**SVKM’s NMIMS**

**Mukesh Patel School of Technology Management & Engineering**

**Computer Engineering Department**

Program: B. Tech. Semester: VIII

**Course: Software Project Management**

**Experiment No.02**

PART A

(PART A: TO BE REFFERED BY STUDENTS)

**A.1 Aim:** Finalise the problem statement and conduct the requirement gathering

**A.2 Prerequisite: -** C, C++, Software Development Life Cycle /Literature survey

**A.3 Outcome:**

**After successful completion of this experiment students will be able to**

1. Identify ambiguities, inconsistencies and incompleteness from a requirements specification
2. Identify and state functional requirements
3. Identify and state non-functional requirements
4. Frame the problem statement for the MINI project

**A.4 Theory:**

Requirements identification is the first step of any software development project. Until the requirements of a client have been clearly identified, and verified, no other task (design, coding, testing) could begin. Usually business analysts having domain knowledge on the subject matter discuss with clients and decide what features are to be implemented.

In this experiment we will learn how to identify functional and non-functional requirements from a given problem statement. Functional and non-functional requirements are the primary components of a Software Requirements Specification.

Requirements

Somerville defines "requirement" as a specification of what should be implemented. Requirements specify how the target system should behave. It specifies what to do, but not how to do. Requirements engineering refers to the process of understanding what a customer expects from the system to be developed, and to document them in a standard and easily readable and understandable format. This documentation will serve as reference for the subsequent design, implementation and verification of the system.

It is necessary and important that before we start planning, design and implementation of the software system for our client, we are clear about it's requirements. If we don't have a clear vision of what is to be developed and what all features are expected, there would be serious problems, and customer dissatisfaction as well.

Characteristics of Requirements

Requirements gathered for any new system to be developed should exhibit the following three properties:

Unambiguity: There should not be any ambiguity what a system to be developed should do. For example, consider you are developing a web application for your client. The client requires that enough number of people should be able to access the application simultaneously. What's the "enough number of people"? That could mean 10 to you, but, perhaps, 100 to the client. There's an ambiguity.

Consistency: To illustrate this, consider the automation of a nuclear plant. Suppose one of the clients say that if the radiation level inside the plant exceeds R1, all reactors should be shut down. However, another person from the client side suggests that the threshold radiation level should be R2. Thus, there is an inconsistency between the two end users regarding what they consider as threshold level of radiation.

Completeness: A particular requirement for a system should specify what the system should do and also what it should not. For example, consider a software to be developed for ATM. If a customer enters an amount greater than the maximum permissible withdrawal amount, the ATM should display an error message, and it should not dispense any cash.

Categorization of Requirements

Based on the target audience or subject matter, requirements can be classified into different types, as stated below:

User requirements: They are written in natural language so that both customers can verify their requirements have been correctly identified

System requirements: They are written involving technical terms and/or specifications, and are meant for the development or testing teams

Requirements can be classified into two groups based on what they describe:

Functional requirements (FRs): These describe the functionality of a system -- how a system should react to a particular set of inputs and what should be the corresponding output.

Non-functional requirements (NFRs): They are not directly related what functionalities are expected from the system. However, NFRs could typically define how the system should behave under certain situations. For example, a NFR could say that the system should work with 128MB RAM. Under such condition, a NFR could be more critical than a FR.

Non-functional requirements could be further classified into different types like:

Product requirements: For example, a specification that the web application should use only plain HTML, and no frames

Performance requirements: For example, the system should remain available 24x7

Organizational requirements: The development process should comply to SEI CMM level 4

Functional Requirements: Identifying Functional Requirements

Given a problem statement, the functional requirements could be identified by focusing on the following points:

Identify the high level functional requirements simply from the conceptual understanding of the problem. For example, a Library Management System, apart from anything else, should be able to issue and return books.

Identify the cases where an end user gets some meaningful work done by using the system. For example, in a digital library a user might use the "Search Book" functionality to obtain information about the books of his interest.

If we consider the system as a black box, there would be some inputs to it, and some output in return. This black box defines the functionalities of the system. For example, to search for a book, user gives title of the book as input and get the book details and location as the output.

Any high level requirement identified could have different sub-requirements. For example, "Issue Book" module could behave differently for different class of users, or for a particular user who has issued the book thrice consecutively.

**A.5 Task to be completed in PART B**

**A.5.1. Task 1:**

**Every student needs to follow following steps and record the findings in appropriate section of PART B**

1. Identify the **Scope** of the problems selected
2. Identify the end user of the solution
3. Identify the functional requirements of the project
4. Identify the **non-functional** requirements of the project
5. Identify the **feasibility** of the project(Economical feasibility)
6. Frame the final problem statement.
7. List down the list of user requirements, system requirements.
8. Identify the ambiguities, inconsistencies, incompleteness from the requirements gathered.
9. List down the development plan for the selected problem statements. SDLC(Software development life cycle)

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**PART B**

|  |  |
| --- | --- |
| Roll No.: B084 | Name: Vedica Kandoi |
| Program: B. Tech Computer Engineering | Division: B |
| Batch: B3 | Date of Experiment: 06/01/2023 |
| Date of Submission: | Grade: |

**B.1 Tasks given in PART A to be completed here**

1. Identify the **Scope** of the problems selected

Communicating with and voicing complaints to the government may be a daunting task for many citizens of the country, due to long queues and longer wait times. This portal aims to bridge that gap between the citizens and the governing body, by addressing all issues regarding various departments in India. The portal can take any complaints and forward it to the concerned authorities for redressal. Thus, it can be used to address any type of complaint, from banal to urgent as well as bypass the long queues.

1. Identify the end user of the solution

End users are the citizens of India familiar with the internet.

1. Identify the functional requirements of the project

a. Login and registration from both the citizens’ end and department officials’ end.

b. Complaint form/window to file the complaint.

c. Forwarding the complaints to the respective department officials.

d. Citizens can review complaint status using unique complaint no.

e. Citizens close the complaint themselves along with a feedback form.

f. All complaints can be viewed by the user even after the complaint is closed.

1. Identify the **non-functional** requirements of the project

a. Complaint should be filed within 5 seconds.

b. Complaint should be forwarded to the respective department official within 60 seconds of filing.

c. The site should load within 3 seconds when the number of concurrent users exceeds 10000.

1. Identify the **feasibility** of the project (Economical feasibility)

a. It will allow Indian citizens to lodge complaints easily, and also allow them to track the status of their complaint.  
b. It will allow the govt officers to process complaints more efficiently.

1. Frame the final problem statement.

The Sarkar Sahayog Portal acts as an interface between the citizens and the government of India. The citizens can file their complaints on the portal, which will be forwarded to the respective department officials. The complaint status will be updated every 24 hours until the complaint is closed by the citizens themselves.

1. List down the list of user requirements, system requirements.

**User requirements:**

a. The portal should only be accessible to Indian citizens or government officials, taking verification in form of Aadhar number (for citizens), or employee ID (for government officials).

b. The portal should be publicly hosted on the Internet.

c. The portal would be accessible for English speakers.

d. There should be ability to create account and login both for users and officials.

e. The portal should show status of complaint redressal.

f. A complaint should not be closed until the user deems it resolved, or a lot of time (6 months) has passed in inactivity.

**System requirements:**

The system needs many terminals with web browsers that is HTML and database compatible. At the back-end, it must have a server that can handle high amounts of traffic and a robust, scalable database system.

1. Identify the ambiguities, inconsistencies, incompleteness from the requirements gathered.

**Ambiguity:**

a. If two people put forth the same complaint, and two different government officials are assigned to handle the complaints, they might have a different way of resolving the same complaint.

b. Each complaint has to be dealt with separately, even if they’re same.

**Inconsistency:**

a. The complaints will be classified under some predefined subcategories, and they will be forwarded to the respective department. But if a user writes a general complaint which does not belong to any particular category, there’ll be no department to forward it to.

b. A user might never close the complaint even after it is resolved.

**Incompleteness:**

a. A complaint written by the user might not be specific enough to interpret what exactly the user demands out of it.

1. List down the development plan for the selected problem statements. SDLC(Software development life cycle)

* Planning
* Analysis
* Design
* Development
* Testing
* Implementation

**B.2 Observations and Learning:**

We finalized the problem statement and conducted the requirements gathering.

**B.3 Conclusion:**

**After successful completion of this experiment we are able to**

1. Identify ambiguities, inconsistencies and incompleteness from a requirements specification
2. Identify and state functional requirements
3. Identify and state non-functional requirements
4. Frame the problem statement for the MINI project

**B.4 Question of curiosity:**

1. Justify the statement, *“User requirements must be understood well before beginning the software development”*

Having no official specifications means the following:

* There’s no clear understanding as to what constitutes a finished product or even feature.
* The client doesn’t know what to expect by the end of development and what they’re paying for.
* Developers are left hanging, having to figure out the specifics of features based on what was said and how they themselves understood it.
* The finished product may be buggy and full of errors.

1. What is SRS? Download sample SRS format of any organization and prepare the relevant part of the same for this experiment.

A software requirements specification (SRS) is a document that describes what the software will do and how it will be expected to perform. It also describes the functionality the product needs to fulfill all stakeholders (business, users) needs.

1. What is feasibility study? When is the feasibility study done?

A feasibility study is an assessment of the practicality of a proposed plan or project. A feasibility study analyzes the viability of a project to determine whether the project or venture is likely to succeed. The study is also designed to identify potential issues and problems that could arise from pursuing the project. Information such as resource availability, cost estimation for software development, benefits of the software to the organization after it is developed and cost to be incurred on its maintenance are considered during the feasibility study. The objective of the feasibility study is to establish the reasons for developing the software that is acceptable to users, adaptable to change and conformable to established standards. Various other objectives of feasibility study are listed below.

1. Differentiate between functional requirements and nonfunctional requirements.

|  |  |
| --- | --- |
| **Functional Requirements** | **Non-functional requirements** |
| Functional requirements help to understand the functions of the system. | They help to understand the system's performance. |
| Functional requirements are mandatory. | While non-functional requirements are not mandatory. |
| They are easy to define. | They are hard to define. |
| They describe what the product does. | They describe the working of product. |
| It concentrates on the user's requirement. | It concentrates on the expectation and experience of the user. |
| It helps us to verify the software's functionality. | It helps us to verify the software's performance. |

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*