1.UML class diagram

a. UC 1 & UC2 :Sign-up and Login

I created an external entity called User, which is users who have not registered the system, and another entity called Account.

The class-Account can perform registration and verified login and is the highest level, which means it is inherited by all stakeholders, such as the class-Fan and class-Team, etc.

b.UC3: Process policy/rule

In the use case requirement, I combined the scheduling and budget rules into a case, which means IFA to distinguish which category the policy belongs to during operation. But in the class diagram, I divided them into class-BudgetRule and class-SchedulePolicy (in the top-left of the diagram) because it will be easier to understand.

Both classes have aggregation relationships to class-IFAAdmin and IFA admins can manage those policies by using CRUDBudgetRule() and CRUDSchedulePolicy(). The main difference between CRUDBudgetRule() and CRUDSchedulePolicy() is parameters. The use case mentions that when an IFA admin wants to update policy related to scheduling, he/she first needs to input the Admin-ID, the reason, and the date, and then be verified by other IFA people to finish this operation. So in the CRUDSchedulePolicy(), there are three parameters and return a Boolean value to show whether this operation is failed or not.

c. UC4 : Audit Budget

Class-Team and class-BudgetReport(in top-right of the diagram), and the relationship between two classes belongs to the composition relationship, which means that when class-Team does not exist, class-BudgetReport will not exist. The team needs to submit reports to IFA for auditing so there will be an undirected association between class-IFA admin and class-Team.

In this use case, the class team needs to add, query, update, and delete budget reports, send budget reports to IFA for auditing, and receive a result of auditing, so they are represented by three operations: CRUDBudget(), SendBudgetReport(), and GetAuditResult().

On the class-IFA admin side, IFA admin uses CRUDBudgetRule() to manage budget rules and SetBudgetRule to select the required rules, then finally the ActivateAuditing() uses BudgetRule as parameter to generate the audited report. This report will use the SendAuditingReport() to return class-Team.

d. UC5: Process game schedule

In this use case, first, class-referees create their own schedules and save them into the database through SaveRefereeSchedule(). In addition, IFA will generate schedules by obtaining referees ’schedules from RetrieveRefereeSchedule () to ActivateScheduling (). Finally, the produced schedule is sent to class-Team and class-Referee via SendSchedule().

e. UC6: Support Fans

Class-Fan(in the middle of the diagram) is able search any game statistics by using SearchingGameReport() and interacts with Class-SocialNetwork by using AddingComment() or FollowSocialNetworkPage().

f. UC7: Manage team’s resource

Above the class-IFAAdmin in the diagram, there is a class-ExternalTeam used to indicate that after the class-IFAAdmin's VerifiedTeamRegister (), the team can become a class-Team and is able to use ManageResource() to manage the resource. In class-Team, its relationship with the player and coach is aggregation. A team can manage multiple players and coaches and they still exist when the team does not exist in this system. In addition, class-Team has composition relationships with class-SocialNetwork and resources, such as transaction, budget, and stadium. When this a team is not in the system, resource and class-SocialNetwork will also disappear from the system.

g. UC8: Manage social media

The class-SocialNetwork(in the bottom-right of the digram) has CRUDPost() to manage posts and SendFollowPost() to notify fans newest information.

h. UC9: Communicate within stakeholders

Class-GameNotification(in the left-bottom of the diagram) is a class that can receive the private message from stakeholders when it uses GetMessage() and this class can assign who will receive the notification by using AssignStakholder() and SendNotification() sends the notification to stakeholders.

I. UC10: Report game’s event

Class-Referee has RecordGameReport() that every event happened in a game and the class-GameReport sends the game report to the class-StatisticsTable, which is a class for fans to search interesting data, and the real-time notification to Class-GameNotification.

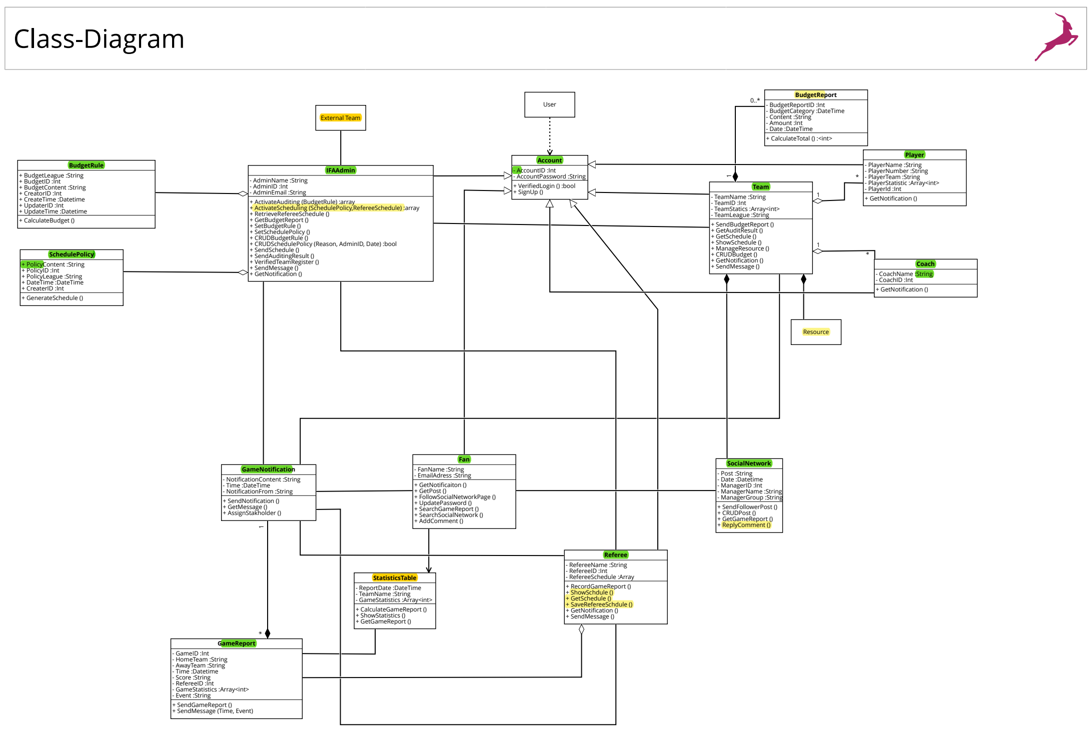

Figure 1. Class diagram