# 3. Design

## 3.1 Introduction

Design means transforming users need into suitable form, which helps the programmer in software coding and implementation. Users need is used for creating a design, as it must full all the requirements. It is the third phase of waterfall model after analysis.

Advantages of designing:

* Easy to use for the users.
* Fulfill the requirement.
* Good design leads to good relationship between customer and the software developer.

## 3.2 Structural Design

Structural design is the overview of the program, how the program will work and what are the requirements for it. It also helps in the architectural design,

There are many examples of structural design like class diagram, flowchart and data flow diagram. For this project, I have created class diagram and flowchart.

### Class diagram

Class diagram is the detailed view of the classes how it interacts with others classes and what methods are used in that class. It helps to understand the relation as well as help in coding.

**Justification:**

The class diagram I have, created shows the relation between the classes and it also shows the methods and datatypes used in each class, which can help me in coding with less errors.

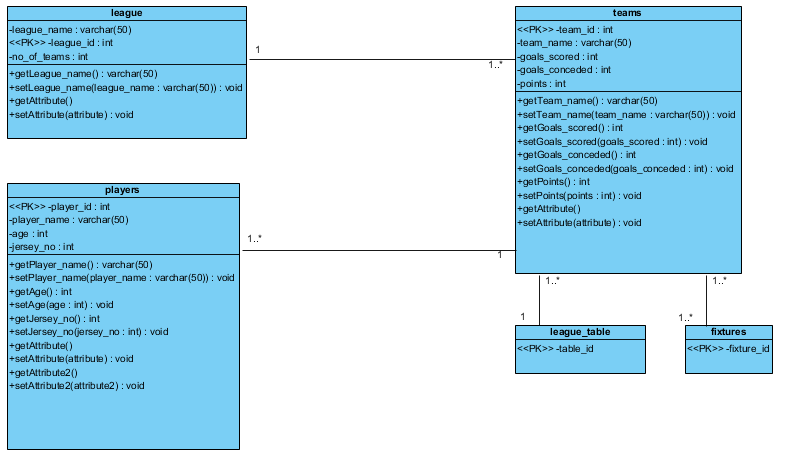
**Notation used:**

*Aggregation notation:* Used to configure objects for making complex type object.

*Composition notation:* Used for composite objects.

*Association notation:* Used to inter-relate class.

**Diagram:**

****

**Explanation:**

This is the relation between the entities used in my program.

### Data flow diagram (DFD)

Data flow diagram shows how the data flows in the system. It can also help us to determine the outputs as well as point where errors can take place in the program.

For this project, I have created DFD to know the detailed information how the data is going to flow.

**Justification:**

A data flow diagram can help a lot in the software development as it can overview how the data is flowing in the program which can also show the interaction between the classes.

**Notation used:**

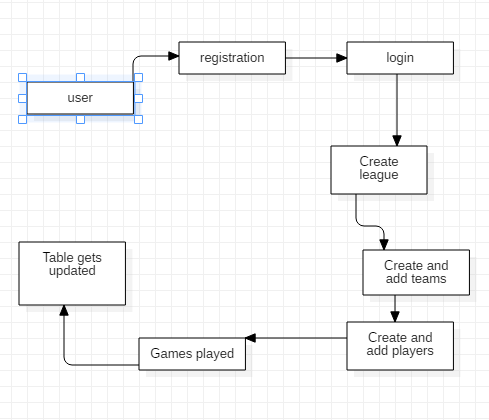
*Process Notation:* helps in data flow

*Data stores Notation:* repositories of the system.

*Data flow notation:* helps in determining flow of the data.

*External entity notations:* information about objects outside the system

**Diagram:**

****

**Explanation:**

This is how the data flows in the process and how it is executed properly.

## 3.3 Behavioral Diagram

Behavioral diagrams helps in visualizing, specifying, constructing, and documenting the dynamic aspect of the system. Behavioral diagram can be categorized as sequence diagrams, activity diagram, interaction diagram and state-chart diagram.

For this project, I have created use case and sequence diagram.

### Sequence diagram

Sequence diagram describes the interaction among classes in terms of an exchange of messages overtime. They are also called event diagrams.

I have created sequence diagram to have overview how the objects are interacting with each other and how they deal with each other.

**Justification:**

I have created sequence diagram to show the interaction between the objects and show how the data flows. It can clarify the users about how the program is going to work out.

**Notation used:**

*Lifeline:* This notation helps to represent the all instance in every interaction.

*Message:* This notation help to send the text.

*Message reply:* This notation help to reply the text to the object.

*Self-message:* This notation helps to write the text to own self-object.

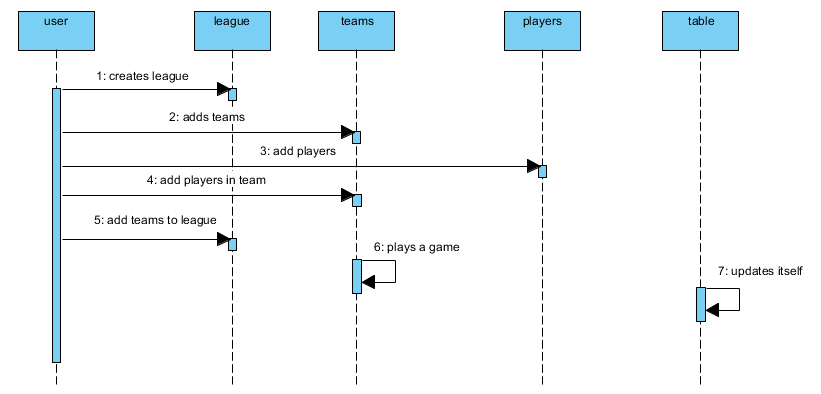
*Actor:* This notation is for the object, which is important for the system to run.

*Combine fragment:* This type of notation help to write down the if/else statement.

*Object:* This is the class like structure without which system flow is not possible.

This notation helps to represent the all instance in every interaction.

**Diagram:**

****

**Explanation:**

This is how the whole process is carried out and how the league is organized properly.

### Activity diagram

Activity diagram is the process of showing how a task is carried out by process in their specified classes. They show how the task are operated.

For this project I have created activity diagram and shown a full process of a task done.

**Justification:**

With the help of activity diagram, we can have more detailed information about how a task is carried out.

**Notation used:**

*Start Point:* This is the point which represents the initial action state.

*Action State:* This point represents the non interruptible actions of the step.

*Action flow:* This notation helps to flow the object from one action state to another.

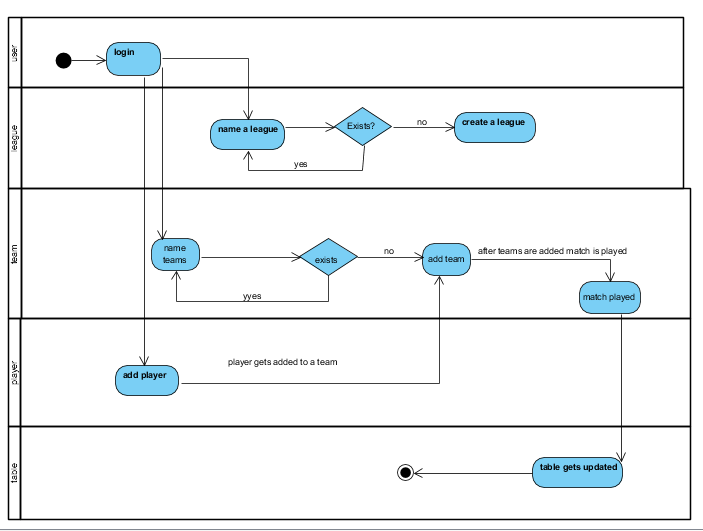
*Decision branching:* This notation helps to decide like if/else condition.

***Synchronization:***

*Fork node:* This notations help to split the one action state into two or many.

*Join node:* This notations helps to combine the two or many action state to one action state.

**Diagram:**

****

**Explanation:**

This is how the full process is carried out in the system.

## 3.4 Database management system

It is the system responsible for the creation of database as well as manage the software. It also helps us to perform CRUD operations. There are advantages of using DBMS :

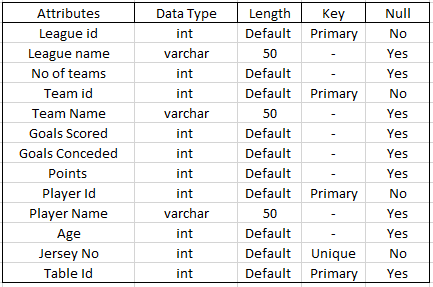
1. It helps to control redundancy.
2. Avoid inconsistency.
3. Integrity can be enforced.
4. More secure

### Data Dictionary:

Data dictionary is the file that contains the data of the data or metadata. It helps to know the type, length and many more information about the data.

**Justification:**

I have created data dictionary to avoid further problems that may occour during the coding.



### ER (Entity relationship):

Entity relationship gives the knowledge about the relation between entities.

**Justification:**

I have five entities for this project they are:

1. League
2. Teams
3. Players
4. Table
5. Fixtures

**Notation Used:**

*Fields:* This notation is used to classify the entity and their columns.

***Keys***

*Primary key:* This key uniquely identifies the column. It is unique as well.

*Foreign key:* This key are created any time an attribute relates to another entity that has same character from the related entity.

***Cardinality and Cordiality***

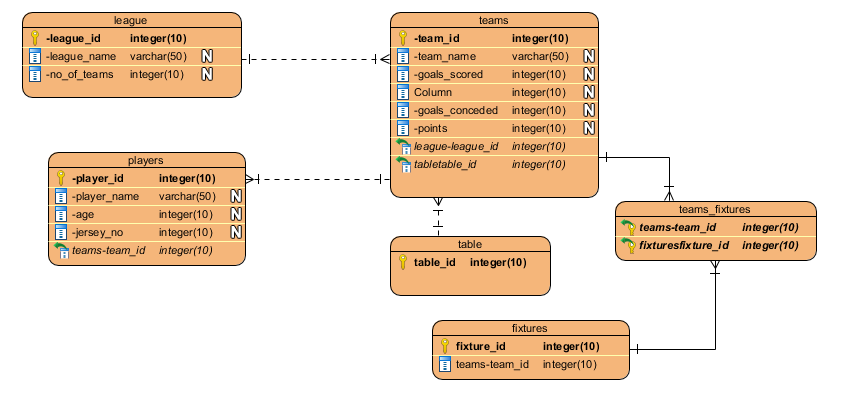
Cardinality and Cordiality are the notations that are used to relate the one instance of one entity to another instance of another entity.

There are 3 types of notation

*One to many:* This notation helps to relate the one instance of one entity with the many instance of another entity.

*Many to many:* This notation helps to relate the maximum number of times an instance in one entity with other many instance of another entity.

*Many to one:* This notation helps to relate the maximum number of times an instance in one entity with one instance of other entity.



## 3.5 Architecture

It is the fundamental structure that must be made to give shape to the system. It helps to design the system, edit if necessary and proceed to develop the system.

**Justification**: