**T.Y. B. Tech.**

**CS 303: Software Engineering Laboratory**

Assignment No: 3

**Player Value Analyser**

**System Requirement Specification**

**12-09-2017**

**Version 1.0**

|  |  |  |  |
| --- | --- | --- | --- |
| Project Group Information | | | |
| Roll. No. | **Gr. No.** | **Name** | **Roles** |
| 18 | **151488** | **Anup Mahindre** | **Analyst** |
| 31 | **151600** | **Hiranyey Gajbhiye** | **Developer** |
| 33 | **151482** | **Neeraj Ganu** | **Leader** |
| 34 | **151724** | **Sanket Ostwal** | **Designer** |

**Approved By: Dr. M. R. Dube**

**Academic Year: 2017-18 Semester: I**

**Table of Contents**

|  |  |  |
| --- | --- | --- |
| **Sr. No.** | **Title** | **Page** |
| 1 | Introduction |  |
| 2 | Terms of Reference |  |
| 3 | Problem Description |  |
| 4 | Functional Hierarchy |  |
| 5 | User Interfaces |  |
| 6 | Hardware Interfaces |  |
| 7 | Software Interfaces |  |
| 8 | Logical Databases |  |
| 9 | Non-Functional Requirements |  |

# INTRODUCTION

The introduction of the Software Requirements Specification (SRS) provides an overview of the entire SRS with purpose, scope, definitions, acronyms, abbreviations, references and overview of the SRS. The aim of this document is to gather and analyze and give an in-depth insight of the complete **Player Value Analyzer** software system by defining the problem statement in detail. Nevertheless, it also concentrates on the capabilities required by stakeholders and their needs while defining high-level product features. The detailed requirements of the Marvel Electronics and Home Entertainment are provided in this document.

|  |  |
| --- | --- |
| Item | Description |
| Purpose | **To jot down the System Requirement Specifications of the Player Value Analyser. The purpose of the SRS to give a complete technical background of our system and its likely implementation.** |
| Audiences | **Developers, Clients and other concerned audiences.** |
| SRS Scope | **This SRS is also aimed at specifying requirements of software to be developed but it can also be applied to assist in the selection of in-house and commercial software products. The standard can be used to create software requirements specifications directly or can be used as a model for defining a organization or project specific standard. It does not identify any specific method, nomenclature or tool for preparing an SRS.** |
| Project Scope | **Primarily, the scope pertains to the football player market for making the Player Value Analyzer project live. It focuses on the company, the stakeholders and applications, football club financers and mass media companies.** |

**References:**

1. Statement of Work
2. Feature Set
3. Standard Football Market Model

# TERMS OF REFERENCE

1. **Background**
   1. The football market is a multibillion-dollar industry and has a lot of business activity. The Football Clubs in the world are constantly trading players for money.
   2. There is also a lot of media interest in these events and this makes it all a very important part of the world’s headlines.
   3. In such cases a sense of authenticity is required as the player values can be very big and no assurances over the performances can be made.
   4. The Clubs ae ready to pay a significant amount of money so as to know how much the real value of a player will be.
   5. The Player Value analyser will consider a player’s rating and his statistics to find out a correct value of the player.
   6. The project will help all interested parties and might even help clubs save money from a risky or unworthy investment.
2. **Objectives**
   1. The Objective of the project is to find the correct approximation of the value of a football player.
   2. Stage-wise objectives.

* Collecting Player Data
* Analysing Player Data
* Finding a pattern and plotting values
* Formulating significant factors contributing to Player’s value
* Displaying Player Value and Stats
  1. Target Audience is
* Football Clubs
* Football Media Companies
* Fans and general interested users
  1. Resources required would be clearer along the course of the implementation but the basic necessities include player data, machine learning techniques and UI design.

1. **Issues**
   1. Efficiency – Efficiency of the algorithm and its outcome to predict values.
   2. Relevance – Proving relevance to stakeholders.
   3. Effectiveness – Exact outputs of the project and realization of benefits.
   4. Impact – the market is ever growing and the product’s use will never cease to exist.
   5. Sustainability – The project if successful will have wide ranging benefits and will become self-sustainable soon after its initiation.
2. **Methodology**
   1. Data Acquisition, Filtering, Data Mining, Data Analysis, Data Sorting & Classification
   2. Stakeholder involvement at initial and final stages will ensure smooth implementation
   3. The planning and designing phase and implementation phase will each take about 1 month.
   4. The information collection tools to be used throughout the project for monitoring purposes are Media Reports and similar online resources
   5. Data analysis rules
3. **Expertise**

The expertise needed for doing a project defines a set of professional requirements for the individuals and teams involved in project implementation. It will be the basis for team building, including training and skill assessment.

The Expertise section of a Project Terms of Reference template should identify the following:

* 1. The type of work involved in the project is Data Analysis and Machine Learning along with UI development.
  2. The type of skills and abilities required to do project work are Machine Learning in Python, Data Processing, DBMS, UI development, etc.
  3. 4 students from T.Y, B. Tech will be involved in the development of the project.
  4. The period of engagement of each team member is about the same, roughly 3 months.
  5. A description of the duties and responsibility per teammate has been provided in earlier documents and will be further described in the succeeding documentation.
  6. The relationship between the team members, including leadership roles are specified in the following table.

|  |  |
| --- | --- |
| Name | Roles |
| Anup Mahindre | Analyst |
| Hiranyey Gajbhiye | Developer |
| Neeraj Ganu | Leader |
| Sanket Ostwal | Designer |

1. **Reporting**

Reports provide valued information about project performance over a certain period. Reporting is a process that starts once a project is launched and continues until the project is completed and its product is handed over. Reporting requirements will define how to write and submit project reports and what information to include. The Reporting Requirements section of a Terms of Reference template should clearly specify the requirements for the reporting process, and might include the details of:

* 1. Table of contents for project reports/ Rules for composing annexes
  2. Report templates/ The language to be used in reports
  3. Computer software programmes to be used/ Submission dates
  4. People responsible for reporting and approving

1. **Work Plan**

A work plan is a kind of strategy that aims to help solve problems throughout a project and boost employee drive and focus. It determines what actions need to be taken to start, implement, and complete the project within a specified time period and under defined budget. It is often used as a general guide for developing a project implementation plan. The Work Plan section of a Project Terms of Reference template should set out the activities and necessary resources required for achieving the project’s results and purpose. It should therefore include a summary of the anticipated work and time schedule, which are based upon the following:

* 1. An analysis of the issues, in terms of the evaluation criteria
  2. The proposed implementation methodology/ The reporting requirements

# PROBLEM DESCRIPTION

|  |  |
| --- | --- |
| The problem of | Effectively determining the value that a player brings to the club. |
| Affects | Clubs, Clubs financial teams, Players |
| The impact of which is | The current scenario is that player prices are inflated, so clubs have to borrow money or get funding from sponsors to make big transfers possible, however this isn’t sustainable, since not all players bring actual value to the club and increased borrowing causes financial instability and overall, also decreases fair-play in the sense that clubs that secure more money end up on the top. |
| A successful solution would | A successful solution will give an idea about the real value of a player based on his performance and ratings from recent time and hence help bringing down the overall inflation in the transfer market, helping improve stability and fair-play. |

|  |  |
| --- | --- |
| For | Football Clubs |
| Who | Needs it for determining values of players they want to bring in/ put on transfer list. |
| The ‘Player Value Analyzer’ | is a statistical and analytical tool |
| That | Determines player value based on statistics and performance rather than hype in football world. |
| Unlike | Scouts and advisors |
| Our product | Won’t have biases and will not have prejudices. |

# FUNCTIONAL HIERARCHY

|  |  |  |  |
| --- | --- | --- | --- |
| Goal-ID | 1 | Build Player Profile | Description |
| Objective ID | 1 | Get data | |
| Process ID: 1 | Getting the statistical data about players |
| Process ID: 2 | Cleaning data from any abnormalities |
| Objective ID | 2 | Analyse Data |  |
| Process ID: 1 | Get information about data |
| Process ID: 2 | Ascertain the correctness of data |

|  |  |  |  |
| --- | --- | --- | --- |
| Goal-ID | 2 | Classify players | Description |
| Objective ID | 1 | Extract information | |
| Process ID: 1 | Extract player positions |
| Process ID: 2 | Extract player abilities |
| Objective ID | 2 | Group players |  |
| Process ID: 1 | Group players according to position |
| Process ID: 2 | Check if player abilities and positions match and verify grouping |

|  |  |  |  |
| --- | --- | --- | --- |
| Goal-ID | 3 | Processing and learning player profiles | Description |
| Objective ID | 1 | Pre-process data | |
| Process ID: 1 | Normalize data and convert categorical data into numerical values |
| Process ID: 2 | Generate features according to domain knowledge |
| Objective ID | 2 | Select and fit a model | |
| Process ID: 1 | Selecting appropriate statistical model |
| Process ID: 2 | Fitting and optimising the model |

|  |  |  |  |
| --- | --- | --- | --- |
| Goal-ID | 4 | Processing queries and new data | Description |
| Objective ID | 1 | Process queries | |
| Process ID: 1 | Standardize query format |
| Process ID: 2 | Execute the necessary steps for prediction |
| Objective ID | 2 | Handle new data | |
| Process ID: 1 | Apply steps from previous pre-processing and get the new data in proper form |
| Process ID: 2 | Append new data to dataset |

|  |  |  |  |
| --- | --- | --- | --- |
| Goal-ID | 5 | Demonstrate player statistics | Description |
| Objective ID | 1 | Get player statistics | |
| Process ID: 1 | Search for statistics |
| Process ID: 2 | Decide on how to transmit the statistics |
| Objective ID | 2 | Display player statistics | |
| Process ID: 1 | Select which statistics to display |
| Process ID: 2 | Select and generate proper statistical graphs/representations |

|  |  |  |  |
| --- | --- | --- | --- |
| Goal-ID | 6 | Estimate player value | Description |
| Objective ID | 1 | Check player data | |
| Process ID: 1 | Fetch player data |
| Process ID: 2 | Feed the data to the model |
| Objective ID | 2 | Generate predicted value | |
| Process ID: 1 | Get the predicted value from the model |
| Process ID: 2 | Process and transmit it to the client |

|  |  |  |  |
| --- | --- | --- | --- |
| Goal-ID | 7 | Training and refining model upon feedback | Description |
| Objective ID | 1 | Get feedback | |
| Process ID: 1 | Get feedback from user and from the model |
| Process ID: 2 | Process feedback |
| Objective ID | 2 | Apply required steps for improvement | |
| Process ID: 1 | Determine feedback response |
| Process ID: 2 | Apply necessary changes to model |

# USER INTERFACES

*5.1 Abbreviated UI, it is the junction between a user and a computer program. An interface is a set of commands or menus through which a user communicates with a program. A command-driven interface is one in which you enter commands. A menu-driven interface is one in which you select command choices from various menus displayed on the screen.*

*The user interface is one of the most important parts of any program because it determines how easily you can make the program do what you want. A powerful program with a poorly designed user interface has little value. Graphical user interfaces (GUIs) that use windows, icons, and pop-up menus have become standard on personal computers.*

*GUI is a program interface that takes advantage of the computer's graphics capabilities to make the program easier to use. Well-designed graphical user interfaces can free the user from learning complex command languages. On the other hand, many users find that they work more effectively with a command-driven interface, especially if they already know the command language.*

*Graphical user interfaces, such as Microsoft Windows and the one used by the Apple Macintosh, feature the following basic components:*

* *Pointer: A symbol that appears on the display screen and that you move to select objects and commands. Usually, the pointer appears as a small angled arrow. Text -processing applications, however, use an I-beam pointer that is shaped like a capital I.*
* *Pointing device: A device, such as a mouse or trackball, that enables you to select objects on the display screen.*
* *Icons: Small pictures that represent commands, files, or windows. By moving the pointer to the icon and pressing a mouse button, you can execute a command or convert the icon into a window. You can also move the icons around the display screen as if they were real objects on your desk.*
* *Desktop: The area on the display screen where icons are grouped is often referred to as the desktop because the icons are intended to represent real objects on a real desktop.*
* *Windows: You can divide the screen into different areas. In each window, you can run a different program or display a different file. You can move windows around the display screen, and change their shape and size at will.*
* *Menus: Most graphical user interfaces let you execute commands by selecting a choice from a menu.*

*In addition to their visual components, graphical user interfaces also make it easier to move data from one application to another. A true GUI includes standard formats for representing text and graphics. Because the formats are well-defined, different programs that run under a common GUI can share data. This makes it possible, for example, to copy a graph created by a spreadsheet program into a document created by a word processor.*

*5.2 Characteristics of Successful User Interfaces*

* ***Clear****: Clarity is the most important element of user interface design. Indeed, the whole purpose of user interface design is to enable people to interact with your system by communicating meaning and function. If people can’t figure out how your application works or where to go on your website they’ll get confused and frustrated.*
* ***Concise****: Clarity in a user interface is great, however, you should be careful not to fall into the trap of over-clarifying. It is easy to add definitions and explanations, but every time you do that you add mass. Your interface grows. Add too many explanations and your users will have to spend too much time reading through them. Keep things clear but also keep things concise. When you can explain a feature in one sentence instead of three, do it. When you can label an item with one word instead of two, do it. Save the valuable time of your users by keeping things concise. Keeping things clear and concise at the same time isn’t easy and takes time and effort to achieve, but the rewards are great.*
* ***Familiar****: Many designers strive to make their interfaces ‘intuitive’. But what does intuitive really mean? It means something that can be naturally and instinctively understood and comprehended. But how can you make something intuitive? You do it by making it ‘familiar’. Familiar is just that: something which appears like something else you’ve encountered before. When you’re familiar with something, you know how it behaves – you know what to expect. Identify things that are familiar to your users and integrate them into your user interface.*
* ***Responsive****: Responsive means a couple of things. First of all, responsive means fast. The interface, if not the software behind it, should work fast. Waiting for things to load and using slaggy and slow interfaces is frustrating. Seeing things load quickly, or at the very least, an interface that loads quickly (even if the content is yet to catch up) improves the user experience. Responsive also means the interface provides some form of feedback. The interface should talk back to the user to inform them about what’s happening. Have you pressed that button successfully? How would you know? The button should display a ‘pressed’ state to give that feedback.*
* ***Consistent****: Consistent interfaces allow users to develop usage patterns – they’ll learn what the different buttons, tabs, icons and other interface elements look like and will recognize them and realize what they do in different contexts. They’ll also learn how certain things work, and will be able to work out how to operate new features quicker, extrapolating from those previous experiences.*
* ***Attractive****: This one may be a little controversial but I believe a good interface should be attractive. Attractive in a sense that it makes the use of that interface enjoyable. Yes, you can make your UI simple, easy to use, efficient and responsive, and it will do its job well – but if you can go that extra step further and make it attractive, then you will make the experience of using that interface truly satisfying. When your software is pleasant to use, your customers or staff will not simply be using it – they’ll look forward to using it. There are of course many different types of software and websites, all produced for different markets and audiences. What looks ‘good’ for any one particular audience will vary. This means that you should fashion the look and feel of your interface for your audience. Also, aesthetics should be used in moderation and to reinforce function. Adding a level of polish to the interface is different to loading it with superfluous eye-candy.*
* ***Efficient****: A user interface is the vehicle that takes you places. Those places are the different functions of the software application or website. A good interface should allow you to perform those functions faster and with less effort. Now, ‘efficient’ sounds like a fairly vague attribute – if you combine all of the other things on this list, surely the interface will end up being efficient? Almost, but not quite. What you really need to do to make an interface efficient is to figure out what exactly the user is trying to achieve, and then let them do exactly that without any fuss. You have to identify how your application should ‘work’ – what functions does it need to have, what are the goals you’re trying to achieve? Implement an interface that lets people easily accomplish what they want instead of simply implementing access to a list of features.*
* ***Forgiving****: Nobody is perfect, and people are bound to make mistakes when using your software or website. How well you can handle those mistakes will be an important indicator of your software’s quality. Don’t punish the user – build a forgiving interface to remedy issues that come up. A forgiving interface is one that can save your users from costly mistakes. For example, if someone deletes an important piece of information, can they easily retrieve it or undo this action? When someone navigates to a broken or non-existent page on your website, what do they see? Are they greeted with a cryptic error or do they get a helpful list of alternative destinations?*

|  |  |  |  |
| --- | --- | --- | --- |
| UI-ID | UI Name | Type | Scope |
| 1 |  | **Input** |  |
| 2 |  | **Forms** |  |
| 3 |  | **Navigation** |  |
| 4 |  | **Command** |  |
| 5 |  | **NL** |  |
| 6 |  | **Menu** |  |
| 7 |  |  |  |

# HARDWARE INTERFACES

*.*

|  |  |
| --- | --- |
| Profile | Description |
| Processor |  |
| RAM |  |
| Server-Side Technology |  |
| Client-Side Technology |  |
| External Devices |  |
|  |  |
|  |  |

# SOFTWARE INTERFACES

|  |  |
| --- | --- |
| Profile | Description |
| Front-end Capabilities | **Browser, HTML 5 support** |
| Back-end Capabilities | **PHP** |
| Programming Languages | **Python, PHP** |
| Operating Environment | **Any** |
| Software Platform | **Browser** |
| Database Servers | **MySQL** |
| Framework Resources | **NA** |
| API (If Any) | **Football data APIs** |
| Other Services/Resources | **NA** |
| Communication Interfaces | **Email** |

# LOGICAL DATABASES

*This section specifies the logical requirements for any information that is to be placed into a database. This may include following parameters:*

*• Types of information used by various functions*

*• Frequency of use*

*• Accessing capabilities*

*• Data entities and their relationships*

*• Integrity constraints*

*• Data retention requirements*

|  |  |  |
| --- | --- | --- |
| Database Name | Parameter | Scope |
| Player Basic | Player Id, Name, Age, Nationality, Position, Club, Height | Input Data |
| Player Attribute | Player Id, Rating, preferred Foot, Work rate, Week Foot, Skill move, Ball Control, Dribbling, Marking, Tackle, Acceleration, Speed, Stamina, Shot, Heading | Input Data |
| GK Attribute | Player Id, Positioning, Diving, Kicking, Handling, Reflexes, Penalties | Input Data |
| Price | Player id, Base Price | Output Data |
| Window | Player Id, Sold Price, Manager | Updatable record |

# NON-FUNCTIONAL REQUIREMENTS

* *Reliability: Specify the factors required to establish the required reliability of the software system at time of delivery. If you have MTBF requirements, express them here. This doesn’t refer to just having a program that does not crash. This has a specific engineering meaning.*
* *Availability: Specify the factors required to guarantee a defined availability level for the entire system such as checkpoint, recovery, and restart. This is somewhat related to reliability. Some systems run only infrequently on-demand (like MS Word). Some systems have to run 24/7 (like an e-commerce web site). The required availability will greatly impact the design. What are the requirements for system recovery from a failure? “The system shall allow users to restart the application after failure with the loss of at most 12 characters of input”.*
* *Security: Specify the factors that would protect the software from accidental or malicious access, use, modification, destruction, or disclosure. Specific requirements in this area could include the need to:*
  + *Utilize certain cryptographic techniques*
  + *Keep specific log or history data sets*
  + *Assign certain functions to different modules*
  + *Restrict communications between some areas of the program*
  + *Check data integrity for critical variables*
* *Maintainability: Specify attributes of software that relate to the ease of maintenance of the software itself. There may be some requirement for certain modularity, interfaces, complexity, etc. Requirements should not be placed here just because they are thought to be good design practices. If someone else will maintain the system*
* *Portability: Specify attributes of software that relate to the ease of porting the software to other host machines and/or operating systems. This may include:*
  + *Percentage of components with host-dependent code*
  + *Percentage of code that is host dependent*
  + *Use of a proven portable language*
  + *Use of a particular compiler or language subset*
  + *Use of a particular operating system*
* *Correctness - extent to which program satisfies specifications, fulfills user’s mission objectives*
* *Efficiency - amount of computing resources and code required to perform function*
* *Flexibility - effort needed to modify operational program*
* *Interoperability - effort needed to couple one system with another*
* *Reliability - extent to which program performs with required precision*
* *Reusability - extent to which it can be reused in another application*
* *Testability - effort needed to test to ensure performs as intended*
* *Usability - effort required to learn, operate, prepare input, and interpret output*

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **ID** | **Characteristic** | **H/M/L** | **1** | **2** | **3** | **4** | **5** | **6** | **7** | **8** | **9** | **10** | **11** | **12** |
| 1 | Correctness | H |  | 2 |  |  |  |  |  |  |  |  |  |  |
| 2 | Efficiency | L |  |  |  |  |  |  |  |  |  | 10 |  |  |
| 3 | Flexibility | L |  |  |  |  |  |  |  |  |  |  | 11 |  |
| 4 | Integrity/Security | H |  |  |  |  |  | 6 |  |  |  |  |  |  |
| 5 | Interoperability | M |  |  |  |  |  |  |  |  | 9 |  |  |  |
| 6 | Maintainability | H |  |  |  |  | 5 |  |  |  |  |  |  |  |
| 7 | Portability | M |  |  |  |  |  |  | 7 |  |  |  |  |  |
| 8 | Reliability | H | 1 |  |  |  |  |  |  |  |  |  |  |  |
| 9 | Reusability | L |  |  |  |  |  |  |  |  |  |  |  | 12 |
| 10 | Testability | M |  |  |  |  |  |  |  | 8 |  |  |  |  |
| 11 | Usability | H |  |  | 3 |  |  |  |  |  |  |  |  |  |
| 12 | Availability | M |  |  |  | 4 |  |  |  |  |  |  |  |  |