Report on OOP project

Project name: University System

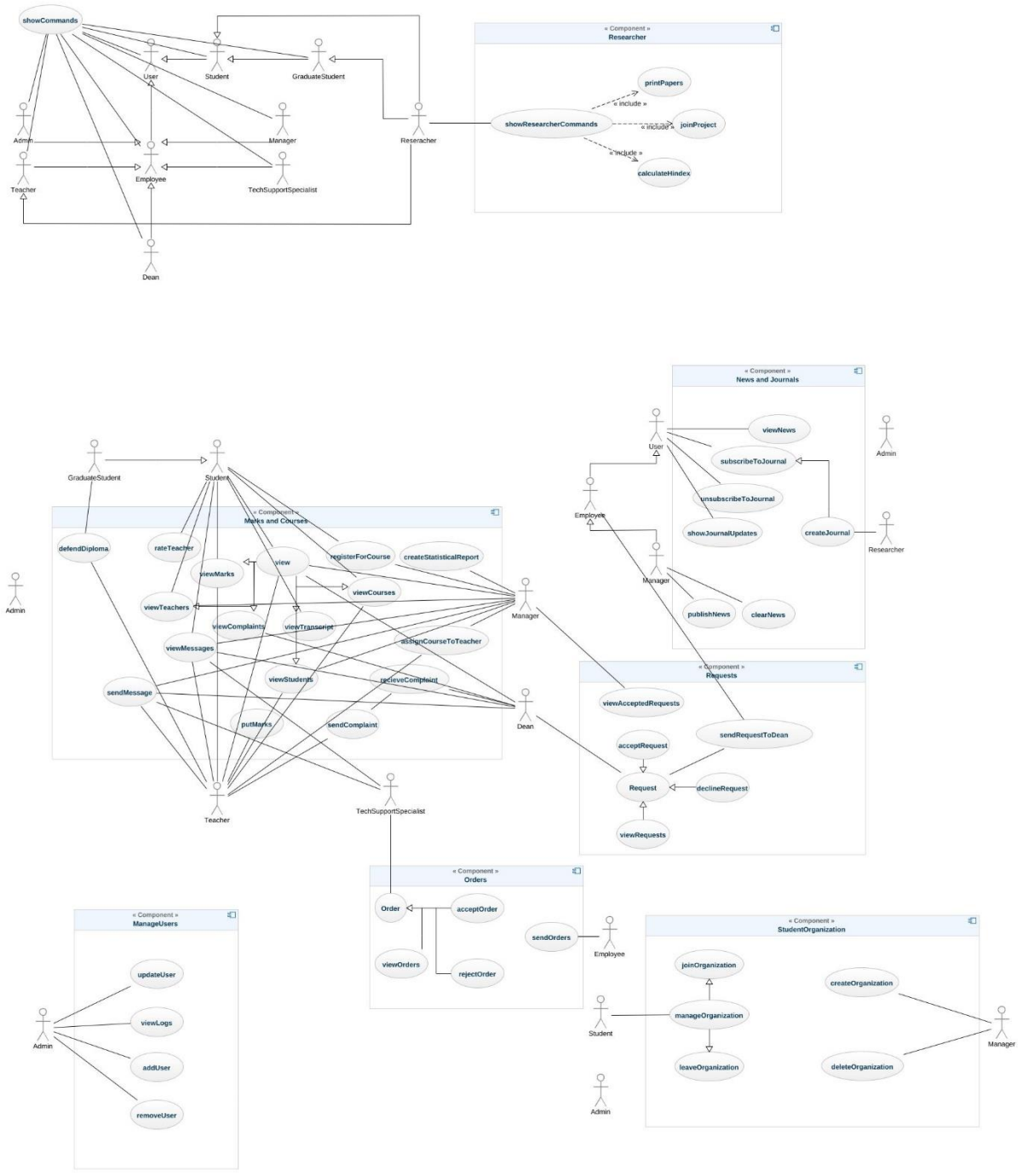
Team “False promise”:
Adilov Amir (Team leader)
Zhumadildinov Ilias
Nurushev Aldiyar

1.Introduction

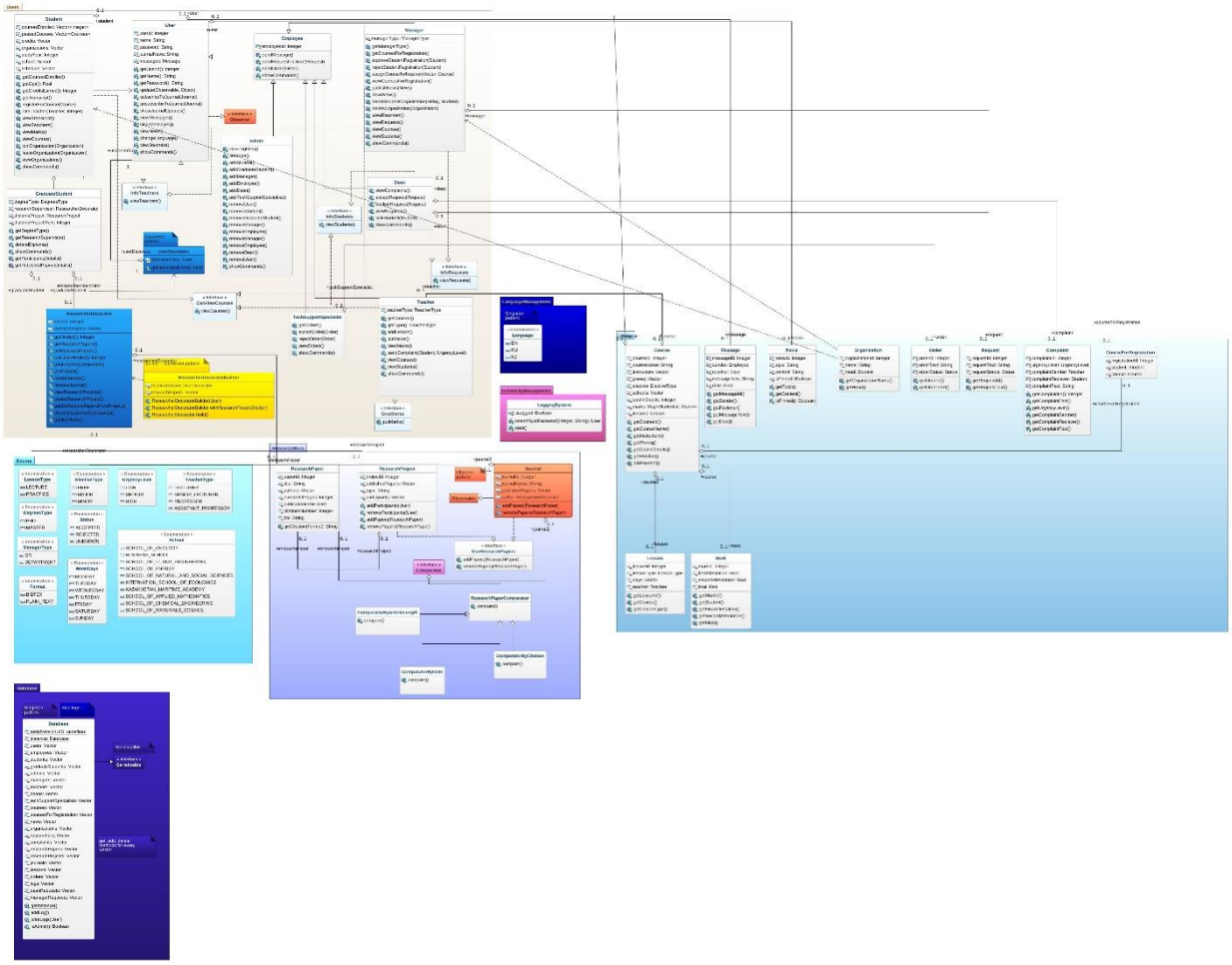
The project undertaken by the team aimed to create a comprehensive University Management System using Java, incorporating various features such as user authentication, course management, grading system, and research activities. The development process was divided into three main parts: designing UML diagrams, converting them into Java code, and creating comprehensive documentation.

2.UML Diagrams

Use case diagram:



Class diagram:



We initiated the project by constructing comprehensive diagrams to gain a clear understanding of our system's architecture. Prioritizing my preference for working with classes, I opted to create the class diagram first. This graphical representation captured the structure of classes, interfaces, their fields, methods, and the relationships between them. Additionally, we crafted a Use-case diagram to illustrate the interactions and actions performed by various actors within the system. These diagrams served as vital blueprints, guiding our subsequent development phases.

3.Classes:

1. **Admin:**
    - Admins have broad access, allowing them to manage journals, messages, news, and users.
    - They can perform administrative tasks such as adding/removing users, creating courses, and viewing system logs.

2. **Dean:**
    - Deans possess administrative powers and additional responsibilities related to student management.
    - They can handle student requests, including accepting or declining requests and addressing student complaints.
3. **Employee:**
    - Employees have functionalities similar to admins and can perform tasks such as sending requests, orders, and accessing the researcher menu.
    - They also have responsibilities related to teaching, including viewing courses and managing student-related tasks.
4. **Manager:**
    - Managers inherit functionalities from employees and have additional management-related tasks.
    - They can oversee and manage students, courses, teachers, organizations, and handle various administrative duties.
5. **Researcher:**
    - Researchers have specific functionalities related to academic research.
    - They can view and participate in research projects, manage research papers and journals, calculate H-Index, and perform other research-related tasks.
6. **Student:**
    - Students have functionalities focused on academic and organizational aspects.
    - They can manage their academic profile, view grades, register for courses, join or leave organizations, and interact with teachers.
    - Also we have a graduate student version that can defend diploma.
7. **Employee:**
    - This class includes specific commands related to employees, such as sending complaints, putting marks, and accessing the researcher menu.

Each class has a specific role and set of responsibilities within the university system, reflecting the diverse functions and tasks carried out by different user roles.

4.Documentation:

Package researchWorks

package researchWorks

All Classes and Interfaces | Interfaces | Classes

Class	Description
ComparatorByArticleLength	Comparator for comparing ResearchPaper objects based on their title lengths.
ComparatorByCitation	Comparator for comparing ResearchPaper objects based on their citation numbers.
ComparatorByDate	Comparator for comparing ResearchPaper objects based on their publication dates.
Journal	Represents a scientific journal with published research papers.
ResearchPaper	Represents a research paper with information such as title, authors, publication date, etc.
ResearchPaperComparator	Abstract class for comparing ResearchPaper objects.
ResearchProject	Represents a research project with published research papers, a topic, and participants.
UseResearchPapers	Interface for classes that can use research papers.

Module UniversityManagement

Package utility

package utility

Classes

Class	Description
Complaint	Represents a complaint in a university management system.
Course	Represents an academic course in a university management system.
CourseForRegistration	Represents a registration record for a student enrolling in a course.
Lesson	Represents a lesson or class session in a university management system.
Mark	Represents academic marks for a student in a specific course or assessment.
Message	Represents a message in a university management system.
News	Represents a news item in a university management system.
Order	Represents an order or request in a university management system.
Organization	Represents an organization or club in a university management system.
Request	Represents a request or application in a university management system.

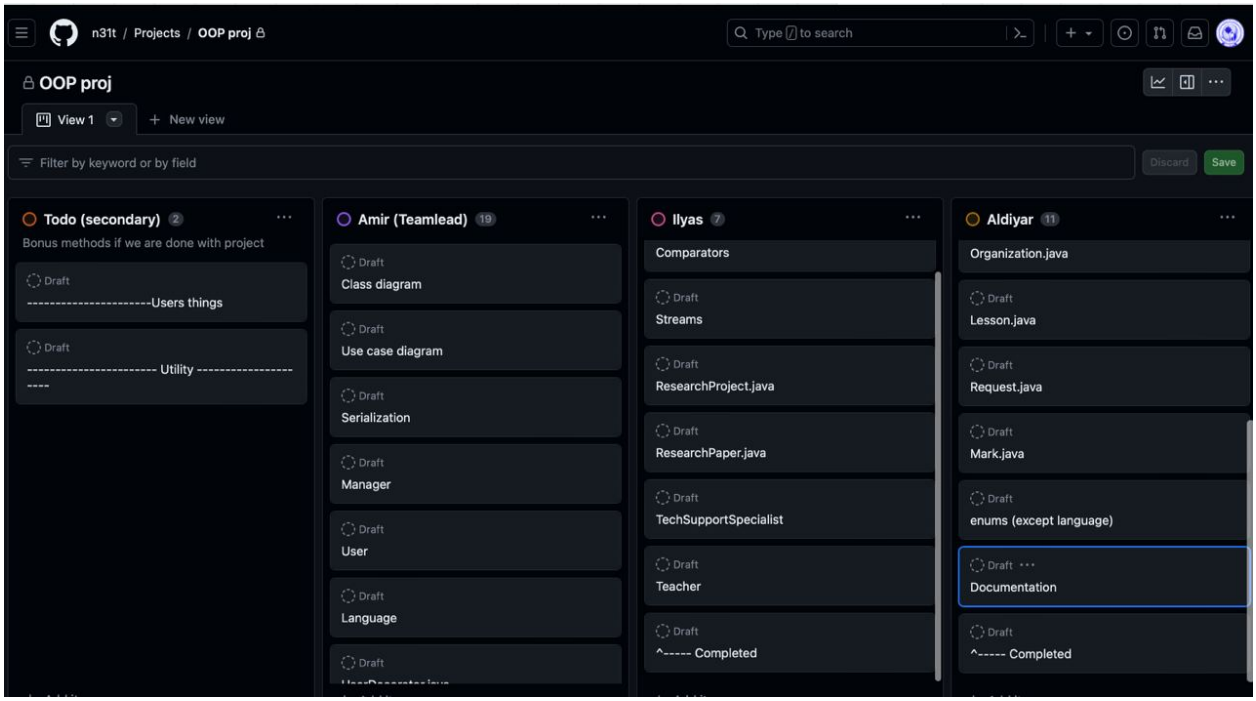
Package researchWorks

package researchWorks

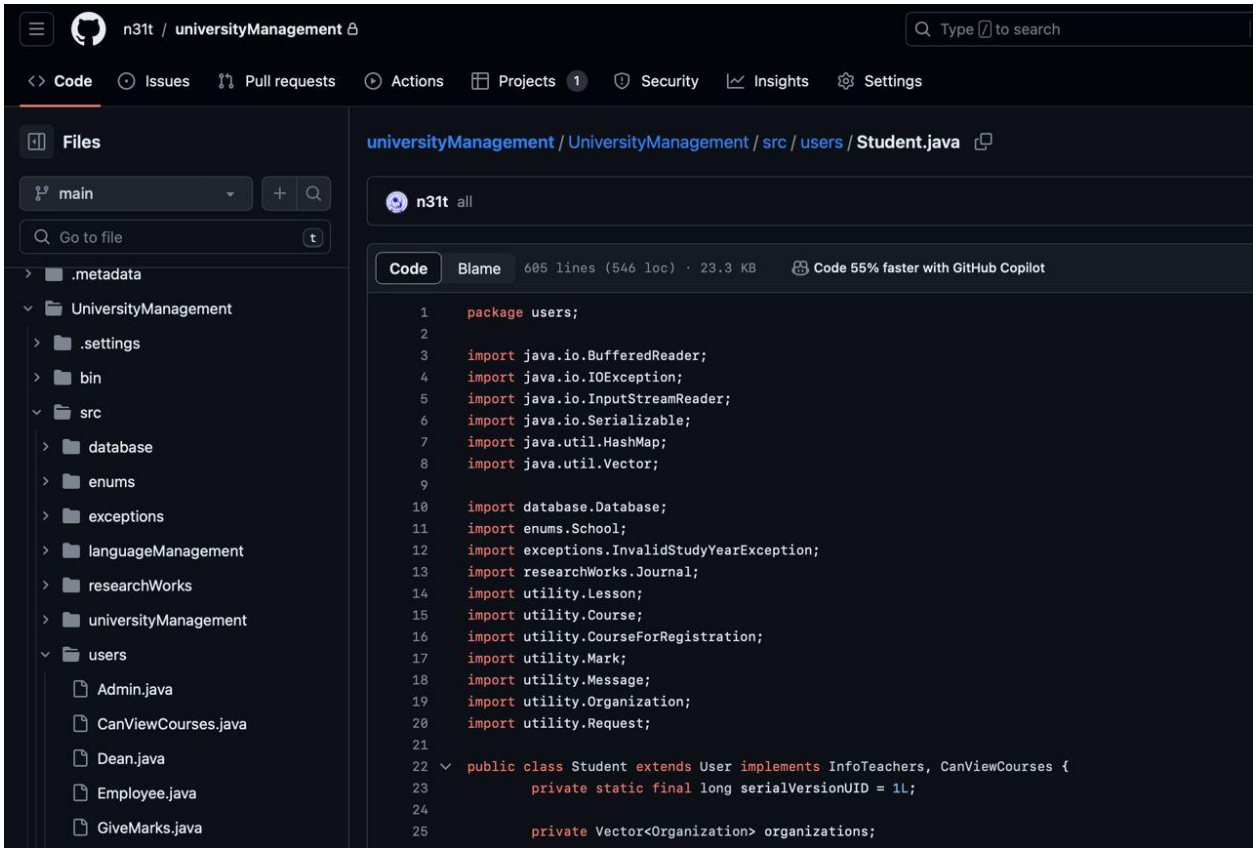
All Classes and Interfaces | Interfaces | Classes

Class	Description
ComparatorByArticleLength	Comparator for comparing ResearchPaper objects based on their title lengths.
ComparatorByCitation	Comparator for comparing ResearchPaper objects based on their citation numbers.
ComparatorByDate	Comparator for comparing ResearchPaper objects based on their publication dates.
Journal	Represents a scientific journal with published research papers.
ResearchPaper	Represents a research paper with information such as title, authors, publication date, etc.
ResearchPaperComparator	Abstract class for comparing ResearchPaper objects.
ResearchProject	Represents a research project with published research papers, a topic, and participants.
UseResearchPapers	Interface for classes that can use research papers.

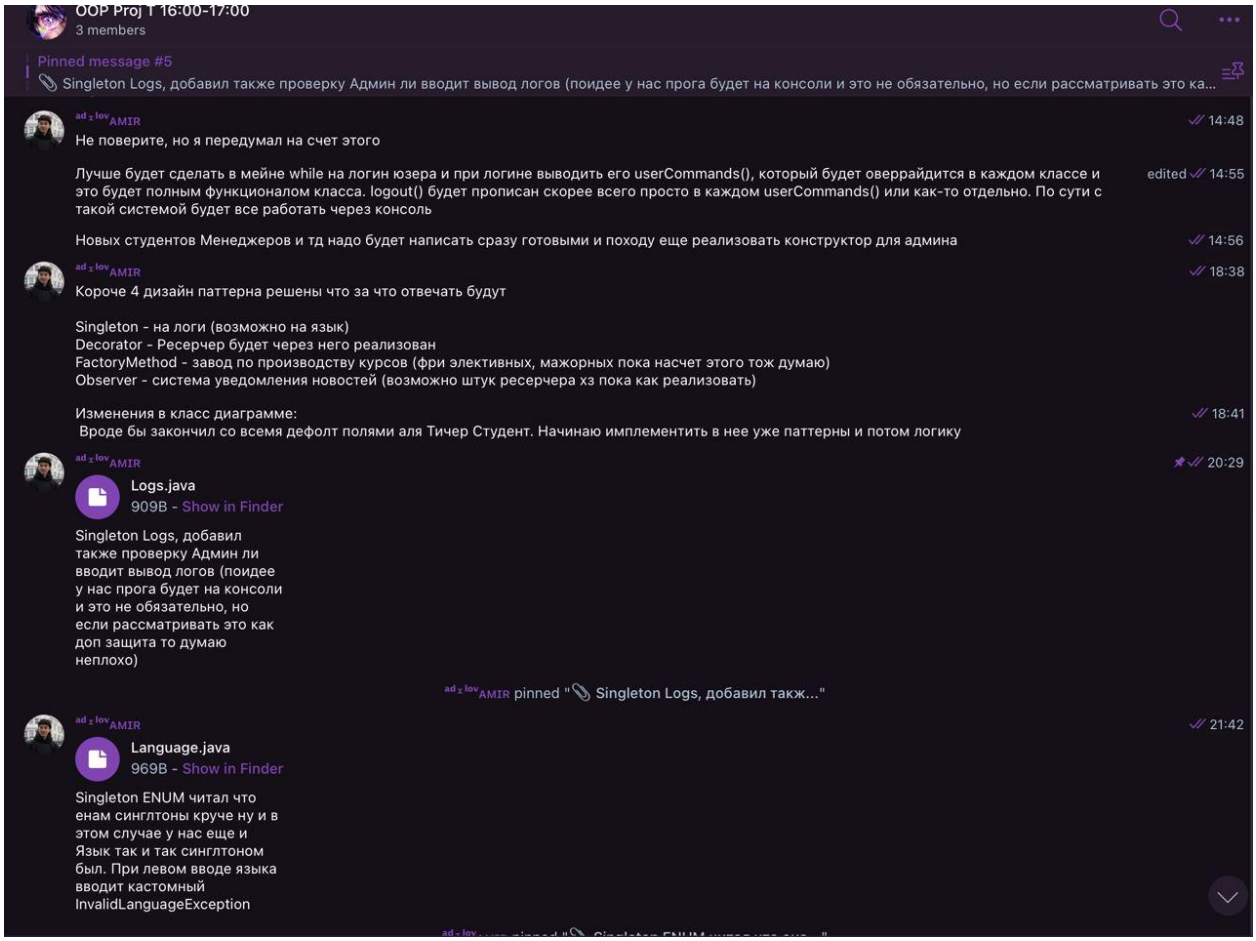
5.Github Repository
Canban Desk:



The Kanban board shows who was working on what. And finally realized it. This is a very convenient way of separating responsibilities, tested by many people.



6.Problems and Communication:



The biggest problems were at the beginning and on the last day of delivery of the project. On the first day, I spent a lot of time thinking through the logic of the code and trying to compare everything with each other.

Due to certain reasons, the working atmosphere in the team began only at the end of the project, so I (the team leader) did most of the project. We had to give up some functions because we didn't have much time left. We chose functionality rather than optimization. The project turned out to be something new for all of us. Everyone faced their problems differently and each person had their own approach to solving them.