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.

GLOSSARY

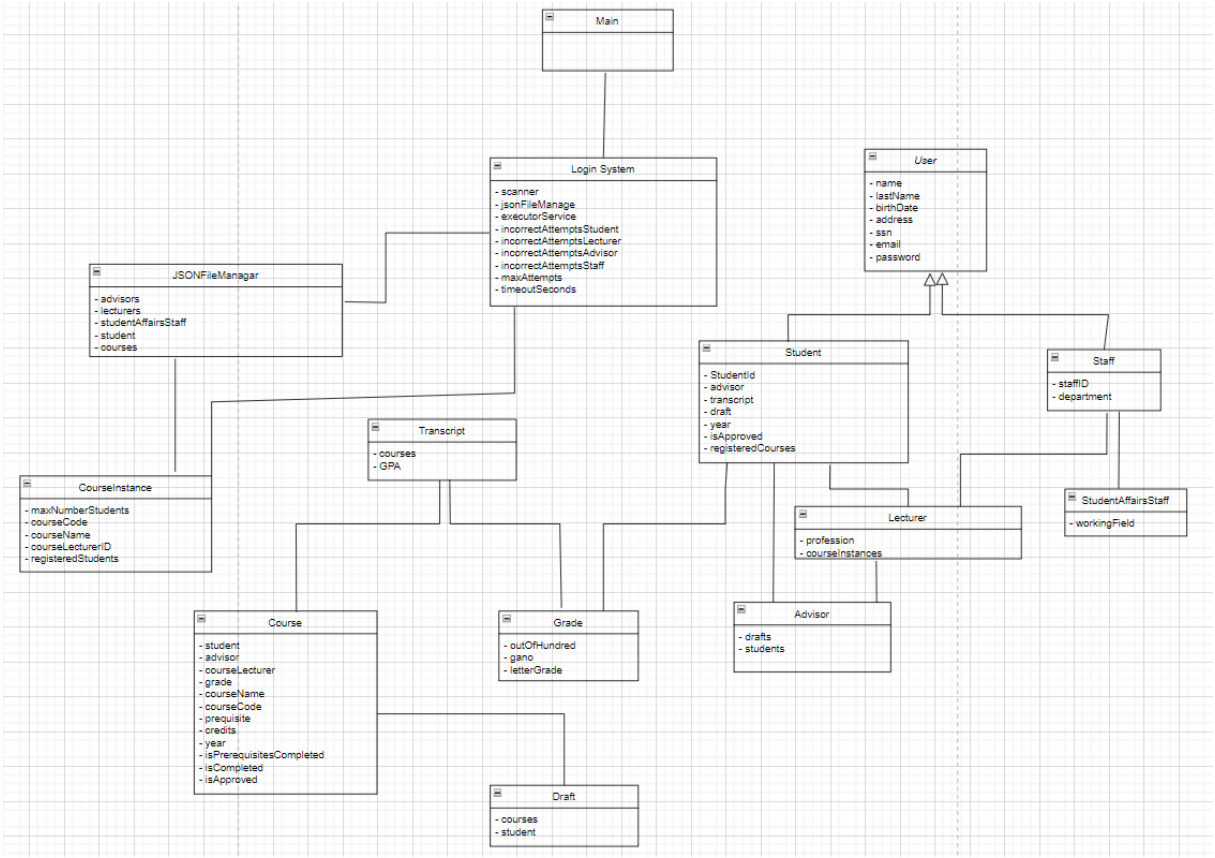
1. **User Management:** Concepts related to user registration, login, role assignment, and password reset functionalities.
2. **User Roles:** Allowing users to have different roles (e.g., student, advisor, administrator) and defining specific permissions for each role.

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 "register to a course" tab to view courses.

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

4) Student submits draft to their advisor for approval.

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

6) Student logs out.

Alternative Flow:

- Step 1: If student enters the username or password incorrectly three times, the system warns the student and gives a timeout, then use case returns to step 1.
- Step 2: If student opens “transcript” tab, the system shows students’ transcript, then use case returns to step 2.
- Step 3: If student enters the course incorrectly, the system warns the student, then use case returns to step 3.
- Step 4: If student submits incorrectly, the system warns the student and does not send it to the advisor, then use case returns to step 4.

SYSTEM SEQUENCE DIAGRAM (SSD)

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

2) Students enters the “register to a course" tab to view available courses.

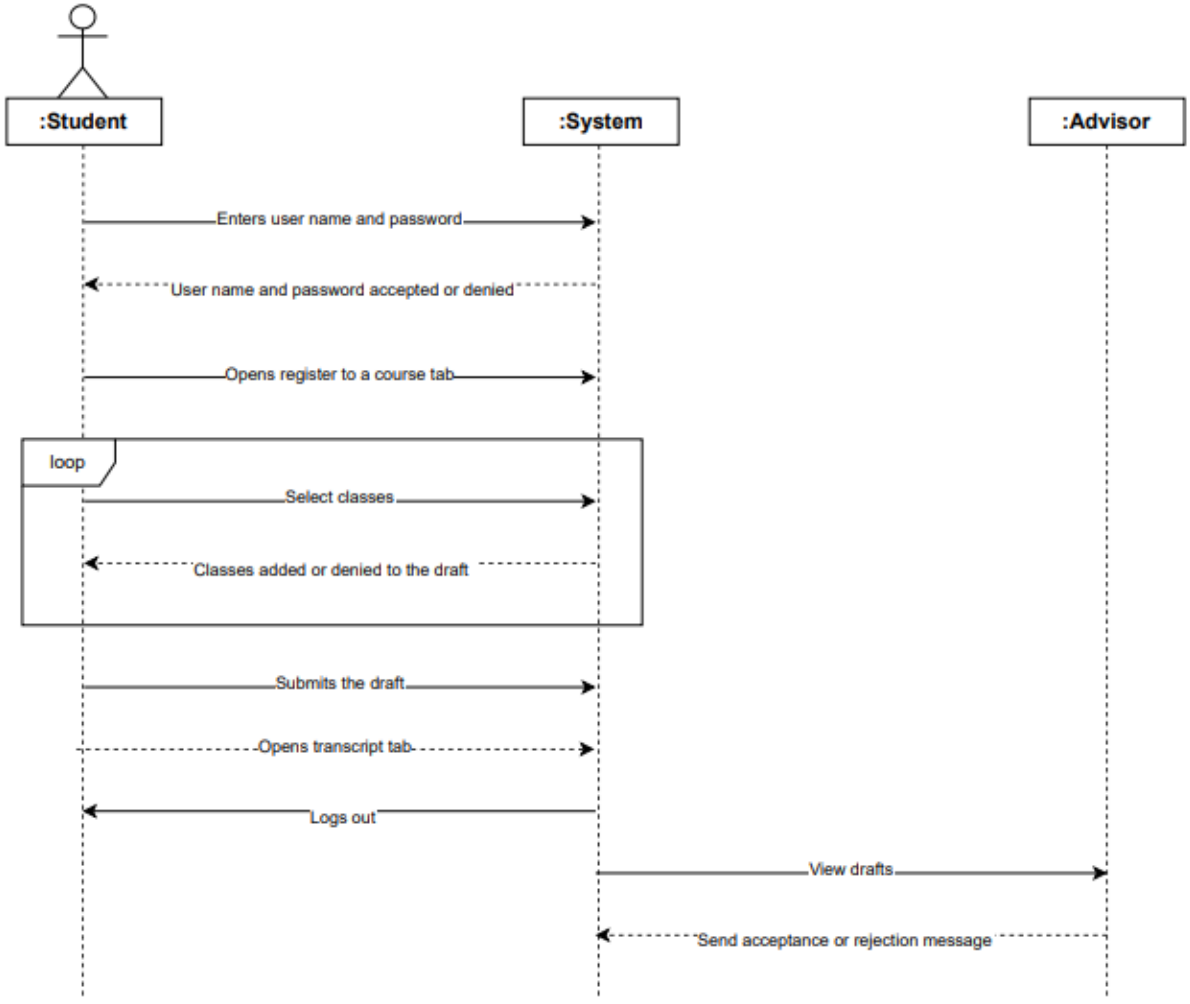
3) After selecting the courses, student submits the drafts for approval if there no conflicts (misspelling etc.).

4) If students open “transcript” tab, the system shows students’ transcript.

5) Students log out.

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

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



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 in to the website with their username and password.

2) The advisor opens "approve student registrations" tab to view registrations.

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

4) Advisor logs out.

Alternative Flow:

- Step 1: If advisor enters the username or password incorrectly three times, the system warns the advisor and gives a timeout, then use case returns to step 1.
- Step 2: If student opens “profession” tab, the system shows advisor’s profession, then use case returns to step 2.
- Step 3: If there is no drafts to approve currently, the systems gives a warning about nothing to see.

SYSTEM SEQUENCE DIAGRAM (SSD)

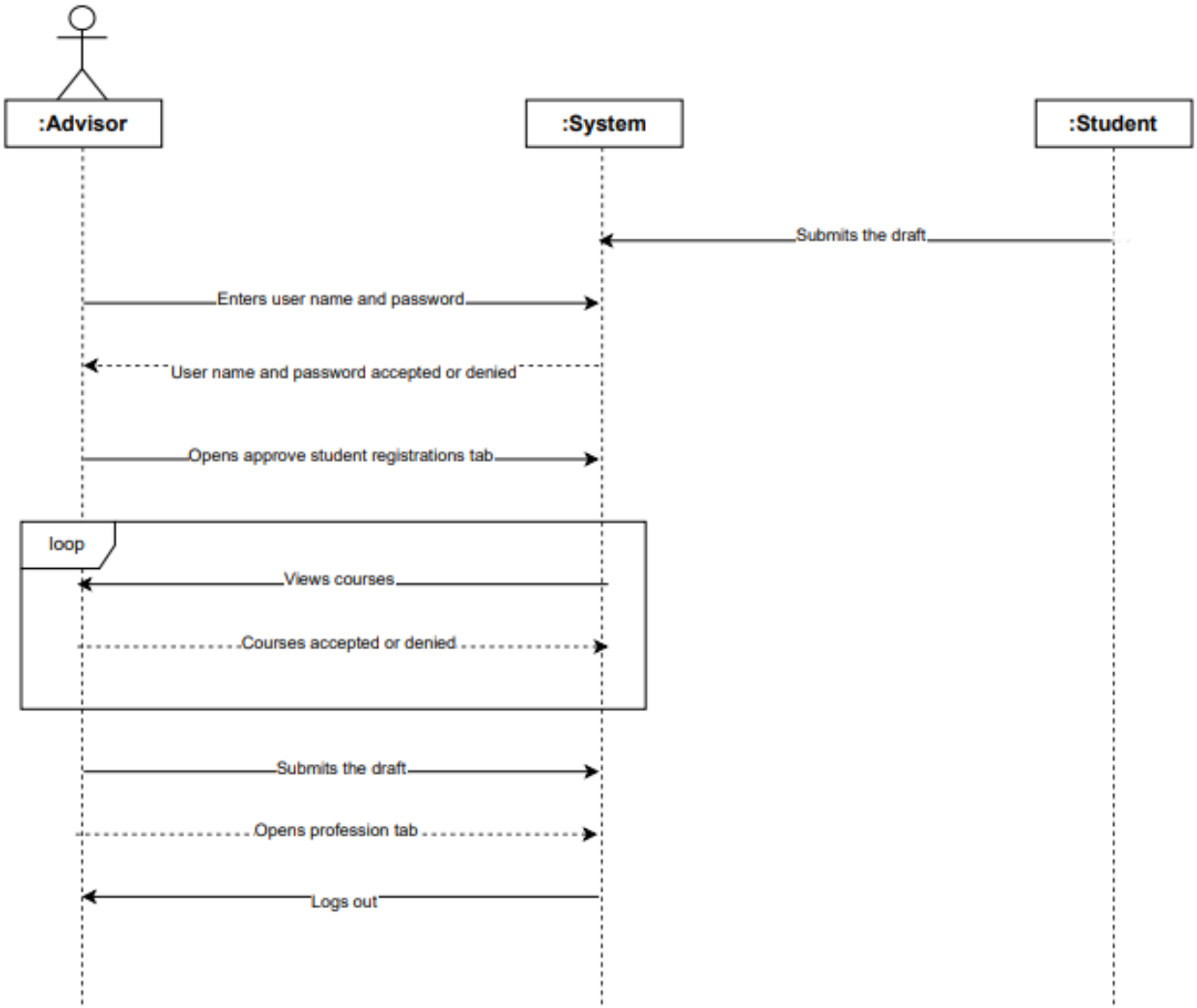
1) Advisors log in with using their username and password if there are no conflicts.

2) Advisors open "approve student registrations" tab to view registrations.

3) Advisors evaluate the drafts chosen by the students and decide whether they will be accepted or rejected.

4) If advisors open “profession” tab, the system shows advisor’s profession.

5) Advisors log out.



DESCRIBING TEAMWORK

Team Member’s Responsibilities

- Yiğit Tuncer (150121073): Wrote a unit test for the student class. Implemented a year system as a prerequisite to taking classes. Updated and added features to the student class registration interface. Implemented course acceptance/rejection logic instead of the draft being completely accepted or rejected. Added JAVADOC comments to some of the classes, created a draft class instead of using a 2D array of courses. Implemented input sanitation to prevent runtime errors with taking input. The domain model documented for RAD document.
- Hasan Pekedis (150120068): Implemented JSONFileManager class for JSON file manuplation and data management, created arrays for all componenets. All student, advisor, lecturer, student affairs staff and course JSON files are designed and samples created. LoggerSystem class created and tested for event tracking and debugging, logger messages added in LoginSystem. Timer implementation completed, used TimeUnit library to implement. JUnit test created for the Lecturer class and passed all tests. List of requirements and important concepts(glossary) documented for RAD document.
- Ahmet Arda Nalbant (150121004): Wrote a unit test for Advisor class with using JUnit library and passed all test. Wrote the Use Cases (one for student and one for advisor) for Iteration 1 and updated it for Iteration 2 Draw the System Sequence Diagram (SSD) for Iteration 1 and updated it for Iteration 2 with using drawio. Implemented advisor approval or rejection function to the code for Iteration 1. Implemented a new feature that when a user enters the password or user name incorrectly for many times, it gives a timeout (like in BYS).

- Hasan Özeren (150121036): In the first iteration, I created the course class as code and enabled the selection of courses through this class and wrote a unit test for the course class. In addition, I created and prepared the DSD document in the first iteration. In the second iteration, I created

the prerequisite logic for the courses and ensured that the courses were selected accordingly. In addition, I prepared 2 DSDs for Advisor and Student.

- Umut Bayar (150120043): Added new methods to the Student Affairs Staff class, which was one of the two courses wroted in the first iteration, and the other class was the staff class. Added all cases related to student affairs staff with the methods added to the Login system these are information updating and various information viewing. Wrote a unit test for the Staff class using the JUnit library for two iterations and passed all test. Wrote the Design Class Diagram for Iteration 1 and updated it for Iteration 2 .

- Niyazi Ozan Ateş (150121991): In the first iteration I have created the LoginSystem class. The purpose of this class is to give each person the functionality to login and provide the actions they can take, which my team-members have created. Also, I have created the classes, attributes, and methods for the DCD. Where I did this also in the second iteration. In the second iteration beside doing the same for the DCD, I have created the CourseInstance class which connects the students with the lecturers. Also, I have gave the lecturers the option to control their courses. By choosing a course, the lecturers can get information about the courses and the students that take that course, grade any student, and pass/fail any student for a particular course. Also, to help with the prerequisite, I added the isCompleted variable to the Course class that says if a student has completed that course.
- Mehmet Sina Çağlar (150123821): In the documentation process, I played a role in crafting relationship arrows, including association, aggregation, composition and inheritance for the UML class diagram. In the subsequent iteration, I have updated the domain model for the ultimate project. With the aim of demonstrating the implementation of polymorphism, I have introduced an abstract method within the User.java class. Additionaly, I introduced the

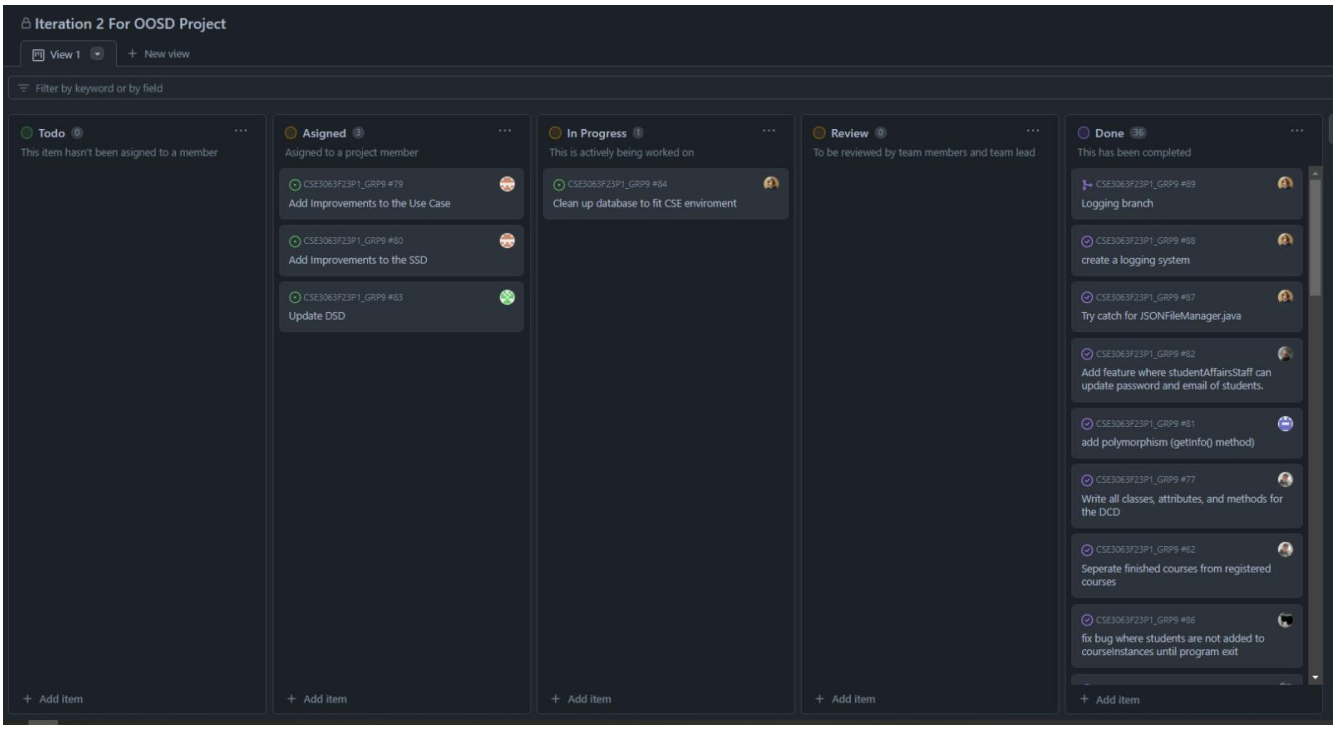
Transcript.java class, designed to provide detailed information about the student. Furthermore, I implemented the Grade.java class that helps for converting between various grade types seamlessly, which I made sure by implementing a unit test for this class.

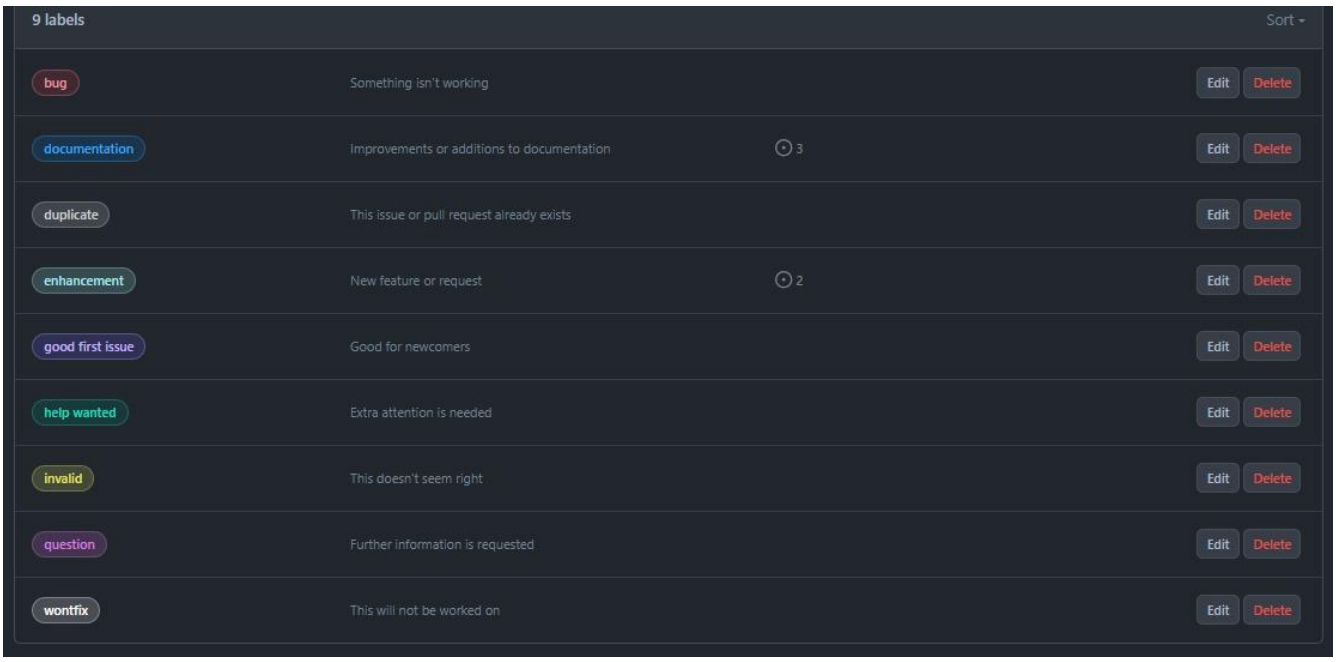
The Project Management

We used a Kanban board to organize tasks. Each new idea, feature or bug fix got turned into a git issue and assigned to a project member. We have 5 columns on our Kanban board to organize tasks. To-do column for keeping track of unassigned issues, assigned for issues assigned to a team member, a In Progress column to show what who is working on at the current moment, a review column to show issues that are complete but should be reviewed by other team members before completion, and finally a done section for issues that are complete. While working on the project we sometimes worked on diverging features that required us to branch the project and then one we had a working feature or change, merge into the main branch. Sometimes we had to help each other out on some features.

The Tools

Kanban board




9 labels
Sort
bug
Something isn't working
Edit
Delete
documentation
Improvements or additions to documentation
3
Edit
Delete
duplicate
This issue or pull request already exists
Edit
Delete
enhancement
New feature or request
2
Edit
Delete
good first issue
Good for newcomers
Edit
Delete
help wanted
Extra attention is needed
Edit
Delete
invalid
This doesn't seem right
Edit
Delete
question
Further information is requested
Edit
Delete
wontfix
This will not be worked on
Edit
Delete