# Analysis

Analysis is the first stage of software development either desktop or web where detailed examination is done. Analysis is done before the program is created, with the given scenario or the scenario of behavior of the program. It helps to determine requirement, the specified requirement by the user and finally checking the feasibility if the data that has been analyzed.

There are few points why analysis is needed:

1. Helps to know about future problem.
2. Potential upgrades in the program.
3. Fulfill all the requirements.
4. Requirements required (hardware, software).

There are many types of analysis process but for this project, I have chosen SWOT analysis, which means Strength, Weakness, Opportunity and Threat.



* Strength: It shows the strength of the company thing that company can do the best. Those qualities that our company can do better than the competitors as well as it includes the resources within like skill of the staff, capital, intellectual property etc.
* Weakness: It includes the limitations in our company that needs to be focused on such as limited resource.
* Opportunities: It means all those parts in which the company can shine and be at it best such as having less competitors in a certain area or in products.
* Threat: Threats refer to that problem that can harm the company. There are two types of threats :

External threat: These are those threats that can be affected by the outer means such as culture, trends, competitors etc.

Internal threat: These are those threats that can occur within the company such as low budget, less skillful, limited resources, etc.

# Feasibility study

After the analysis is done, it is not fixed if the software that has been developed will be a success or not and further can it satisfy the user or not. There are certain feasibility that can help to determine the feasibility of the program. They are:

1. Economic Feasibility: Considering the budget the application should be built. The expenses should be considered as the budget is limited and later either the sale of program can benefit the company or not.
2. Technical Feasibility: Only with proper technical supplies, we can completely build the program demanded by the user as per wish. In lack of any technology, the program may not be a perfect one.
3. Schedule Feasibility: We should work as per schedule made which should be done properly. In case of any issues, the project may not be completed in time.
4. Operational Feasibility: It refers to the skill of solving problems.
5. Legal Feasibility: It determines if the project developed is not against the law of the country or against ethical rules.

# Requirement analysis

## Functional

They refer to those requirements that have their own specific role in the project. It can change as per the projects. They are one of the most important function of the program and determine the application. It fulfills the requirement of the user or client.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **ID** | **Function** | **Data Required** | **Dependency** | **Remarks** |
| F01 | Login | Username and Password | N/A | It helps to secure the system |
| F02 | Create a league | N/A | F01 | To start a league. |
| F03 | CRUD for teams | N/A | F02 | To organize the teams properly. |
| F04 | CRUD for players | Data of teams | F03 | To assign players as per their teams |
| F05 | Automated fixtures | Data of teams | F02 | Fixtures will be automatically scheduled. |
| F06 | Automated Table update | Fixture results | F05 | Tables updated after games |
| F07 | Data in games | Automated Fixtures | F05 | Data about games to be inserted |

## Non-functional

These features help to support the functional requirement or enhance the functional requirement. They are not important to be executed.

The non-functional requirement are:

|  |  |  |  |
| --- | --- | --- | --- |
| **ID** | **Function** | **Purpose** | **Remarks** |
| F01 | Security | Secure the system | Increases security. |
| F02 | Efficient | Increase productivity | Helps to obtain best results. |
| F03 | Accessible | Easy to use | Friendly to use |
| F04 | Extensible | Further upgrades | Enhance features |
| F05 | Testable | Find solutions easily | Less error free and no bugs |

# MOSCOW Prioritization

It is one of the most popular technique used which helps to determine the important factor or the needs and separate which is not needed. It helps us to classify which is important and which is not so that we can allocate our time and effort.

The classification are:

1. Must have: these are the must have in the project that cannot be replaced or removed. They are the core of the program
2. Should have: these are those features that needs to be included but is not a must as without it the program can still run properly and needs to be included if possible.
3. Could have: They can be replaced or even removed as their presence doesn’t make a significant change in the program.
4. Won’t have: They are the requirements that should be removed from the project and don’t have any impact on the efficiency of the program.

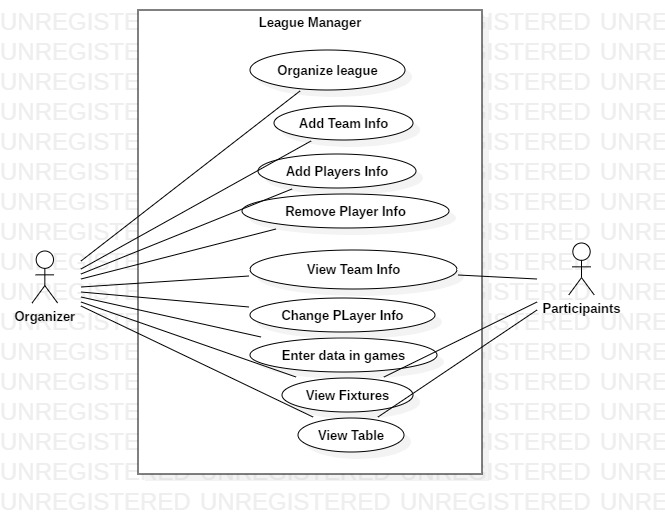
|  |  |  |
| --- | --- | --- |
| **ID** | **Title** | **Priority** |
| FR01 | Login | Must Have |
| FR02 | Create a league | Must Have |
| FR03 | CRUD for teams | Must Have |
| FR04 | CRUD for players | Must Have |
| FR05 | Automated fixtures | Should Have |
| FR06 | Automated Table update | Should Have |
| FR07 | Data in games | Must Have |

FUNCTIONAL MOSCOW TABLE

|  |  |  |
| --- | --- | --- |
| **ID** | **Title** | **Priority** |
| FR01 | Security | Must Have |
| FR02 | Efficient | Must Have |
| FR03 | Accessible | Must Have |
| FR04 | Extensible | Should Have |
| FR05 | Testable | Must Have |

NON-FUNCTIONAL MOSCOW TABLE

# Use Case



Use case diagram are those diagram that shows the basic context of the program which is developed at the initial phase of software development. It shows the interaction of user in the program. The diagram consists of actor, use case and communication link.

|  |  |  |
| --- | --- | --- |
| **ID** | **Title** | **Description** |
| 1 | Organize League | Only organizers can organize a league |
| 2 | Add team info | Teams are needed for a league |
| 3 | Add Player Info | Add details about players of the team |
| 4 | Remove player info | Remove player info that is mistaken |
| 5 | View team info | Everyone should be able to see the information of the team |
| 6 | Change Player info | Change information about players |
| 7 | Enter data in game | Events that happened in game |
| 8 | View fixtures | Everyone should see the information of the team. |
| 9 | View table | Everyone should see the table |

# NLA

NLA stands for Natural Language Analysis it is we classify the noun, verb and adjective from the given scenario. From the filtered process nouns are classes, verbs are method and adjective are attributes.

## Assuming Our Scenario

There will be a league organizer with a specified name and the teams can be added. There will be players in the teams that represent the team individually. The fixtures among the teams will be automatically created and the table will be updated as well. The events that happens in the team will be recorded by the admin only not by others. Participants can look at their fixtures, table as well as their info about the team. Whereas the organizer can have a lot of access such as CRUD for team and players.

## Unfiltered Selection

Noun: league, admin, team, player, match table, event, fixtures, participants, record.

Verb: organize, specified, add, record, create, CRUD,result

## Filtered Selection

|  |  |
| --- | --- |
| **Class/Noun** | League, Team, Player, Fixtures, Table |
| **Method/Verb** | CRUD on teams/players, result, organize |

