**[Report Title]**

**TECHNICAL REPORT**



**SUBMITTED BY**

[Student Name]

[AG #]

**ADVISED BY**

[Supervisor Name]

**A TECHNICAL REPORT SUBMITTED IN PARTIAL FULFILLMENT OF REQUIREMENT FOR THE DEGREE OF**

*MASTER OF SCIENCE*

*IN*

*COMPUTER SCIENCE*

**DEPARTMENT OF COMPUTER SCIENCE**

**FACULTY OF SCIENCES**

**UNIVERSITY OF AGRICULTURE FAISALABAD**

**DECLARATION**

I hereby declare that the contents of the report [“**Report Title**”] are project of my own research and no part has been copied from any published source (except the references). I further declare that this work has not been submitted for award of any other diploma/degree. The university may take action if the information provided is found false at any stage. In case of any default the scholar will be proceeded against as per UAF policy.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

[ Student Name]

**CERTIFICATE**

To,

The Controller of Examinations,

University of Agriculture,

Faisalabad.

The supervisory committee certify that **[Your Name] [AG #]** has successfully completed his project in partial fulfillment of requirement for the degree of M.Sc. Computer Scienceunder our guidance and supervision.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

[Supervisor Name]

Supervisor

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

[Member Name]

Member

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Dr. Muhammad Ahsan Latif

Incharge,

Department of Computer Science

**ACKNOWLEDGEMENT**

I thank all who in one way or another contributed in the completion of this report. First, I thank to ALLAH ALMIGHTY, most magnificent and most merciful, for all his blessings. Then I am so grateful to the Department of Computer Science for making it possible for me to study here. My special and heartily thanks to my supervisor, *[Supervisor Name]* who encouraged and directed me. His/her challenges brought this work towards a completion. It is with his/her supervision that this work came into existence. For any faults I take full responsibility. I am also deeply thankful to my informants. I want to acknowledge and appreciate their help and transparency during my research. I am also so thankful to my fellow students whose challenges and productive critics have provided new ideas to the work. Furthermore, I also thank my family who encouraged me and prayed for me throughout the time of my research. May the Almighty God richly bless all of you

**ABSTRACT**

An Abstract is a summary of the whole technical report. it’s also the last thing you will write but comes first in the document.The Abstract tells the reader the main points about your technical project. Readers may not have a technical background. The Abstract gives them an overview and can help them decide which specific sections to focus on. An academic abstract typically outlines four elements relevant to the completed work: The development/research focus (i.e. statement of the problem(s)/research issue(s) addressed), the methodology used, the results/findings of the work done and the main conclusions and recommendations. Write at least 200 words as abstract of your report.

Table of Contents

[Chapter 1 - INTRODUCTION 5](#_Toc536663558)

[1.1 Background: 5](#_Toc536663559)

[1.2 Description: 5](#_Toc536663560)

[1.3 Problem Statement: 5](#_Toc536663561)

[1.4 Scope: 5](#_Toc536663562)

[1.5 Objectives: 6](#_Toc536663563)

[1.6 Feasibility: 6](#_Toc536663564)

[1.7 Requirements: 7](#_Toc536663565)

[1.7.1 Functional Requirements 7](#_Toc536663566)

[1.7.2 Non- Functional Requirements 7](#_Toc536663567)

[1.7.3 Hardware Requirements 8](#_Toc536663568)

[1.7.4 Software Requirements 8](#_Toc536663569)

[1.8 Stakeholders: 8](#_Toc536663570)

[Chapter 2 – MATERIALS & METHODS 9](#_Toc536663571)

[2.1 Process Model: 9](#_Toc536663572)

[2.2 Tools & Technologies 9](#_Toc536663573)

[2.3 Design: 9](#_Toc536663574)

[2.3.1 Use Case Diagrams: 10](#_Toc536663575)

[2.3.3 Sequence Diagram: 16](#_Toc536663577)

[2.3.4 Class Diagram: 19](#_Toc536663578)

[2.3.5 Data Flow Diagram: 20](#_Toc536663579)

[2.3.6 ER Diagram: 24](#_Toc536663580)

[2.3.7 Database Model: 24](#_Toc536663581)

[2.3.8 Architecture: 25](#_Toc536663582)

[Chapter 3 - RESULTS & DISCUSSION 27](#_Toc536663583)

[3.1 Testing: 27](#_Toc536663584)

[3.2 Test Cases: 27](#_Toc536663585)

[3.3 Conclusion: 29](#_Toc536663587)

[Chapter 4 - USER MANUAL 30](#_Toc536663588)

[References 31](#_Toc536663589)

List of Figures

[Figure ‎1.1 Stakeholders 8](#_Toc536625484)

[Figure ‎2.1 Agile Activities 9](#_Toc536625485)

[Figure 2.‎2 Use Case Diagram 12](#_Toc536625486)

[Figure 2.3 Sequence Diagram 19](#_Toc536625487)

[Figure 2.4 Class Diagram 20](#_Toc536625488)

[Figure ‎2.5 Context Diagram 22](#_Toc536625489)

[Figure ‎2.6 Level 0 DFD 22](#_Toc536625490)

[Figure 2.7 Level 1 DFD 23](#_Toc536625491)

[Figure ‎2.8 Entity Relationship Diagram 24](#_Toc536625492)

[Figure 2.9 Database Model 25](#_Toc536625493)

[Figure ‎2.10 Applications's Architecture 26](#_Toc536625494)

[Figure ‎4.1 Signing in 30](#_Toc536625495)

List of Tables

[Table 2. 1: Add User 15](#_Toc536658797)

[Table 3. 1: User login Test Case 29](#_Toc536658805)

# Chapter 1 - INTRODUCTION

## 1.1 Background:

A background of a project is just a simple and short statement of the project, meaning why we need to initiate it and what problems and needs will be addressed once it’s been implemented successfully. The purpose of the background is to give an overview of the project for deciding on the need to do the project and for initiating the planning process. When you write a background for your project your primary focus should be placed on giving a general idea and explaining the key prerequisites. Write at least 200 words.

## 1.2 Description:

A description of a project is a narrative containing a more detailed explanation of the project’s goals and objectives, the definition of the business needs and problems to be addressed, potentials pitfalls and challenges, implementation methods and approaches to be applied, people and organizations interested in and/or impacted by the project. The purpose of the description is to create a foundation for further development and implementation of the project. When you develop a description, you should use accurate and specific information to explain the objectives, desired outcome and implementations methods of your future project. Write at least 250 words

## 1.3 Problem Statement:

A problem statement is a clear concise description of the issue(s) that need(s) to be addressed by a problem-solving team. Issue Statement - one or two sentences that describe the problem using specific issues. It is not a "lack of a solution" statement. For example, our problem is that we don't have an ERP system

Part of initiating a software development project is to do a reality check to determine whether or not the project even makes sense. Your main goal should be to define and justify the best implementation solution for your project.

## 1.4 Scope:

Project scope is the work that needs to be accomplished to deliver a product, service, or result with the specified features and functions. Scope Plays a Vital Role in Projects. "Scope" includes the expected work effort and results for a given project and must be documented and accepted before the project begins. A well-written scope statement is crucial to a project. You create a project scope statement to establish a solid agreement between the project team and the customer by clarifying, identifying, and relating the work of the project to the business owner's objectives. [1]

## 1.5 Objectives:

Goals and objectives define what has to be done.  A goal is simply a broad statement of what you want to do. The objectives are sub-goals, more detailed, that explain what must be done to achieve the goal. Your project should have only one goal but may have several objectives.

* Goal (more broad): We want to move the office to Houston, Texas.
* Objective (more specific): Locate an office in Houston.
* Objective (more specific): Arrange for personnel and equipment transfer
* Objective (more specific): Transfer equipment and furnishings

## 1.6 Feasibility:

A feasibility study is performed by a company when they want to know whether a project is possible given certain circumstances. Feasibility studies are undertaken under many circumstances – to find out whether a company has enough money for a project, to find out whether the product being created will sell, or to see if there are enough human resources for the project. A good feasibility study will show the strengths and deficits before the project is planned or budgeted for. By doing the research beforehand, companies can save money and resources in the long run by avoiding projects that are not feasible. There are many different types of feasibility studies; here is a list of some of the most common:

**1.6.1 Technical Feasibility** – Does the team have the technological resources to undertake the project? Are the processes and procedures conducive to project success?

**1.6.2 Schedule Feasibility** – Does the team currently have the time resources to undertake the project? Can the project be completed in the available time?

**1.6.3 Economic Feasibility** – Given the financial resources of the company/team, is the project something that can be completed? The economic feasibility study is more commonly called the cost/benefit analysis.

**1.6.4 Cultural Feasibility** – What will be the impact on both local and general cultures? What sort of environmental implications does the feasibility study have?

**1.6.5 Legal/Ethical Feasibility** – What are the legal implications of the project? What sort of ethical considerations are there? You need to make sure that any project undertaken will meet all legal and ethical requirements before the project is on the table.

**1.6.6 Resource Feasibility** – Do you have enough resources, what resources will be required, what facilities will be required for the project, etc.

**1.6.7 Operational Feasibility** – This measure how well you will be able to solve problems and take advantage of opportunities that are presented during the course of the project

## Requirements:

### 1.7.1 Functional Requirements

Any requirement which precisely specifies what the system should do is called functional requirement of the system. In other words, a functional requirement will describe a particular behavior of function of the system when certain conditions are met, for example: “Send email when a new customer signs up” or “Open a new account”.

(Write down all functional requirements of your system in the format given below. Given is an example :)

**FR01:** Provide user name and password to log in

|  |  |
| --- | --- |
| FR01-01 | System shall get Username and Password from user |
| FR01-02 | System should authenticate user name and password |
| FR01-03 | System shall let the user to log in if information is valid |
| FR01-04 | If information is not valid then system will display message to get the account by admin |

**FR02:** Create user account

|  |  |
| --- | --- |
| FR02-01 | System shall allow admin to create accounts for faculty members |
| FR02-02 | System shall collect necessary details in this regard. |

### 1.7.2 Non- Functional Requirements

Any requirement which specifies how the system performs a certain function is called non-functional requirement. In other words, a non-functional requirement will describe how a system should behave and what limits there are on its functionality. Non-functional requirements generally specify the system’s quality attributes or characteristics, for example: “Modified data in a database should be updated for all users accessing it within 2 seconds.” Typical non-functional requirements include: Performance – for example: response time, throughput, utilization, static volumetric, Scalability, Capacity, Availability, Reliability, Recoverability, Maintainability, Serviceability, Security, Regulatory, Manageability, Environmental, Data Integrity, Usability, Interoperability. [2]

**NFR01:** System shall remain available 24/7 to its users.

**NFR02:** System shall have two types of users i.e., admin and client.

**NFR02:** System shall provide tooltip for every option/button.

### 1.7.3 Hardware Requirements

(List minimum hardware requirement to run your project on user’s side. Below is the hardware requirement example for a website)

Processor: Pentium(R) Core i3 CPU or more

Hard Disk: 40GB or more

RAM: 256MB or more

### 1.7.4 Software Requirements

(List minimum software requirement to run your project on user’s side. Below is the software requirement example for a website)

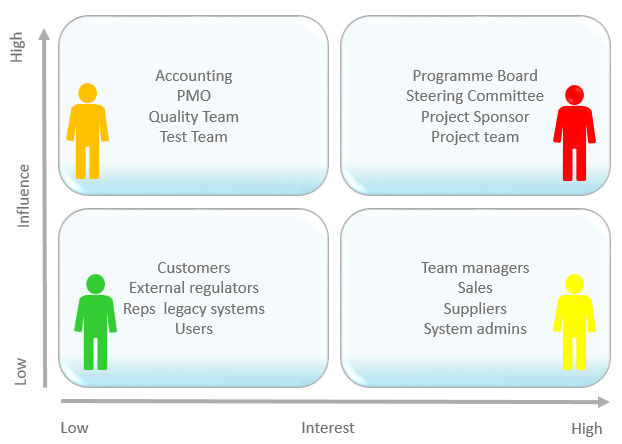
Operating System: Windows10, Windows 8.1, Windows 8, Windows 7

Browser: Google Chrome, Firefox, Mozilla etc.

## Stakeholders:

Stakeholders are different people who would be interested in the software or Stakeholders are all those with an interest or role in the project or who are impacted by the project.

We need to understand that it is the actual user who will eventually use the system and hence accept or reject the product. Therefore, ignoring the needs of any user class may result in the system failure. Below picture depicts involvement level of different categories of stakeholders. Identify your project’s stakeholders carefully.

****

*Figure 1.1 Stakeholders*

# Chapter 2 – MATERIALS & METHODS

## 2.1 Process Model:

(Describe the significance of process model you choose. For example: if we Chose Agile then briefly describe about its significance and how Agile fits with your project) Agile model is Rapid Application Development model. It is a type of incremental model. In Agile model the components or functions are developed in parallel as if they were mini projects. The developments are time boxed, delivered and then assembled into a working prototype. This can quickly give the customer something to see and use and to provide feedback regarding the delivery and their requirements. [2]

*Figure 2.1 Agile Activities*

## 2.2 Tools & Technologies

Briefly describe tools to be used in project development (one line each). For example: MS Visual Studio, NetBeans, PHP, Adobe Photoshop, MySQL etc.

## 2.3 Design:

Note: Given below are some generic software design diagrams. However more design/UML content can be added according to the requirement and nature of your project.

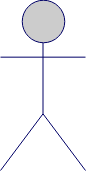
### 2.3.1 Use Case Diagrams:

A use case is a functionality the users need from the system. A use case diagram depicts the relationships among the actors and use cases. Generally, a single use case is supposed to cover all the actions or events that an actor can perform on the system at one go. The size of use case should not be very large or very small. For example, *Add User, Manage Profile, View Sales Report, Update Order* etc. are good medium size use cases. Whereas *Enter Password, Display Error Message* etc. are very small use cases and *Manage Sale & Purchase* is a very large use case.

The components in a use case diagram include:

Actors:

Actors are first thing you need to find for the use case diagram. Actors represent external entities of the system. These can be people or things, such as external hardware that interact with the system. For example, if an online store is being modeled there can be more than one actor that interacts with the store functionality. Such as the Customer and stocker will be the actors in the system. It is represented simply by a stick figure with its name at the bottom of it.



*Actor*

Use Cases:

Use cases are functional parts of the system. They figure out what actions/functionalities a user will perform. Use cases are basically the functional requirements that you have pointed out in the functional and non-functional requirements topic. For example: The customer "browses the catalog", "chooses items to buy", and "pays for the items". Here browse catalog, buy item and pay for item are the use cases. Many actors can share a single use cases. The notation for a use case is an ellipse. As it is displayed below:



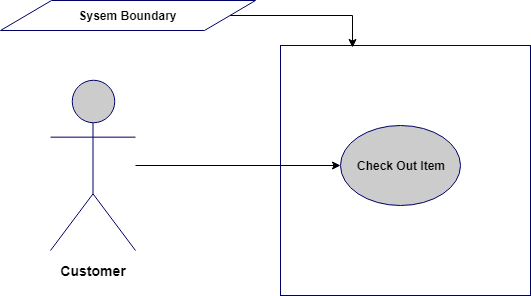
Associations:

Associations between actors and use cases are indicated in use case diagrams by solid lines. An association exists whenever there is direct interaction between actor and use case.   Associations are modeled as lines connecting use cases and actors to one another, with an arrowhead on one end of the line.



System boundary:

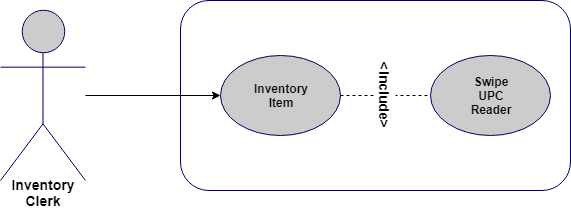
System’s boundary is drawn by a rectangle that contains use cases. The actors are placed outside the system boundary and use cases inside it.



**Relationship between Use cases:**

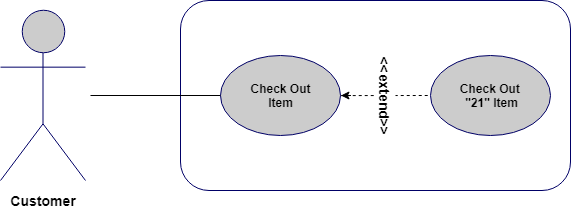
1. **Include/Uses:**

Include relationship is a relationship in which one use case (the base use case) includes the functionality of another use case (the inclusion use case). <<include>> use cases must be **used** by the use cases that **use** them before the latter can be complete. It is displayed in the diagram editor as a dashed line with an open arrow pointing from the base use case to the inclusion use case. The keyword «include» is attached to the connector.

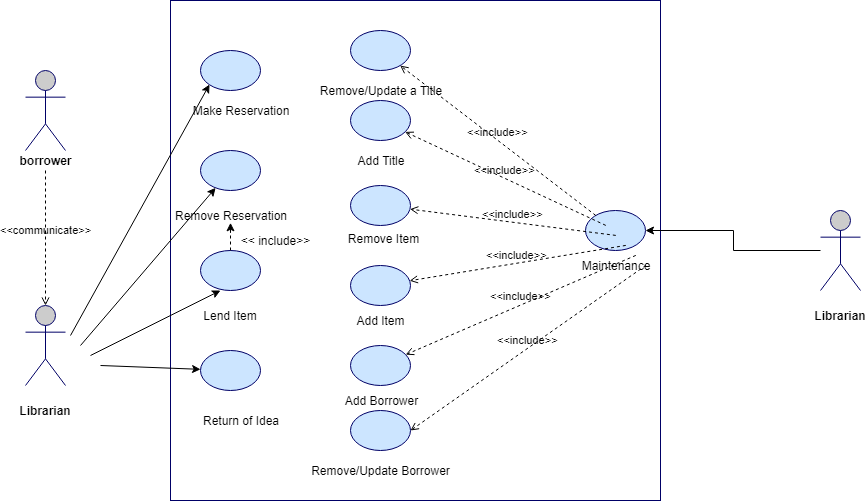


1. **Extend:**

A use case **extends** another use case to do more than the latter. It extends the functionality of one use case to further level. is displayed in the diagram editor as a dashed line with an open arrowhead pointing from the extension use case to the base use case. The arrow is labeled with the keyword «extend».



**Use Case Diagram:**





*Figure 2.2 Use Case Diagram*

**2.3.2 Usage Scenario**

Usage scenario is the actual text-based representation of the use case, among various representation methods discussed above. A usage scenario is likely to have various sections depending upon the level of details required in a given system. There is no fixed standard so far for number of sections in a use case (usage scenario).

Following is a typical table structure for usage scenario, note that it is not mandatory to write a usage scenario in the table format only, and it is likely that you will find different structures for the same representation.

|  |  |  |
| --- | --- | --- |
| **Use Case Title** | **Write Title Here (must match use case title in use case diagram)** | |
| **Abbreviated Title** |  | |
| **Use Case Id** |  | |
| **Requirement Id** |  | |
| **Description:** | | |
| **Pre Conditions:** | | |
| **Task Sequence** | | **Exceptions** |
|  | |  |
|  | |  |
|  | |  |
| . | |  |
| . | |  |
| **Post Conditions:** | | |
| **Unresolved issues:** | | |
| **Authority:** | | |
| **Modification history:**  **Author:**  **Description:** | | |

Let’s explore each section from the template provided above. (you have to make only usage scenario tables, below is description of its each part)

* **Use Case Title**

It is the name or label of the use case for which we are writing the usage scenario. Generally it must start with a Verb and it should consist of 2 to 4 words e.g. *Add User, Manage User Roles* etc.

|  |  |
| --- | --- |
| **Use Case Title** | **Add User** |

* **Use Case Id**

Sometimes use cases are indexed for better reference in overall project documentation/artifacts. This can be any form of series e.g. 1, 2, 3 etc. Priority based use case id is another famous use for this section. Use cases are indexed to present their importance in the system. You would want to set ascending or descending rating or priority for all use cases i.e. the most important use cases are ranked higher so that the project team knows what should be implemented first in the upcoming phases/deliverable of the project.

* **Requirement Id**

The purpose of this section is same as ‘Use Case Id’ section. The number written against this section represents the corresponding functional requirements this use case belongs to. It is compulsory to index all the functional requirements properly before this section can be used:

|  |  |
| --- | --- |
| **Requirement Id** | 3 |

* **Description**

It should be a very brief description of the use case under discussion. Generally this portion should consist of 2/3 lines.

|  |  |
| --- | --- |
| **Description** | This use case is about adding a new user to existing system with the privileges defined at time of user account creation. |

* **Pre-Conditions**

This section should enlist what must be true before this use case can be performed.

|  |
| --- |
| **Pre Conditions:**   * 1. All must-required information about the new user should be available.   2. Database should be available in online mode. |

* **Task Sequence & Exceptions**

This is the most important section of the usage scenario. It is also referred as Success Scenario, Actions, (or simply) Scenario. This should be a list of actor’s interaction with the use case.

|  |  |
| --- | --- |
| **Task Sequence** | **Exceptions** |
| 1. Administrator opts to Add a new user account. |  |
| 1. System asks for necessary information |  |
| 1. Administrator provides all the required information and opts to complete the operation. |  |
| 1. System after confirmation adds the new account. |  |
| 1. System sends the account creation email to the administrator’s email id and user’s email address. |  |
| 1. A log is saved on the successful operation of the use case. |  |

Alternate scenarios could be more than one; in this case, it will be better to make a bold heading to show each alternate scenario separately. Again, there can be multiple ways to show the alternate scenarios.

Exceptions are any unhandled scenarios that must be discussed under this section. Sometimes there are ambiguous situations in start of project that might hurdle the flow of events in the Task Sequence portion. In such situations the details are provided in the Exceptions section. Generally, this section should be left blank as in case of final project, you will get fixed requirements at the start and thus there should be no ambiguity.

* **Post Conditions**

The conditions that must be true depending upon the successful use case are mentioned in this section.

|  |
| --- |
| **Post Conditions:**   * + - A new user account is successfully created. |

* **Unresolved Issues**

In addition to the Exceptions portion, we write unresolved issues (if any) in this section, so that in later phases (when the situation gets more clear).

Just like Exceptions section, this will be generally left blank (or its row can be deleted from the use case table-structure.

* **Authority**

The role that is allowed to perform this use case, in our current example it will be Administrator.

* **Modification History**

If a use case is updated in later stages of the project development, the versioning information should be mentioned in this section (version can be a series such as 1.0, 2.0, 3.0 or 1.1, 1.2, 1.3 etc.)

* **Author**

It means the author of this usage scenario. Put your project/group id in this section.

* **Description**

Any more details about author, revision of the use case should be provided in this section.

So final usage scenario is as follows:

Table 2. 1: Add User

|  |  |  |
| --- | --- | --- |
| Use Case Title | Add User | |
| Use Case Id | 1 | |
| Requirement Id | 3 | |
| Description: This use case is about adding a new user to existing system with the privileges defined at time of user account creation. | | |
| Pre Conditions:   * 1. All must-required information about the new user should be available.   2. Database should be available in online mode. | | |
| Task Sequence | | Exceptions |
| 1. Administrator opts to add a new user account. | |  |
| 1. System asks for necessary information. | |  |
| 1. Administrator provides all the required information and opts to complete the operation. | |  |
| 1. There is a problem in the data provided; some data needs to be corrected.    * Administrator checks the available information and corrects the error.    * Administrator continues from the step 3. | |  |
| 1. System after confirmation adds the new account. | |  |
| 1. System sends the account creation email to the administrator’s email id and user’s email address. | |  |
| 1. A log is saved on the successful operation of the use case. | |  |
| Post Conditions:   * + A new user account is successfully created. | | |
| Unresolved issues: | | |
| Authority: Administrator | | |
| Modification history: 1.0  Author: <Project or Group ID>  Description: | | |

**Important Points:**

* + - There should be a usage scenario for each use case from use case diagram e.g. if there are (let say) 10 use cases in use case diagram, then there must be 10 separate usage scenarios (i.e. one for each).
    - Title of use cases in use case diagram and in the usage scenario must be same.
    - As said earlier, there is no standard for any fixed sections in usage scenario, so if you don’t have anything to write in a particular section, then just leave it blank or delete its row/cell from the table. It is important to note that some sections are very common across different representations of usage scenarios and such sections should not be removed/kept blank at all. These sections are: Use Case Title, Pre-Conditions, Post Conditions, Task Sequence, Author etc. Again, it all depends upon the situation and level of details required in a given system.
    - Generally, a good approach for Task Sequence portion is not to mention very small GUI level details such as:

‘Administrator clicked on Submit button’

OR

‘System shows the confirmation message’

### 2.3.3 Sequence Diagram:

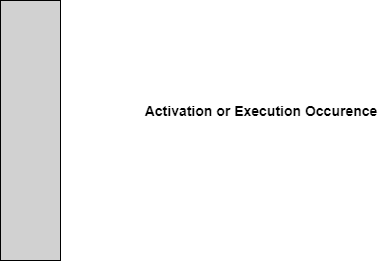
A sequence diagram simply depicts interaction between objects in a sequential order i.e. the order in which these interactions take place. We can also use the terms event diagrams or event scenarios to refer to a sequence diagram. Sequence diagrams describe how and in what order the objects in a system function. These diagrams are widely used by businessmen and software developers to document and understand requirements for new and existing systems.

**Sequence Diagram Notations:**

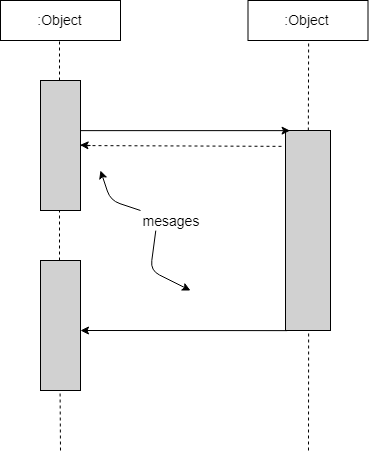
**Actors –** An actor in a UML diagram represents a type of role where it interacts with the system and its objects. It is important to note here that an actor is always outside the scope of the system we aim to model using the UML diagram.  
**Class Roles or Participants:**  
Class roles describe the way an object will behave in context. Use the UML object symbol to illustrate class roles, but don't list object attributes.



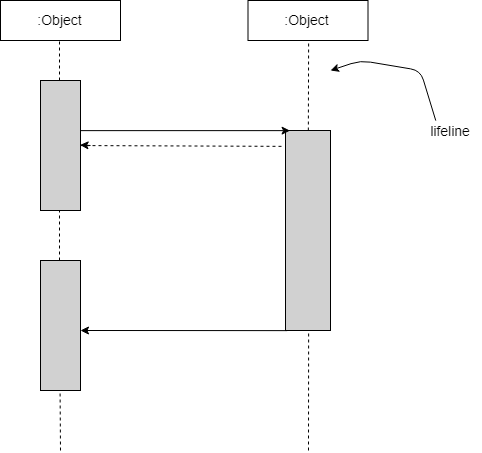
**Activation or Execution Occurrence:**  
Activation boxes represent the time an object needs to complete a task. When an object is busy executing a process or waiting for a reply message, use a thin gray rectangle placed vertically on its lifeline.



**Messages**  
Messages are arrows that represent communication between objects. Use half-arrowed lines to represent asynchronous messages. Asynchronous messages are sent from an object that will not wait for a response from the receiver before continuing its tasks. For message types, see below.



**Lifelines**  
Lifelines are vertical dashed lines that indicate the object's presence over time.

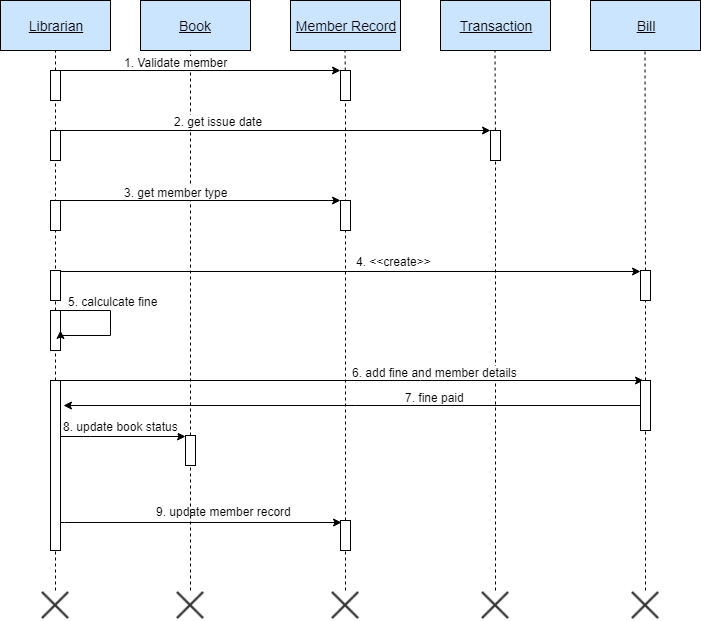


**Destroying Objects**  
Objects can be terminated early using an arrow labeled "<< destroy >>" that points to an X. This object is removed from memory. When that object's lifeline ends, you can place an X at the end of its lifeline to denote a destruction occurrence.

**Loops**  
A repetition or loop within a sequence diagram is depicted as a rectangle. Place the condition for exiting the loop at the bottom left corner in square brackets [ ].

**Sequence Diagram Example (for item return use case):**

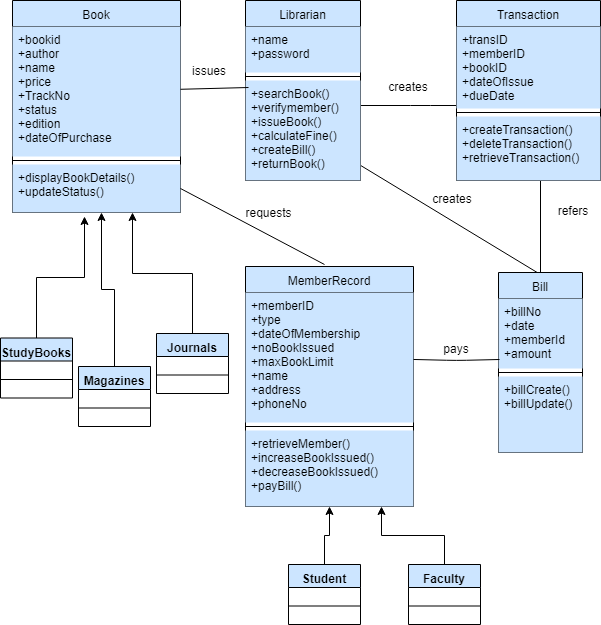
Note: Make sequence diagram for each use case illustrated in use case diagram

****

*Figure 2.3 Sequence Diagram*

### 2.3.4 Class Diagram:

A class diagram 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. For further reading visit: <https://creately.com/blog/diagrams/class-diagram-relationships/>. Below is the class diagram for library management system discussed above.

****

*Figure 2.4 Class Diagram*

### 2.3.5 Data Flow Diagram:

A data flow diagram (DFD) maps out the flow of information for any process or system. It uses defined symbols like rectangles, circles and arrows, plus short text labels, to show data inputs, outputs, storage points and the routes between each destination. A Data Flow Diagram (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. It can be manual, automated, or combination of both.

It 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. It may be used as a communications tool between a systems analyst and any person who plays a part in the system that acts as the starting point for redesigning a system. [3]

It is usually beginning with a context diagram as the level 0 of DFD diagram, a simple representation of the whole system. To elaborate further from that, we drill down to a level 1 diagram with lower level functions decomposed from the major functions of the system. This could continue to evolve to become a level 2 diagram when further analysis is required. Progression to level 3, 4 and so on is possible but anything beyond level 3 is not very common. Please bear in mind that the level of details for decomposing particular function really depending on the complexity that function. For further reading use the link given below:

<https://www.visual-paradigm.com/guide/data-flow-diagram/what-is-data-flow-diagram/>

#### DFD Diagram Notations

#### External Entity

An external entity can represent a human, system or subsystem. It is where certain data comes from or goes to. It is external to the system we study, in terms of the business process. For this reason, people used to draw external entities on the edge of a diagram.



#### Process

A process is a business activity or function where the manipulation and transformation of data takes place. A process can be decomposed to finer level of details, for representing how data is being processed within the process. 



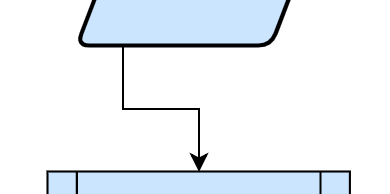
#### Data Store

A data store represents the storage of persistent data required and/or produced by the process. Here are some examples of data stores: membership forms, database table, etc. 



#### Data Flow

A data flow represents the flow of information, with its direction represented by an arrow head that shows at the end(s) of flow connector.



**Context Diagram:**

****

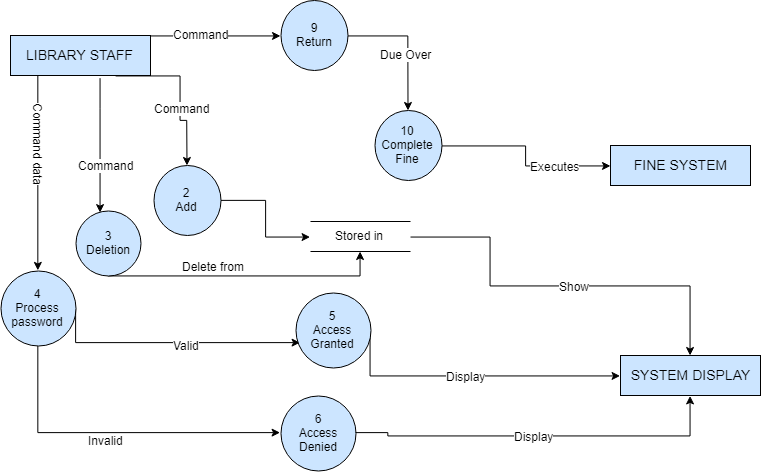
*Figure 2.5 Context Diagram*

**Level 0:**

****

*Figure 2.6 Level 0 DFD*

**Level 1:**

****

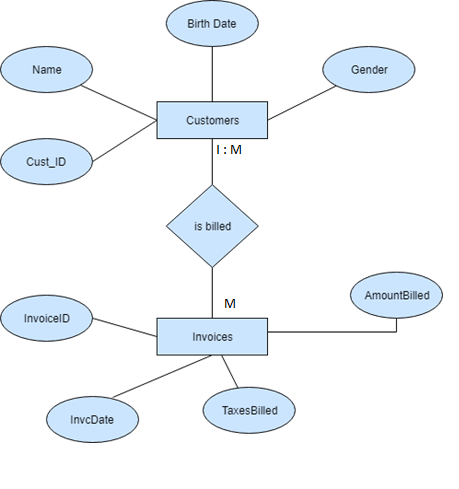
*Figure 2.7 Level 1 DFD*

**Also include Data Dictionary in this section.**

### 2.3.6 ER Diagram:

An entity relationship model, also called an entity-relationship (ER) diagram, is a graphical representation of entities and their relationships to each other, typically used in computing in regard to the organization of [data](https://www.webopedia.com/TERM/D/data.html) within [databases](https://www.webopedia.com/TERM/D/database.html) or information systems. An entity is a piece of data-an [object](https://www.webopedia.com/TERM/O/object.html)or concept about which data is stored. To learn more about ERD visit: <https://www.smartdraw.com/entity-relationship-diagram/>

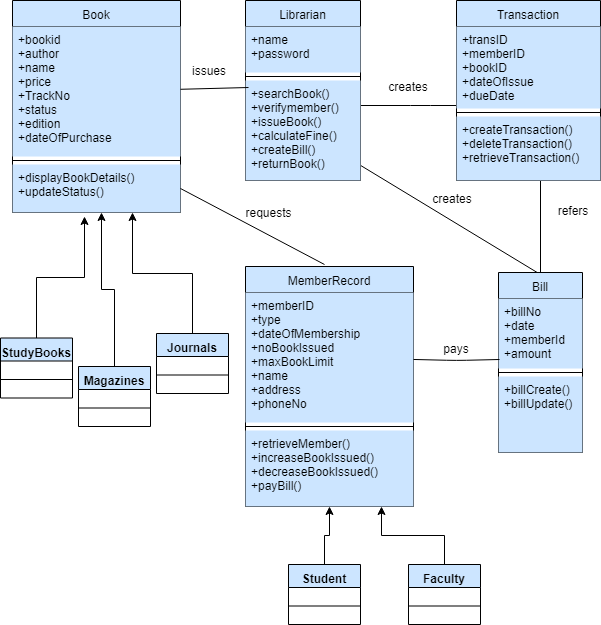
Below is example of ERD between customer and invoices in any xyz E-Billing system.



*Figure 2.8 Entity Relationship Diagram*

### 2.3.7 Database Model:

A database model shows the logical structure of a database, including the relationships and constraints that determine how data can be stored and accessed. Individual database models are designed based on the rules and concepts of whichever broader data model the designers adopt. Most data models can be represented by an accompanying database diagram. Below is an example for library management system. [4] To read more about designing database model visit: <https://www.datanamic.com/support/lt-dez005-introduction-db-modeling.html>



*Figure 2.9 Database Model*

### 2.3.8 Architecture:

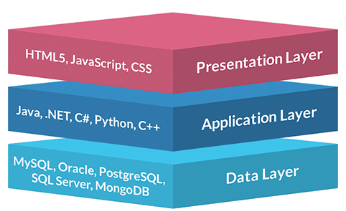
**3-Tier: (or N- Tier/multitier, chose whatever best suits your project nature)**

A 3-tier architecture is a type of software architecture which is composed of three “tiers” or “layers” of logical computing. They are often used in applications as a specific type of client-server system. 3-tier architectures provide many benefits for production and development environments by modularizing the user interface, business logic, and data storage layers. Doing so gives greater flexibility to development teams by allowing them to update a specific part of an application independently of the other parts. This added flexibility can improve overall time-to-market and decrease development cycle times by giving development teams the ability to replace or upgrade independent tiers without affecting the other parts of the system.

**Presentation Tier-** The presentation tier is the front-end layer in the 3-tier system and consists of the user interface. This user interface is often a graphical one accessible through a web browser or web-based application and which displays content and information useful to an end user. This tier is often built on web technologies such as HTML5, JavaScript, CSS, or through other popular web development frameworks, and communicates with other layers through API calls.

**Application Tier-** The application tier contains the functional business logic which drives an application’s core capabilities. It’s often written in Java, .NET, C#, Python, C++, etc.

**Data Tier-** The data tier comprises of the database/data storage system and data access layer. Examples of such systems are MySQL, Oracle, PostgreSQL, Microsoft SQL Server, MongoDB, etc. Data is accessed by the application layer via API calls.

****

*Figure 2.10 Application Architecture*

# Chapter 3 - RESULTS & DISCUSSION

In this chapter discuss overall performance or all functional and non-functional requirements you listed in chapter no. 1 as this section will verify the performance measures proposed for this project. For this software testing plays a vital role.

## 3.1 Testing:

Software testing is a process, to evaluate the functionality of a software application with an intent to find whether the developed software met the specified requirements or not and to identify the defects to ensure that the product is defect free in order to produce the quality product. In this regard, Test case writing is a major activity and considered as one of the most important parts of software testing. It is used by the testing team, development team as well as the management. If there is no documentation for an application, we can use test case as a baseline document. Below are some suggestions for writing good test cases:

## 3.2 Test Cases:

**Test cases need to be simple and transparent**

Create test cases that are as simple as possible. They must be clear and concise as the author of the test case may not execute them.

Use assertive language like go to the home page, enter data, click on this and so on. This makes the understanding the test steps easy and tests execution faster.

**Create Test Case with End User in Mind**

The ultimate goal of any software project is to create test cases that meet customer requirements and is easy to use and operate. A tester must create test cases keeping in mind the end user perspective

**Avoid test case repetition.**

Do not repeat test cases. If a test case is needed for executing some other test case, call the test case by its test case id in the pre-condition column

**Do not Assume**

Do not assume functionality and features of your software application while preparing test case. Stick to the User Requirement Specification Documents.

**Ensure 100% Coverage**

Make sure you write test cases to check all software requirements mentioned in the specification document.

**Test Cases must be identifiable.**

Name the test case id such that they are identified easily while tracking defects or identifying a software requirement at a later stage. [2]

**Test Case Template Help:**

* Test case ID: Unique ID is required for each test case.
* Test Title/Name: Test case title. E.g. verify login page with a valid username and password.
* Test priority (Low/Medium/High): This is very useful while test execution. Test priority for business rules and functional test cases can be medium or higher whereas minor user interface cases can be of a low priority. Test priority should always be set by the reviewer.
* Requirements: Requirements for which this test case is being written for. Preferably the exact section number of the requirement doc.
* Test Summary/Description: Describe the test objective in brief.
* Test Execution Date: Date when the test was executed.
* Pre-conditions: Any prerequisite that must be fulfilled before the execution of this test case. List all the pre-conditions in order to execute this test case successfully.
* Dependencies: Mention any dependencies on the other test cases or test requirement.
* Test Steps: List all the test execution steps in detail. Write test steps in the order in which they should be executed. Make sure to provide as many details as you can. Tip – In order to manage a test case efficiently with a lesser number of fields use this field to describe the test conditions, test data and user roles for running the test.
* Test Data: Use of test data as an input for this test case. You can provide different data sets with exact values to be used as an input.
* Expected Result:  What should be the system output after test execution? Describe the expected result in detail including message/error that should be displayed on the screen.
* Post-condition: What should be the state of the system after executing this test case?
* Actual result: Actual test result should be filled after test execution. Describe the system behavior after test execution.
* Status (Pass/Fail): If actual result is not as per the expected result, then mark this test as failed. Otherwise, update it as passed.
* Notes/Comments/Questions: If there are some special conditions to support the above fields, which can’t be described above or if there are any questions related to expected or actual results then mention them here. [3]

**Test Case: User Login:**

Below is test case format:

Table 3. 1: User login Test Case

|  |  |
| --- | --- |
| Test Case ID: | 1 or TC-1 |
| Test Case Title: | To verify the Login functionality of the application |
| Test Case Priority: | High |
| Requirement: | User Login |
| Test Description: | This test will verify the user login process. |
| Test Date: | mm/dd/yyyy |
| Pre-Conditions: | 1. Run the application.  2. Click Sign in button. |
| Dependencies: | Internet Availability |
| Test Steps: | 1. Enter Valid user name and password and click Login  2. Click Sign Out  2. Without entering user name click sign in  3. Without entering password click sign in  4. Enter wrong password or user name and click sign in |
| Test Data | Email id and password of user |
| Expected Results: | 1. System should open home page.  2. Login page should be displayed.  2. An error message should be shown to enter user name  3. An error message should be shown to enter password  4. Error message should be shown to enter correct password and user id |
| Actual Results: | As above |
| Post Conditions: | System shows Dash board page of signed in user. In case of unauthorized sign in attempt system shows the message “Invalid username/password”. |
| Status: (Pass/Fail) | Pass |
| Other Comments: | None |

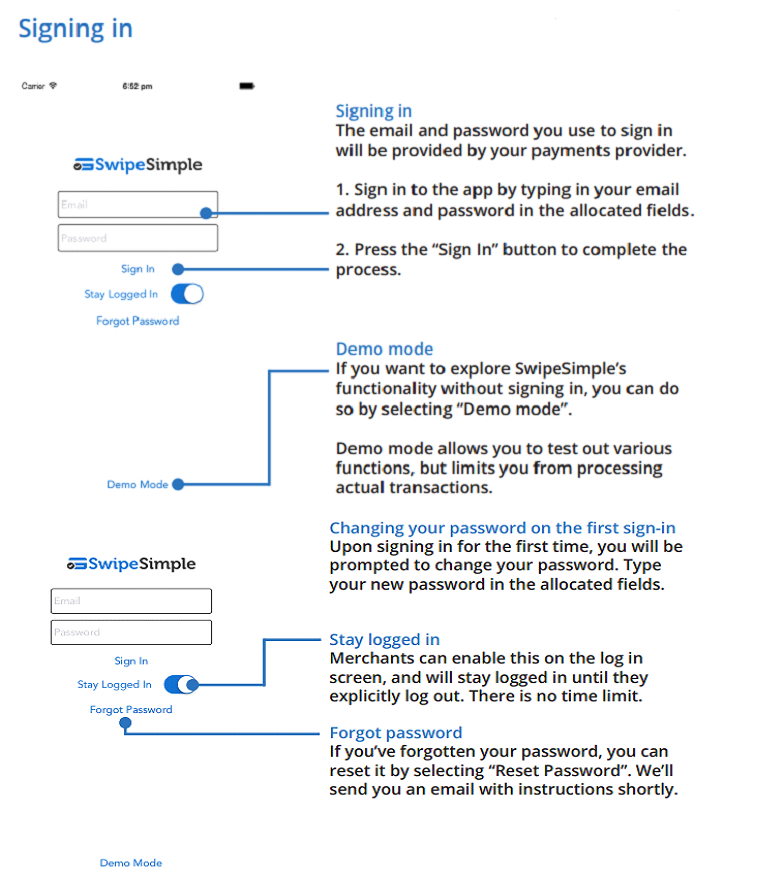
Similarly, continue Test Cases designing as above for your application. Create a test case for each of the usage scenario provided in initial phases. [4]

## 3.3 Conclusion:

In the end of this chapter, this section will conclude the overall performance and results of test cases, how many of them resulted in Pass or satisfactory status. Also, it will include the look and feel or UI/UX aspects of the software. Whether UI is user friendly or much technical to use. Discuss all possible aspects. (At least 350 words)

# Chapter 4 - USER MANUAL

Paste screen shots of your application’s user interface and briefly describe use/function of each option/button/link/field etc. present there. Following is example of a mobile app sign in page.



*Figure 3.1 Signing in*

**Follow UAF reference style to cite book, research paper (journal as well as conference paper) and website references. At least 3 to 4 reference must be present in Chapter 1 and 2.**

**Formatting instructions:**

Following formatting is already applied on the document. However, it is explicitly mentioned below:

* Minimum length of the Report: 30 pages
* Font style : Times New Roman
* Paragraph font size: 12pt
* Main Heading Size: 14pt + Bold (before and after spacing 12pt)
* Sub Heading Size: 13pt + Bold (before and after spacing 8pt)
* Sub sub heading size 12pt + bold (before and after spacing 6pt)
* Paragraph Alignment: Justified
* Picture/Chart Alignment: Center
* Picture/chart/table heading font: Times New Roman
* Picture/chart/table heading font size: 10pt, Italic, center alignment
* Picture caption goes under the picture without any extra line and line space
* Table caption goes above the table without any extra line and line space
* Table heading: Times New Roman, 10pt, Bold
* Table text: Times New Roman, 10pt
* Line Spacing: 1.15
* Left/Right/Top/Bottom Margins: 1 inch
* Table of Contents, List of Tables, List of Figures Heading: Times New Roman. 14pt, Bold
* Table of Contents, List of Tables, List of Figures: Times New Roman. 12pt
* For Table of Contents, List of Tables & List of Figures use Roman number as page number format in footer (center align)
* For Introduction onwards, use integer number as page number format in footer (center align)

**For Submission:**

**Three hard binded copies of technical report are required with memory card (attached with each copy) containing technical report and project (exe file + all source code).**