

# DEATH-MATCH

## DETAILED DESIGN REPORT



### PROJECT GROUP MEMBERS

AHMET UĞUR e1502806

İSMAİL SAMET SORKUN e1502657

MEHMET E. ŞENER e1503010

MURAT EZGİ BİNGÖL e1502210

## Index

1	Introduction.....	5
1.1	Problem Definition .....	5
1.2	Purpose.....	6
1.3	Scope .....	6
1.4	Overview.....	6
1.5	Definitions, Acronyms and Abbreviations.....	7
1.6	References.....	7
2	System Overview .....	8
2.1	Design Overview.....	8
2.2	Goals and Objectives .....	8
3	Design Considerations .....	9
3.1	Design Assumptions, Dependencies and Constraints .....	9
3.1.1	Time Constraints.....	9
3.1.2	Performance Constraints.....	9
3.2	Design Goals and Guidelines .....	10
3.2.1	Extreme Programming.....	10
3.2.2	Usability .....	11
3.2.3	Functionality.....	11
3.2.4	Reliability .....	11
4	Data Design.....	12
4.1	Data Description .....	12
4.2	Data Dictionary .....	18
5	System Architecture .....	23
5.1	Architectural Design .....	23
5.2	Description of Components.....	25
5.2.1	Main Game Module.....	25
5.2.2	Sound Engine Module .....	27
5.2.3	Match Engine Module .....	28
5.2.4	Graphic Engine Module .....	29
5.2.5	Data Handler Module .....	29
5.2.6	Forum Module.....	30

5.2.7	GUI Module .....	31
6	User Interface Design .....	31
6.1	Overview of User Interface .....	31
6.2	Screen Images.....	34
6.3	Screen Objects and Actions .....	39
7	Detailed Design.....	40
7.1	Main Game Module.....	40
7.1.1	GameMain Class .....	41
7.1.2	Update Class .....	42
7.1.3	Manager Class .....	44
7.1.4	Team Class .....	44
7.1.5	YouthTeam Class .....	45
7.1.6	Transfer Class.....	46
7.1.7	Staff Class.....	48
7.1.8	Matches Class .....	48
7.1.9	Player Class.....	50
7.1.10	SeniorPlayer Class.....	51
7.1.11	YouthPlayer Class .....	52
7.2	Sound Engine Module .....	53
7.2.1	MatchSounds.....	53
7.2.2	MenuSounds.....	55
7.2.3	GameMusics .....	57
7.3	Match Engine Module .....	59
7.3.1	Match.....	60
7.3.2	MatchSimulate .....	61
7.4	Graphic Engine Module .....	63
7.4.1	FlashHandler Class.....	63
7.5	Data Handler Module .....	65
7.5.1	Executer .....	65
7.6	Forum Module.....	66
7.6.1	Forum .....	66
7.7	GUI Module .....	68

7.7.1	Interface .....	68
8	Libraries and Tools.....	70
8.1	PHP .....	70
8.2	MVC .....	71
8.3	Kohana.....	72
8.4	WampServer.....	72
8.5	Sqlyog .....	73
8.6	NuSphere PhpEd.....	73
8.7	ORM Library.....	73
9	Time Planning (Gannt Chart) .....	75
9.1	Term 1 Gannt Chart.....	75
9.2	Term 2 Gannt Chart.....	76
10	Conclusion .....	77

# 1 Introduction

## 1.1 Problem Definition

Online games are becoming more and more popular recently with the increasing internet user numbers and football manager games are also very popular in last decade and they have lots of fans. On the other hand, Facebook and Facebook applications is the most attractive part of the internet with the huge number of participants. However, there is not any popular football manager game in Facebook like the games Farmville or Texas Hold'em Poker.

In this project, our goal is to create an interactive, multiplayer game which will be played on Facebook. The game will be a football management like game. In Facebook applications, this type of games is rare. So our main purpose is to design a football manager game which will be played on Facebook. Differently from the known popular online manager games, this game will also have visual effects. In this project, the general goals are:

- To implement a multiplayer environment
- Provide virtual reality by using flash animations
- To implement a simple and user-friendly interface
- Integrating AI into the game

## 1.2 Purpose

The purpose of this document is to give a detailed design for Best-Eleven Football Manager Game project that will be developed by team members of Death-Match under the Computer Engineering Design I course. It includes initial design concepts and descriptions of the proposed software system design of the project Best-Eleven Football Manager and the software development methods and strategies underlying.

## 1.3 Scope

This document describes how the software system will be structured for Best-Eleven Football Manager project. This document is prepared according to *IEEE Recommended Practice for Software Design Descriptions* document.

The design decisions that we make while preparing this document will be a very useful guide for us in the future work. In addition to this, these design decisions can be changed later.

## 1.4 Overview

In this document, design considerations about Best-Eleven Football Manager project are explained. General description of the software system is given. Moreover, detailed explanations and diagrams about data design, architectural design, and user interface design are given. Libraries and tools that will be used within the project are given. In addition, a detailed explanation for components is given. Time planning that will be followed by group members can be found for both terms.

## 1.5 Definitions, Acronyms and Abbreviations

OPM: Object Relational Mapping

SQL: Structured Query Language

PHP: Hypertext Preprocessor

MVC: Model View Controller

## 1.6 References

### **Document:**

IEEE Std. 1016-1998: IEEE Recommended Practice for Software Design Descriptions.

### **Libraries and Tools:**

<http://www.php.net/>

<http://en.wikipedia.org/wiki/Model%E2%80%93View%E2%80%93Controller>

<http://kohanaframework.org/>

<http://www.wampserver.com/>

[http://en.wikipedia.org/wiki/Object-relational\\_mapping](http://en.wikipedia.org/wiki/Object-relational_mapping)

<http://www.webyog.com/en/>

<http://www.nusphere.com/>

## 2 System Overview

### 2.1 Design Overview

To construct our design we will have a database to hold all information about users, teams and players. The user interactions and information will be handled by PHP, and Kohana which is a PHP5 framework. In the match important positions will be animated with the help of Flash. The design will be explained more detailed in design considerations and data design sections of this document.

### 2.2 Goals and Objectives

Since Best-Eleven Football Manager game will be a Facebook game, a lot of people will be playing it online. In the game, all users begin with a team and try to lead leagues and championships by improving players and using necessary tactics. Users will also think about transfers from other teams to improve the strength of their teams. With the transfer issues, a strong economy becomes more important. Users must think all these issues and they will also compete with each other in leagues, and championships. While playing their games they will also enjoy online competition.

There are some other online football manager games on internet like Hattrick and SoccerManager. They have thousands of users, but they do not have enough visual effects. Unlike these games we will have animations which will show the important positions during a match.



Our aim is to design an online football manager game better than other available games on internet like Hattrick and SoccerManager, and unlike these games our game will be played on Facebook.

## **3 Design Considerations**

### **3.1 Design Assumptions, Dependencies and Constraints**

#### **3.1.1 Time Constraints**

Project is decided to be finished in seven months according to the schedule. For the end of the first semester we have to prepare a prototype. All members of the project have own weekly program so homework and exam because of the other courses. In order to prepare a demo and finalize the project successfully, we have to divide the project in some tasks and each member of the project has one task for two weeks period. After the tasks are finished we make meeting and there will be new tasks for each members. This period will be continuing when the Project is finished. The time planning of the project and the demo includes which properties of the project are also defined in the Time Planning section.

#### **3.1.2 Performance Constraints**

Because Best-11 Football Manager is a web Project the performance is one of the important tasks. The chosen framework Kohana is one of the quickness frameworks for php. But database that will be used initially will keep a small data. So, it will not affect the

performance. However, later large data can be added to game. For large amount of data performance will be an important issue.

## 3.2 Design Goals and Guidelines

### 3.2.1 Extreme Programming

Extreme Programming emphasizes teamwork. Managers, customers, and developers are all equal partners in a collaborative team. Extreme Programming implements a simple, yet effective environment enabling teams to become highly productive. The team self-organizes around the problem to solve it as efficiently as possible. Extreme Programming improves a software project in five essential ways; communication, simplicity, feedback, respect, and courage.

Every small success deepens their respect for the unique contributions of each and every team member. With this foundation Extreme Programmers are able to courageously respond to changing requirements and technology.

The most surprising aspect of Extreme Programming is its simple rules. Extreme Programming is a lot like a jig saw puzzle. There are many small pieces. Individually the pieces make no sense, but when combined together a complete picture can be seen. The rules may seem awkward and perhaps even naive at first, but are based on sound values and principles.

Because Best-11 Football Manager is web application it must be programmed in one of the web development technique. Extreme programming is the most suitable technique for this web Project and for group members.

### 3.2.2 Usability

One of the most important design goals is usability for the project because; users want to play simple and usable games. Best-11 Football Manager will have many users; many football management game players think that the interface and the functions are complicated. They need to play for a period in order to understand them. Therefore, we will design a simple and understandable game interface. Moreover, the features and the game modes that will be available will not be complicated.

### 3.2.3 Functionality

Functionality is another important issue that should be considered. We should keep the interface and features simple in order to users can play the game easily. Best-11 Football Manager has different functions. For example, during matches we are planning to add two modes. 2D animated match and commentator mode.

### 3.2.4 Reliability

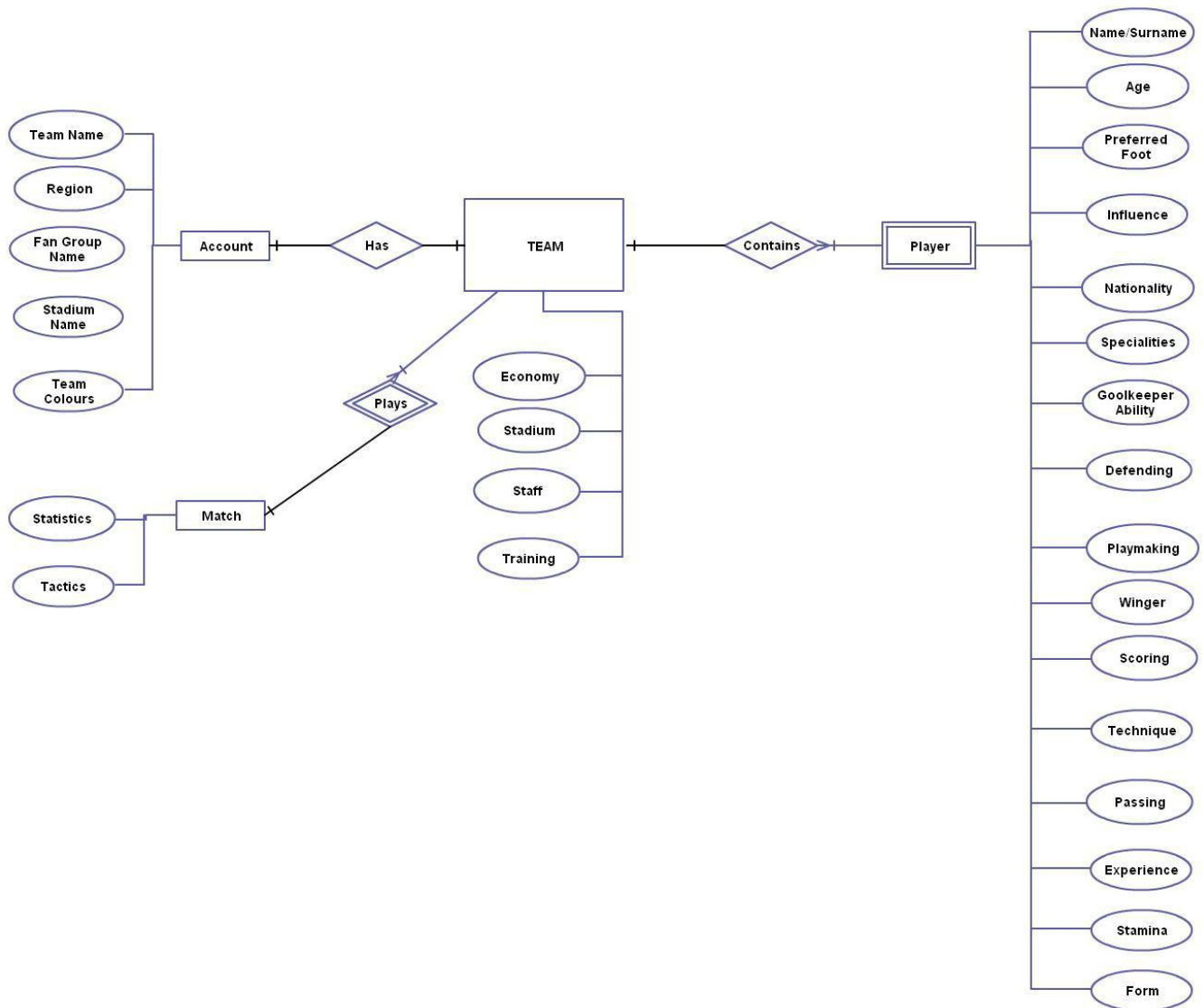
Because Best-11 Football Manager will be a online multiplayer football management game we have to design it with minimum faults. It will be used various testing strategies at the milestones that we reach while designing the project in order to improve the performance and decrease the number of errors that will occur. Another issue about the reliability is that the data that will be used should be realistic.

## **4 Data Design**

### **4.1 Data Description**

All of the data about users, teams, players and leagues are stored in database. Each new user will have a new team with new players.

Visual effects and animations will be loaded in match by main application.



- Entity Relationship Diagram -

There will be four data objects in our database:

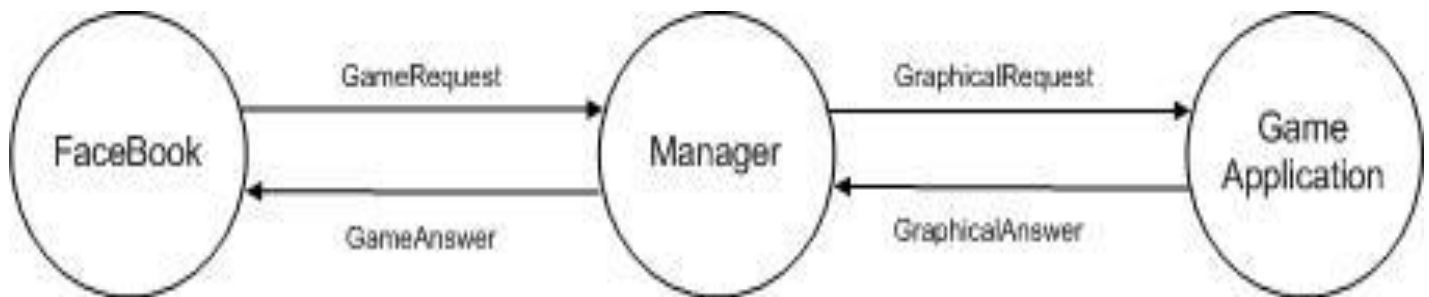
- Account
- Team
- Player
- Match

Data flow is explained with three different levels of diagrams. The data flow will be understood by showing level-0, level-1, and level-2 diagrams and explaining them.

### Level-0

The project is in its primitive form. User will send request to enter Facebook and then manage the team by sending requests to game application. Facebook accept the entrance request and game takes requests and answers them accordingly.

This level is represented by the following diagram:



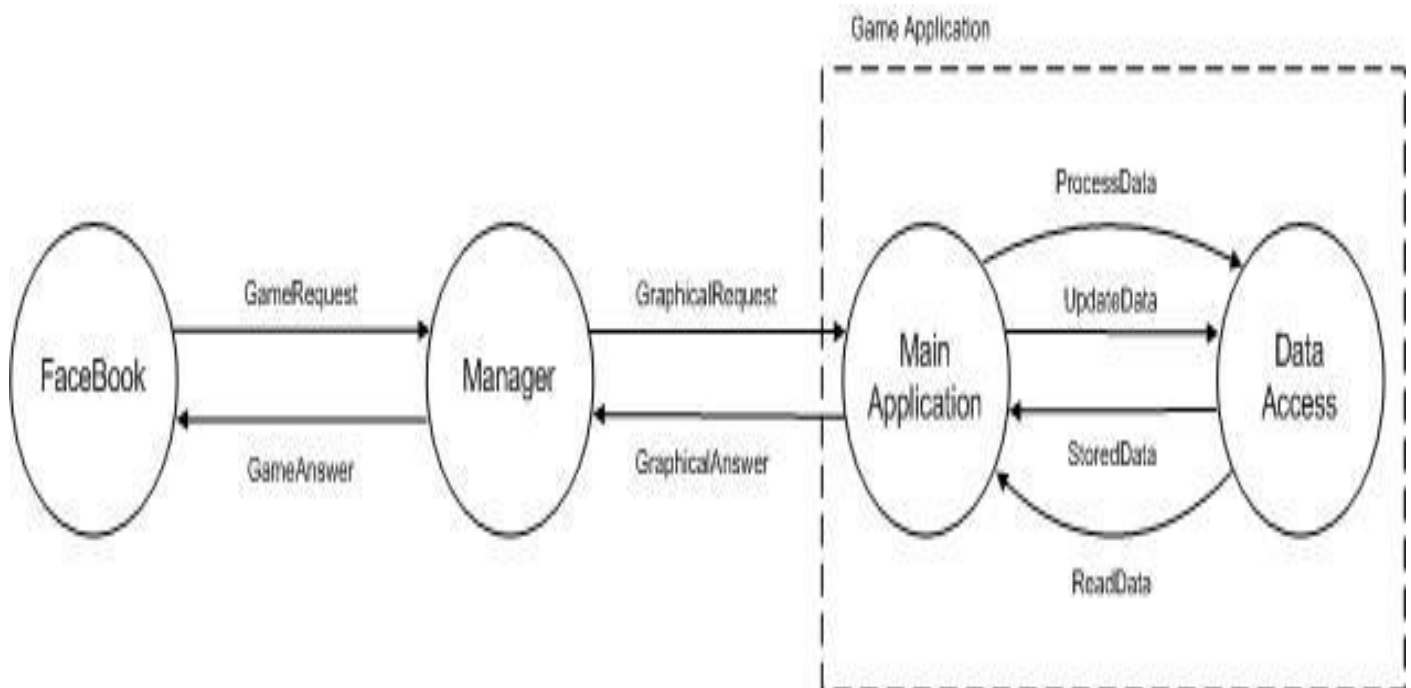
- Data Flow Diagram -

### Level-1

In this data flow diagram we opened game application to two components, main application and data access.

Main application will take request from manager instead of game application and it will handle data issues with data access by sending requests for needed data and taking data.

This level is represented by the following diagram:



- Data Flow Diagram -

## Level-2

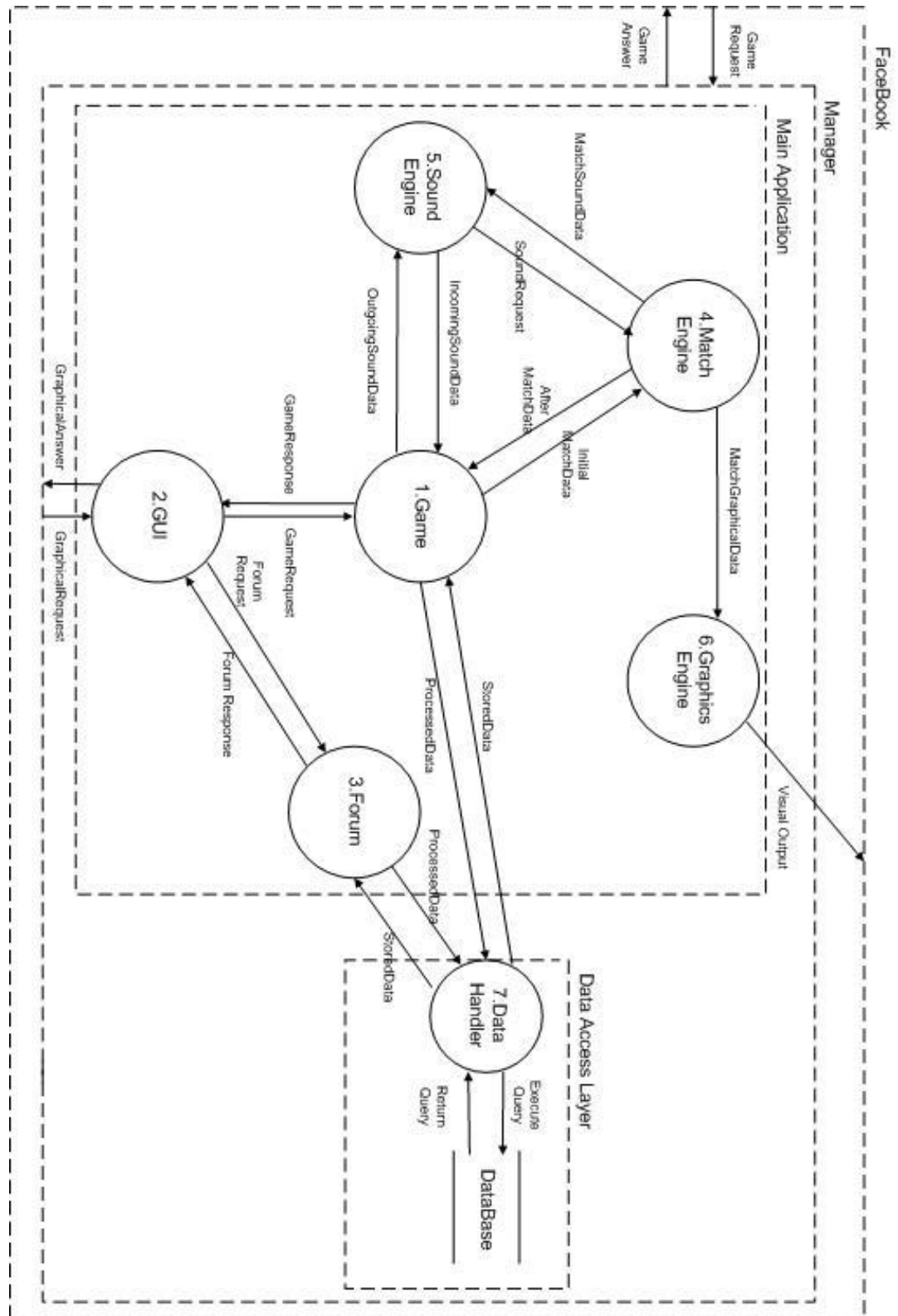
In this diagram data flow is explained more detailed. Main application will be represented by 6 parts.

Users access to Forum and the Game by GUI. Both Game and Forum takes necessary data from database and sends back the processed data by taking requests from user.

Game process also sends user requests to SoundEngine and MatchEngine. SoundEngine and MatchEngine also send and take requests between each other. In addition, MatchEngine sends all graphical results to GraphicEngine. Then graphical results are displayed to the user as 2D images.

This level is represented by the following diagram:





-Data Flow Diagram-

## 4.2 Data Dictionary

<b>Name</b>	GraphicalRequest
<b>Alias</b>	None
<b>Where/How Used</b>	GUI(input) Manager(output)
<b>Description</b>	GraphicalRequest = Mouse or Keyboard Call

<b>Name</b>	GraphicalAnswer
<b>Alias</b>	None
<b>Where/How Used</b>	Manager(input) GUI(output)
<b>Description</b>	GraphicalAnswer = GUI Screen

<b>Name</b>	VisualOutput
<b>Alias</b>	None
<b>Where/How Used</b>	Manager(input) GraphisEngine(output)
<b>Description</b>	VisualOutput = Match image

<b>Name</b>	GameResponse
<b>Alias</b>	None
<b>Where/How Used</b>	GUI(input) Game(output)
<b>Description</b>	GameResponse = Output of GUI

<b>Name</b>	GameRequest
<b>Alias</b>	None
<b>Where/How Used</b>	Game(input) GUI(output)
<b>Description</b>	GameRequest = Data values determined by Managers interaction

<b>Name</b>	OutgoingSoundData
<b>Alias</b>	None
<b>Where/How Used</b>	SoundEngine(input) Game(output)
<b>Description</b>	OutgoingSoundData = Data values to determine output of SoundEngine

<b>Name</b>	IncomingSoundData
<b>Alias</b>	None
<b>Where/How Used</b>	Game(input) SoundEngine(output)
<b>Description</b>	IncomingSoundData = Values that are determined by sound input

<b>Name</b>	InitialMatchData
<b>Alias</b>	None
<b>Where/How Used</b>	MatchEngine(input) Game(output)
<b>Description</b>	InitialMatchData = Initial values to start a new match

<b>Name</b>	AfterMatchData
<b>Alias</b>	None
<b>Where/How Used</b>	Game(input) MatchEngine(output)
<b>Description</b>	AfterMatchData = Values that are determined by the result of match

<b>Name</b>	StoredData
<b>Alias</b>	None
<b>Where/How Used</b>	Game(input) DataHandler(output)
<b>Description</b>	StoredData = Data values that are read from database

<b>Name</b>	ProcessedData
<b>Alias</b>	None
<b>Where/How Used</b>	DataHandler(input) Game(output)
<b>Description</b>	ProcessedData = Data values that are sent to database

<b>Name</b>	SoundRequest
<b>Alias</b>	None
<b>Where/How Used</b>	MatchEngine(input) SoundEngine(output)
<b>Description</b>	SoundRequest = A request determined by the Manager's interaction with SoundEngine

<b>Name</b>	MatchSoundData
<b>Alias</b>	None
<b>Where/How Used</b>	SoundEngine(input) MatchEngine(output)
<b>Description</b>	MatchSoundData = Data values that will determine the output of SoundEngine

<b>Name</b>	MatchGraphicalData
<b>Alias</b>	None
<b>Where/How Used</b>	GraphicsEngine(input) MatchEngine(output)
<b>Description</b>	MatchGraphicalData = Data values that will determine the output of GraphicsEngine

<b>Name</b>	ExecuteQuery
<b>Alias</b>	None
<b>Where/How Used</b>	Database(input) DataHandler(output)
<b>Description</b>	ExecuteQuery = Query that handles changes in database

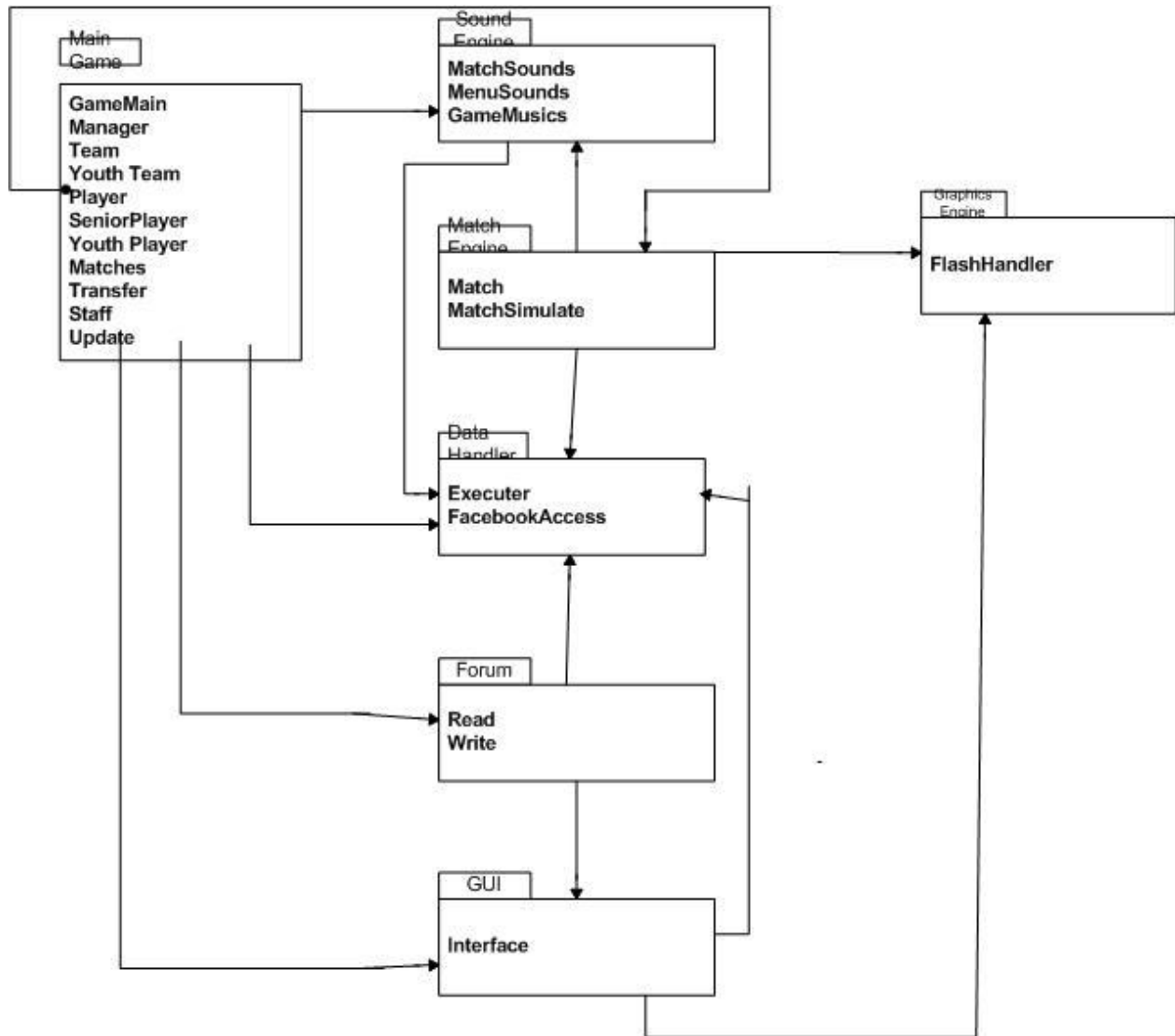
<b>Name</b>	ReturnQuery
<b>Alias</b>	None
<b>Where/How Used</b>	DataHandler(input) Database(output)
<b>Description</b>	ReturnQuery = Return tables that are the result of queries

<b>Name</b>	ForumRequest
<b>Alias</b>	None
<b>Where/How Used</b>	Forum(input) GUI(output)
<b>Description</b>	ForumRequest = A request determined by interaction with Forum

<b>Name</b>	ForumResponse
<b>Alias</b>	None
<b>Where/How Used</b>	GUI(input) Forum(output)
<b>Description</b>	ForumResponse = The response to requests of users from Forum

## 5 System Architecture

### 5.1 Architectural Design



--Modules Diagram--

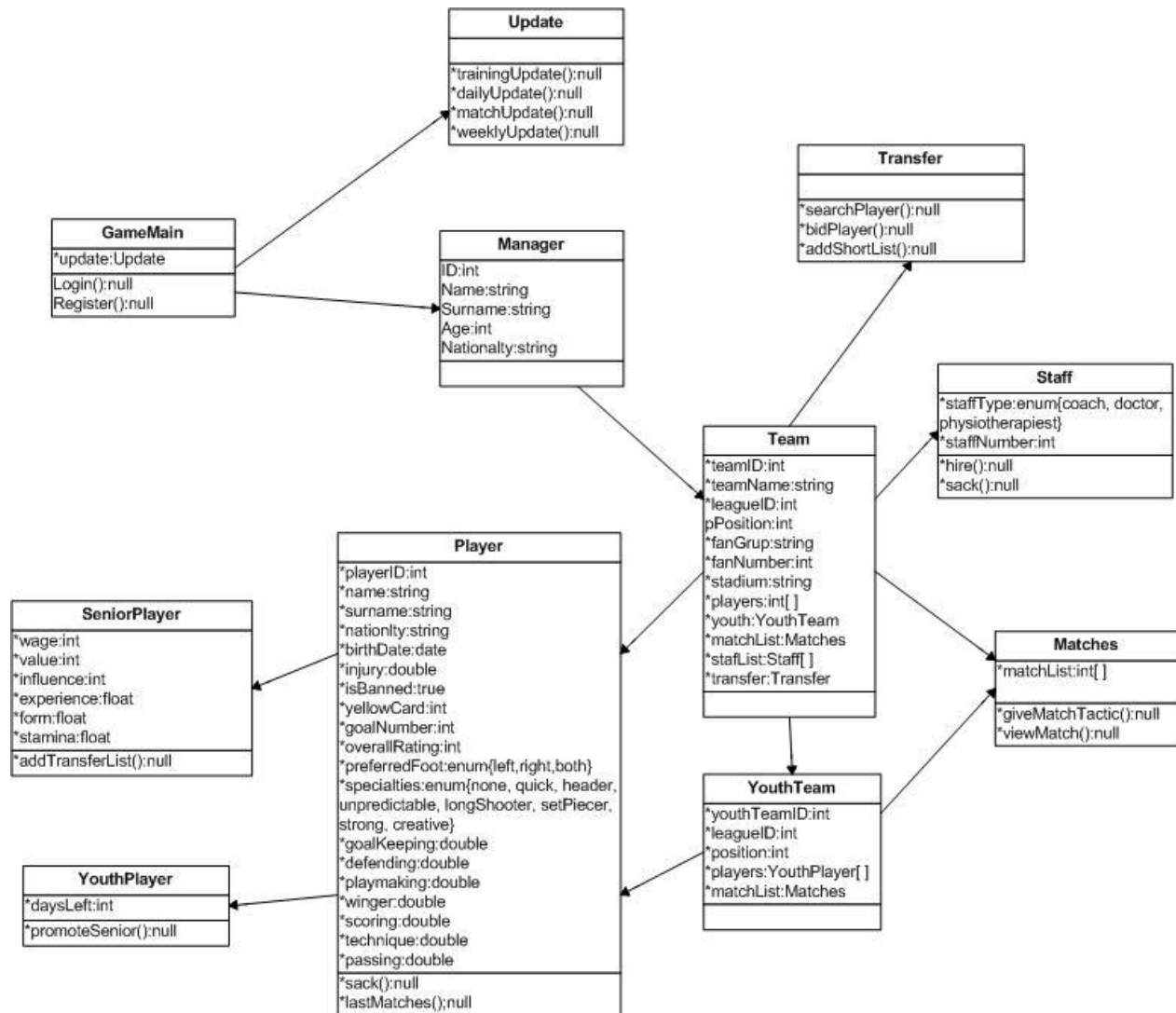
All of the modules that will be used within the project and their relations is shown with the figure above. Basically we have 7 modules:

- Main Game: This module contains, GameMain, Manager, Team, YouthTeam, Player, SeniorPlayer, YouthPlayer, Matches, Transfer, Staff and Update classes.
- Sound Engine: This module contains, MatchSounds, MenuSounds and GameMusics classes.
- Match Engine: This module contains Match and MatchSimulate classes.
- Graphic Engine: This module contains FlashHandler class.
- Data Handler: This module contains Executer and FacebookAccess classes.
- Forum: This module Forum class.
- GUI: This module contains Interface class.



## 5.2 Description of Components

### 5.2.1 Main Game Module



---Main Game Module Class Diagram---

The class diagram hierarchy given above is the classes related to the Main Game. Some important about the classes are as follows:

- Game Main class is the main class for logging and registering to the game and it has update object.
- Manager class keeps information about users.
- Team class keeps information of current team like teamID, players, matches etc. This class has association with Player, Staff, Transfer, Matches, YouthTeam classes.
- YouthTeam class is just like Team class but it is for Youth Team and has fewer attributes than Team class.
- Matches class has matchList which contains match ids of current team. This class controls match actions.
- Staff class has a staff type and changeable staff numbers.
- PlayerClass is the parent class of both SeniorPlayer and YouthPlayer classes. PlayerClass mainly includes the attributes of a player like wage, shooting, passing etc.

## 5.2.2 Sound Engine Module

MatchSounds
fileFormat : string name : string
play() : void configureAudioEffect() : void mute() : void

MenuSounds
fileFormat : string name : string
play() : void configureAudioEffect() : void mute() : void

GameMusics
fileFormat : string name : string
play() : void pause() : void configureAudioEffect() : void mute() : void

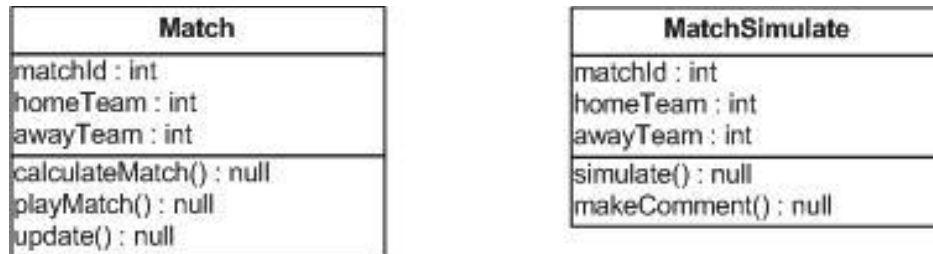
---Sound Engine Module Class Diagram---

- MatchSound class is for play sounds during matchess. This class can be mutable like all classes in this module.
- MenuSound class is used for playing the sound files saved when clicking the menu buttons.
- GameMusic class is used for playing the sound files that were arranged as game music during whole game.

The three classes have the same type variables :

- fileFormat : This variable is a string type variable. It keeps the extension of the sound file that will be played. (like .wav, .mp3 etc.)
- name : This variable is a string type variable. It keeps the name of the sound file without extension.

### 5.2.3 Match Engine Module



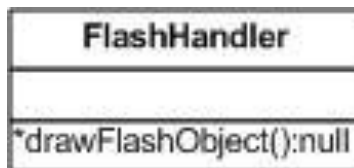
--Match Engine Module Class Diagram--

- Match class has the functions that calculate the matches' results or the situations of the footballers. In addition to calculations, the class plays matches in commentary mode. The "update" function starts the update operations that will be executed.
- MatchSimulate class is used for simulating the expected results before matches. The "makeComment" function comments the result of the simulated match. By showing these comments to the users, it helps to the manager for their selections of tactics.

These two classes have the same type variables.

- `matchId` : This variable is an integer type variable. It keeps the id of the match that will be simulated or calculated.
- `homeTeam` : This variable is an integer type variable. It keeps the id of the homeTeam which plays in the simulated or calculated match.
- `awayTeam` : This variable is an integer type variable. It keeps the id of the awayTeam which plays in the simulated or calculated match.

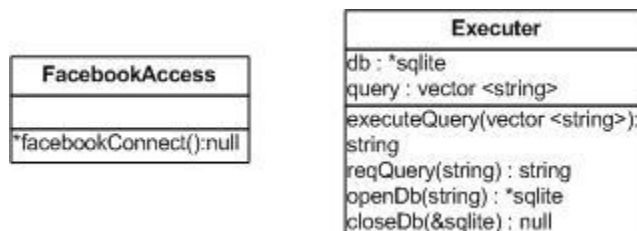
## 5.2.4 Graphic Engine Module



--Graphic Engine Module Class Diagram—

- FlashHandler class : The class is used to communicate with flash and create flash objects. The whole game and the whole buttons used in the game window will be created by this class.  
The class has no variables.

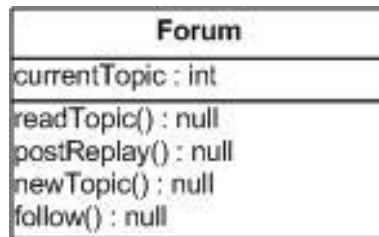
## 5.2.5 Data Handler Module



--Data Handler Module Class Diagram—

- FacebookAccess class : This class is used to connect the game with facebook platform.
- Executer class : This class is used for the SQL queries and the database connections where the objects and the required data is hold in it.

### 5.2.6 Forum Module

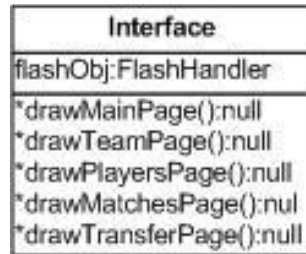


--Forum Module Class Diagram—

- Forum class : This class is used for managing whole information about the forum. It holds the functions that were required for creating and editing the forum topics, and making them available to read by the users.

The only variable that the class has is currentTopic which is an integer type variable. It keeps the id of the topic which is edited or viewed at that time.

## 5.2.7 GUI Module



--GUI Module Class Diagram—

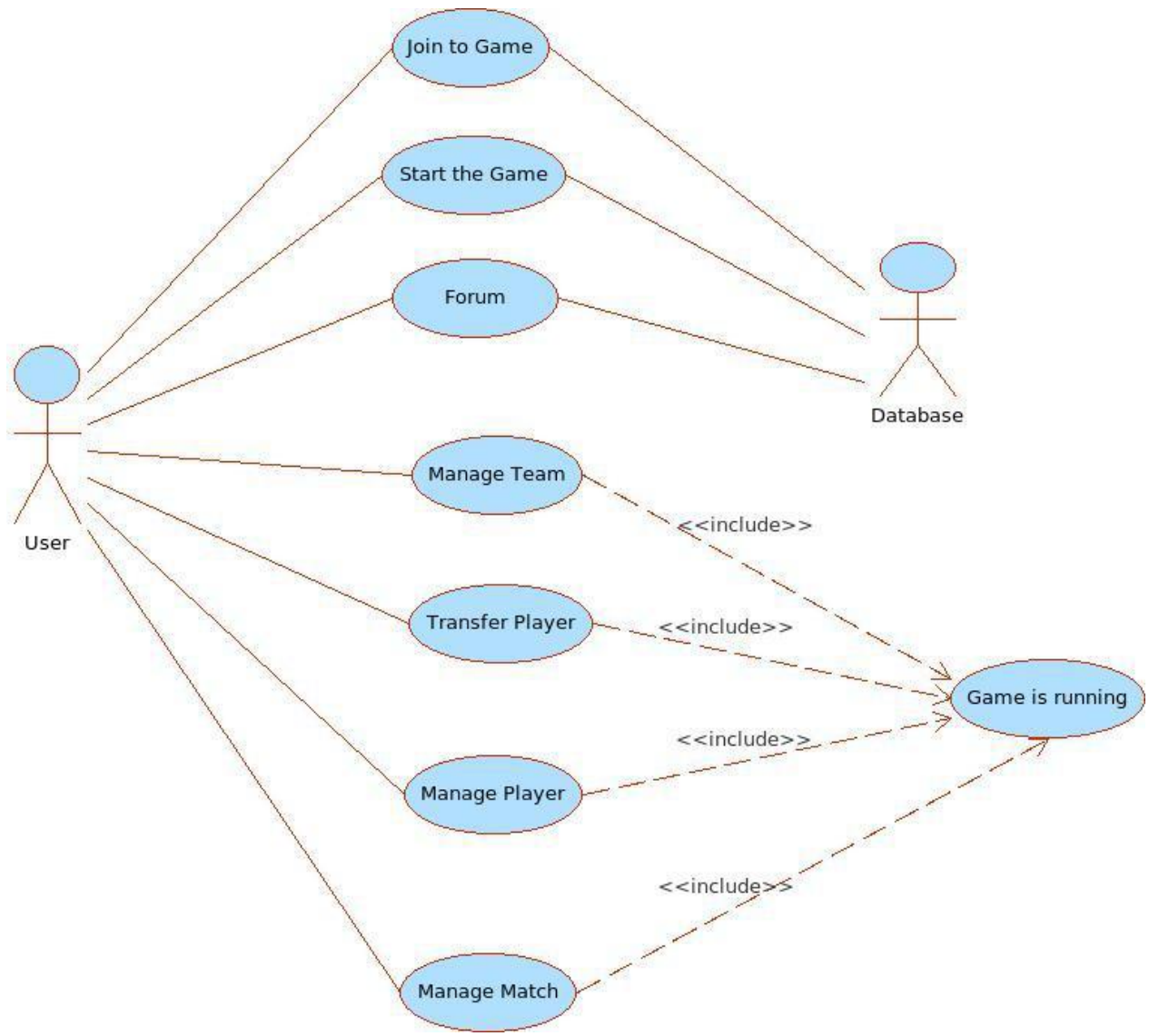
- Interface class : This class is used for preparing graphical user interfaces. It will create HTML pages that hold flash objects. The only part that will be in touch with the users is that interface part.

The only variable of that class is “flashObj” which is an instance of the class FlashHandler.

## 6 User Interface Design

### 6.1 Overview of User Interface

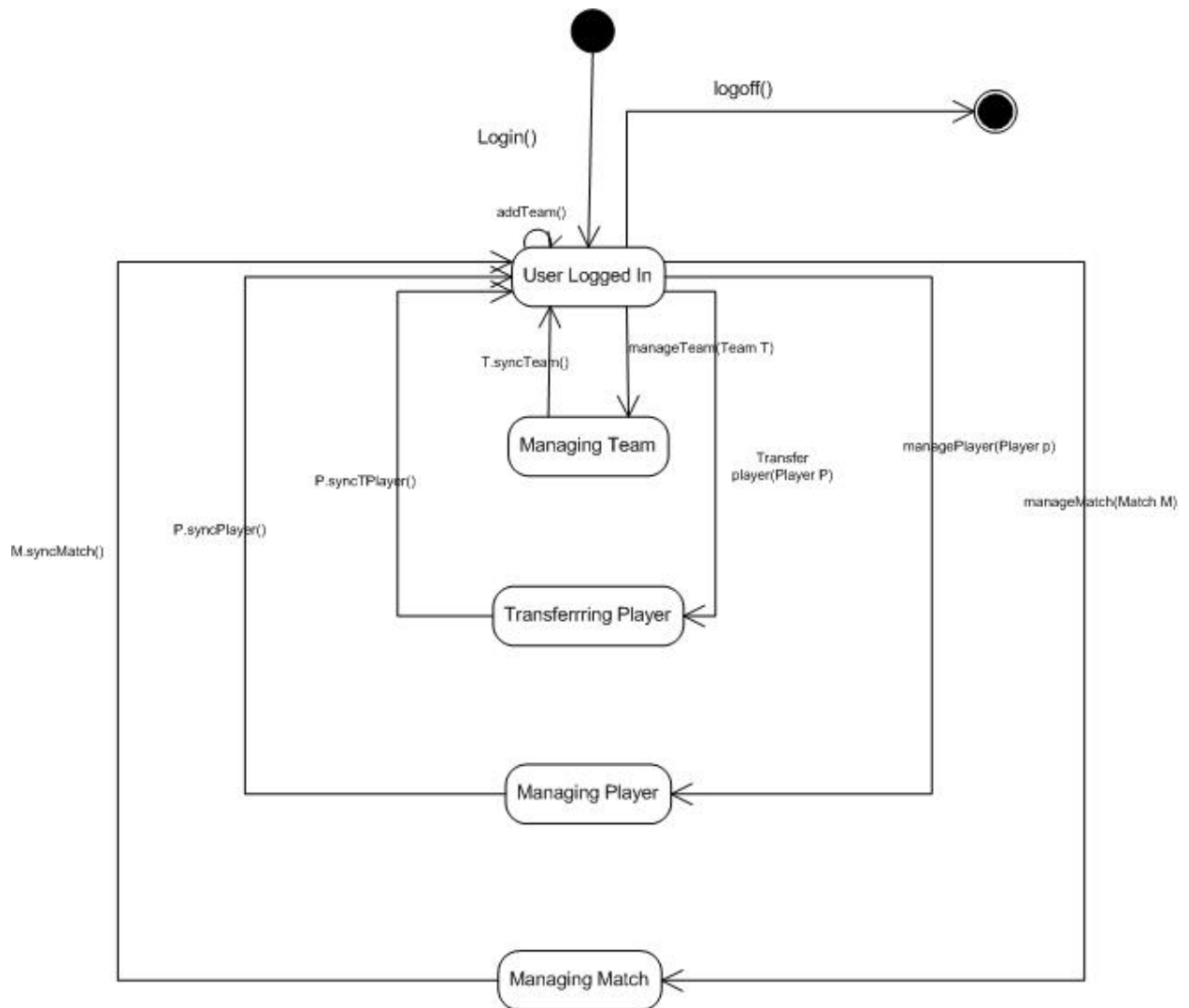
We designed the interface of Best-11 Football Manager according to the use case diagram below which was constructed during the requirement analysis stage.



- Use Case Diagram -

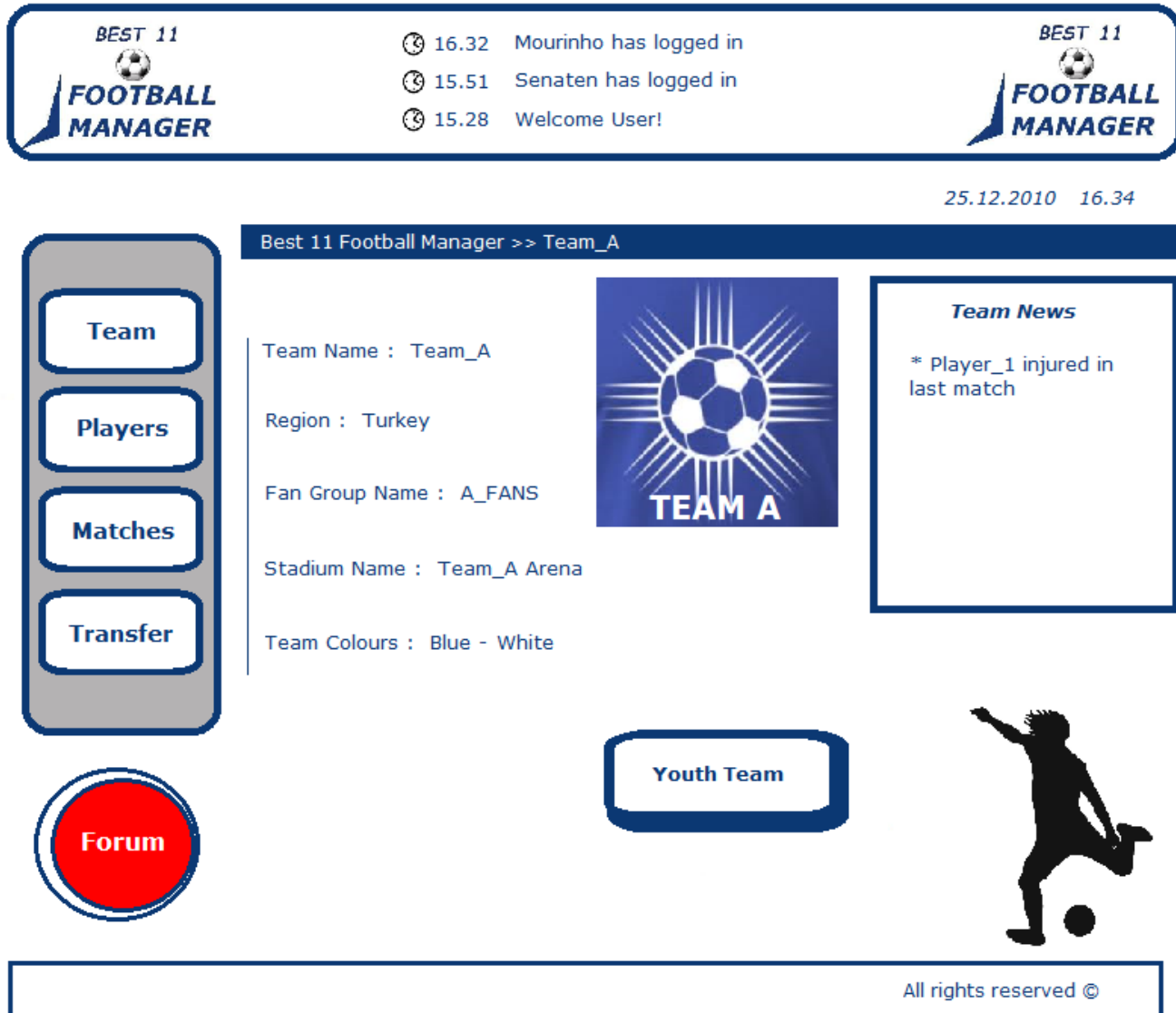


Users will be able to use our system according to the state transition diagram below which was constructed during the requirement analysis stage.



- State Transition Diagram -

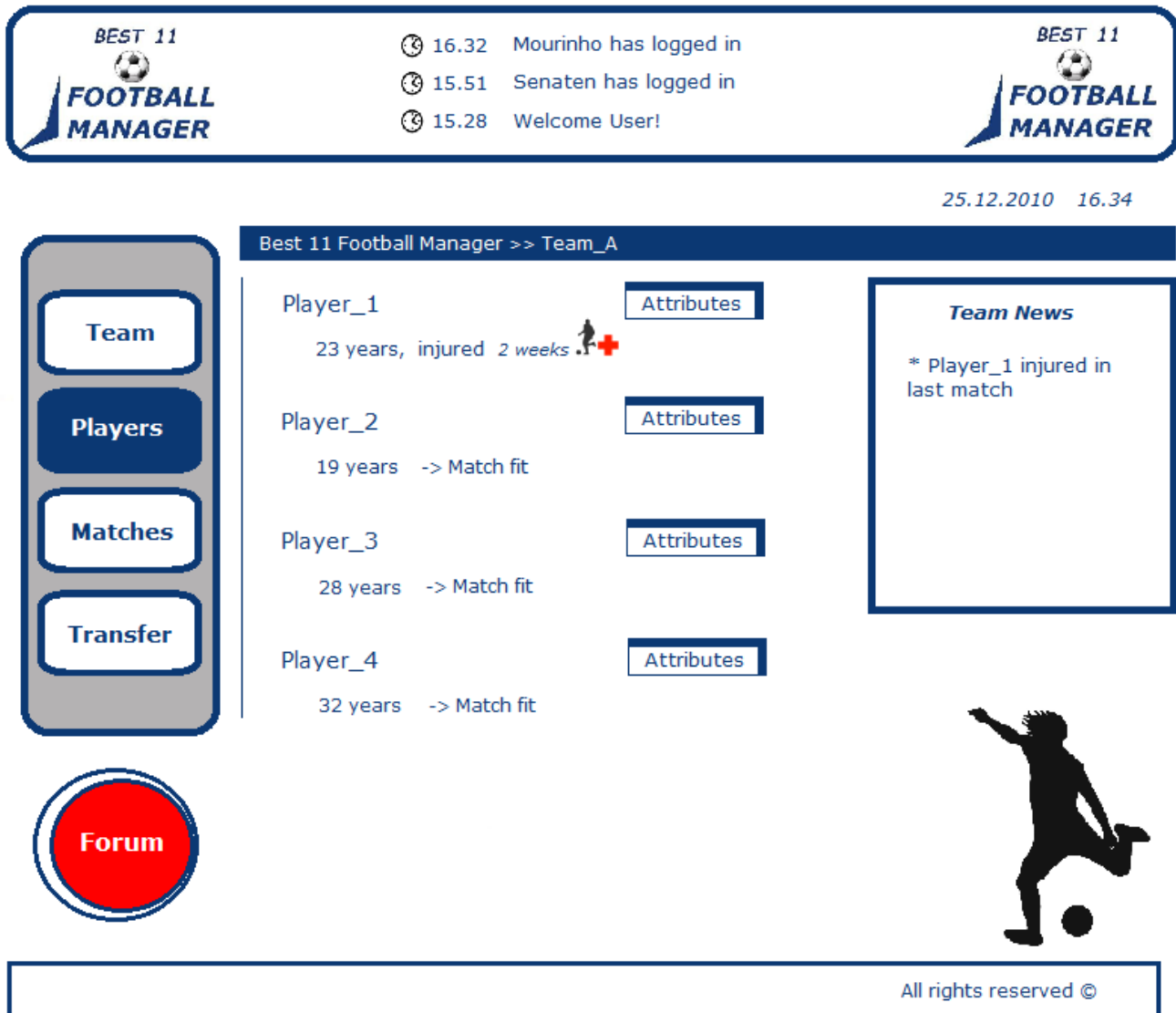
## 6.2 Screen Images



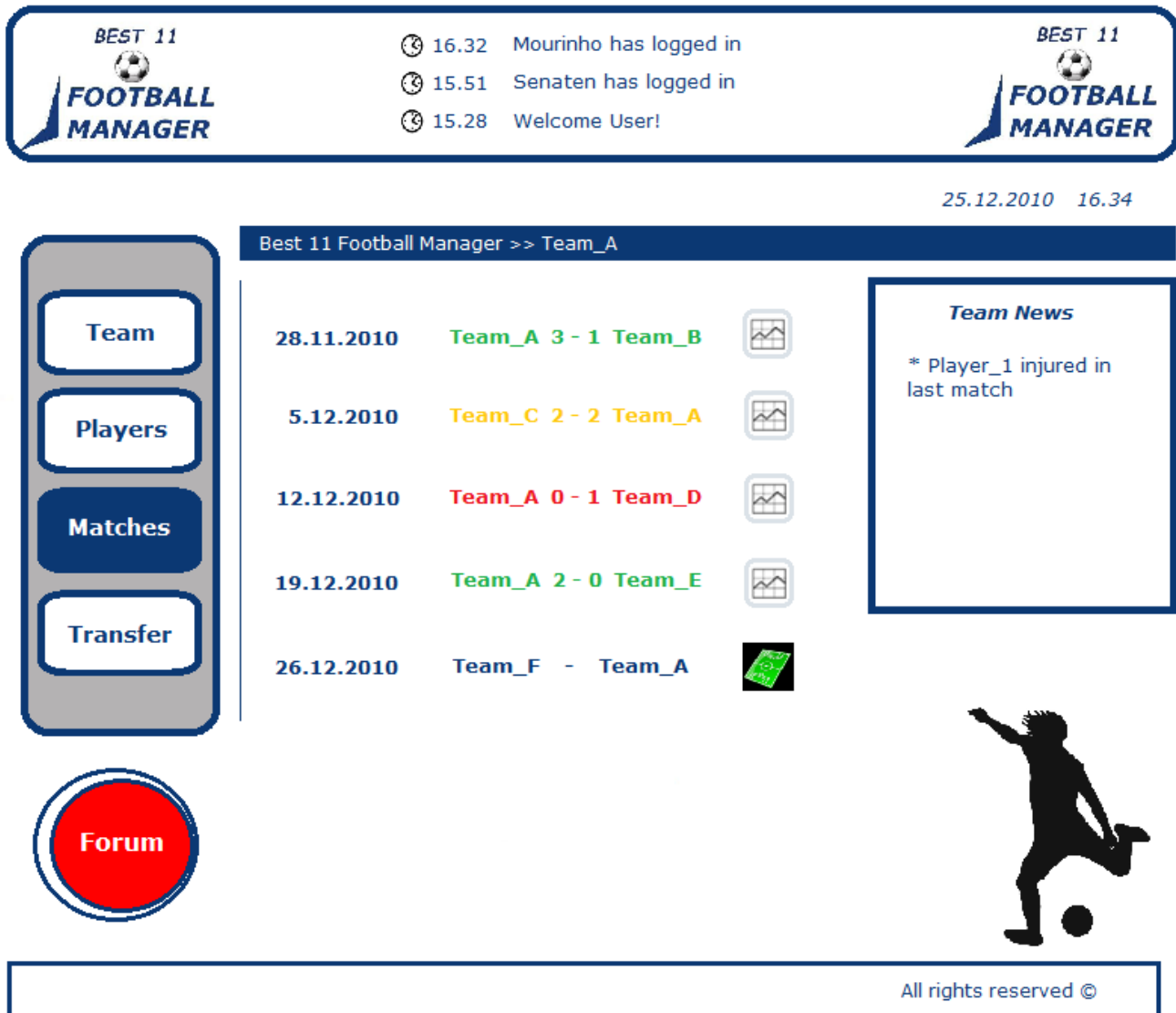
This screen will be main page screen seen by user.



This screen will be team page screen seen by user.




This screen will be players page screen seen by user.



This screen will be matches page screen seen by user.

BEST 11



FOOTBALL  
MANAGER

16.32

Mourinho has logged in


15.51

Senaten has logged in

15.28

Welcome User!

BEST 11



FOOTBALL  
MANAGER

25.12.2010 16.34

Team

Players

Matches

Transfer

Best 11 Football Manager >> Team\_A

Transfer Listed? ☒ Yes ☐ No

Age

Anything

Value

Between

-

Current Bid

Between

-

Attributes

Select Attributes


CLEAR

SEARCH

Team News

\* Player\_1 injured in last match

Forum



All rights reserved ©

This screen will be transfer page screen seen by user.

- ✓ All images designed by Death-Match group members.

## 6.3 Screen Objects and Actions

### Team

User will access different team pages from here such as:

- ❖ Economy
- ❖ Stadium
- ❖ Staff
- ❖ Training

In Economy page, total incomes and outcomes of two weeks will be seen and compared.

In Stadium page, recent seat plan of the stadium will be seen and stadium improvements will be managed.

In Staff page, club staffs will be seen and changed.

In Training page, training schedule will be set.

### Players

In this page, any player can be selectable to be analyzed by selecting “Attributes”.

When a player selected, its profile will appear on the screen. Each player’s estimated value and wage can be seen from there. Also, each player can be added to transfer list from there.

## **Matches**

The matches will be seen and chosen from here in form of fixture as sorted. When one of the previous matches is chosen, statistics of this match will be retrieved. When one of the future matches is chosen, tactics pages of this match will be retrieved. Matches can be watched by selecting the match which is currently playing.

## **Transfer**

User can search player through transfer market with selected attributes. Also user can observe the transfer listed players and select them from this page. They can make bids for these players.

## **Forum**

Different forum categories appear when it is selected by the user. After selecting category, user can read existing threads or they can add new threads. When the user selects one of the existing threads from this page, they can read posts and add new posts.

# **7 Detailed Design**

## **7.1 Main Game Module**

There are eleven classes named GameMain, Update, Manager, Team, Transfer, Staff, Matches, YouthTeam, Player, YouthPlayer and SeniorPlayer in the Main Game Module.



### 7.1.1 GameMain Class

- **Classification :**

GameMain is a class under the Main Game Module.

- **Definition**

The main class that builds the start of the game.

- **Responsibilities**

The responsibilities and abilities of the class are:

1. Register to the game
2. Login to the game
3. Start the update operations

- **Composition**

The class has 2 functions:

1. login() : the fuction called for the login operations that the user makes. Until the operation ends, a “loading page” page appears on the screen.
2. register() : the function called for the register operation that the user makes. The user information is taken via facebook when the user allows the game to reach their information. If they don’t allow, the sart operation of the game is cancelled for the reason they can not play without register.

- **Uses/Interactions**

When the user log in, the class immediately calls the class Manager and with this operation, the game starts.

In addition, the class keeps the instance of class Update for the specified updates.

- **Processing**

When user logged in, facebook request for starting the game. After this request server loads needed data about current user's team.

At register process, system adds new team for user to database after facebook request.

- **Interface/Exports**

This component does not take any inputs. It only takes request from facebook and return with login or register user interface.

### 7.1.2 Update Class

- **Classification**

Update is a class under the Main Game Module.

- **Definition**

The class manages all of the update operations.

- **Responsibilities**

The responsibility and ability of the class is calling the suitable update function according to the different time periods decided before.

- **Constraints**

The update operations will be done at specified times independent from the user operations.

- **Composition**

The class has 4 functions:

1. `trainingUpdate()` : by looking the matches played that week, it decides the amount of need of ability training of the players. The decrease of ability because of the age will be handled by this function.
2. `dailyUpdate()` : the injury states of the players will be changed. The stadium construction progress will be controlled by this function.
3. `weeklyUpdate()` : economical updates will be controlled by this function.
4. `matchUpdate()` : the scoreboard, league position and the point chart will be controlled and updated by this function. Also, the injured and banned players at current match will be updated with this function. The experience will be updated just after the match.

- **Resources**

Server automatically do this operations at a specified times, so all processor and memory activities will be handled by the server.

### 7.1.3 Manager Class

- **Classification**

Manager is a class under the Main Game Module.

- **Definition**

The class keeps the information of the manager like name, surname, etc.

- **Constraints**

Every manager has to have a team, i.e. there will be no manager without team or more than one teams. So, manager class has the instance of team class

### 7.1.4 Team Class

- **Classification**

Team is a class under the Main Game Module.

- **Definition**

The class keeps the information of the team like team ID, team name etc.

- **Composition**

The class keeps the informations about team and also keeps player list, staff list, match list and an instance of youthTeam class.

- **Constraints**

The staff list and the instance of YouthTeam class are the forced resources for this class. The staff list must be included in the class even if it is empty.

Every team has exactly one youth team.

Every team has a player list and this player list must have at least 11, at most 40 players. The players in the player list must be over age 18.

### 7.1.5 YouthTeam Class

- **Classification**

YouthTeam is a class under the Main Game Module.

- **Definition**

The class is similar to the Team class. That controls youth teams of the teams.

- **Constraints**

Every youth team has a player list and this player list must have at least 11, at most 40 players. The players in the list must be under age 21.

### 7.1.6 Transfer Class

- **Classification**

Transfer is a class under the Main Game Module.

- **Definition**

All of the transfer operations except adding to transfer list will be done with this class. Adding to transfer list operation will be handled by Player class.

- **Responsibilities**

The responsibility and the ability of the class is running the functions about the transfer operations when user wants to make transfers or searches players on the transfer list. Also, the function to create a short list of desired players is done within this class.

- **Constraints**

To make transfer, the money of the team should be more than or equal to bid player. In addition, the number of players should be less than 40 in the player list of the senior team.

There is no transfer for the youth team.

- **Composition**

The class has 3 functions:

1. `searchPlayer()` : searches players in the transfer list that covers the given attribute by users.
2. `bidPlayer()` : bids to the player that user wants to transfer.
3. `addShortList()` : adds the players in the transfer list to transfer short list of the user.

- **Interface/Exports**

`searchPlayer` interface is loaded by the `searchPlayer()` function of the class. After the execution with given attributes, the players returned from the function is listed in this interface.

### 7.1.7 Staff Class

- **Classification**

Staff is a class under the Main Game Module.

- **Definition**

The class for the numbers of the different type staffs in the team.

- **Responsibilities**

The operations of the staffs like hiring or sacking are done within the class

- **Composition**

The class has 2 functions:

1. hire() : increases the staffNumber of current staff for hiring new staff.
2. sack() : decreases the staffNumber of current staff for sacking one staff.

### 7.1.8 Matches Class

- **Classification**

Matches is a class under the Main Game Module.



- **Definition**

The class to make operations about past and future matches.

- **Responsibilities**

The responsibilities and availabilities of this class is keeping the match list an an integer array of matchIDs and inspecting them to give tactics.

- **Constraints**

User can not give tactic to team for number of players below 11.

- **Composition**

The class has 2 functions:

1. giveMatchTactic() : user creates tactics for the future games with this function.
2. viewMatch() : the user can view the result and statics of the past games with this function.

- **Interface/Exports**

giveMatchTactic() function of the class loads the tactics interface.

viewMatch() function of the class loads the viewMatch interface.

### 7.1.9 Player Class

- **Classification**

Player is a class under the Main Game Module.

- **Definition**

The class keeps the information of the player like name, surname, etc.

- **Responsibilities**

The responsibilities of the class are to keep information and attributes of player. Also, sacking the player operation is executed within this class. Viewing past matches performances of the player is handled.

- **Constraints**

The players played in national matches can not be sacked.

When the number of players in the team is below 12, user can not sack a player.

- **Composition**

The class has 2 functions:

1. sack() : sakes the player from his team.

2. lastMatches() : calculating overall performance of the player and shows the statistics of the player.

The class has 2 subclasses as SeniorPlayer class and YouthPlayer class. SeniorPlayer has extra attributes and they can be listed for transfer. YouthPlayer can be promoted to senior team.

### 7.1.10 SeniorPlayer Class

- **Classification**

SeniorPlayer is a subclass of Player class under the Main Game Module.

- **Definition**

The class of players which play in the main team.

- **Responsibilities**

The responsibility of the class is keeping information and attributes of senior player. Also, adding the player to the transfer list operation is executed within this class.

- **Constraints**

When the number of players is below 12, the player can not be listed in transfer list.

- **Composition**

The class has one function :

addTransferList(): The function adds the player to the transfer list.

### 7.1.11 YouthPlayer Class

- **Classification**

YouthPlayer is a subclass of Player class under the Main Game Module.

- **Definition**

The class of players which are playing in the youth team.

- **Responsibilities**

The responsibility of the class is keeping information and attributes of youth player.

Also, the class promotes the player to the senior team.

- **Constraints**

The player must be over 18 to be promoted to the senior team. Also, he must be below 21 to stay in the youth team.

To promote the player to the senior team, the number of players in the youth team must be over 11 and the number of the players in the senior team must be below 40.

- **Composition**

The class has one function :

promoteSenior() : promotes the player to the senior team.

## 7.2 Sound Engine Module

There are three classes in sound engine module named MatchSounds, MenuSounds and GameMusics.

### 7.2.1 MatchSounds

- **Classification**

It includes three functions and two string components. The functions are play, configureAudioEffect and mute, and the strings are fileFormat and name.

- **Definition**

MatchSounds plays sounds during matches.

- **Responsibilities**

The responsibilities of MatchSounds are taking the sound requests from Match Engine and playing the relevant sound file.

- **Constraints**

Match Engine sends request for sound to MatchSounds and takes result by playing the sound file for matches. This file will be kept in database and only called when match starts. So there will be no timing and storage constraint because programmers will select the sound files and these files will not be changed by users.

- **Composition**

Name: Holds the name of sound file without file format.

FileFormat: Holds the format of the sound file.

Play: The function that plays the sound file.

ConfigureAudioEffect: The function that selects the audio effect.

Mute: The function that mutes the sound.

- **Uses/Interactions**

Play function takes name and fileFormat concatenates them and gives the name to facebook to play that sound file.

- **Resources**

PHP will prompt the web browser to activate an audio application and this application will play our sound file, so all processor and memory activities will be handled by the web browser.

- **Processing**

When a match is played in game, MatchEngine requests to play the sound file which is meant to be played in matches. MatchSounds takes the request and calls play function which will make the related part of the code work. This code says facebook to play the sound file and the file is played with the help of an audio application from web browser. If the user mutes match sound later this code segment will be disabled.

- **Interface/Exports**

This component does not take any inputs. It only takes request from MatchEngine and return with playing or muting the sound in match. The sound file will be selected by programmers initially.

## 7.2.2 MenuSounds

- **Classification**

It includes three functions and two string components. The functions are play, configureAudioEffect and mute, and the strings are fileFormat and name.

- **Definition**

MenuSounds plays sounds when a menu button is pressed by user.

- **Responsibilities**

The responsibilities of MenuSounds are taking the sound requests of Main Game and playing the relevant sound file.

- **Constraints**

Main Game sends request for sound to MenuSounds and takes result by playing the sound file. This file will be kept in database and only called when a menu button is pressed. So there will be no storage constraint because programmers will select the sound files and these files will not be changed by users. If users press menu buttons continuously it will cause a lot of requests and conflict of sounds.

- **Composition**

Name: Holds the name of sound file without file format.

FileFormat: Holds the format of the sound file.

Play: The function that plays the sound file.

ConfigureAudioEffect: The function that selects the audio effect.

Mute: The function that mutes the sound.

- **Uses/Interactions**

Play function takes name and fileFormat concatenates them and gives the name to facebook to play that sound file.



- **Resources**

PHP will prompt the web browser to activate an audio application and this application will play our sound file, so all processor and memory activities will be handled by the web browser.

- **Processing**

Main Game requests to play the sound file which is meant to be played when user presses a menu button. MenuSounds takes the request and calls play function which will make the related part of the code work. This code says facebook to play the sound file and the file is played with the help of an audio application from web browser. If user mutes later this code segment will be disabled.

- **Interface/Exports**

This component does not take any inputs. It only takes request from Main Game when a menu button is pressed by user and return with playing or muting the sound. The sound file which will be played will be selected by programmers initially.

### 7.2.3 GameMusics

- **Classification**

It includes four functions and two string components. The functions are play, configureAudioEffect, pause and mute, and the strings are fileFormat and name.

- **Definition**

GameMusics plays the arranged music during game.

- **Responsibilities**

The responsibilities of GameMusics is playing or muting the music with requests from Main Game.

- **Constraints**

Main Game sends request for music to GameMusics and takes result by playing the music file. This file will be kept in database and only called when the game starts. So there will be no timing and storage constraint because programmers will select the sound files and these files will not be changed by users.

- **Composition**

Name: Holds the name of sound file without file format.

FileFormat: Holds the format of the sound file.

Play: The function that plays the sound file.

Pause: The function that pauses playing of sound file.

ConfigureAudioEffect: The function that selects the audio effect.

Mute: The function that mutes the sound.

- **Uses/Interactions**

Play function takes name and fileFormat concatenates them and gives the name to facebook to play that sound file.

- **Resources**

PHP will prompt the web browser to activate an audio application and this application will play our sound file, so all processor and memory activities will be handled by the web browser.

- **Processing**

When the game begins, Main Game requests to play the music file which is meant to be played during game. GameMusics takes the request and calls play function which will make the related part of the code work. This code says facebook to play the sound file and the file is played with the help of an audio application from web browser if pause or mute functions weren't called yet.

- **Interface/Exports**

This component does not take any inputs. It only takes request from Main Game and return with playing the related sound file during all game. The sound files will be selected and arranged by programmers initially.

## 7.3 Match Engine Module

There are two classes in match engine module named Match and MatchSimulate

### 7.3.1 Match

- **Classification**

It includes three functions and three integer components. The functions are calculateMatch , playMatch, and update and the integers are matchId, homeTeam and awayTeam.

- **Definition**

Match class is the base of the game engine.

- **Responsibilities**

The responsibilities of Match class is take the each team values from database then according to playing match algorithm give the return values to match simulate class.

- **Constraints**

Match Class sends request to database and take the properties of each team and player and in order to simulate match according to time efficiency is the most important constraint whole of the project.

- **Composition**

matchId: Holds the id of the playing match.

homeTeam: Holds the id of the home team.

awayTeam: Holds the id of the away team.

calculateMatch: The function that takes the values from database for playMatch function.

playMatch: The function that plays the match according to time and used algorithm.

updateMatch: The function that updates the team and player properties

- **Uses/Interactions**

PlayMatch function concatenates the database values and simulation values.

- **Processing**

When match is started calculateMatch sends request to database and take the properties of each team and player. After this the playMatch is start to match. In the end of the match update function updates the values of each teams and players

- **Interface/Exports**

This component only takes request from GUI and play the match according the values graphic engine simulate the match.

### 7.3.2 MatchSimulate

- **Classification**

It includes two functions and three integer components. The functions are simulate and makeComment and the integers arematchId, homeTeam and awayTeam.

- **Definition**

MatchSimulate is the base of the graphic engine and send values to simulation or commentary mode.

- **Responsibilities**

The responsibilities of MatchSimulate is to dynamically send the data to graphic engine.

- **Constraints**

There is no constraint for match simulation.

- **Composition**

matchId: Holds the id of the playing match.

homeTeam: Holds the id of the home team.

awayTeam: Holds the id of the away team.

simulate: The function that when the match is playing dynamically simulate the match details..

makeComment: The function that when the match is playing dynamically comment the match details..

- **Uses/Interactions**

This class interact the match engine and and graphic engine.

- **Processing**

When match is started playMatch sends data to matchSimulate class and this class interact with graphic engine. Give the coordinates of each player and the condition of each team.

- **Interface/Exports**

This component only takes request from match class and simulate the match according the values to graphic engine.

## **7.4 Graphic Engine Module**

### **7.4.1. FlashHandler Class**

- **Classification**

FlashHandler is a class under the Graphic Engine Module.

- **Definition**

The class is the part of the project that the operations of flash are executed.

- **Responsibilities**

The class is used to communicate with flash and create flash objects. The tactic and matches interface and the whole buttons used in the game window will be created by this class.

- **Composition**

The class has one function:

drawFlashObject() : draws all of the flash objects used in the game. The object created will be decided according to the current interface.

- **Uses/Interactions**

drawFlashObject do not take any arguments. Function decides drawing objects according to current interface.

- **Processing**

PHP will call drawFlashObject function for interface objects or playing matches then send the objects within an interface to web browser.

- **Interface/Exports**

This component does not take any inputs. It only takes request from PHP and return flash objects with in interface.



## 7.5 Data Handler Module

### 7.5.1 Executer

- **Classification**

It includes four functions and database connection, and query components.

- **Definition**

Executer class is the base of the game data handler module.

- **Responsibilities**

The responsibilities of executer class takes the each team and player data from database.

- **Constraints**

The constraint of data handler module is the time because the size of queries in the database.

- **Composition**

db: Holds the id of the playing match.

query: Holds the id of the home team.

ORM library is used for queries execution.

- **Processing**

When user login to application from facebook users facebook id will be user id.

According to this id team and player data are sent to GUI and when the match engine starts the data values also execute for the the match. Also when match is ended data queries are updated.

- **Interface/Exports**

This component takes request from GUI and Match Engine.

## 7.6 Forum Module

### 7.6.1 Forum

- **Classification**

It includes four functions and one integer components. The functions are readTopic, postReply,newTopic and follow, and the integer is currentTopic.

- **Definition**

Forum lets user talk and discuss the issues which are related with game play, transfers, hints (etc).

- **Responsibilities**

The responsibilities of Forum are taking requests from Main Game, shaping forum accordingly, giving results to database.

- **Constraints**

Main Game sends request for using forum and results changes the database. If the number of user that reaches forum is become larger, than it will slow the data transfer a little bit, because each user will try to reach and change same info at the same time.

- **Composition**

CurrentTopic: Holds the number of current topic.

ReadTopic: The function that shows the current topic.

PostReply: The function that creates reply to user posts in the current topic.

NewTopic: The function for opening a new topic.

Follow: The function that makes users follow the current topic.

- **Uses/Interactions**

Forum handles the forum information on database and makes users to access and change this information. It can only be reached by Main Game.

- **Resources**

Forum takes requests from Main Game by users then takes information from database. When user changes this information the information on the database will be changed. Forum will only work with information on database so it will not have so much memory and processor work. However, user will reach and, changes same data, so changing information will be handled by letting only one user to change information at same time.

- **Processing**

When a user requests to reach forum, Forum will bring the forum information from database. CurrentTopic will change according to user and readTopic function will open the current topic and shows the related information on screen. PostReply will take the information from user and append this information to the end of current topic. NewTopic will take information from users and open a new topic. The information on database will be changed according to these new changes. Follow function will make user see the other information in the current topic which cannot be seen in screen like scrolling mouse wheel in web pages.

- **Interface/Exports**

This component does not take any inputs. It only takes request from Main Game and return with showing or changing related information which are kept in database.

## 7.7 GUI Module

### 7.7.1 Interface

- **Classification**

It includes five functions and one FlashHandler components. The functions are drawMainPage, drawTeamPage, drawPlayersPage, drawMatchesPage, and drawTransferPage, and the FlashHandler is flashObj.

- **Definition**

GUI generates all pages as main page, team page, player pages, matches page and transfer page related with game via Flash.

- **Responsibilities**

The responsibilities of GUI are creating HTML pages that hold flash objects and preparing graphical user interfaces.

- **Constraints**

GUI sends request for graphics to Graphics Engine and takes result by getting the page information. This information kept in database and there will be no constraint because programmers will set the page information and this information cannot be change by users.

- **Composition**

flashObj: an instance of the class FlashHandler

drawMainPage: The function that draws the main page

drawTeamPage: The function that draws the team page

drawPlayersPage: The function that draws the players page

drawMatchesPage: The function that draws the matches page

drawTransferPage: The function that draws the transfer page

- **Uses/Interactions**

All functions create a graphical user interface for users.

- **Resources**

GUI takes request from Main Game by users then takes information from database.  
All memory activities will be handled by web browser.

- **Processing**

Main game requests to draw pages which will be user interface during the game when the game begins. flashObj will be created as an instance of the class FlashHandler. drawMainPage will prepare main page for the game. drawTeamPage will prepare team page, drawPlayers page will prepare players page, drawMatchesPage will prepare matches page and drawTransferPage will prepare transfer page for the game.

- **Interface/Exports**

This component does not take any inputs. It only takes request from Main Game and return with showing the related page in the game.

## 8 Libraries and Tools

### 8.1 PHP

**Hypertext Pre-processor** is a widely used, general-purpose scripting language that was originally designed for web development to produce dynamic web pages. For this purpose, PHP code is embedded into the HTML source document and interpreted by a web server with a PHP processor module, which generates the web page document. As a general-

purpose programming language, PHP code is processed by an interpreter application in command-line mode performing desired operating system operations and producing program output on its standard output channel. It may also function as a graphical application. PHP is available as a processor for most modern web servers and as a standalone interpreter on most operating systems and computing platforms.

PHP language is one of the most powerful languages for Facebook applications. So the Best-11 Football Manager is decided to write with PHP.

## 8.2 MVC

**Model–View–Controller (MVC)** is a software's architecture, currently considered an architectural pattern used in software engineering. The pattern isolates "domain logic" (the application logic for the user) from the user interface (input and presentation), permitting independent development, testing and maintenance of each (separation of concerns).

The **model** manages the behaviour and data of the application domain, responds to requests for information about its state (usually from the view), and responds to instructions to change state (usually from the controller). In event-driven systems, the model notifies observers (usually views) when the information changes so that they can react.

The **view** renders the model into a form suitable for interaction, typically a user interface element. Multiple views can exist for a single model for different purposes. A view port typically has a one to one correspondence with a display surface and knows how to render to it.

The **controller** receives input and initiates a response by making calls on model objects. A controller accepts input from the user and instructs the model and view port to perform actions based on that input.

An **MVC** application may be a collection of model/view/controller triads, each responsible for a different UI element. The Swing GUI system, for example, models almost all interface components as individual MVC systems.

MVC is one of the milestones of the Best-11 Football Manager game. It facilitates the Project development.

## 8.3 Kohana

Kohana is an elegant MVC PHP5 framework that provides a rich set of components for building web applications. It requires very little configuration, fully supports UTF-8 and I18N, and provides many of the tools that a developer needs within a highly flexible system. The integrated class auto-loading, cascading file system, highly consistent API, and easy integration with vendor libraries make it viable for any project, large or small.

The expedition, ORM and the helper library of the Kohana is the reason why this framework is selected to the Project.

## 8.4 WampServer

WampServer is a Windows web development environment. It allows you to create web applications with Apache, PHP and the MySQL database.

Best-11 Football manager will be developed in localhost before the put it into server. Because of this reason WampServer is useful for the Project



## 8.5 Sqlyog

SQLyog MYSQL GUI is the most powerful MYSQL manager and admin tool, combining the features of MYSQL Administrator, phpMyAdmin and other MYSQL Front Ends and MYSQL GUI tools.

WampServer also includes MySQL database which is phpMyAdmin but the usage of SQLyog is more useful than phpMyAdmin for this reason SQLyog will be used in the Project.

## 8.6 NuSphere PhpED

**NuSphere PhpED** is today's top integrated development environment for php. Suitable both for small individual works and large multi-developer projects, PhpED considerably boost up the development process.

PhpED is a robust tool featuring full-cycle functionality for developing web-sites and web-applications. Balanced combination of advanced code editor, reliable dbg debugger, productive database connectivity client and fast and secure deployment abilities make PhpED a complete solution for most sophisticated developer needs.

## 8.7 ORM Library

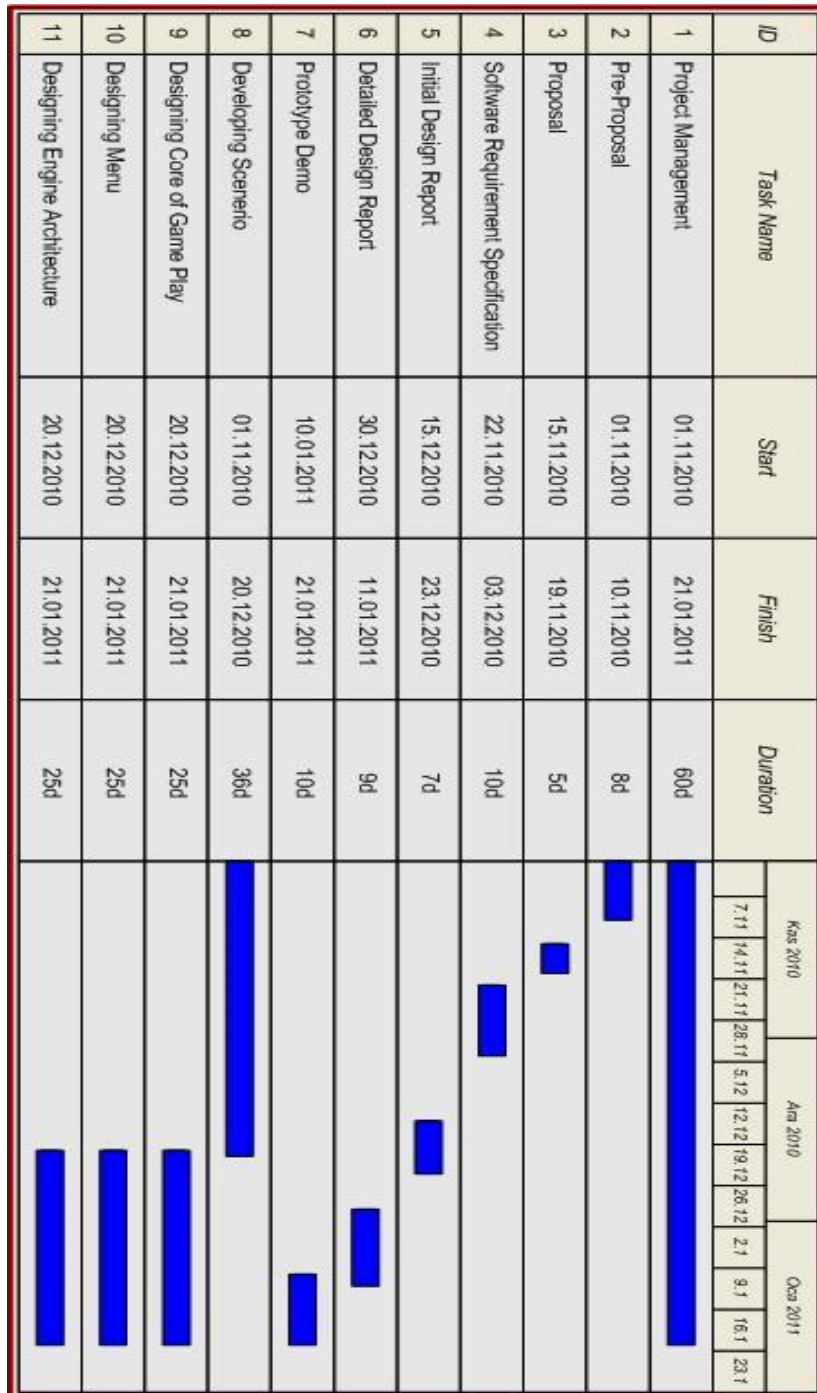
**Object-relational mapping (ORM)** in computer software is a programming technique for converting data between incompatible type systems in object-oriented programming languages. This creates, in effect, a "virtual object database" that can be used from within the programming language. There are both free and commercial packages available that perform object-relational mapping, although some programmers opt to create their own ORM tools.

Also Kohana has its own ORM library. Some of the ORM advantages in this project are:

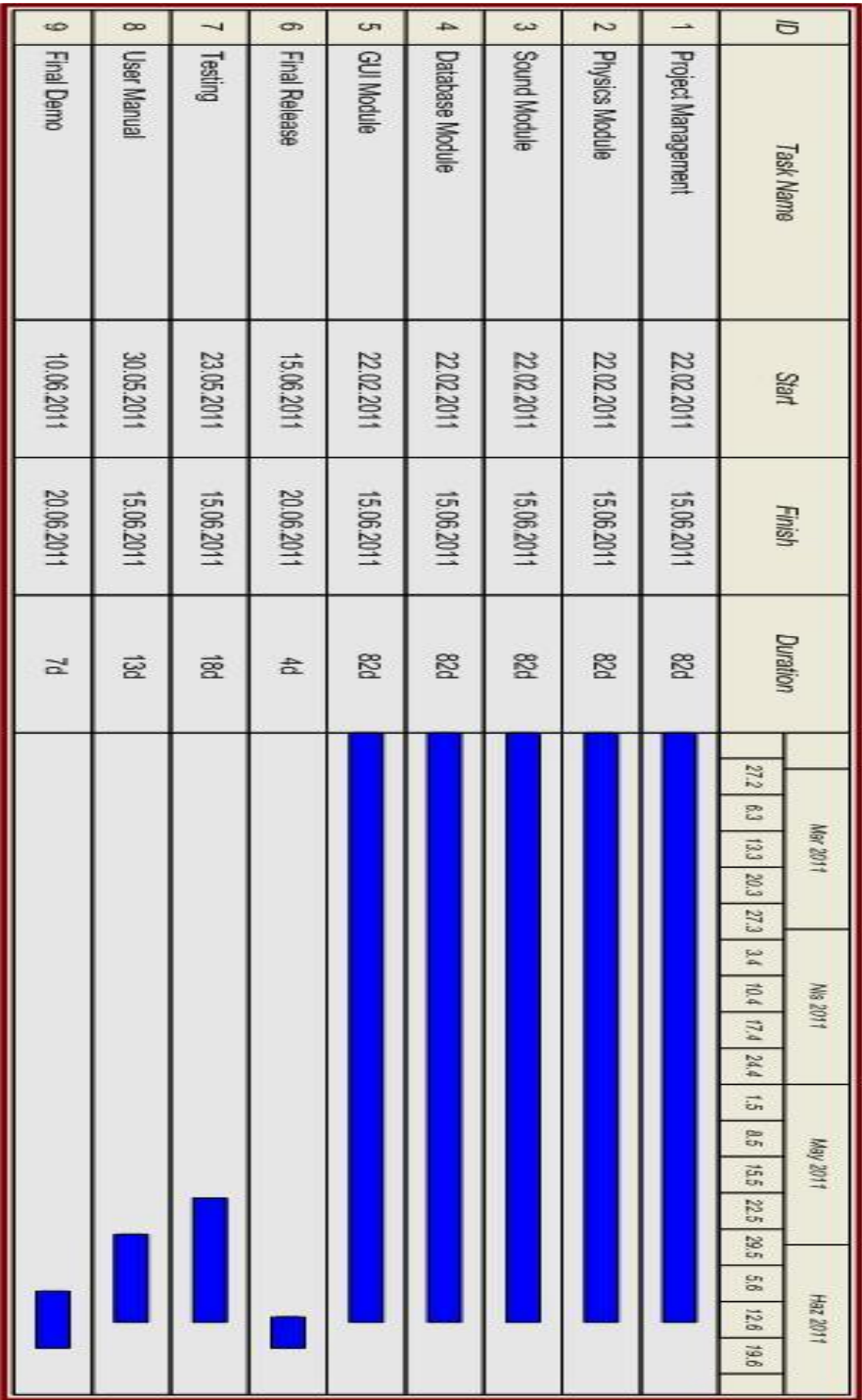
1. Facilitates implementing the Domain Model pattern. This one reason supercedes all others. In short using this pattern means that you model entities based on real business concepts rather than based on your database structure. ORM tools provide this functionality through mapping between the logical business model and the physical storage model.
2. Huge reduction in code. ORM tools provide a host of services thereby allowing developers to focus on the business logic of the application rather than repetitive CRUD (Create Read Update Delete) logic.
3. Changes to the object model are made in one place. Once you update your object definitions, the ORM will automatically use the updated structure for retrievals and updates. There are no SQL Update, Delete and Insert statements strewn throughout different layers of the application that need modification.
4. Rich query capability. ORM tools provide an object oriented query language. This allows application developers to focus on the object model and not to have to be concerned with the database structure or SQL semantics. The ORM tool itself will translate the query language into the appropriate syntax for the database.
5. Concurrency support. Support for multiple users updating the same data simultaneously.

## 9 Time Planning (Gantt Chart)

### 9.1 Term 1 Gantt Chart



9.2 Term 2 Gannt Chart



## 10 Conclusion

This Detailed Design Report was prepared to create a bridge between the design and implementation parts of the project. This document specifies the detailed software design specifications, which are organized according to the specifications in IEEE Recommended Practice 1016-1998, for Best-11 Football Manager game.

This report includes data flow models, class diagrams, interface features, entity relationship diagrams and use cases. The possible design and other constraints that can be faced are explained. The tools and the libraries that will be used to develop the project are given. A brief detailed design explained. We have explained the works that we have done so far and we give the time planning that will be followed by group members.