**GAASYS**

Software Engineering: Catherine Woods

Submitted By: Alan Moriarty (SD\_KCPGD\_Y2)

Submitted on: 13/05/2016

Table of Contents

[**1.** **Introduction** 3](#_Toc432079050)

[**2.** **Functional Requirement** 4](#_Toc432079051)

[**3.** **User Requirements** 5](#_Toc432079052)

[**3.1.** **The system performs team’s administration** 5](#_Toc432079053)

[**3.2.** **The system performs system management** 5](#_Toc432079054)

[**3.3.** **The system performs analysis management** 5](#_Toc432079055)

[**4.** **System Requirements** 6](#_Toc432079056)

[**4.1.** **System Level Use Case Diagram** 6](#_Toc432079057)

[**4.2.** **Manage Teams** 7](#_Toc432079058)

[**4.2.1.** **Register Team** 7](#_Toc432079059)

[**4.2.2.** **Change Team** 9](#_Toc432079060)

[**4.2.3.** **Remove Team** 11](#_Toc432079061)

[**4.3.** **Manage Fixtures** 13](#_Toc432079062)

[**4.3.1.** **Generate Fixtures** 13](#_Toc432079063)

[**4.3.2.** **Amend Fixtures** 16](#_Toc432079064)

[**4.3.3.** **Enter Results** 19](#_Toc432079065)

[**4.3.4.** **Show League Table** 22](#_Toc432079066)

[**4.4.** **Analysis** 24](#_Toc432079067)

[**4.4.1.** **Team Profile** 24](#_Toc432079068)

[**4.4.2.** **Weekly Fixtures** 26](#_Toc432079069)

[**5.** **System Model** 29](#_Toc432079070)

[Level 0 - DFD Diagram 29](#_Toc432079071)

[Level 1 - DFD Diagram 30](#_Toc432079072)

[Level 2 - DFD Diagram – Process P1 31](#_Toc432079073)

[Level 2 - DFD Diagram - Process P2 32](#_Toc432079074)

[Level 2 - DFD Diagram - Process P3 33](#_Toc432079075)

[**6.** **Data Model** 34](#_Toc432079076)

[**7.** **Database Schema** 35](#_Toc432079077)

[**8.** **Program Specifications** 37](#_Toc432079078)

[**9.** **Conclusion** 38](#_Toc432079079)

[**10.** **References:** 39](#_Toc432079080)

# **Introduction**

This System is a GAA League Administration System. All teams playing in the league must be registered before fixtures are generated. Once fixtures are generated, the administration stuff will record fixture date and time for each fixture. When results are entered, team points are allocated. Team points determine the team position in the league. There will also be an option to analyze team performance and generate weekly fixtures/daily listings.

# **Functional Requirement**

# **User Requirements**

The following user requirements have been identified for the system (GAASys):

## **The system performs team’s administration**

* + 1. GAASys will allow the details of each new team to be recorded
    2. GAASys will allow a team’s details to be changed
    3. GAASys will allow the removal of a team from the league

## **The system performs system management**

* + 1. GAASys will generate fixtures for the forthcoming season
    2. GAASys will allow fixture details to be changed
    3. GAASys will process the results of a fixture
    4. GAASys will generate the league table
    5. GAASys will store last year’s fixtures and reset them for the upcoming season

## **The system performs analysis management**

* + 1. GAASys will generate a profile of each team
    2. GAASys will generate a weekly fixtures list

# **System Requirements**

The system requirements for the proposed system:

## **System Level Use Case Diagram**

|  |
| --- |
|  |

Team

Administrator

## **Manage Teams**

Each team must be registered in order to compete in the league.

### **Register Team**

Administrator

Team

<<Includes>>

<<Extends>>

|  |  |  |
| --- | --- | --- |
| **Use Case Name** | **Register Team** | |
| **Use Case Id** | 1 | |
| **Priority** | High | |
| **Source** | Team | |
| **Primary Business Actor** | Administrator | |
| **Other Participating Actors** | Team | |
| **Description** | This function registers a team in the Team’s File. | |
| **Preconditions** | The team are not already registered. | |
| **Trigger** |  | |
| **Typical Scenario** | **Actor Action** | **System Response** |
|  | **Step 1:** The administrator requests a team registration form.  **Step 5:** The administrator inputs the team’s following details and clicks the register button.   * Name * Venue * Email * ContactName * ContactNumber | **Step 2:** The system generates the next TeamId.  **Step 3:** The system checks that the next TeamId is not greater than 12 and a new TeamId is generated. (Last TeamId incremented by 1 or new TeamId of ‘0001’).  **Step 4:** The user interface is displayed.  **Step 6:** Inputted details are validated as follows:   * All fields are required * Email format must be valid * Team must not already be registered   **Step 7:** The system retrieve’s the last TeamId from the Team’s File.  **Step 8:** The team is given the following attributes and sets them all to zero:   * Won * Lost * Drawn * Scores For * Scores Against * Points   **Step 9:** The system saves the team’s details in the Team’s file.  **Step 10:** A confirmation message is displayed.  **Step 11:** The team’s form is displayed. |
| **Alternate Scenarios** | **Actor Action** | **System Response** |
| Invalid data entered.  Twelve teams are already registered |  | **Step 5:** An appropriate error message is displayed.  **Step 6:** The user interface is reset.  **Step 6:** An appropriate error message is displayed.  **Step 7:** The team’s form is displayed. |
| **Conclusions** | The team is registered. | |
| **Post conditions** | This team can now be included in the Fixtures File. | |
| **Business Rules** | A maximum of 12 teams can be registered in the Team’s File.  A team cannot be registered in the Team’s File unless all teams have 0 points. | |
| **Implementation Constraints** |  | |

### **Change Team**

Administrator

Team

<<Includes>>

<<Extends>>

|  |  |  |
| --- | --- | --- |
| **Use Case Name** | **Change Team** | |
| **Use Case Id** | 2 | |
| **Priority** | Medium | |
| **Source** | Teams | |
| **Primary Business Actor** | Administrator | |
| **Other Participating Actors** | Team | |
| **Description** | This function updates a team’s details in the Team’s File. | |
| **Preconditions** | The team is registered in the Team’s File. | |
| **Trigger** |  | |
| **Typical Scenario** | **Actor Action** | **System Response** |
|  | **Step 1:** The administrator requests the Change Team function.  **Step 3:** The administrator selects the desired team.  **Step 5:** The administrator makes the necessary changes to the team’s details and confirms the change. | **Step 2:** The system retrieves a summary of all teams from the Team’s File and displays a list of all the teams on the User Interface.  **Step 4:** The system retrieves and displays the selected team’s details from the Team’s File. The TeamId cannot be altered. Only the following details can be changed:   * Name * Venue * Email * ContactName * ContactNumber   **Step 6:** The system validates the inputted details:   * All fields are required * Email format must be valid   **Step 7:** The team’s details are updated in the Team’s File.  **Step 8:** A confirmation message is displayed.  **Step 9**: The user interface is reset. |
| **Alternate Scenarios** | **Actor Action** | **System Response** |
| Invalid data entered. |  | **Step 7:** An appropriate error message is displayed.  **Step 8:** The system retrieves a summary of all teams from the Team’s File and displays a list of all the teams on the User Interface. |
| **Conclusions** | The team’s details are changed. | |
| **Post conditions** |  | |
| **Business Rules** |  | |
| **Implementation Constraints** |  | |

### **Remove Team**

Administrator

Team

|  |  |  |
| --- | --- | --- |
| **Use Case Name** | **Remove Team** | |
| **Use Case Id** | 3 | |
| **Priority** | Low | |
| **Source** | Teams | |
| **Primary Business Actor** | Administrator | |
| **Other Participating Actors** | Team | |
| **Description** | This function removes a team from the Team’s File. | |
| **Preconditions** | The team is registered. | |
| **Trigger** |  | |
| **Typical Scenario** | **Actor Action** | **System Response** |
|  | **Step 1:** The administrator requests the Remove Team function.    **Step 3:** The administrator selects the desired team.  **Step 5:** The administrator confirms the removal. | **Step 2:** The system retrieves a summary of all teams from the Team’s File and displays a list of all the teams on the User Interface.  **Step 4:** The system retrieves and displays the selected team’s details from the Team’s File.  **Step 6:** The team is removed from the Team’s File.  **Step 7:** A confirmation message is displayed.  **Step 8**: The user interface is reset. |
| **Alternate Scenarios** | **Actor Action** | **System Response** |
|  |  |  |
| **Conclusions** | The team is removed from the Team’s File. | |
| **Post conditions** | The team cannot be included in the Fixtures and Analysis. | |
| **Business Rules** | All team’s must have 0 Points. | |
| **Implementation Constraints** |  | |

## **Manage Fixtures**

### **Generate Fixtures**

Administrator

Team

<<Includes>>

<<Extends>>

|  |  |  |
| --- | --- | --- |
| **Use Case Name** | **Generate Fixtures** | |
| **Use Case Id** | 3.2.1 | |
| **Priority** | High | |
| **Source** | Team’s | |
| **Primary Business Actor** | Administrator | |
| **Other Participating Actors** | Team | |
| **Description** | Generates fixtures for the upcoming season | |
| **Preconditions** | The previous season is finished.  There are 12 teams registered. | |
| **Trigger** | The date is the 01/01/ | |
| **Typical Scenario** | **Actor Action** | **System Response** |
|  | **Step 1:** The administrator requests that the fixtures be generated. | **Step 2:** The system retrieves details of all teams from the Team File in order of TeamID.  **Step 3:** The system initializes the FixtureID to 1.  **Step 4:** Read the first/Next TeamID (Team1)  **Step 5:** Read the next TeamId (Team2)  **Step 6:** If Team1ID is not equal to Team2Id write a fixture to the Fixture File:  **Step 7:** Assign Team1\_Goals, Team1\_Pts, Team2\_Goals, Team2\_Pts to a default value of zero.  **Step 8:** Assign FixtureDate to the first Thursday in April.  **Step 9:** If either team already has a FixtureDate assigned to that date already then the FixtureDate is assigned to seven days later.  **Step 10:** Repeat step 9 until a FixtureDate is found where no team has a game already scheduled.  **Step 11:** Assign FixtureTime to default value of 7:30pm.  **Step 12:** Assign Venue to team that has had the least number of home games.  **Step 13:** If both teams have the same number of home games then a team is randomly selected to be at home.  **Step 14:** Write fixture details in Fixture File   * FixtureID * Team1Id * Team2Id * Team1 Goals * Team1 Points * Team2 Goals * Team2 Points * Venue * FixtureDate * FixtureTime   **Step 14:** Repeat step 6 until last team is reached.  **Step 15:** Repeat step 4 until all teams are processed.  **Step 14:** Display confirmation message. |
| **Alternate Scenarios** | **Actor Action** | **System Response** |
|  | **Step 2:** The administrator reviews fixtures, if any fixture venue clashes with a team’s request the administrator allocates a more appropriate fixture to satisfy all involved. |  |
| **Conclusions** | The fixtures have been generated | |
| **Post conditions** | Schedule fixtures can now take place | |
| **Business Rules** | Only an administrator can generate fixtures | |
| **Implementation Constraints** |  | |

### **Amend Fixtures**

Administrator

Team

<<Includes>>

<<Extends>>

|  |  |  |
| --- | --- | --- |
| **Use Case Name** | **Amend Fixtures** | |
| **Use Case Id** | 4.3.2 | |
| **Priority** | Low | |
| **Source** | Fixtures | |
| **Primary Business Actor** | Administrator | |
| **Other Participating Actors** | Team | |
| **Description** | This function updates the Fixtures File. | |
| **Preconditions** | That the fixture is already generated. | |
| **Trigger** | There is a change in the fixture details. | |
| **Typical Scenario** | **Actor Action** | **System Response** |
|  | **Step 1:** The administrator requests the amend fixtures function.  **Step 3:** The administrator clicks on the desired team.  **Step 5:** The administrator clicks on the desired fixture to amend.  **Step 7:** The administrator makes the necessary changes to the fixture details:   * Venue * FixtureDate * FixtureTime | **Step 2:** The system retrieves a list of all teams from the teams file and displays it on the user interface.  **Step 4:** The system retrieves a list of all fixtures of the selected team from the Fixtures File and displays it on the User Interface.  **Step 6:** The system retrieves the full details of the selected fixture from the fixture file.  **Step 8:** The system validates the inputted details:   * All fields are required * FixtureDate must be in the future * FixtureDate must not clash with another fixture   **Step 9:** The system saves the fixture details in the Fixture File.  **Step 10:** Confirmation message is displayed to the screen. |
|  | **Actor Action** | **System Response** |
| Invalid data entered. | **Step 7:** The administrator inputs the team’s following details again.   * FixtureID * Team1Id * Team2Id * Team1\_Goals * Team1\_Pts * Team2\_Goals * Team2\_Pts * Venue * FixtureDate * FixtureTime | **Step 6:** Invalid data is entered.  **Step 7:** Error message is displayed. |
| **Conclusions** | The fixture details are now updated | |
| **Post conditions** |  | |
| **Business Rules** | Only the administrator can amend a fixture.  Only a fixture without results can be amended. | |
| **Implementation Constraints** |  | |

### **Enter Results**

Administrator

Team

<< Includes >>

<<Extends>>

|  |  |  |
| --- | --- | --- |
| **Use Case Name** | **Enter Results** | |
| **Use Case Id** | 3.2.3 | |
| **Priority** | High | |
| **Source** | Team’s | |
| **Primary Business Actor** | Administrator | |
| **Other Participating Actors** | Team | |
| **Description** | When a match is completed both teams will email the final score of the game to the administrator who in turn inputs the results to the system. | |
| **Preconditions** | The game takes place | |
| **Trigger** | The game takes place | |
| **Typical Scenario** | **Actor Action** | **System Response** |
|  | **Step 1:** Each team email the administrator the result of the fixture.  **Step 2:** The administrator requests the results form from the system.  **Step 4:** The administrator inputs the fixtures results on the form in the following format(x stands for the number that will be inserted):   * Team1 Goals x * Team1 Points x * Team2 Goals x * Team2 Points x | **Step 3:** The system retrieves the results form and displays the User Interface.  **Step 5:** The system validates the inputted details:   * All fields are required * The format is valid   **Step 6:** The system converts the scores of each team by converting the scores into points by multiplying the number of goals by 3 and adding the result to the number of points. ((Goalsx3)+Points).  **Step 7:** Won, Lost, Draw, Scores For, Scored Against attributes updated.  IF PTS\_T1 > PTS\_T2  Points + 2 for T1  IF PTS\_T2 > PTS\_T1  Points + 2 FOR T2  IF PTS\_T1 = PTS\_T2  Points + 1 for T1 AND T2  **Step 8:** The system saves the team details in the team file.  **Step 9:** Confirmation message is displayed to the screen. |
| **Alternate Scenarios** | **Actor Action** | **System Response** |
| **1.**  **2.** | **Step 1:** Each team does not email the administrator the result of the fixture.  **Step 2:** The administrator emails the team/teams asking for the result.  **Step 3:** Carry on from step 3 above.  **Step 1:** The game did not go ahead.  **Step 3:** Carry on from step 1 above. | **Step 2:** The system re-schedules the game for the following Thursday night after the last set of fixtures at 7:30pm. |
| **Conclusions** | The results for the matches have been inputted to the system. | |
| **Post conditions** | The league table con now be generated. | |
| **Business Rules** | Only an administrator can enter results to the system. | |
| **Implementation Constraints** |  | |

### **Show League Table**

Administrator

Team

|  |  |  |
| --- | --- | --- |
| **Use Case Name** | **Show League Table** | |
| **Use Case Id** | 3.3.4 | |
| **Priority** | High | |
| **Source** | Team’s | |
| **Primary Business Actor** | Administrator | |
| **Other Participating Actors** | Team | |
| **Description** | When the results of the fixtures have been inputted into the system a programme will be able to generate a league table | |
| **Preconditions** | Results have been inputted into the system. | |
| **Trigger** | Results have been inputted into the system. | |
| **Typical Scenario** | **Actor Action** | **System Response** |
|  | **Step 1:** The administrator request’s the system run the programme to show the league table. | **Step 2:** The system generates the league table from the programme and displays the result to the screen in a grid view. The table contains the following headings   * Won * Lost * Drawn * Scores For * Scores Against * Points   The table contains this information for each team based on the games played and the results of those game being entered in the Enter Results requirement. |
| **Alternate Scenarios** | **Actor Action** | **System Response** |
|  |  |  |
| **Conclusions** | The league table is displayed | |
| **Post conditions** |  | |
| **Business Rules** |  | |
| **Implementation Constraints** |  | |

## **Analysis**

### **Team Profile**

Administrator

Team

|  |  |  |
| --- | --- | --- |
| **Use Case Name** | **Team Profile** | |
| **Use Case Id** | 3.3.1 | |
| **Priority** | Low | |
| **Source** | Team’s | |
| **Primary Business Actor** | Administrator | |
| **Other Participating Actors** | Team | |
| **Description** | This function displays details of the team | |
| **Preconditions** | Team must be registered | |
| **Trigger** | Requested to display team profile by administrator | |
| **Typical Scenario** | **Actor Action** | **System Response** |
|  | **Step 1:** The administrator requests the team profile function.  **Step 3:** The administrator clicks on the desired team.  **Step 4:** The administrator closes the system. | **Step 2:** The system retrieves a list of all the teams from the Teams File and displays it on the User Interface.  **Step 4:** The system retrieves the team profile of the selected team from the Teams file. The team profile includes   * TeamId * Name * Venue * Email * Won * Lost * Drawn * Scores For * Scores Against * Points   **Step 5:** The system closes. |
| **Alternate Scenarios** | **Actor Action** | **System Response** |
|  |  |  |
| **Conclusions** | Specific team profile is displayed | |
| **Post conditions** |  | |
| **Business Rules** | The team must be already registered to view. | |
| **Implementation Constraints** |  | |

### **Weekly Fixtures**

Administrator

Team

|  |  |  |
| --- | --- | --- |
| **Use Case Name** | **Weekly Fixtures** | |
| **Use Case Id** | 3.3.2 | |
| **Priority** | Low | |
| **Source** | Team’s | |
| **Primary Business Actor** | Administrator | |
| **Other Participating Actors** | Team | |
| **Description** | This function displays a specific week fixture only. | |
| **Preconditions** | Fixtures must have been generated | |
| **Trigger** | Administrator requests weekly fixtures | |
| **Typical Scenario** | **Actor Action** | **System Response** |
|  | **Step 1:** The administrator requests the weekly fixtures function.  **Step 3:** The administrator clicks on the desired fixture week.  **Step 5:** The administrator closes the system. | **Step 2:** The system retrieves a list of all the fixture weeks on the User Interface.  **Step 4:** The system retrieves that week’s fixtures from the Team File and displays it on the User Interface. It will include the following   * FixtureId * Team1Name * Team2Name * Venue * FixtureDate * FixtureTime   **Step 6:** The system closes. |
| **Alternate Scenarios** | **Actor Action** | **System Response** |
|  | **Step 1:** There are no fixtures for that week. |  |
| **Conclusions** | A specific weeks fixtures are displayed | |
| **Post conditions** | The fixtures have already been generated | |
| **Business Rules** |  | |
| **Implementation Constraints** |  | |

# **System Model**

## 5.1. Level 0 - DFD Diagram

Fixture Results

GAASYS

Team

League Table

## 5.2. Level 1 - DFD Diagram

Team

Team Details

P1

Process Teams

Team Details

D1

Teams File

Uses

Fixture Details

D2

Fixtures File

P2

Process

Fixtures

## 5.3. Level 2 - DFD Diagram – Process P1

Change Team Request

P1.2

Change Team

Team

Register

Team

Request

Team Details

Remove Team Request

Team

New

Team Details

New Team Details

Team Details

P1.3

Remove Team

P1.1

Register Team

Team Details

D1

Teams File

## 5.4. Level 2 - DFD Diagram - Process P2

P2.2

Amend Fixtures

Team

Amended Fixture Details

Fixture Details

Result Details

Fixture Details

P2.1

Generate Fixtures

P2.3

Enter Results

D2

Fixture File

Team Details

Result Details

D1

Team File

D1

Team File

P2.4

Show League Table

## 5.5. Level 2 - DFD Diagram - Process P3

P3.1

Team Profile

Team

Team Details

D3

Fixtures File

Fixture Details

D1

Team File

P3.2

Weekly Fixtures

# **Data Model**

|  |
| --- |
| Home Team |
| TeamID (PK)  Name  Venue  Email  Won  Lost  Drawn  ScoresFor  ScoresAgainst  Points  ContactName  ContactNumber |

|  |
| --- |
| Away Team |
| TeamID (PK)  Name  Venue  Email  Won  Lost  Drawn  ScoresFor  ScoresAgainst  Points  ContactName  ContactNumber |

11..11

🡨

🡨

11..11

Played By

Played By

1

1

|  |
| --- |
| Fixtures |
| FixtureID (PK)  Team1Goals  Team1Points  Team2Goals  Team2Points  Venue  FixtureDate  FixtureTime |

# **Database Schema**

Relation: HomeTeam

Attributes:

TeamId numeric(5) NOT NULL auto increment,

Name varchar(30) NOT NULL,

Venue varchar(30) NOT NULL,

Email varchar(30) NOT NULL,

ContactName varchar(30) NOT NULL,

ContactNumber varchar(15) NOT NULL,

Won numeric(2) NOT NULL DEFAULT 0,

Lost numeric(2) NOT NULL DEFAULT 0,

Drawn numeric(2) NOT NULL DEFAULT 0,

ScoresFor numeric(3) NOT NULL DEFAULT 0,

ScoresAgainst numeric(3) NOT NULL DEFAULT 0,

Points numeric(3) NOT NULL DEFAULT 0,

Primary Key: TeamId

Relation: AwayTeam

Attributes:

TeamId numeric(5) NOT NULL auto increment,

Name varchar(30) NOT NULL,

Venue varchar(30) NOT NULL,

Email varchar(30) NOT NULL,

ContactName varchar(30) NOT NULL,

ContactNumber varchar(15) NOT NULL,

Won numeric(2) NOT NULL DEFAULT 0,

Lost numeric(2) NOT NULL DEFAULT 0,

Drawn numeric(2) NOT NULL DEFAULT 0,

ScoresFor numeric(3) NOT NULL DEFAULT 0,

ScoresAgainst numeric(3) NOT NULL DEFAULT 0,

Points numeric(3) NOT NULL DEFAULT 0,

Primary Key: TeamId

Relation: Fixtures

Attributes:

FixturesId numeric(5) NOT NULL auto increment,

Team1Id numeric(5) NOT NULL,

Team2Id numeric(5) NOT NULL,

Team1Goals numeric(2) NOT NULL,

Team1Points numeric(2) NOT NULL,

Team2Goals numeric(2) NOT NULL,

Team2Points numeric(2) NOT NULL

Venue varchar(30) NOT NULL,

FixtureDate date NOT NULL,

FixtureTime char(5) NOT NULL,

Primary Key: FixturesId

Foreign Key: Team1Id REFERENCES HomeTeam

Foreign Key: Team2Id REFERENCES AwayTeam

# **Program Specifications**

# **Conclusion**

My project allows the user to generate a league table using the data inputted by the administrator in the Enter Results section. It has a Register, change and remove team function to manage the teams. It generates the fixtures and allows them to be amended. Once the game is played the results are entered and the league table will be updated. The team profile and weekly fixtures functions are for viewing purposes only which add extra functionality.

# **References:**