println(ind + ". " + c.getTitle());
			ind++;
		}

		int chCourse = sc.nextInt();
		Course course = UniSystem.getDatabase().getCourses().get(chCourse-1);
		System.out.println("Choose student whom want add to course : ");
		ind = 1;
		for(Student s : UniSystem.getDatabase().getStudents()) {
			System.out.println(ind + ". " + s.getName() + " " + s.getSurname());
			ind++;
		}
		int chStudent = sc.nextInt();
		Student student = UniSystem.getDatabase().getStudents().get(chStudent-1);
		if(student.getCourses().contains(course)) {
			System.out.println(student.getName() + " has already registred for " + 
course.getTitle() + "\n\n");
			addingCourse = false;
		}else {
			((Manager)user).addCoursetoStudent(course, student);
			System.out.println(course.getTitle()  + " was successfully added to " + 
student.getName() + "\n\n");
			addingCourse = true;
		}
	}
private void DropCourseFromStudent() throws Exception {
		System.out.println("Choose student to drop from the course : ");
		int ind = 1;
		for(Student s : UniSystem.getDatabase().getStudents()) {
			System.out.println(ind + ". " + s.getName() + " " + s.getSurname());
			ind++;
		}
		int chStudent = sc.nextInt();
		Student student = UniSystem.getDatabase().getStudents().get(chStudent-1);
		if(student.getCourses().size()==0) {
			System.out.println(student.getName() + " has no lesson!");
			droppingCourse = false;
		}else {
			System.out.println("Choose students's course that you want to drop : ");
			ind = 1;
			for(Course c : student.getCourses()) {
				System.out.println(ind + ". " + c.getTitle());
				ind++;
			}
			int chCourse = sc.nextInt();
			Course course = student.getCourses().get(chCourse-1);
			((Manager)user).dropCoursefromStudent(course, student);
			System.out.println(course.getTitle()  + " was saccessfully droped for + " + 
student.getName() + "\n\n");
			droppingCourse = true;
		}
	}

private void reportNews() throws Exception {
		News n = new News();
		System.out.println("In which language will news be reported? ");
		Language[] ll = Language.values();
		int ind = 1;
		for (Language dir : ll) {
			System.out.println(ind + ". " + dir);
			ind++;
		}
		int chLanguage = sc.nextInt();
		Language l = (Language)Array.get(ll, chLanguage-1);
		System.out.print("Enter news title: ");
```

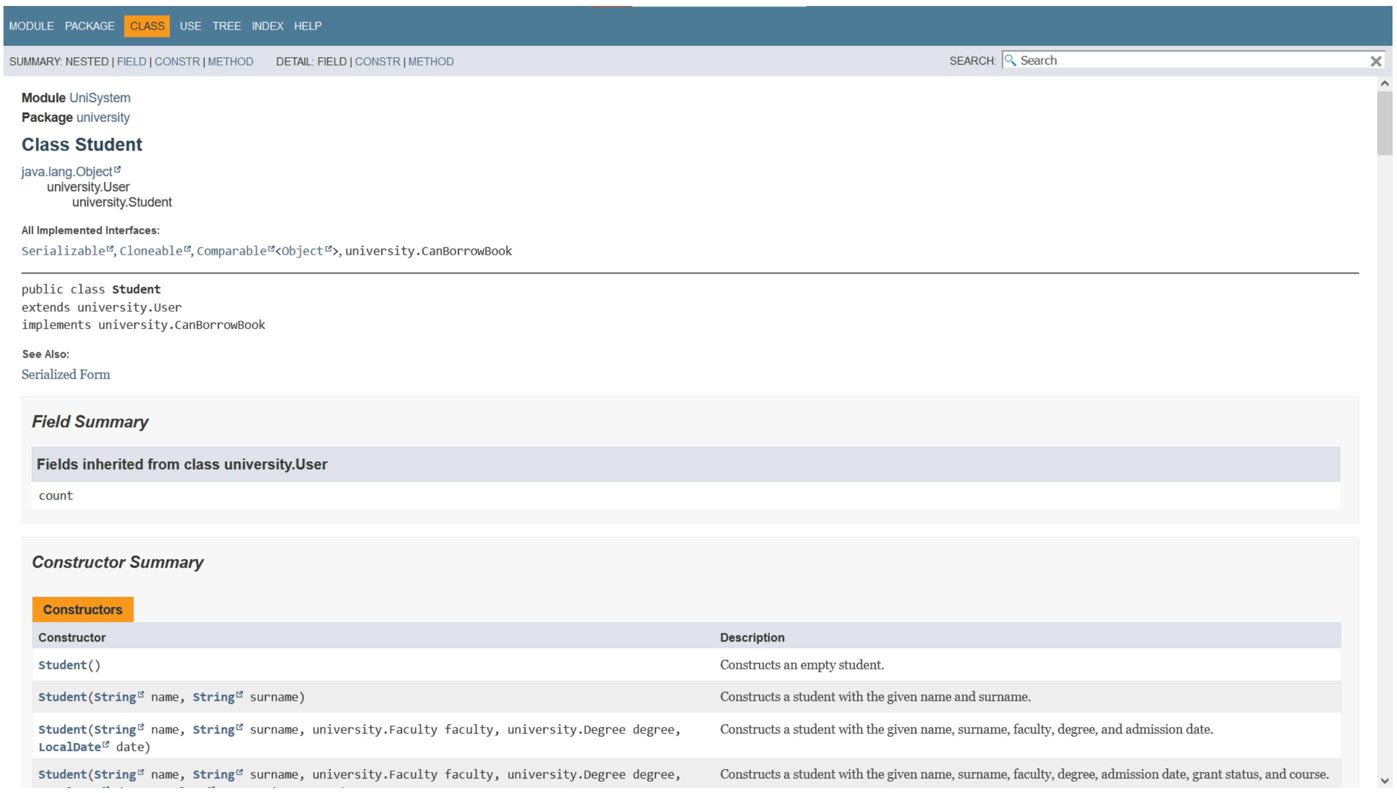
```
            String tit = sc.next();
            n.setTitle(tit);
            if(n.getTitle().length()!=0) {
                  System.out.print("Enter information about news: ");
                  String inf = sc.next();
                  n.setInfo(inf);
                  if(n.getInfo().length()>0) {
                        ((Manager)user).reportNews(n);
                        System.out.println("News was successfully reported!");
                        reportingNews = true;
                  }
            }
      }

private void removeNews() throws Exception {
            System.out.println("Choose news that you want remove: ");
            if(UniSystem.getDatabase().getNews().size() == 0) {
                  System.out.println("No news in system");
            }else {
                  int ind = 1;
                  for(News n: UniSystem.getDatabase().getNews()) {
                        System.out.println(ind + ". " + n.getTitle());
                        ind++;
                  }
                  int chNews = sc.nextInt();
                  News news = UniSystem.getDatabase().getNews().get(chNews-1);
                  ((Manager)user).removeNews(news);
                  if(chNews > 0) {
                        System.out.println("News successfully deleted!");
                  }
            }
      }

private void viewStudentInfo() throws Exception {
            System.out.println("select the student from whom you want to view the full information :
");
            int ind = 1;
            for(Student s : UniSystem.getDatabase().getStudents()) {
                  System.out.println(ind + ". " + s.getName() +  " "  + s.getSurname());
                  ind++;
            }
            int chStudent = sc.nextInt();
            Student s = UniSystem.getDatabase().getStudents().get(chStudent-1);
            System.out.println(s.toString());
      }

private void viewTeacherInfo() throws Exception {
            System.out.println("select the teacher from whom you want to view the full information :
");
            int ind = 1;
            for(Teacher s : UniSystem.getDatabase().getTeachers()) {
                  System.out.println(ind + ". " + s.getName() +  " "  + s.getSurname());
                  ind++;
            }
            int chTeacher = sc.nextInt();
            Teacher t = UniSystem.getDatabase().getTeachers().get(chTeacher-1);
            System.out.println(t.toString());
      }

private void manageRequest() throws Exception{
            System.out.println("Choose request that you want execute: ");
            int ind = 1;
            for(Request r : UniSystem.getDatabase().getRequests()) {
                  System.out.println(ind + ". " + r.toString());
                  ind++;
            }
            int chReq = sc.nextInt();
            Request r = UniSystem.getDatabase().getRequests().get(chReq-1);
            r.setStatus(RequestStatus.COMPLETED);
            System.out.println(r.toString()  + "request successfully accepted");
      }
```

```
public void action() throws Exception {
		// TODO Auto-generated method stub
		try {
			menu : while(true){
				displayMenu();
				int choice = sc.nextInt();
				if(choice==1){
					addCourseToStudent: while(true){
						addCourseToStudent();
						if(addingCourse) {
							System.out.println("Whould you like to add another 
course to this student? \n 1.Yes \n 2.Return back \n");
							choice = sc.nextInt();
							if(choice==1) continue addCourseToStudent;
							if(choice==2) continue menu;
							break;
						}else {
							System.out.println("Whould you like to try again? \n 
1.Yes \n 2.Return back \n");
							
							choice = sc.nextInt();
							if(choice==1) continue addCourseToStudent;
							if(choice==2) continue menu;
							break;
						}
					}
				}else if(choice==2) {
					dropCourseFromStudent: while(true) {
						DropCourseFromStudent();
						if(droppingCourse) {
							System.out.println(" Whould you like to drop another 
course for this student? \n 1.Yes \n 2.Return back \n");
							choice = sc.nextInt();
							if(choice==1) continue dropCourseFromStudent;
							if(choice==2) continue menu;
							break;
						}else {
							System.out.println("You have already back to Main 
Menu");
							
							continue menu;
						}
						
					}
				}else if(choice==3) {
					reportNews : while(true) {
						reportNews();
						if(reportingNews) {
							reportingNews = false;
							System.out.println(" Whould you like to report another 
news? \n 1.Yes \n 2.Return back \n");
							
							choice = sc.nextInt();
							if(choice==1) continue reportNews;
							if(choice==2) continue menu;
							break;
						}
						break;
					}
				}else if(choice==4) {
					removeNews : while(true) {
						removeNews();
						
						System.out.println("Whould you like to remove another news? \n 
1.Yes \n 2.Return back \n ");
						choice = sc.nextInt();
						if(choice==1) continue removeNews;
						if(choice==2) continue menu;
						break;
						
					}
				}else if(choice==5) {
					viewStudentInfo : while(true) {
						viewStudentInfo();
```

```
                                                System.out.println("Whould you like to view info another 
student? \n 1.Yes \n 2.Return back \n");
                                                choice = sc.nextInt();
                                                if(choice==1) continue viewStudentInfo;
                                                if(choice==2) continue menu;
                                                break;
                                        }
                                }else if(choice==6) {
                                        viewTeacherInfo : while(true) {
                                                viewTeacherInfo();
                                                System.out.println("Whould you like to view info another 
teacher? \n 1.Yes \n 2.Return back \n");
                                                choice = sc.nextInt();
                                                if(choice==1) continue viewTeacherInfo;
                                                if(choice==2) continue menu;
                                                break;
                                        }
                                }else if(choice==7) {
                                        manageRequest : while(true) {
                                                manageRequest();
                                                System.out.println("Whould you like to accept another request? 
\n 1.Yes \n 2.Return back \n");
                                                choice = sc.nextInt();
                                                if(choice==1) continue manageRequest;
                                                if(choice==2) continue menu;
                                                break;
                                        }
                                }else if(choice==8) {
                                        exit();
                                }
                                break;
                        }
                } catch (Exception e) {
                        System.out.println("Something bad happened... \n Saving resources...");
                        e.printStackTrace();
                }
        }
```

6. Documentation

We used Javadoc for generating Java code documentation. Below is the documentation of a Student class.


MODULE PACKAGE CLASS USE TREE INDEX HELP
SUMMARY: NESTED | FIELD | CONSTR | METHOD DETAIL: FIELD | CONSTR | METHOD
SEARCH: Search

Module UniSystem
Package university

Class Student

java.lang.Object
university.User
university.Student

All Implemented Interfaces:
Serializable, Cloneable, Comparable<Object>, university.CanBorrowBook

```
public class Student
extends university.User
implements university.CanBorrowBook
```

See Also:
Serialized Form

Field Summary

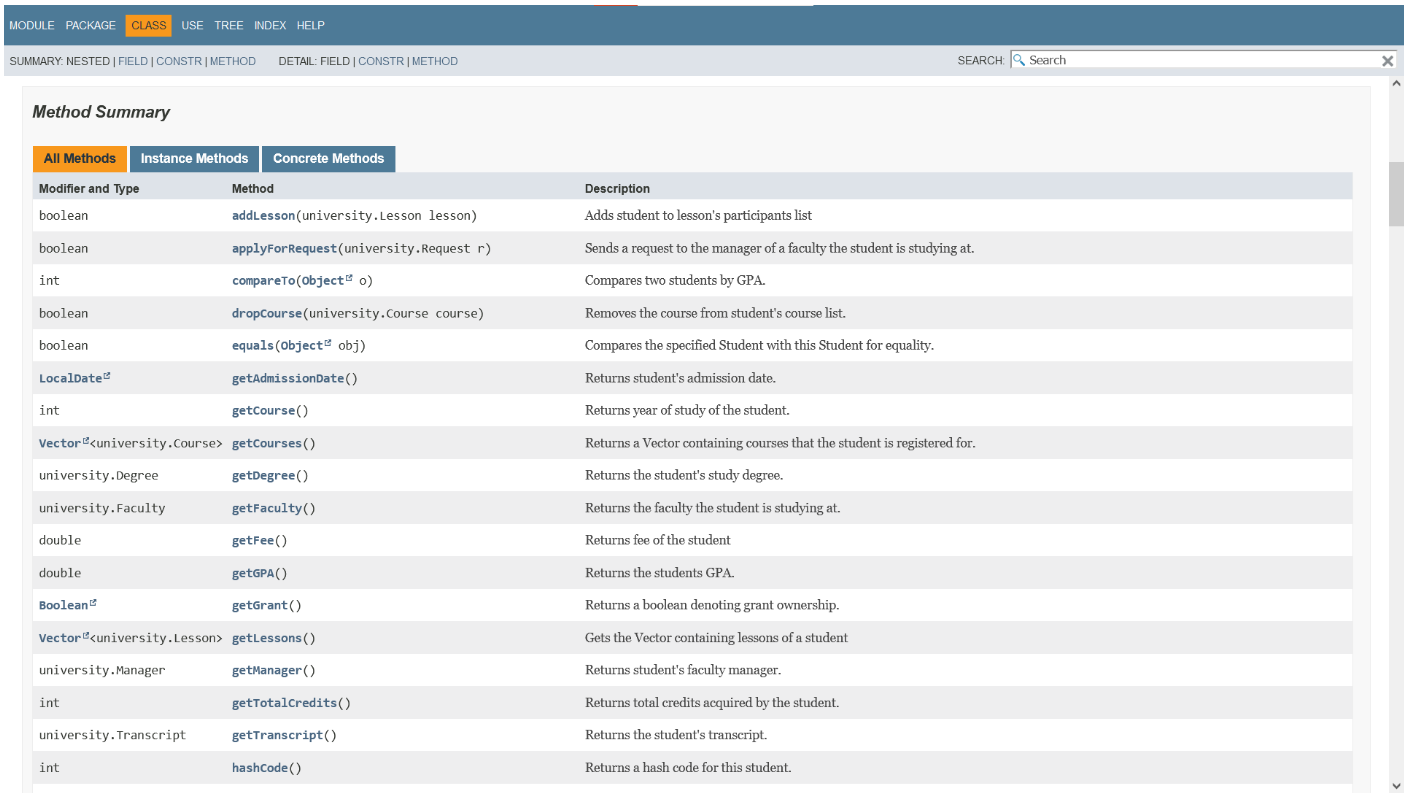
Fields inherited from class university.User

count

Constructor Summary

Constructors

Constructor	Description
`Student()`	Constructs an empty student.
`Student(String name, String surname)`	Constructs a student with the given name and surname.
`Student(String name, String surname, university.Faculty faculty, university.Degree degree, LocalDate date)`	Constructs a student with the given name, surname, faculty, degree, and admission date.
`Student(String name, String surname, university.Faculty faculty, university.Degree degree,`	Constructs a student with the given name, surname, faculty, degree, admission date, grant status, and course.


MODULE PACKAGE CLASS USE TREE INDEX HELP
SUMMARY: NESTED | FIELD | CONSTR | METHOD DETAIL: FIELD | CONSTR | METHOD
SEARCH: Search
All Methods
Instance Methods
Concrete Methods

Method Summary

Modifier and Type	Method	Description
`boolean`	`addLesson(university.Lesson lesson)`	Adds student to lesson's participants list
`boolean`	`applyForRequest(university.Request r)`	Sends a request to the manager of a faculty the student is studying at.
`int`	`compareTo(Object o)`	Compares two students by GPA.
`boolean`	`dropCourse(university.Course course)`	Removes the course from student's course list.
`boolean`	`equals(Object obj)`	Compares the specified Student with this Student for equality.
`LocalDate`	`getAdmissionDate()`	Returns student's admission date.
`int`	`getCourse()`	Returns year of study of the student.
`Vector<university.Course>`	`getCourses()`	Returns a Vector containing courses that the student is registered for.
`university.Degree`	`getDegree()`	Returns the student's study degree.
`university.Faculty`	`getFaculty()`	Returns the faculty the student is studying at.
`double`	`getFee()`	Returns fee of the student
`double`	`getGPA()`	Returns the students GPA.
`Boolean`	`getGrant()`	Returns a boolean denoting grant ownership.
`Vector<university.Lesson>`	`getLessons()`	Gets the Vector containing lessons of a student
`university.Manager`	`getManager()`	Returns student's faculty manager.
`int`	`getTotalCredits()`	Returns total credits acquired by the student.
`university.Transcript`	`getTranscript()`	Returns the student's transcript.
`int`	`hashCode()`	Returns a hash code for this student.

7. Problems

The main problem we faced was code sharing – we usually did it through Telegram, so imports and path building was very awkward. Another problem was with user's menu implementation, which, however, helped us fix the system.

8. Conclusion

We have implemented basic functionality needed for the system to function properly, reached some sophistication level and did so by trial and error. There are still plenty of features that can be implemented and require some additional skills, which, hopefully, we will acquire later on, but it is at it is by this moment.

Participating in the project truly was a nice opportunity to challenge our hard and soft skills, and we did have a great time working together as a team and combining our ideas into one great chunk we all worked hard on and admired. This all left us curious and connected, great thanks to our project tracker Pakizar Shamoi! (It also showed us that it is a good idea to start using GitHub for further projects like this))