ANALYSIS

# 2.1 Introduction

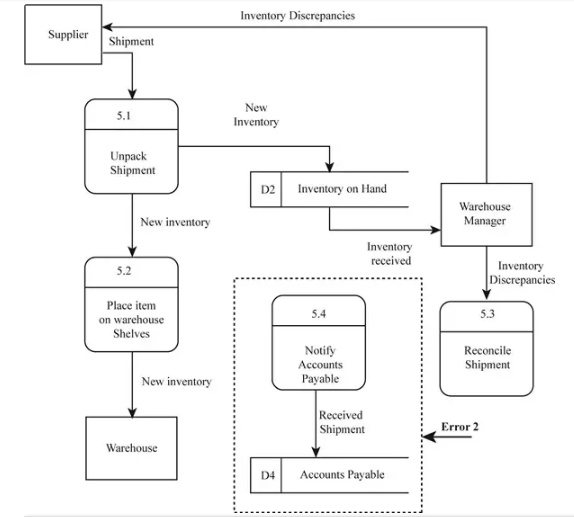
Analyzing is the method to identify and documenting of requirement of the proposed system. The analysis first phase is to feasibility study after that we model the system in use case diagram and class diagram which is also known as system modelling.

# 2.2 Analysis Methodologies

There are mainly three types of methodologies used namely Hard, Soft and Combined Approach. In this project, Hard approach has been implemented. The Hard systems approach can be used to address both qualitative and quantitative problems. It involves a step by step procedure, which can be iterative, and the process should be revised if new information to light later stage in the process changes the situational perspective.

Different stages should be implemented in Hard approach, initially we should identify the problem. After that, situation should be described as it currently is using diagrams such as DFD, should also the system as it would be ideally and the constraints preventing the system operating in this way at present. Identify metrics by which you will know if you have achieved your goal and generate ideas of possible routes to attain the ideal situation. Evaluate how these routes identified will behave in practice, this may involve pilot studies, feasibility trials or tests. Then, should decide which of these routes should be pursued and follow this route and monitor and evaluate the outcomes.

A Data Flow Diagram or DFD is traditional visual representation of the information flows within a system. A neat and clear DFD can depict a good amount of the system requirements graphically which can be manual, automated or combination of both. DFD shows how information enters and leaves the system, what changes the information and where information is stored. The purpose of a DFD is to show the scope and boundaries of a system as a whole.



Justification over choosing Hard Approach:

Hard systems approaches assume objective reality of systems in the world, Well-defined problem to be solved, Technical factors foremost, Scientific approach to problem-solving and moreover an ideal solution. It is also the more traditional way of viewing systems in Computing Science. It is the rigid techniques and procedures to provide unambiguous solutions to well-defined data and processing problems, focused on computer implementations.

So, in hard approach problem has a definite solution, the problem has a number of achievable goals, they answer the ‘how’ questions and moreover has a deterministic complexity.

# 2.3 Information Gathering

In order to determine the requirements of a system, information must be gathered from the customer.  Ideally, the information obtained will enable a well-defined, accurate, and complete description of how the business functions as well as the people, functions and data involved.  However, this is not always the case, and information is often misinterpreted or omitted entirely.

There are many techniques that can be employed when gathering information. The type of information you are trying to obtain, as well as the people providing the information, will determine which techniques you should use.

Different Methods of gathering information include interviews, Questionnaires, observation and study of existing organizational documents, forms and reports. For this project of “Inventory Management System” Interviews which is a method has been used.

Questions were asked and pointed out during the interviews with the customer and employees for the data gathering and various result were noticed such as the problems related to the manual work being done, problems related to the inventory levels and item details.

# 2.4 Feasibility study

Feasibility study is defined as the study of the extent in the real world scenario to which a project can be performed successfully. This determines whether the solution to a problem is practical and workable. For software development, information like resources availability, cost and time estimation, benefits to the society are considered. The aim of this study is to develop a software which is acceptable by users, scalable, comfortable and complies with the set standards and existing system. Different types of feasibility study are listed below with the relation with project:

* Economic feasibility (Cost-benefit Analysis): The project requires some of the components that may costs money which may be the barrier to this first phase of development but can be included in further improvements.
* Operational feasibility: This is related to if the project will be able to solve the problems and take advantages of opportunities.
* Technical feasibility: The technology of the internet is sufficient to develop this project and the skill is developing gradually on the subject matter.
* Time feasibility: The project is scheduled with the Gantt chart along with Work Breakdown Structure, it seems that the tasks are being performed as planned.

# 2.5 SRS (System requirements specification)

SRS is a document that enlists the features and functionalities of a system/software. This document comprehensively describes the intended purpose of the system and what the system will do and what is to be expected from the system.

## 2.5.1 Functional Requirement

In software engineering, functional requirements are the features that are intended to be incorporated in the system being built. Functional requirements when built into a system makes the system functional as per the needs of the user. Functional requirements can relate to technical details, software-hardware or specific functionality that defines what a system is supposed to accomplish. Functional requirements for the system being built is depicted as follows along with its rationale, dependencies and description.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ID | Title | Description | Rational | Dependency |
| FR1 | Registration | Registration is required to have login in the system | To create new user | n/A |
| FR2 | Login | Login is required to have access in the system | To use the system as registered user. | FR1 |
| FR3 | Create items | Inventory is created for items to be placed. | To place the items | FR2 |
| FR4 | Update items | Inventory need to be updated. | To update the items. | FR2 |
| FR5 | Delete items | Able to Delete items | To delete items | FR2 |
| FR6 | View Items | Able to view the items | To view items | FR2 |
| FR7 | Authenticate Users | Users is checked if genuine. | To check the user status. | FR1,FR2 |
| FR8 | Order history | Able to view history of items. | To see the history. | FR2,FR7 |
| FR9 | Stock levels | Able to check the levels of items. | To view the levels of stock. | FR2,FR7 |
| FR10 | Supplier Details | Able to view supplier details | To view details of supplier | FR2,Fr7 |

## 2.5.2 Non Functional Requirement

Non-functional Requirement postulates, in terms of constraints or prerequisites, how the system performs. As important the functional requirements are, non-functional requirements play equal and vital role in overall efficiency and increase functionality and usability of the system covering the various non-functional requirements.

Here enlisted are some of the non-functional requirements that the system being developed:

|  |  |  |  |
| --- | --- | --- | --- |
| ID | Title | Description | Dependency |
| NF1 | Security | System must ensure data security | FR1,Fr2 |
| NF2 | Performance | Should perform as expected | N/A |
| NF3 | Scalability | The system has to grow. | N/a |
| NF4 | Reliability | Should deliver the accurate and reliable data. | N/A |
| NF5 | Maintainability | Should be easy to be maintained | N/A |
| NF6 | Usability | Acceptance of end users | FR1.FR2,FR7 |
| NF7 | Documentation | In order to know about the system | N/A |
| NF8 | Availability | Convenience of the user | N/A |
| NF9 | Legal | System must be legal | N/A |
| NF10 | Privacy | System should keep user information private. | Fr1,FR2,FR7 |

## 2.5.3 MOSCOW Prioritization

After the requirement gathering, they have to be segregated based upon their priorities. Requirements should be prioritized as the stakeholders can't always get what they desire because of limitations in the technology. This is because of limited time frame and budget. So the requirements that a system must have in order to achieve its basic goal should be incorporated in first delivery and others can be integrated in future improvements.

There are many prioritization techniques like: Ranking, Numerical Assignment, Bubble Sort Technique, Hundred Dollar Method, Analytic Hierarchy Process, Five Whys and so on. Among them, we are using MoSCoW prioritization technique to prioritize our functional and non-functional requirements.

This technique uses 4 priority groups instead of numbers: Must have, Should have, Could have and Would have and was developed in 1994 by Dai Clegg.

Must have (M) - these requirements have to be incorporated in the final solution as this has basic role to achieve the system's aims and objectives.

Should have (S) - requirements of high-priority better if included within the delivery time frame but can be included in future increment if time frame is limited.

Could have (C) - requirements that are nice to have in the system but can be accepted in not included. If possible these can be incorporated.

Would have(W) – requirements that is considered ok not to be included in the current version but can be added in future increments.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ID | Requirements | MOSCOW | Rational | Remarks |
| FR1 | Registration | M | To register a new user | New user must register to the system |
| FR2 | Login | M | To Login the user | User must be logged in |
| NF1 | Performance | M | To run the system smoothly | System must run Smoothly. |
| NF3 | Reliability | C | Accurate information | Precise info. |
| NF9 | Maintainability | C | Maintenance should performed. | Maintenance could be done regularly. |
| NF10 | Privacy | M | Confidentiality should be maintained. | Must keep information private. |
| FR9 | Stock levels | S | Should provide the info regarding the levels. | Should have it now or developed later. |

## 2.5.4 Hardware and software specification:

The system being developed requires the below mentioned software and hardware specifications:

Hardware

* RAM: 2 GB minimum
* Processor: Core 2 Duo and above
* Hard disk space: 300GB

Software

* Operating system: Windows 8,Linux
* Browser: Chrome, Mozilla
* Database: SQL

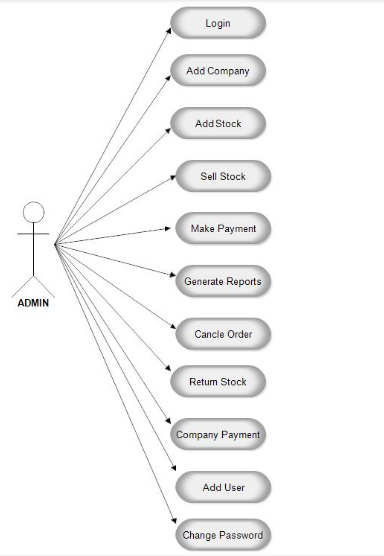
# 2.6 Use Case Diagram

A graphical representation that depicts the interactions of the system elements with each other. To identify, simplify, and organize the system requirements use cases are used during analysis. System is something that is being built or developed. It is a UML (Unified Modeling Language) diagram and is a standard notation for real-world systems and objects modelling. It uses its own notations to depict the interactions which are mentioned below:

* Boundary- it defines the system that is of interest
* Actors - people involved with the system and has defined roles who interact with the system
* User cases --These are the functions of the system which an actor performs according to the role designated.
* Relationship -the notation that depicts the relationship between actors and use cases.

Why use Use-case Diagrams?

* To know the details about the user-interaction goals with the system
* To segregate the requirements into the smaller portions of the scenarios where the understanding becomes easier and clear.
* The needs of the systems, requirements can be gathered based upon the functionalities and the features that is operated in the system.
* To understand better about the system work flow and processes
* Defines the overall development cycle by helping to plan (time, budget, priorities, and complexities), test-cases designations and final user documentation.



# 2.7 System Architecture

## 2.7.1 NLA

Firstly, for the development of any project assigned it is essential to find out the required candidate classes and methods. So, in order to recognize the possible classes and methods we will be using NLA or Natural Language Analysis. NLA is basically a procedure of inspecting the given scenario in order to identify all the possible methods and classes in a systematic way. NLA involves series of steps before determining the final results. Nouns and verbs are acknowledged from the scenario, repeated ones are excluded and eventually final results are extracted.

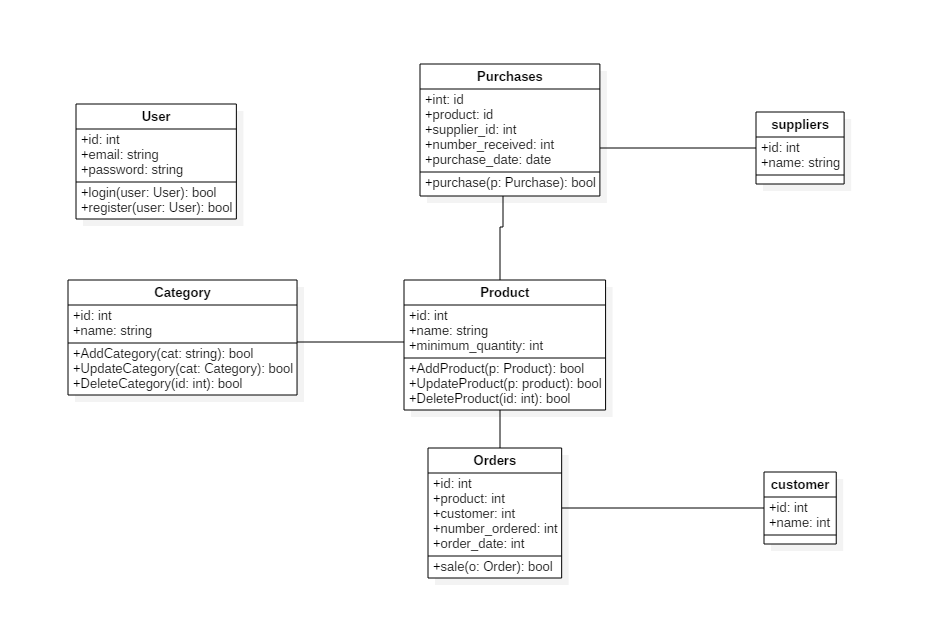
In NLA, classes are dictated by the nouns whereas the methods/functions are derived from the verbs. The reason behind the implementation of this process along with the designing of the diagrams before even we start the coding is to understand the flow of code and to successfully accomplish the mentioned goals smoothly.

## 2.7.2 Initial class diagram

Class diagram basically a blueprint of a system. It is a graphical representation of the relationships between the classes, objects with their attributes and methods or functions. Class diagrams are very effective tool to over the system that too with detailed information and modelling clarity Classes are similar to entities in the database system with set of methods and attributes. Objects can be derived from the class as its instance to run an d do many different things in the system

Classes are depicted as rectangular boxes with 3 sections: top level for the class name, middle section for the variables, and the last section for the methods. Relationships like association, aggregation, and composition can also be represented in the diagram by their specific notations.

Class diagrams are the source from which a full-fetched complete system is developed by implementing it. Implementation is aided by the diagrams as it has the details about the entities, relations, visibility, methods and attributes. This is later turned into a complete functional system.



## 2.7.3 System Architecture

It is known as the conceptual model, stating the overall view of the system. I have chosen the three- tier architecture which is a client-server architecture for the development of data access, computer data storage, functional process logic and user interface and to maintain the independent modules found on separate platform.

Following are the three tiers in this architecture:

* Presentation tier
* Application tier
* Data tier

Because of the following reasons, I have decided to use three-tier architecture in my project

* Applying object-oriented concept is easier.
* Updating data provider queries is easy and fast.
* In order to maintain, understand the large and complex project is not difficult.
* Security for the database can be provided in Application layer.

