Analysis

# 2.1 Introduction to analysis

Analysis is a comprehensive inspection of parts and functions within the software. Specifying possible conditions and incorporating user provided information define what the software is intended to be used for and the path it will later be developed on. Conflicting decisions, project alteration are mitigated with the help of cooperative communication among team members.

## Activities involved in analysis

**Problem Investigation**

* It predicts the characteristics of system based on previous experience, which include certain problems or features and requirements for a new system.
* It can lead to analysis of areas that would otherwise go unnoticed by inexperienced analyst. However, if shortcuts are taken and bias is introduced in conducting the investigation, then requirement Anticipation can be half-baked.

**Solution Investigation**

* It is studying the current system and documenting its features for further analysis.
* It is at the heart of system analysis where analyst documenting and describing system features using fact-finding techniques, prototyping, and computer assisted tools.

**Requirement Specification**

* It includes the analysis of data which determine the requirement specification, description of features for new system, and specifying what information requirements will be provided.
* It includes analysis of factual data, identification of essential requirements, and selection of Requirement-fulfilment strategies.

**Feasibility analysis**

A feasibility study is a test of a system proposal according to its workability. Impact on the organization, ability to meet user needs, and effective use of resources. It consists of Technical, Operational, Schedule and Economical.

Feasibility analysis implemented for the project

|  |  |  |
| --- | --- | --- |
| Factors for feasibility analysis | Determination of the analysis | Justification |
| Technical Feasibility | Is it capable in terms of hardware, software, personnel and expertise to handle the project? | Application will be compatible with x86 and x64 systems. Basic performing system will be branched out for employees. |
| Economic Feasibility | What is the ROI?  What is the project financial structure? | Faster transaction. Low chance of error. |
| Operational Feasibility | How do the end-users feel about a new process that may be implemented? | Skeumorphic design will allow users to understand items in a glance. |
| Schedule Feasibility | How long the system will take to develop? | Grantt chart and milestone tracking will allow to view project timeline. |
| Legal Feasibility | How does it comply with local laws and regulation standard? | Licenses and copyrights will be handed over after project completion. |

**Final Specification**

Specifications would state what the system would achieve. Specifications drawn up are improved for implementation. Finally, software requirement specification written would be given to user and an agreement is reached.

**Hardware Study**

* Determine Hardware and Software required to execute the application.
* Determine Response time, Volume of data to be processed, Frequency of reports and then pick the hardware.

**System Design**

This stage involves Logical Design of the System, identifying Objects, Database Design, Program Specification, Implementation and Test Plan.

**System Modification**

* This involves fixing errors
* Adding/Deleting features as required by users
* Tuning the System features.

# Requirement Analysis

Any project’s requirements need to be well thought out, balanced and clearly understood by all involved, but perhaps of most importance is that they are not dropped or compromised halfway through the project.

Typically, functional requirements will specify a behaviour or function, for example:

“Display the name, total size, available space and format of a flash drive connected to the USB port.” Other examples are “add customer” and “print invoice”.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **ID** | **Title** | **Description** | **Rational** | **Dependency** |
| FR1 | User Registration | Create a User profile | User credentials | N/A |
| FR2 | User Login | Verify user access | Check user credential for access | FR1 |
| FR3 | Edit User | Update or edit incorrect user detail | Change user profile | FR2 |
| FR4 | Delete User | Delete user from system | User deletion | FR2 |
| FR5 | Employee Registration | Add a new employee | Employee enters credentials | FR2 |
| FR6 | Employee Login | Verify employee access | Check employee credential for access | FR2 |
| FR7 | Edit Employee | Change employee information | Edit employee credentials | FR5 |
| FR8 | Add Item | Shop item to be added | Purchasable goods to be added | FR5 |
| FR9 | Edit Item Detail | Change item details | Change Item | FR8 |
| FR10 | Remove Item | Delete item from the stock | Item deletion | FR8 |
| FR11 | Generate transaction report | Preview transaction per customer | Generate transaction in real time | FR12 |
| FR12 | Preview history | View purchase history | Customer history | FR11 |
| FR13 | Generate sales per day | View sales amount per day | Preview sales | FR11/FR12/FR8 |

# Non-functional Requirements

Non-functional requirements cover all the remaining requirements, which are not covered by the functional requirements. They specify criteria that judge the operation of a system, rather than specific behaviours, for example: “Modified data in a database should be updated for all users accessing it within 2 seconds.”

Some typical non-functional requirements are:

* Performance – for example Response Time, Throughput, Utilization, Static Volumetric
* Scalability
* Capacity
* Availability
* Reliability
* Recoverability
* Maintainability
* Serviceability
* Security
* Regulatory
* Manageability
* Environmental
* Data Integrity
* Usability
* Interoperability

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **NID** | **Function** | **Rational** | **Dependency** | **Remarks** |
| N01 | Security | Data privacy and integrity | N08 | Potential security threats minimized |
| N02 | Reusability | Used for later future | N03, N08 | Code architecture will be developed upon as a base |
| N03 | Maintainability | Regular maintenance and new functionality to be introduced | N05, N07 | Changing the system and maintaining accordingly. |
| N04 | Visibility | Skeuomorphic design for easier UI experience. | N05, N07 | Easy view and understanding from icon and buttons. |
| N05 | Reliability | Robustness of system during heavy load. | N07 | Failure during heavy task should be minimized |
| N06 | Availability | System should be available in need or a backup system to be implemented | N01 | System downtime will decrease overall system availability |
| N07 | Environmental | Should complement visibility and reliability of the system | N04, N05 | Easy to use and efficient. |
| N08 | Data integrity | Data protection from potential threats. Data hierarchy implementation. | N01S | Keeping the data encrypted and inaccessible to employees in lower hierarchy. |

# Moscow Prioritisation

In a DSDM project where time has been fixed, it is vital to understand the relative importance of the work to be done in order to make progress and keep to deadlines. Prioritisation can be applied to requirements/User Stories,  
  
tasks, products, use cases, acceptance criteria and tests, although it is most commonly applied to requirements/ User Stories. (User Stories are a very effective way of defining requirements in an Agile style; see later chapter on  
Requirements and User Stories for more information.)  
  
MoSCoW is a prioritisation technique for helping to understand and manage priorities. The letters stand for:

* **M**ust Have
* **S**hould Have
* **C**ould Have
* **W**on’t Have this time

MoSCoW prioritization for functional requirements

|  |  |  |
| --- | --- | --- |
| **ID** | **Title** | **MoSCow** |
| FR1 | User Registration | Must have |
| FR2 | User login | Must have |
| FR3 | Edit User | Must have |
| FR4 | Delete User | Must have |
| FR5 | Employee Registration | Must have |
| FR6 | Employee Login | Should have |
| FR7 | Edit Employee | Should have |
| FR8 | Add Item | Must have |
| FR9 | Edit Item detail | Should have |
| FR10 | Remove Item | Should have |
| FR11 | Generate Transaction Data | Could have |
| FR12 | Preview History | Should have |
| FR13 | Generate Sales Per Day | Should have |

**MosCow prioritization for non-functional requirements**

|  |  |  |
| --- | --- | --- |
| **ID** | **NAME** | **PRIORITY** |
| NFR01 | Security | Must have |
| NFR02 | Reusability | Must have |
| NFR03 | Maintainability | Must have |
| NFR04 | Visibility | Must have |
| NFR05 | Reliability | Should have |
| NFR06 | Availability | Should have |
| NFR07 | Environmental | Must have |
| NFR08 | Data integrity | Must have |

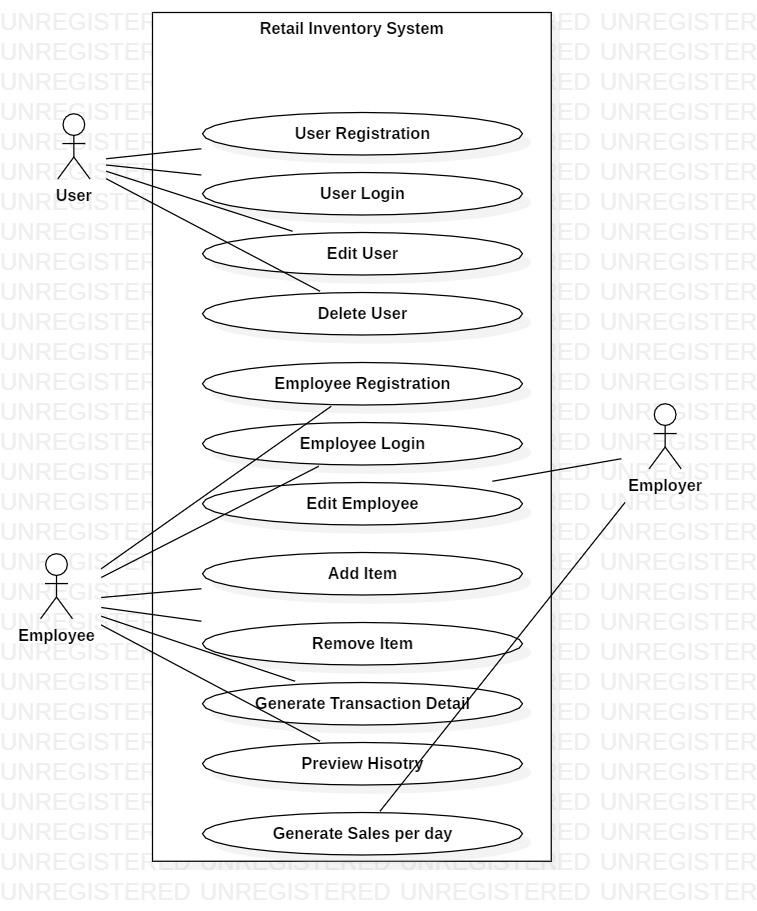
# SRS(Software Requirement Specification)

A software requirements specification (SRS) is a comprehensive description of the intended purpose and environment for software under development. The SRS fully describes what the software will do and how it will be expected to perform.

|  |  |
| --- | --- |
| Operating System | Windows 10 |
| System Type | 32-bit or 64-bit |
| Processor | Intel Core 2 Duo or higher |
| RAM | 2 GB or higher |
| Programming Language Used | Php,html |
| Database | Microsoft SQL Server |

# Use case Diagram

A use case is a methodology used in system analysis to identify, clarify, and organize system requirements. In this context, the term "system" refers to something being developed or operated, such as a mail-order product sales and service Web site. Use case diagrams are employed in UML (Unified Modeling Language), a standard notation for the modeling of real-world objects and systems.



Description of Uuse cases

|  |  |  |
| --- | --- | --- |
| **ID** | **Title** | **Description** |
| 1 | User Registration | Register user with credential in database |
| 2 | User login | Verify and let user access menu items. |
| 3 | Edit User | Change user detail and credentials. |
| 4 | Delete User | Remove user credential from database |
| 5 | Employee Registration | Register an employee with credential to insert in database. |
| 6 | Employee Login | Verify and let employee access employee dashboard |
| 7 | Edit Employee | Change employee details and shifts |
| 8 | Add Item | Add particular item which is in or new to stock |
| 9 | Edit Item detail | Edit Item details. Price,name,brand. |
| 10 | Remove Item | Delete item from database. |
| 11 | Generate Transaction Data | Produce a report of transaction in real time. |
| 12 | Preview History | Preview previous purchases for special discounts |
| 13 | Generate Sales Per Day | Preview all the sales and profit of the day. |

# Class Diagram

In software engineering, a class diagram in the Unified Modeling Language (UML) is a type of static structure diagram that describes the structure of a system by showing the system's classes, their attributes, operations (or methods), and the relationships among objects.

Scenaraio.

In todays scenario, the world we live in is getting smarter each day. The way we do and precive things have transformed into more practical and easier route. From trading goods as a method for payment to using paper currency and now turning paper currency into digital wallets. To assist todays improving transaction procedure this system will keep track of user payment and will accept both cash and digital payment method from their own devices. The retail shop in this theme accpets inventory, analyzes daily profit and provide a reciept to the customer to validate purchase of a product.

|  |  |  |
| --- | --- | --- |
| **List of Candidate Classes (noun)** | **List of Candidate Attributes (Adjective)** | **List of Candidate operations (Verb)** |
| Shop, product, customer,employee, database, owner,report, payment, receipt, owner. | Sales, stock, store, trade | Manage product, information, edit, remove, record, delete, track, add, create, view, search |

Potential class, attributes and operation

|  |  |  |
| --- | --- | --- |
| **List of possible class(noun)** | **List of possible attributes (Adjective)** | **List of possible operations (Verb)** |
| Customer/User, Employee, Owner/Employer | Stock, sales, store | Create, Add, edit, remove, track, view, search |

