**Project Description:**

One of the biggest things missing from collegiate esports competition, where schools compete for cash prizes, is a system that verifies concurrent student participation and student eligibility requirements (e.g., minimum GPA, enrollment status, etc.). For example, Shelton State in 2022, required the student to be scheduled to take a minimum of 12 credit hours and have a minimum GPA of 2.0 to be an official member of the esports program. This, however, is not consistent from school to school.

Although organizations that host seasons and tournaments require these guidelines for payout, it is almost impossible for them to keep up with each individual school’s specific requirements. This creates an opportunity for schools to take advantage of a lapse in eligibility verification. The esports scene needs a system in place that helps verify student eligibility at each school. We propose a system that operates as a pseudo-regulatory body in which students can create an account in which they are able to verify their eligibility against a school’s requirements. Tournament organizers can then verify student eligibility against the system’s database using something like a member ID.

**Milestone 1:**

For the collegiate esports student verification system, several actors interact with the system, namely, students, the school’s administrators, tournament organizers, and the software's backend development.

Some use cases could include:

1. Student registers for an account

a. Steps for use case:

i. Student is prompted to enter several key fields, such as name, birthdate, school, student email

1. During account creation, each field should have criteria required to be met for an acceptable field. Failure to meet criteria should prompt reentry plus explanation of field requirements

ii. System creates an object with each key field

1. Failure to create an object should prompt the system to send notification to user

2. Student logs into their account

a. Steps for use case:

i. Student is prompted to enter email and password

1. Notify student if login succeeds or fails. Allow for 3 maximum attempts before system times out

3. System assigns student a member ID

a. Steps for use case:

i. System generates an ID number

1. ID number will be part of object key fields

2. Checks database to make sure duplicates are not assigned

4. System verifies requirements with school

a. Steps for use case:

i. Upon account creation, system automatically drafts a notice (maybe an email) to school in order to verify credentials

ii. School returns a notice verifying credentials

iii. Student object then updates a key field signifying student status as eligible/ineligible

5. Tournament organizers verify status with system

a. Steps for use case:

i. Tournament organizers can create an organizer account

ii. Organizers can then upload a list of member IDs and receive a report with each student’s status

**Milestone 2:**

Entities such as students, coaches, schools or tournament organizers have well defined interactions. For example, a student is able to create an account and also login in to an account. A tournament organizer is able to upload a list of member IDs and receive a report. These interactions will allow for a detailed static model to be created.

By acting as a pseudo-regulatory body, certain access and privileges must be assigned to each entity in order to maintain data integrity For instance, a coach should have access to view their teams status and eligibility but should not be allowed to manipulate any of the data. Tournament organizers should be allowed to see student eligibility status but should probably have limited access to other personal information. The flow of information will allow for a detailed software context class diagram to be created.

**Milestone 3:**

Within the static model each entity and its corresponding attributes will be precisely described. For example, some of the attributes of a student would be:

* Name as a string
* Student email as a string
* Eligibility status as a boolean

**Milestone 4:**

Referring back to the use cases outlined in Milestone 1, there is a rudimentary outline of the steps being taken for each use case example. The sequence diagram in this milestone will aim to flesh out each of these examples graphically as well as any other use cases the system has.

The collaboration diagram for this system will be a static model at the use case level. An example of item from this diagram would be a graphic representation of request from the system to the school for verification of requirements and an approval returned to the system from the school.

The state chart will be similar to the sequence diagram except it will not be time-sequenced. An example an item in the system’s state chart would be:

Waiting for Tournament Organizer → List Being Processed → Printing Eligibility Status

**Milestone 5:**

At the macro system level, a comprehensive communications diagram will be developed that encapsulates all the individual use cases and how they communicate within the broader esports verification system. This diagram will incorporate all the entities and also reveal the flow and transformation of data throughout the entire system.