**Experiment No-4**

**Title:** **Preparation of Software Requirements Specifications (SRS).**

**Batch: A3 Roll No.: 16010421119 Experiment No: 4**

**Aim: Preparation of Software Requirements Specifications (SRS).**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Resources needed: LaTeX Editor, Internet Browser \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Theory**

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 its 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:

**1. 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.

**2. Consistency:** To illustrate this, consider the automation of a nuclear plant. Suppose one of the clients say that it 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.

**3. Completeness:** A particular requirement for a system should specify what the system should do and also what it should not. For example, consider 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:

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

**2. 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:

**1. 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.

**2. 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.

**IEEE SRS document (Relevant IEEE standards: IEEE-830[5])**

The SRS is a specification of the requirements for the software “product” you will produce in your project. The basic issues to be addressed are:

a) Functionality. What is the software supposed to do?

b) External interfaces. How should the software interact with people, the operating system, hardware, networks, and other software?

c) Performance. What are the requirements for speed, availability, response time, recovery time of various software functions, etc.?

d) Quality Attributes. What are the requirements for portability, correctness, maintainability, security, etc?

e) Design constraints imposed on an implementation. Is there a requirement for a particular programming language? Are there resource limits (such as disk or memory size)? Must it run on a particular operating system? Must it inter-operate with particular web browsers? etc.

The SRS contains requirements, and not your design solutions; it is the “what” and not the “how” of your project. The information is collected from the project client. In a good SRS, the requirements should be: Correct, Unambiguous, Complete, Consistent, and Ranked for importance, Verifiable, and Traceable.

|  |
| --- |
| **1 INTRODUCTION**  1.1 Product Overview  Clearly define the purpose of the software, the environment it will run in, and who will be using it (in terms of their educational level, experience, and technical expertise). Do not go into detail, but outline the general requirements in a way that provides the reasons why specific requirements are later specified in the SRS.  **2 SPECIFIC REQUIREMENTS**  This section of the SRS should contain all of the software requirements to a level of detail sufficient to:  1. enable the SRS to be checked by the originator of the original system requirements (usually the project supervisor),  2. enable designers to design a system to satisfy those requirements, and  3. enable testers to test that the system satisfies those requirements.  Throughout this section, every stated requirement should be externally perceivable by users, or other external systems. This ensures that all features are testable.  **2.1 External Interface Requirements**  **2.1.1** User Interfaces  This should specify the following:  a) The characteristics of the user interface. Include the characteristics necessary to accomplish the software requirements (for example: required screen formats, page or window layouts, content of any reports or menus, or availability of programmable function keys).  b) All the aspects of optimizing the interface with the person who must use the system. This may simply comprise a list of do’s and don’ts on how the system will appear to the user. Like all others, these requirements should be verifiable.  **2.1.2 Hardware Interfaces**  This should specify the characteristics of each interface between the software and the hardware components of the system. This includes configuration characteristics (number of  ports, instruction sets, etc.). It also covers such matters as what devices are to be supported, how they are to be supported, and protocols. Only use this section if your project requires specific hardware to be used – do not, for example, specify the hardware if there is a requirement to establish a network connection.  **2.1.3 Software Interfaces**  This should specify the use of other required software which your software must interface with (for example: a database system, an operating system, or a mathematical package).  For each required software product, the following should be provided:  - Name and Version number.  For each interface, the following should be provided:  - Discussion of the purpose of the interfacing software as related to this software product.  - Definition of the interface in terms of message content and format. It is not necessary to detail any well-documented interface, but a reference to the document defining the interface is required.  **2.1.4 Communications Protocols**  This should specify the various interfaces to communications such as local network protocols, etc. Make reference to any well-defined protocols, specifying exactly which parts or options of that protocol the software needs to support.  **2.2 Software Product Features**  This section should consist of a numbered list of required features.  Typically, all of the requirements that relate to a software product are not equally important. The importance of each feature should be indicated using one of the following terms: essential, important, or desirable.  Each feature requirement should include:  ? ? a description of every input (stimulus) into the system,  ? ? a description of every output (response) from the system,  ? ? a description of every state change within the system,  ? ? a description of all the functions performed by the system in response to an input or in support of an output.  **Functional requirements** should define the actions that must take place in the software in accepting and processing the inputs and in processing and generating the outputs. These are generally listed as “shall” statements starting with “The system shall”. These include  a) Validity checks on the inputs  b) Exact sequence of operations  c) Responses to abnormal situations, including  1) Overflow  2) Communication facilities  3) Error handling and recovery  d) Effect of parameters  e) Relationship of outputs to inputs, including  1) Input/output sequences  2) Formulas for input to output conversion  It may be appropriate to partition the functional requirements into sub functions or sub processes. This does not imply that the software design will also be partitioned that way.  **2.3 Software System Attributes**  **2.3.1 Reliability**  Specify the required reliability of the final software system.  This is particular important for applications such as embedded software. It can be quantified using MTTF (Mean Time To Failure) measurements.  **2.3.2 Availability**  Specify the required availability of the final software system: define requirements such as check pointing, recovery, and restart.  **2.3.3 Security**  Specify the factors that protect the software from accidental or malicious access, use, modification, destruction, or disclosure.  Specific requirements in this area could include the need to  a) Utilize certain crypto graphical techniques;  b) Keep specific log or history data sets;  c) Assign certain functions to different modules;  d) Restrict communications between some areas of the program;  e) Check data integrity for critical variables.  **2.3.4 Maintainability**  This should specify attributes of the software that relate to the ease of maintenance of the software itself. Specify any requirements for certain modularity, interfaces, complexity, etc that make the software easier to maintain.  Requirements should not be placed here just because they are thought to be good design practices.  **2.3.5 Portability**  This should specify attributes of software that relate to the ease of porting the software to other host machines and/or operating systems. This may include the following:  a) Percentage of code that is host dependent;  b) Use of a proven portable language;  c) Use of a particular compiler or language subset;  d) Use of a particular operating system.  **2.3.6 Performance**  This subsection should specify both the static and the dynamic numerical requirements placed on the software or on human interaction with the software as a whole.  Static numerical requirements may include, for example, the following: minimum number of simultaneous users, minimum data storage, etc.  Dynamic numerical requirements may include, for example, the required number of transactions per second for both normal and peak workload conditions, etc.  All of these requirements should be stated in measurable terms.  **2.4 Database Requirements**  This should specify the logical requirements for any information that is to be placed into a database.  This may include the following:  a) Types of information used by various functions;  b) Accessing capabilities;  c) Data entities and their relationships;  d) Integrity constraints. |

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Activities:**

Prepare SRS document for chosen problem definition in LaTeX.

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Results:** SRS Document in given format

\documentclass{article}

\usepackage[utf8]{inputenc}

\title{Software Requirements Specification (SRS)\\for Remote Wildlife Monitoring and Conservation Application}

\author{ Dhairya Satra - 16010421091\\Aarya Tiwari - 16010421119}

\date{31 january 2024}

\begin{document}

\maketitle

\section{Introduction}

\subsection{Product Overview}

The Remote Wildlife Monitoring and Conservation Application is designed to address the increasing threats to wildlife habitats, biodiversity loss, and illegal activities. The application leverages technology, including drones, cameras, sensors, and data analytics, to provide an integrated solution for wildlife monitoring, conservation management, research, education, and stakeholder engagement. This SRS outlines the specific requirements for the development of the application.

\section{Specific Requirements}

\subsection{External Interface Requirements}

\subsubsection{User Interfaces}

a) The user interface shall allow users to display live drone footage, images, and sensor readings from selected wildlife habitats, protected areas, and critical zones.\\

b) Users should be able to navigate, zoom, pan, and interact with aerial maps, layers, markers, and annotations.

\subsubsection{Hardware Interfaces}

This application does not have specific hardware requirements.

\subsubsection{Software Interfaces}

The application shall interface with drone control software for live footage and sensor data.\\

The application shall integrate with external databases for storing and retrieving historical data.\\

It shall be compatible with standard Geographic Information Systems (GIS) for spatial data management.

\subsubsection{Communications Protocols}

The application shall support local network protocols for seamless communication with drones and external databases.

\subsection{Software Product Features}

\subsubsection{Essential Features}

1. The system shall display live drone footage, images, and sensor readings.\\

2. Users shall be able to navigate and interact with aerial maps.\\

3. Historical data, trends, patterns, alerts, and notifications shall be accessible.\\

4. Users shall configure settings, filters, and preferences for data visualization.

\subsubsection{Important Features}

1. The system shall provide data analysis and visualization functionalities.\\

2. Alerts shall be generated for habitat changes, wildlife sightings, and environmental threats.\\

3. The application shall support collaborative data sharing and reporting.

\subsubsection{Desirable Features}

1. Integration with external conservation databases for comprehensive data analysis.\\

2. Support for augmented reality features for enhanced data visualization.

\subsection{Software System Attributes}

\subsubsection{Reliability}

The application shall have a reliability rate of 99.9\%, measured by Mean Time To Failure (MTTF).

\subsubsection{Availability}

The system shall incorporate check pointing, recovery, and restart mechanisms to ensure 24/7 availability.

\subsubsection{Security}

The application shall employ cryptographic techniques to protect data.\\

Access to critical modules and data communication shall be restricted based on user roles.

\subsubsection{Maintainability}

The software shall be designed with modular structures to facilitate ease of maintenance.\\

Regular updates and patches shall be provided for continuous improvement.

\subsubsection{Portability}

The application shall be designed for portability across different operating systems.

\subsubsection{Performance}

The system shall support a minimum of 100 simultaneous users.\\

Dynamic requirements shall include a minimum of 50 transactions per second for normal workload conditions.

\subsection{Database Requirements}

The application shall use a relational database to store information related to habitat monitoring, wildlife sightings, and environmental assessments.\\

Data entities and their relationships shall be defined to maintain integrity.

This Software Requirements Specification outlines the necessary details for the development of the Remote Wildlife Monitoring and Conservation Application. It provides a foundation for system design, development, and testing to ensure the successful implementation of the proposed solution.

\end{document}

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

**Questions:**

1. **Explain various steps involved in Requirement Engineering.**

Requirement Engineering is a systematic process that involves eliciting, analyzing, documenting, validating, and managing requirements for a software system. The main steps involved in Requirement Engineering are as follows:

**Requirement Elicitation:**

Gather requirements from stakeholders using interviews, surveys, and workshops.

**Requirement Analysis:**

Analyze and document requirements, identify conflicts, and refine for clarity.

**Requirement Specification:**

Document requirements (functional/non-functional) using use cases and specifications.

**Requirement Validation:**

Validate requirements for accuracy and correctness through reviews and inspections.

**Requirement Management:**

Manage changes systematically through baselining and change control processes.

**Requirement Traceability:**

Establish traceability links between different requirement levels for better management.

**Requirement Prioritization:**

Prioritize requirements based on importance, criticality, and dependencies.

**Prototyping:**

Create prototypes to visualize and validate user interface-related requirements.

**Requirement Documentation:**

Document requirements comprehensively for a common understanding among stakeholders.

**Communication and Collaboration:**

Facilitate effective communication and collaboration for a shared understanding.

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Outcomes:**

**CO3: Demonstrate requirements, modeling and design of a system.**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Conclusion: (Conclusion to be based on the outcomes achieved)**

In this experiment we learnt, the Requirement Engineering process is vital for successful software development. These streamlined processes contribute to the development of a high-quality software product aligned with stakeholder needs.

Also learnt how to prepare an SRS document.

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

**Grade: AA / AB / BB / BC / CC / CD /DD**

**Signature of faculty in-charge with date**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**References:**

**Books:**

1. Roger S. Pressman, Software Engineering: A practitioners Approach, 7th Edition, McGraw Hill, 2010.
2. Technical report on Guidelines for Documents Produced by Student Projects In Software Engineering based on IEEE standards
3. https://www.sharelatex.com/