**Requirement Analysis Document**

Group Number: 9

Names:

-Yigit Tuncer (150121073)

-Hasan Pekedis (150120068)

-Ahmet Arda Nalbant (150121004)

-Hasan Özeren (150121036)

-Umut Bayar (150120043)

-Niyazi Ozan Ateş (150121991)

-Mehmet Sina Çağlar (150123821)

**DESCRIPTION**

The goal of this Java project is to develop a functional course registration system tailored for your department. This system will provide a user-friendly and efficient platform for students and advisors to manage course registrations in accordance with the department's rules and regulations. It is designed to handle the current needs of students and advisors while remaining adaptable for future expansion to include additional roles like department head, admin, and student affairs. Users will access the system with their assigned usernames and passwords, and the system will implement role-based access control to ensure that each type of user has access only to the features and data relevant to their role. The system will include features such as course selection and registration, enforcement of department rules, a comprehensive course catalog, and an integrated notification system to notify users about important updates. To ensure the successful development of the system, close collaboration with your department and its stakeholders will be necessary, with detailed requirements and specific departmental rules clarified through consultation sessions, including classroom interactions, to make sure the system aligns perfectly with the department's needs. The system aims to streamline the course registration process, increase efficiency, and provide a user-friendly experience for students and advisors, with room for future expansion and adaptation.

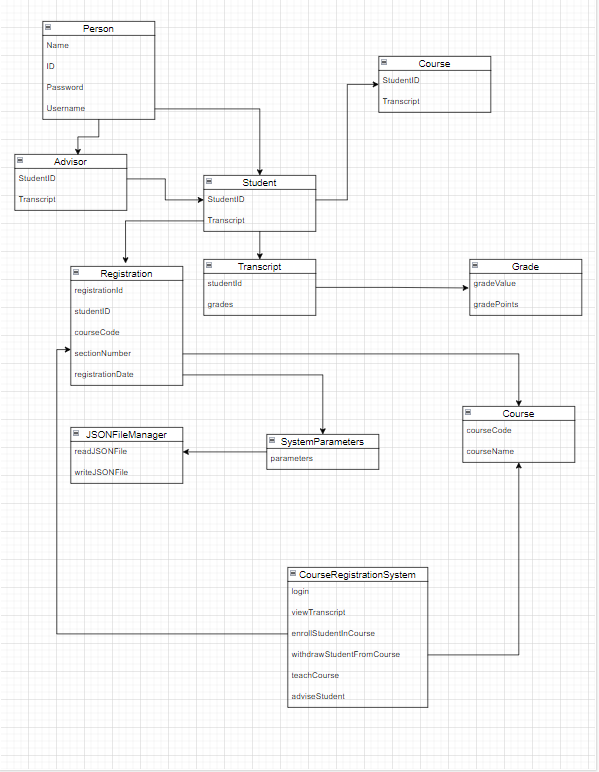
**IMPORTANT CONCEPTS**

1. **User Management:** Concepts related to user registration, login, role assignment, and password reset functionalities.
2. **User Roles:** Allowing users to have different roles (e.g., student, advisor, administrator) and defining specific permissions for each role.
3. **Course Registration:** Understanding the process of users enrolling in courses for specific terms and the associated workflow.
4. **Courses and Programs:** Creating a database or data structure to store information about the courses offered by the department and academic programs.
5. **Registration Rules:** Department-specific rules, policies, and limitations that define how registrations can be made and under what conditions registrations are accepted.
6. **Approval Process:** Defining how advisors approve student registrations and the overall approval workflow.
7. **Course Selection and Cancellation:** Managing course selection, editing existing registrations, and canceling registrations as needed.
8. **Performance and Scalability:** Optimizing system performance and understanding how the system can scale to handle high demands.
9. **Usability and Interface Design:** Designing a user-friendly interface that allows users to interact with the system easily.
10. **Error Handling and Logging:** Dealing with system errors, creating error logs, and monitoring error tracking.
11. **Database Design:** Determining how data related to courses, users, registrations, and rules will be stored in the database (JSON).
12. **Time Management:** Concepts related to terms, course durations, registration deadlines, and time-based processes.

**LIST OF REQUIREMENTS**

|  |  |
| --- | --- |
| **Functional Requirements** | **Non-Functional Requirements** |
| Students and advisors should be able to log into the system with their assigned username and password. | Database operations should support |
| At least two roles should be defined for users in the system: Student and Advisor. It should be easy to add more roles for future requirements, such as Department Head, Admin, and Student Affairs | The system should be compatible with different devices |
| Students should be able to enroll in courses and advisors should be a able to approve students' course selections. | A user-friendly interface should be provided for users |
| Courses and programs available in the system should be viewable and selectable during the registration proces | Critical data should be regularly backed up to prevent data loss |
| Department registration rules should be defined and enforced in the system |  |
| Students should be able to select and register for courses |  |
| They should be able to view past courses and the number of credits they've earned |  |
| Students may need to submit a registration application before making course registrations |  |
| Advisors should have the authority to approve or disapprove students' course registrations |  |

**DOMAIN MODEL**

****

**USE CASES**

**Use Case Name:** Enrolling classes

**Summary:** In order for students to select courses, they must first successfully log in to the system with their username and password, select their courses, save them as a draft and send them to their advisor.

**Subject:** Student

**Basic Flow:**

1) Student logs in to the website with their username and password.

2) Student opens "Enroll to courses" tab at website to view courses.

3) The student selects courses from the course list based on their current curriculum, current semester, and current course progress.

4) System saves courses as draft.

5) Student sends draft to their advisor for approval.

6) Students' course registration procedures have been completed, after the advisor approves the drafts.

**Alternative Flow:**

* Step 1: If student enters the username or password incorrectly, the system warns the student, then use case returns to step 1.
* Step 2: If the student quota of the selected course is full, the system warns the student, then use case returns to step 2.
* Step 4: If there is a conflict in the system, the system cannot save the draft, then use case returns to step 2.
* Step 5: If there is a conflict in the course draft, the student cannot send the course outline to his advisor, then use case returns to step 2.
* Step 6: The student has selected the TE courses and sent them to the advisor for approval. However, if there is not enough demand for that course, the advisor rejects that TE course.

**Use Case Name:** Approving/Rejecting Course Drafts

**Summary:** In order for students to be registered to the courses, the advisor must enter the system and approve or reject the requests from the students.

**Subject:** Advisor

**Basic Flow:**

1) The advisor logs into the system and selects the tab to view students' drafts.

2) The advisor evaluates the drafts chosen by the students and decides whether they will be accepted or rejected.

3) The system sends notification about acceptance or rejection.

**Alternative Flow:**

* Step 1: If there is a conflict in the system, the system cannot view the draft, then use case returns to step 1.

Step 2: The student has selected the TE courses and sent them to the advisor for approval. However, if there is not enough demand for that course, the advisor must informs the students about rejection. Then use case returns to step 2 for rejected students.

**SYSTEM SEQUENCE DIAGRAM (SSD)**

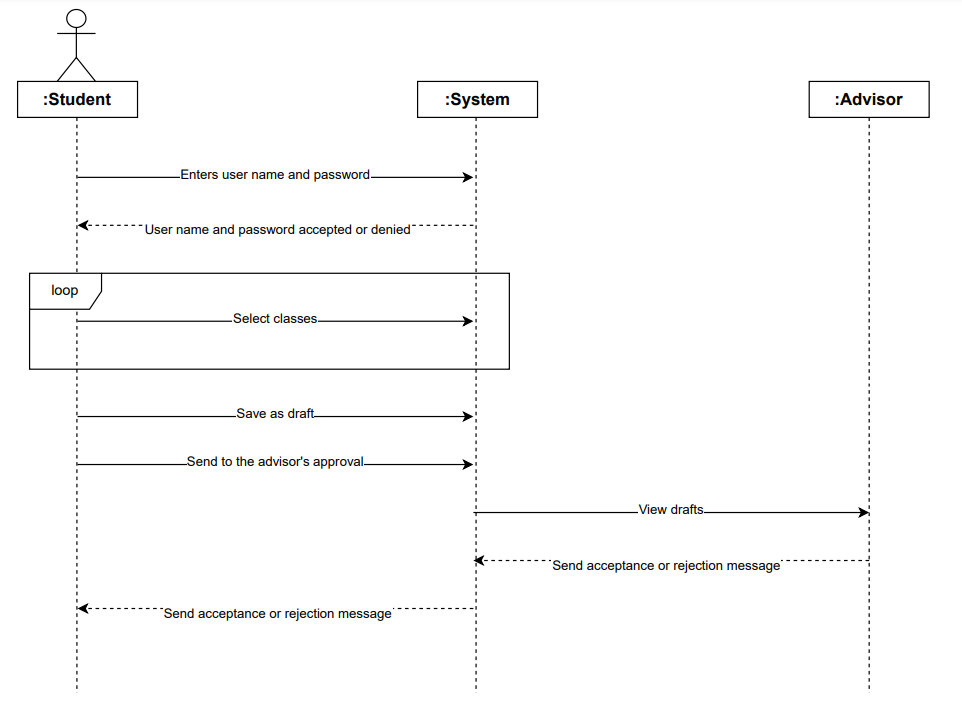
1) The students log in with using their username and password if there are no conflicts.

2) After selecting the courses, students save their chosen courses as drafts.

3) The student submits the drafts for approval from the advisor if there are no conflicts.

4) The advisor reviews the courses the student has selected.

5) The chosen courses have the advisor's approval.

****