**Group No: 01**

**Course No**: CSE 308

**Course Name**: Software Design and Information System Design Sessional.

**Automation of BCB’s Game Development Committee**

**Group Members:**

1005091

1005111

1005113

1005115

1005120

**Introduction:**

Bangladesh cricket is one of the greatest prides for Bangladesh. Though Bangladesh cricket has advanced significantly in international level, unfortunately the domestic cricket management and maintenance has not been well-organized yet. Therefore, in spite of having lots of cricket loving talented players all over the country they are not inspired and utilized properly for sake of the Bangladesh cricket due to lack of seriousness and faulty system in the domestic cricket infrastructure. It is our belief that if we can improve this system to improve the domestic cricket infrastructure then no talent will be waste and our cricket will improve significantly.

**Subsystems:**

Our subsystems are:

* School Cricket Management System
* Youth Cricket Management System
* Facilities and Equipments Management System
* Divisional Cricket Management System
* Academy Operations Management System
* Coach Management System
* Control Management System

**Collaboration and Class Diagrams:**

1. **School Cricket Management System:**
   1. **School Registration**

1.1.1Noun Identification from use-case narrative:

|  |  |
| --- | --- |
| Actor Action  (School Representative, School cricket administrator ) | System Response |
| 1.none | 2. Show registration Form |
| 3. Submit registration form. | 4. Send registration form to School cricket administrator |
| 5. School Cricket Administrator accepts or rejects the Registration request. | 6. If accepts, then the system enrolls the school and save the school info into database. If rejects, then the system ignores the school info. |
|  | 7. Send acceptance/ decline message to School representative. |

1.1.2 Entity:

1. School
2. RegistrationInfo

1.1.3 Collaboration Diagram:

Figure 1.1.3 shows the collaboration diagram for School Registration use case. Here we can see the interaction between RegistrationUI class, ApproveRegistrationUI class, Registration control class, School and RegistraionInfo entities.

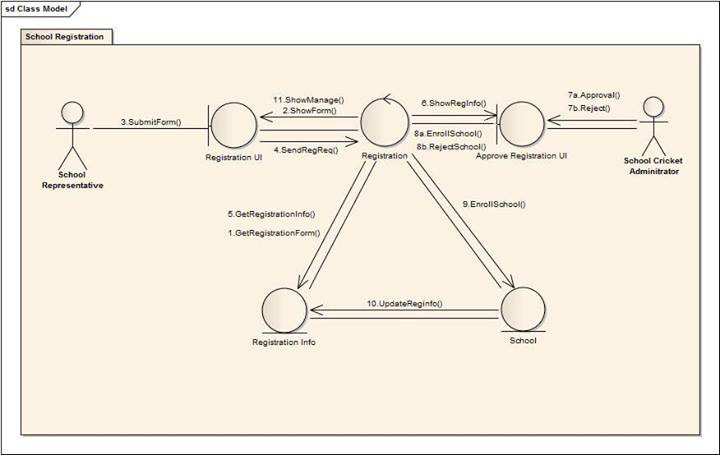


Figure 1.1.3: Collaboration Diagram for School Registration Use Case

1.1.4 Class Diagram:

Figure 1.1.4 shows the class diagram for School Registration use case and we can visualize the static aspects of these building blocks and their relationships.

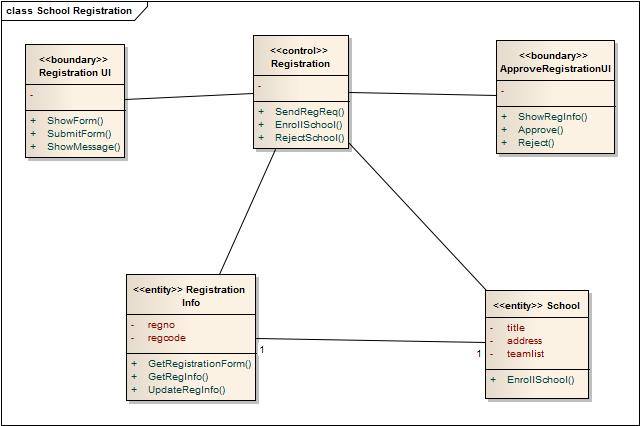


Figure 1.1.4: Class Diagram for School Registration Use Case

* 1. **School Cricket Tournament Arrange**

1.2.1Noun Identification from use-case narrative:

|  |  |
| --- | --- |
| Actor Action  ( School cricket administrator, Program manager ) | System Response |
| 1.none | 2. Get school information |
| 3. none | 4. Get ground database and show them to school cricket administrator |
| 5. School Cricket Administrator schedules tournament | 6. Sends approval request to program manager |
| 7. Program manager approves or rejects the tournament | 8. If accepts then updates school tournament info, school info and ground info. |

* + 1. Entity:

1. SchoolTournament
2. School
3. Ground

1.2.3 Collaboration Diagram:

Figure 1.2.3 shows the collaboration diagram for School Cricket Tournament Arrange use case. Here we can see the interaction between schoolTournamentUI class, ApproveTournamentUI class, schoolTournamentArrange control class, school, schoolTournament and ground entities.

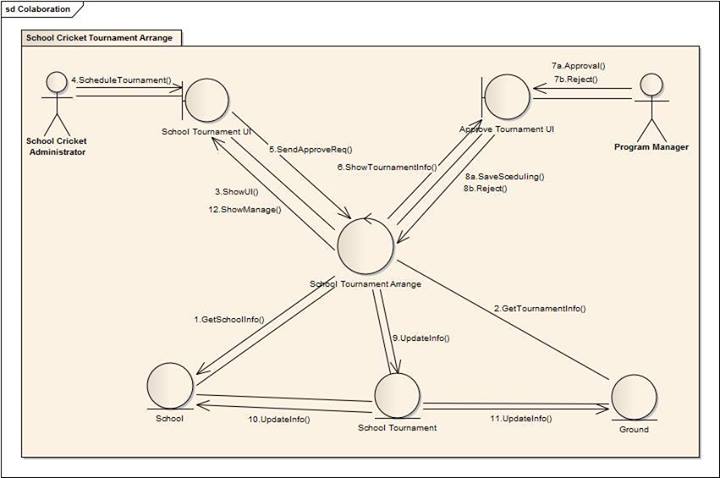


Figure 1.2.3: Collaboration Diagram for School Registration Use Case

1.2.4 Class Diagram:

Figure 1.2.4 shows the class diagram for School Cricket Tournament Arrange use case and we can visualize the static aspects of these building blocks and their relationships.

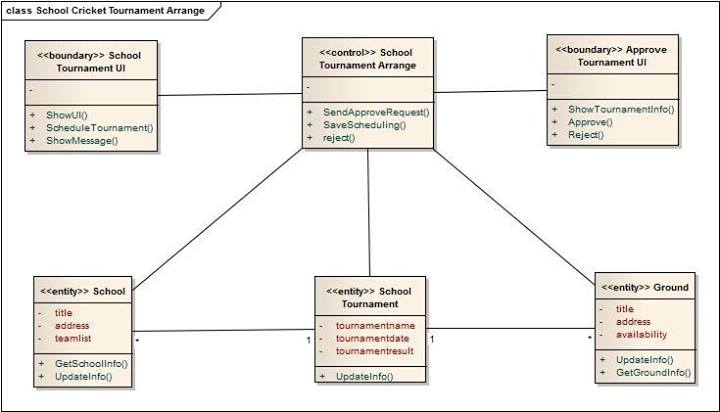


Figure 1.2.4: Class Diagram for School Cricket Tournament Arrange Use Case

1. **Youth Cricket Management System:**
   1. **Arrange Youth Cricket Tournament**

2.1.1Noun Identification from use-case narrative:

|  |  |
| --- | --- |
| Actor Action  (Youth cricket administrator, Program manager ) | System Response |
| 1.none | 2. Get team information |
|  | 3. Get ground database and show them to youth cricket administrator |
| 5. Youth Cricket Administrator schedules tournament | 6. Sends approval request to program manager |
| 7. Program manager approves or rejects the tournament | 8. If accepts then updates youth tournament info, youthTeam info and ground info. |

* + 1. Entity:

1. YouthTeams
2. YouthTournament
3. Ground

2.1.3 Collaboration Diagram:

Figure 2.1.3 shows the collaboration diagram for Arrange YouthCricket Tournament use case. Here we can see the interaction between youthTournamentUI class, ApproveTournamentUI class, youthTournamentArrange control class, youthTeams, youthTournament and ground entities.

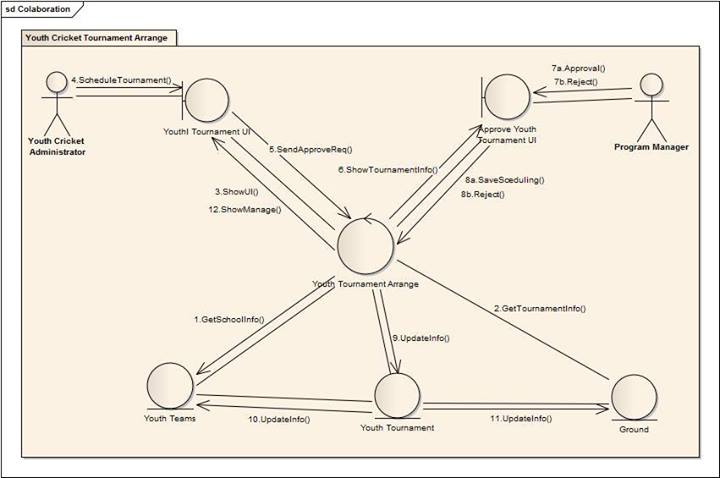


Figure 2.1.3: Collaboration Diagram for Arrange Youth Cricket Tournament Use Case

2.1.4 Class Diagram:

Figure 2.1.4 shows the class diagram for Youth Cricket Tournament Arrange use case and we can visualize the static aspects of these building blocks and their relationships.

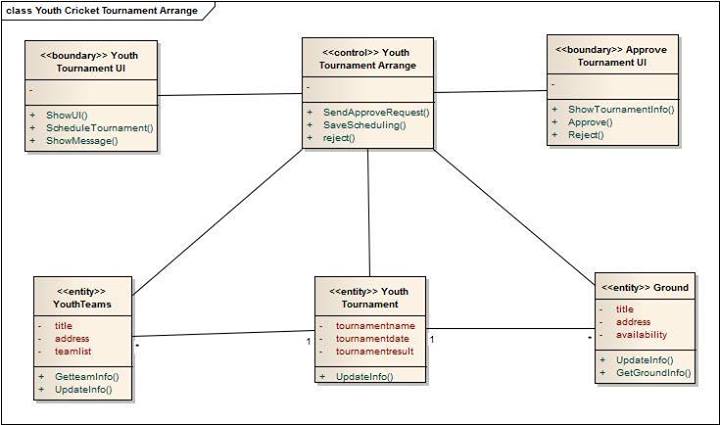


Figure 2.1.4: Class Diagram for Arrange Youth Cricket Tournament Use Case

* 1. **Update Tournament Info**

2.2.1Noun Identification from use-case narrative:

|  |  |
| --- | --- |
| Actor Action  ( Youth cricket administrator ) | System Response |
| 1.none | 2. Get tournaments and show them to youth admin. |
| 3. Youth Cricket Administrator selects tournament | 4. Get tournament information and show them to youth cricket administrator |
| 5. Youth Cricket Administrator updates tournament information | 6. Updates youthTournament and youthTeams database. |

* + 1. Entity:

1. youthTeams

2. youthTournament

3. Tournaments

2.2.3 Collaboration Diagram:

Figure 2.2.3 shows the collaboration diagram for update tournament info use case. Here we can see the interaction between updateTournamentUI class, updateTournament control class, youthTeams,youthTournament and Tournaments entities.

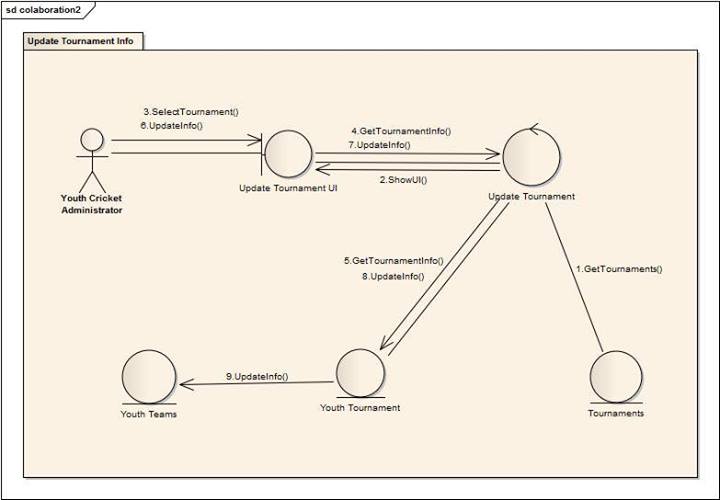


Figure 2.2.3: Collaboration Diagram for update tournament info Use Case

2.2.4 Class Diagram:

Figure 2.2.4 shows the class diagram for for update tournament info use case and we can visualize the static aspects of these building blocks and their relationships.

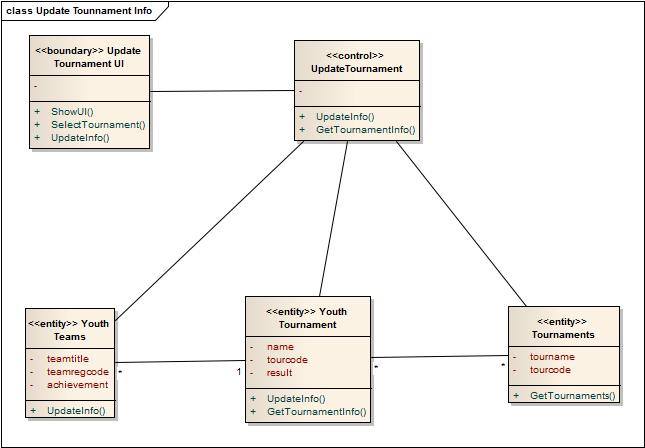


Figure 2.2.4: Class Diagram for update tournament info Use Case

**3. Facility Management System:**

* 1. **Ask For Facility**

3.1.1Noun Identification from use-case narrative:

|  |  |
| --- | --- |
| Actor Action  ( School Cricket Administrator, Youth Cricket Administrator, Facility Administrator ) | System Response |
| 1.none | 2. System will show a window to the school cricket administrator and youth cricket administrator. |
| 3. school cricket administrator, Youth Cricket Administrator send list of facilities to the facility administrator | 4. System will save the list of requests and notify the facility administrator |
| 5. Facility Administrator show the facility requests |  |

3.1.2.Entity:

1.Facility list

3.1.3 Collaboration Diagram:

Figure 3.1.3 shows the collaboration diagram for ask for facilities use case. Here we can see the interaction between facility request UI class, facility request class, Notification UI class and facility list entity.

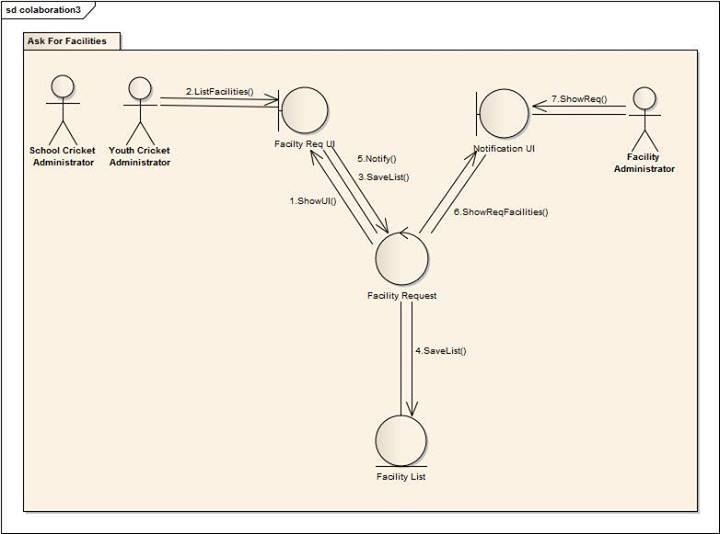


Figure 3.1.3: Collaboration Diagram for Ask for facilities Use Case

3.1.4 Class Diagram:

Figure 3.1.4 shows the class diagram for ask for facilities use case and we can visualize the static aspects of these building blocks and their relationships.

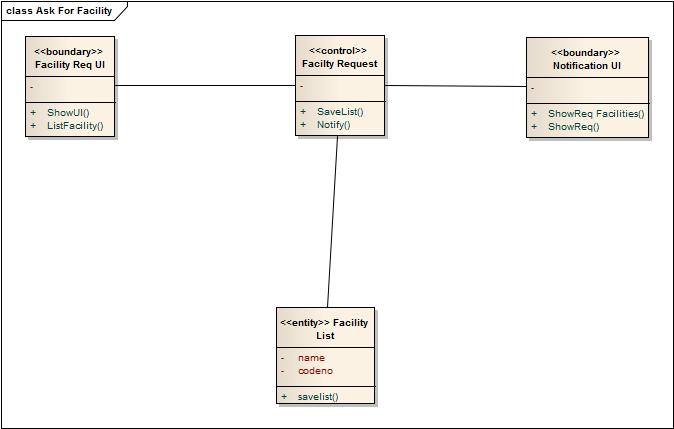


Figure 3.1.4: Class Diagram for ask for facilities Use Case

* 1. **Provide Facilities**

3.2.1Noun Identification from use-case narrative:

|  |  |
| --- | --- |
| Actor Action  ( Facility Admin, Program Manager ) | System Response |
| 1.none | 2. Get facilityList and show list to facility admin |
| 3. Facility admin selects facility | 4. Stores provided facilities to provided facilities and updates facilities and facilityList |
| 5. Facility admin reports about facilities | 6. shows report to program manager |

* + 1. Entity:

1. FacilityList

2. Facilities

3. ProvideFacilities

3.2.3 Collaboration Diagram:

Figure 3.2.3 shows the collaboration diagram for provide facilities use case. Here we can see the interaction between provide facility UI class, facility report UI class, provide Facility control class, Facility List,Facilities and provide Facilities entities.

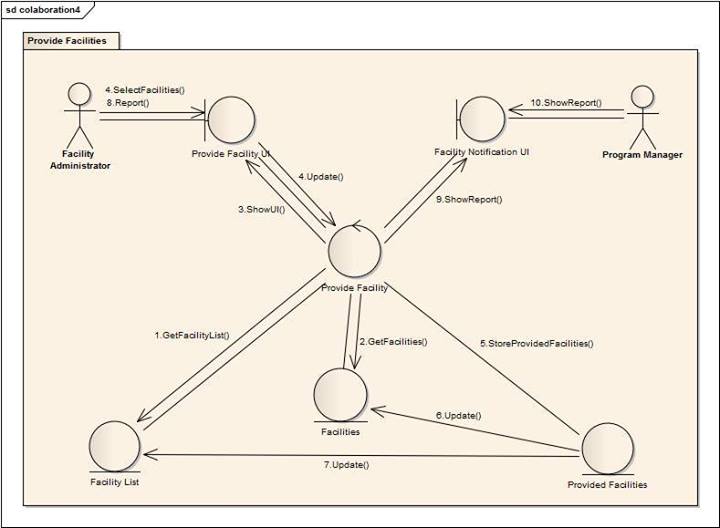


Figure 3.2.3: Collaboration Diagram for Provide Facilities Use Case

3.2.4 Class Diagram:

Figure 3.2.4 shows the class diagram for Provide Facilities use case and we can visualize the static aspects of these building blocks and their relationships.

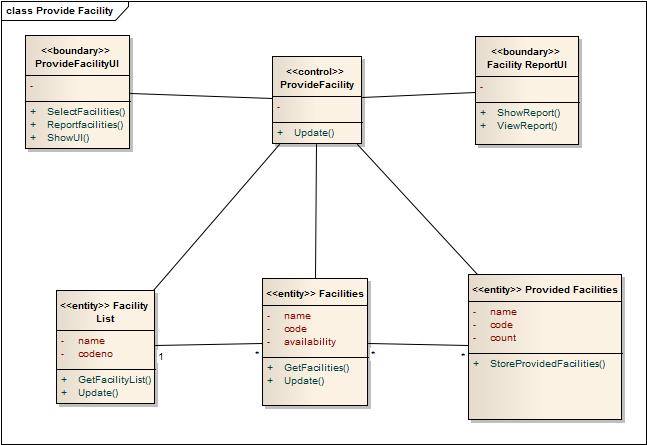


Figure 3.2.4: Class Diagram for Provide facilities Use Case

**4. Academy Operations Management System:**

**4.1: Assign Head Coach**

4.1.1Noun Identification from use-case narrative:

|  |  |
| --- | --- |
| Actor Action  (Operations Manager of Game Development, Academy operations Manager ) | System Response |
| 1.none | 2. Show a window to academy operations manager |
| 3. Submit details coach info. | 4. system shows selected coach info to the operations manager for approval |
| 5. operations manager of game development will approve or reject | 6. If approve then assigns the coach and update head coach and player info. Else rejects |

* + 1. Entity:

1. Player
2. HeadCoach

4.1.3 Collaboration Diagram:

Figure 4.1.3 shows the collaboration diagram for Assign Head Coach use case. Here we can see the interaction betweenAssign Head CoachUI class, Assign Head Coach class, Assign CoachUI class, Head Coachand player entities.

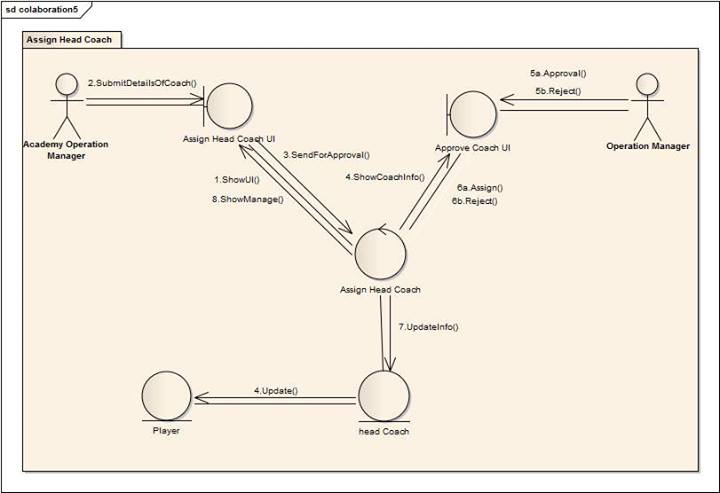


Figure 4.1.3: Collaboration Diagram for Assign Head Coach Use Case

4.1.4 Class Diagram:

Figure 4.1.4 shows the class diagram for Assign Head Coach use case and we can visualize the static aspects of these building blocks and their relationships.

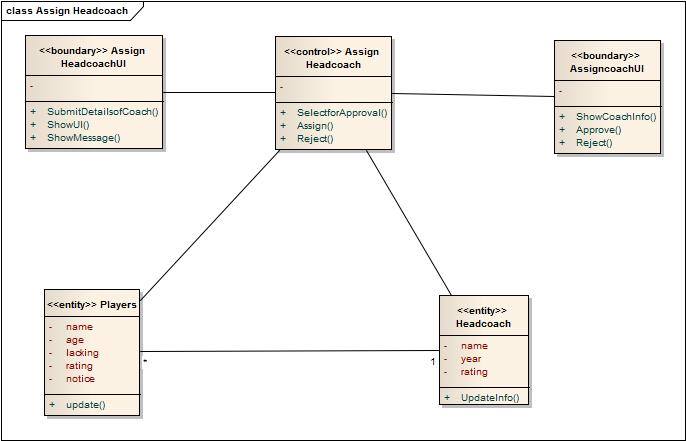


Figure 4.1.4: Class Diagram for Assign Head Coach Use Case

* 1. **Observe Players Lacking**

4.2.1 Noun Identification from use-case narrative:

|  |  |
| --- | --- |
| Actor Action  (Academy operations Manager, Operations Manager of Game Development ) | System Response |
| 1.none | 2. Get Players list and submit to the academy operations manager |
| 3. Academy Operations manager selects player. | 4. system will provide player database |
| 5. Academy Operations Manager reports about players. | 6. System will get and show it to the operations manager of game development. |
| 7. Operations Manager of Game development sends massage to the academy operations manager. | 8. System will show the massage to the academy operations manager. |

4.2.2 Entity:

1. player
2. playerlist
3. Report

4.2.3 Collaboration Diagram:

Figure 4.2.3 shows the collaboration diagram for Observe player lacking use case. Here we can see the interaction between Observe UI class, Observe class, Identifying Lacking class, Player, Player list and Report entities.

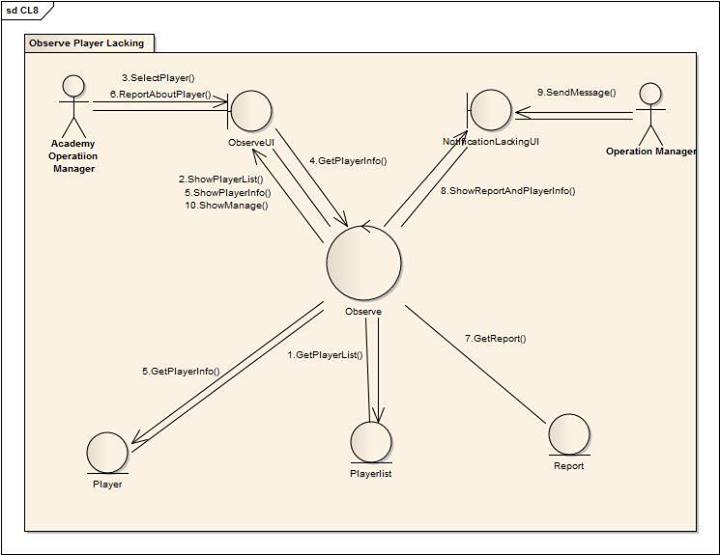


Figure 4.2.3: Collaboration Diagram for Observe player lackingUse Case

4.2.4 Class Diagram:

Figure 4.2.4 shows the class diagram for Observe player lackinguse case and we can visualize the static aspects of these building blocks and their relationships.

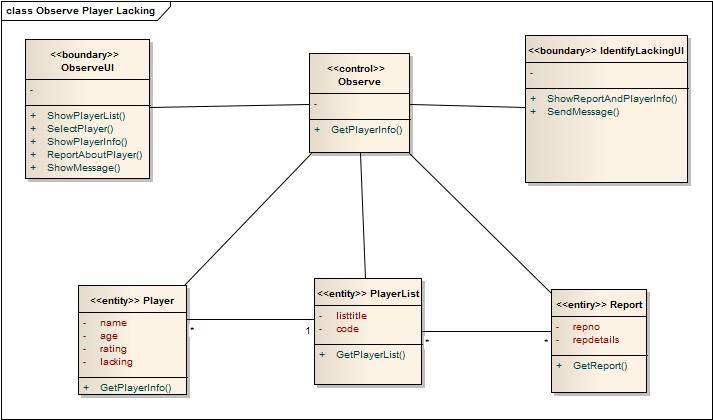


Figure 4.2.4: Class Diagram for Observe player lacking Use Case

1. **Divisional Cricket Management System:**
   1. **Form District Teams**

5.1.1Noun Identification from use-case narrative:

|  |  |
| --- | --- |
| Actor Action  (District Cricket Coach ) | System Response |
| 1.none | 2. System show a window to the district cricket coach |
| 3. District coach adds player info to district team | 4. System will save added player info to the district team and player database |

* + 1. Entity:

1. DistrictTeam

2. player

5.1.3 Collaboration Diagram:

Figure 5.1.3 shows the collaboration diagram for form district team use case. Here we can see the interaction between FormDistrictTeamUI class, FormDisTeam control class, player and DistrictTeam entities.

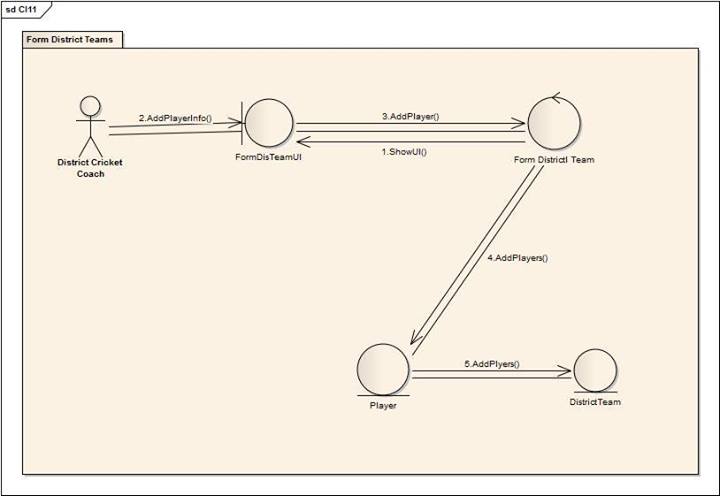


Figure 5.1.3: Collaboration Diagram for form district team Use Case

5.1.4 Class Diagram:

Figure 5.1.4 shows the class diagram for form district team use case and we can visualize the static aspects of these building blocks and their relationships.

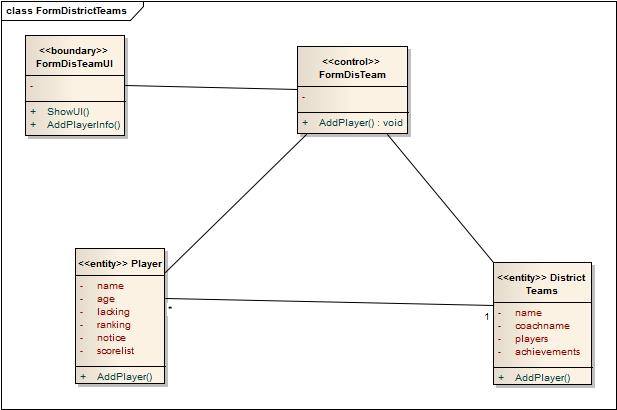


Figure 5.1.4: Class Diagram for form district team Use Case

* 1. **Form Divisional Teams**

5.2.1Noun Identification from use-case narrative:

|  |  |
| --- | --- |
| Actor Action  (Divisional Cricket Coach ) | System Response |
| 1.none | 2. System show the recommended players to divisional coach |
| 3. Divisional coach adds player info to divisional team | 4. System will save added player info to the divisional team and player database |

* + 1. Entity:

1. RecommendList

2. Player

3. DivisionalTeam

5.2.3 Collaboration Diagram:

Figure 5.2.3 shows the collaboration diagram for form divisional team use case. Here we can see the interaction between FormDivTeamUI class, FormDivTeam control class, RecommendList, player and DivisionalTeam entities.

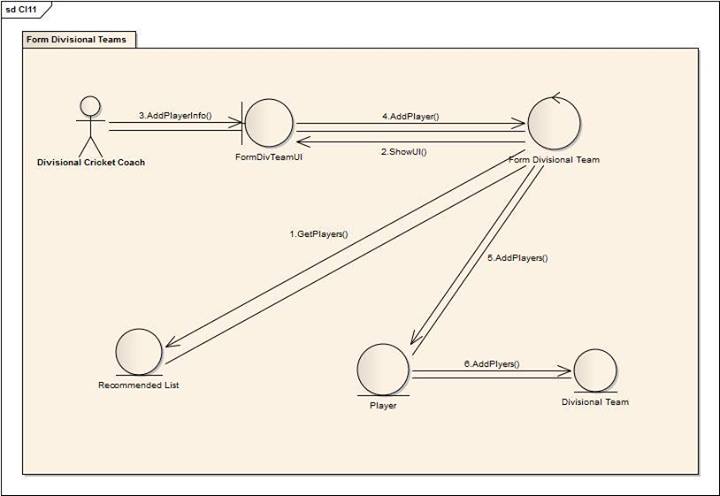


Figure 5.2.3: Collaboration Diagram for form divisional team Use Case

5.2.4 Class Diagram:

Figure 5.2.4 shows the class diagram for form divisional team use case and we can visualize the static aspects of these building blocks and their relationships.

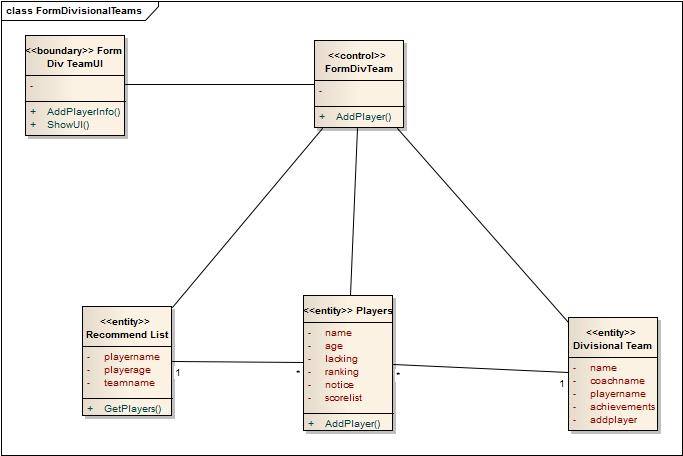


Figure 5.2.4: Class Diagram for form divisional team Use Case

* 1. **Recommend Players**

5.3.1Noun Identification from use-case narrative:

|  |  |
| --- | --- |
| Actor Action  (Divisional Cricket Coach, District Cricket Coach ) | System Response |
| 1.none | 2. System shows the district players to divisional coach |
| 3. District coach selects player | 4. System will show the player info to district coach |
| 6.District coach sends for recommendation to divisional coach | 7.System shows the player info to divisional coach |
| 8.Divisional coach sends feedback to district coach | 9.System shows feedback message to district coach |

* + 1. Entity:

1. DistrictTeam
2. DistrictPlayers

5.3.3 Collaboration Diagram:

Figure 5.3.3 shows the collaboration diagram for Recommend Players use case. Here we can see the interaction between RecommendUI class, GetInfoUI class, Recommend control class, DistrictPlayers and DistrictTeam entities.

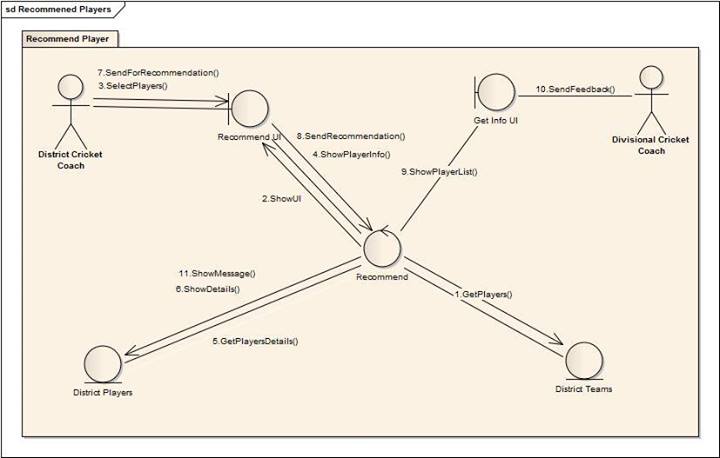


Figure 5.3.3: Collaboration Diagram for Recommend Players use Case

5.3.4 Class Diagram:

Figure 5.3.4 shows the class diagram for Recommend Players use case and we can visualize the static aspects of these building blocks and their relationships.

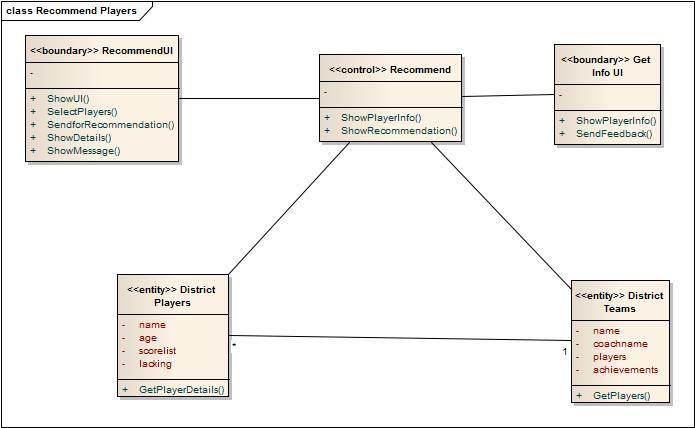


Figure 5.3.4: Class Diagram for Recommend Players Use Case

* 1. **Assign Coach**

5.4.1Noun Identification from use-case narrative:

|  |  |
| --- | --- |
| Actor Action  (Divisional Cricket Manager ) | System Response |
| 1.none | 2. System show a window to the divisional cricket |
| 3. Divisional cricket manager select district coach and divisional cricket coach. | 4. System will save the coach into into the coach database. |

* + 1. Entity:

1. Coaches

5.4.3 Collaboration Diagram:

Figure 5.4.3 shows the collaboration diagram for assign coach use case. Here we can see the interaction between AssignCoachUI class, AssignCoach class and coaches entity.

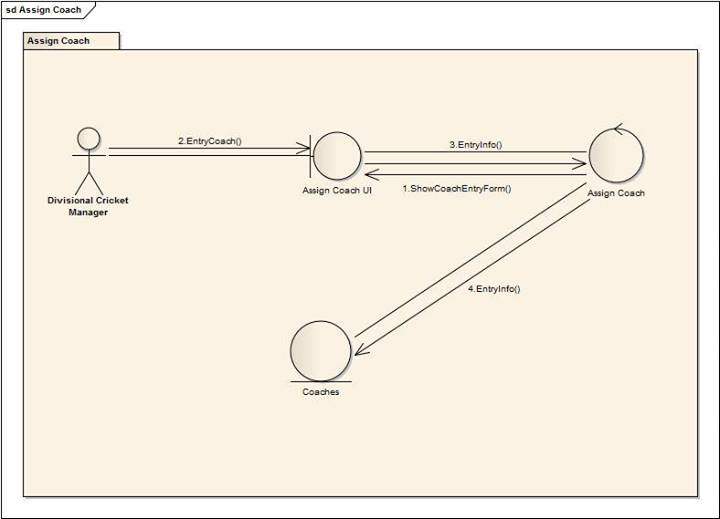


Figure 5.4.3: Collaboration Diagram for Assign Coach Use Case

5.4.4 Class Diagram:

Figure 5.4.4 shows the class diagram for Assign Coach use case and we can visualize the static aspects of these building blocks and their relationships.

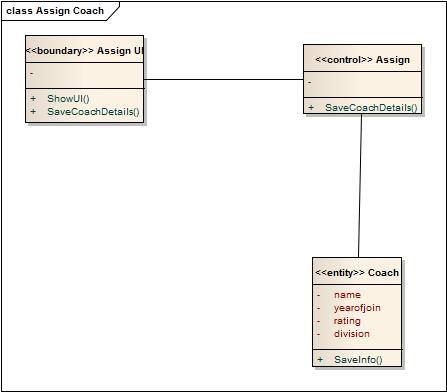


Figure 5.4.4: Class Diagram for Assign Coach Use Case

1. **Coache Management System:**
   1. **Maintain Academy Coaches**

6.1.1Noun Identification from use-case narrative:

|  |  |
| --- | --- |
| Actor Action  (Manager Coach ) | System Response |
| 1.none | 2. System will provide the coaches info to the manager coach |
| 3. Manager coach select coaches from coaches list. | 4. System will get the coaches info |
| 5. Manager coach will update the coaches database. | 6. System will update the coaches info. |

* + 1. Entity:

1. Academy Coaches
2. Academy CoachesList

6.1.3 Collaboration Diagram:

Figure 6.1.3 shows the collaboration diagram for Maintain Academy Coaches use case. Here we can see the interaction between Maintain Coaches UI class, Maintain Coach Class, Academy Coache sand Academy Coaches List entities.

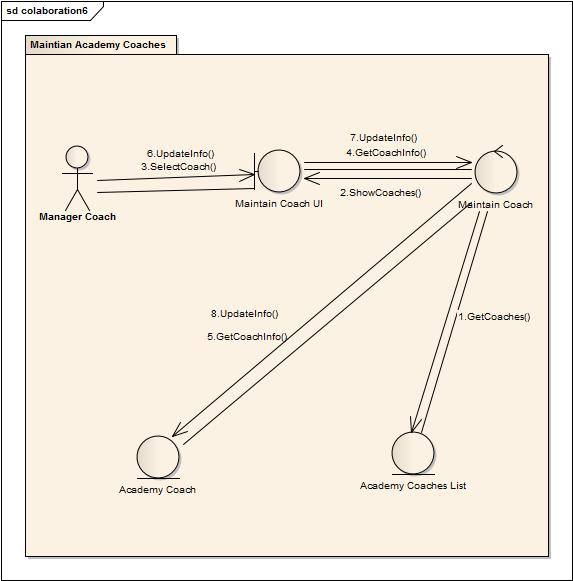


Figure 6.1.3: Collaboration Diagram for Maintain Academy Coaches Use Case

6.1.4 Class Diagram:

Figure 6.1.4 shows the class diagram for Maintain Academy Coaches use case and we can visualize the static aspects of these building blocks and their relationships.

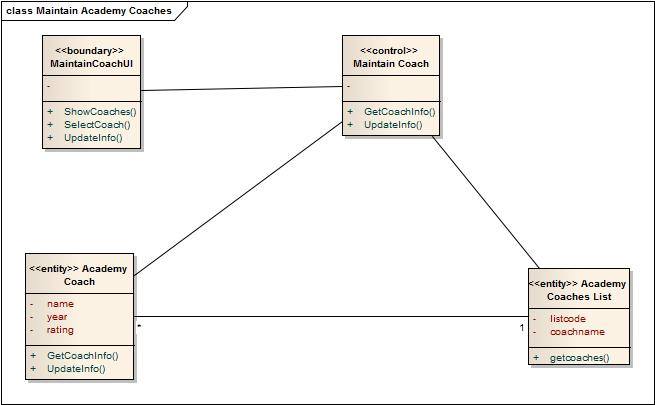


Figure 6.1.4: Class Diagram for Maintain Academy Coaches Use Case

* 1. **Maintain Academy Teams**

6.2.1Noun Identification from use-case narrative:

|  |  |
| --- | --- |
| Actor Action  (Manager Coach ) | System Response |
| 1.none | 2. System will provide the Teams info to the manager coach |
| 3. Manager coach select teams from team list. | 4. System will get the team info |
| 5. Manager coach will update the team database. | 6. System will update the team info. |

* + 1. Entity:

1. Team
2. TeamsList

6.2.3 Collaboration Diagram:

Figure 6.2.3 shows the collaboration diagram for Maintain Academy Teams use case. Here we can see the interaction between MaintainTeamsUI class, MaintainTeam Class, Academy team sand Academy teamsList entities.

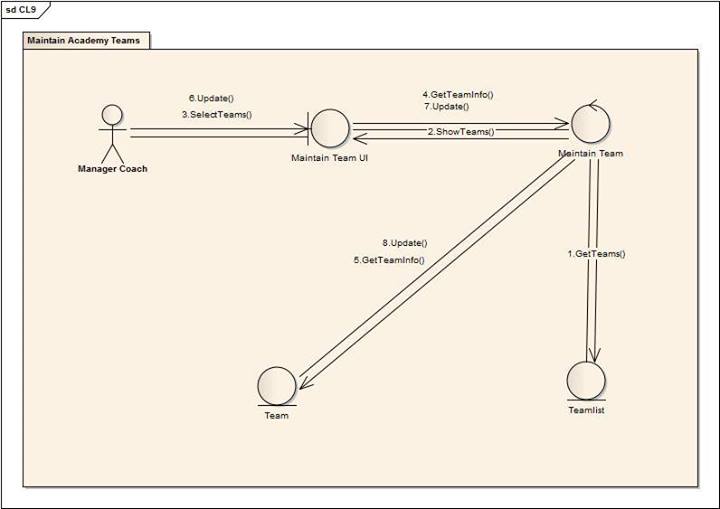


Figure 6.2.3: Collaboration Diagram for Maintain Academy Teams Use Case

6.2.4 Class Diagram:

Figure 6.2.4 shows the class diagram for Maintain Academy teams use case and we can visualize the static aspects of these building blocks and their relationships.

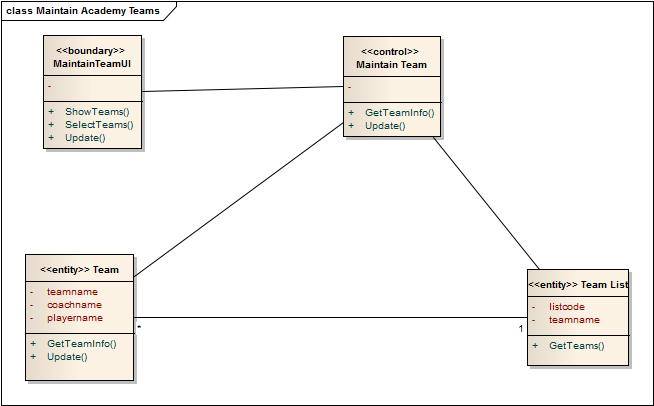


Figure 6.2.4: Class Diagram for Maintain Academy Teams Use Case

* 1. **Notify**

6.3.1Noun Identification from use-case narrative:

|  |  |
| --- | --- |
| Actor Action  (Manager Coach, Operations Manager ofGame Development) | System Response |
| 1.none | 2. System will show a notification window to manager coach. |
| 3. Manager coach sends notifications | 4. System will get the notice details and show manager coach. |
| 5. Operations manager of game development will send feedback to manage coach. |  |

* + 1. Entity:

1. Notification

6.3.3 Collaboration Diagram:

Figure 6.3.3 shows the collaboration diagram for Notify use case. Here we can see the interaction between SendNotificationUI class, SendNotification Class, GetNotificationUI class, Notification entities.

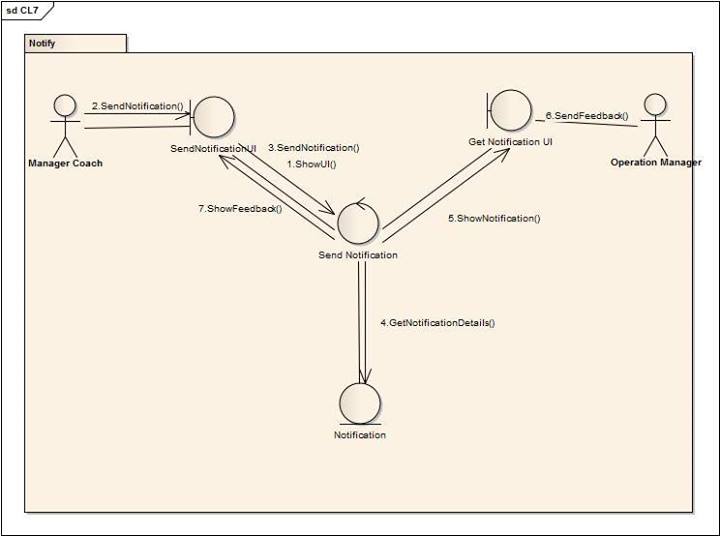


Figure 6.3.3: Collaboration Diagram for Notify Use Case

6.3.4 Class Diagram:

Figure 6.3.4 shows the class diagram for Notify use case and we can visualize the static aspects of these building blocks and their relationships.

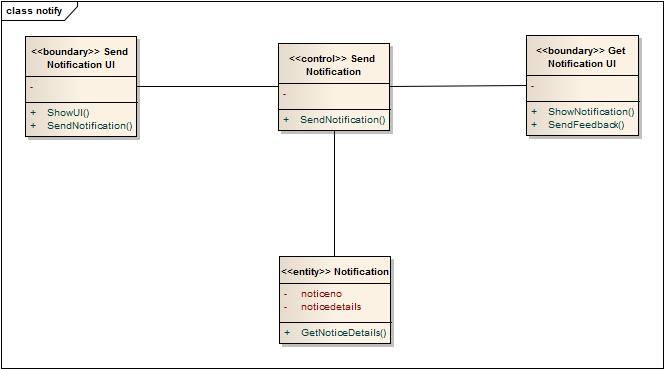


Figure 6.3.4: Class Diagram for Notify Use Case

1. **Control Management System:**
   1. **Observe All Subsystem**

7.1.1Noun Identification from use-case narrative:

|  |  |
| --- | --- |
| Actor Action  (Operations Manager of Game Development ) | System Response |
| 1.none | 2. System will get members and show them to operations manager of Game Development |
| 3. Operations Manager Select members | 4. System will provide a window |
| 5. Operations Manager of Game Development will update member info | 6. System will update member databse |

* + 1. Entity:

1. Member
2. MemberList

7.1.3 Collaboration Diagram:

Figure 7.1.3 shows the collaboration diagram for observe all subsystem use case. Here we can see the interaction between Observe member UI class, Observe Members class, members and Members List entities.

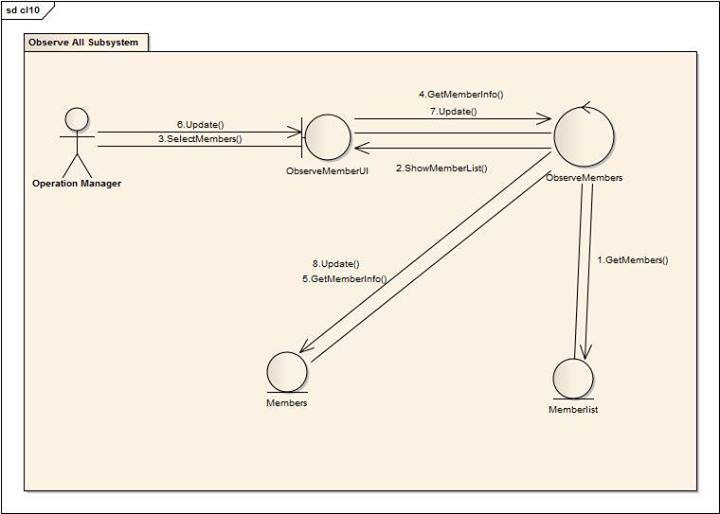


Figure 7.1.3: Collaboration Diagram for Observe all subsystem Use Case

7.1.4 Class Diagram:

Figure 7.1.4 shows the class diagram for Observe all subsystem use case and we can visualize the static aspects of these building blocks and their relationships.

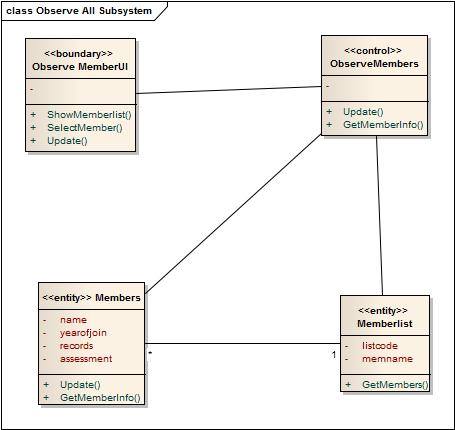


Figure 7.1.4: Class Diagram for Observe all subsystem Use Case

* 1. **Notify**

6.3.1Noun Identification from use-case narrative:

|  |  |
| --- | --- |
| Actor Action  (Operations Manager,  National manager) | System Response |
| 1.none | 2. System will show a notification window to manager coach. |
| 3. Operations manager sends notifications | 4. System will get the notice details and show manager coach. |
| 5. National Manager will send feedback to manage coach. |  |

* + 1. Entity:

1.Notification

7.2.3 Collaboration Diagram:

Figure 6.3.3 shows the collaboration diagram for Notify use case. Here we can see the interaction between Send Notification UI class, Send Notification Class, Get Notice UI class, Notification entities.

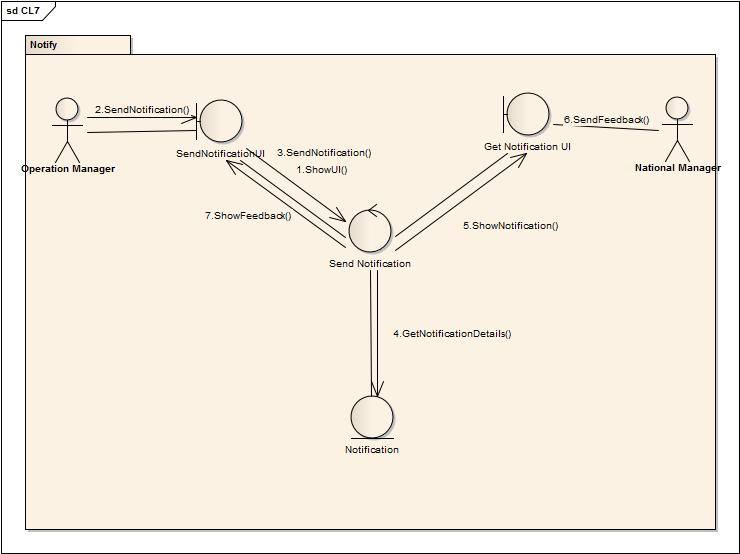


Figure 7.2.3: Collaboration Diagram for Notify Use Case

7.2.4 Class Diagram:

Figure 6.3.4 shows the class diagram for Notify use case and we can visualize the static aspects of these building blocks and their relationships.

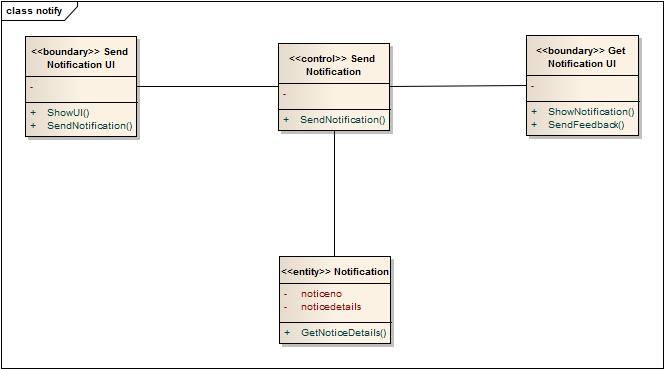


Figure 7.2.4: Class Diagram for Notify Use Case