**Experiment No. 2**

**SEMESTER:** V (2024-2025) **DATE OF DECLARATION:** 02-09-24

**SUBJECT:** SE **DATE OF SUBMISSION:** 9-09-24

**NAME OF THE STUDENT:** Shaun Menezes **ROLL NO:** 40

| | |
|---|---|
| **AIM** | To prepare a SRS (Software Requirement Specification) document for "Inefficiency in Timely Announcement of Holidays for Educational Institutions in Flood-Prone Areas" |
| **LEARNING OBJECTIVE** | The student will describe and use components of standard IEEE SRS document. |
| **LEARNING OUTCOME** | The students will be able identify software requirement specification and formulate it for for the selected case study. |
| **COURSE OUTCOME** | **CSL601.2**: Students will be able identify software requirement specification and formulate it for for the selected case study. |
| **PROGRAM OUTCOME** | **PO1**: Apply the knowledge of mathematics, science, engineering fundamentals, and an engineering specialization to the solution of complex engineering problems.<br><br>**PO2**: Identify, formulate, review research literature and analyze complex engineering problems reaching substantial conclusions using first principal of mathematics, natural science and engineering science.<br><br>**PO9**: Function effectively as an individual, and as a member or leader in diverse teams, and in multidisciplinary settings.<br><br>**PO10**: Communicate effectively on complex engineering activities with the engineering community and with society at large, such as, being able to comprehend and write effective reports and design documentation, make effective presentations, and give and receive clear instructions. |
| **BLOOM'S TAXONOMY LEVEL** | Understand<br><br>Apply |
| **THEORY** | **Requirements**<br><br>Sommerville 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 its requirements. If we do not have a clear vision of what is to be developed and what all features are expected, we would be faced with problems down the road leading 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 about what a system to be developed should do. For example, consider that 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 is the "enough number of people"? That could mean 10 to you, but perhaps 100 to the client. This is an example of ambiguity.

2. Consistency: To illustrate this, consider the automation of a nuclear plant. Suppose that one of the clients say that if the radiation level inside the plant exceeds R1, all reactors should be shut down. However, another person 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 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 customers and analysts can verify that 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 can typically define how the system should behave under certain situations. For example, an NFR can say that the system should work with 128MB RAM. Under such conditions, an NFR can become more critical than an FR.

Non-functional requirements can be further classified into different types, such as:

- **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 can be identified by focusing on the following points:

- Identify the high level functional requirements 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/her 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 provides the title of the inteded book as an input and gets the book details and location as output.
- Any high level requirement identified could have different sub-

requirements. For example, the "Issue Book" module can behave differently for different categories of users or for a particular user who has issued the book thrice consecutively.

**Preparing Software Requirements Specifications**

Once all possible FRs and non-FRs have been identified, which are complete, consistent, and non-ambiguous, the Software Requirements Specification (SRS) is prepared.

IEEE provides a template also available on Moodle course for software engineering , which can be used for this purpose. The SRS is prepared by the service provider and verified by its client. This document serves as a legal agreement between the client and the service provider. Once the concerned system has been developed and deployed, if a proposed feature was not found to be present in the system, the client can point this out from the SRS. Also, if after delivery, the client says a new feature is required, which was not mentioned in the SRS, the service provider can again point to the SRS. The scope of the current experiment, however, does not cover writing a SRS.

**LAB EXERCISE**

# Table of Contents

## Table of Contents for a SRS Document



# 1. INTRODUCTION

### 1.1 PURPOSE

The purpose of this document is to develop an online system that addresses the inefficiency in timely announcements of holidays for educational institutions located in flood-prone areas. The system aims to automate the announcement process, ensuring students, parents, and staff receive timely alerts during potential flood situations.

### 1.2 DOCUMENT CONVENTIONS

This document uses the following conventions.

| | |
|---|---|
| SMS | Short Message Service |
| API | Application Programming Interface |
| UI | User Interface |
| DBMS | Database Management System |

### 1.3 INTENDED AUDIENCE AND READING SUGGESTIONS

- Educational Institutions: To understand how the system will enhance communication and decision-making during floods.
- Government Authorities: For overseeing and managing flood-related notifications.
- Developers: To implement the system based on detailed requirements.
- Testers: To create test cases and validate the system.
- Parents and Students: To comprehend how the system will benefit them in receiving timely notifications.

### 1.4 PROJECT SCOPE

The project aims to design a system that efficiently disseminates holiday announcements to educational institutions in flood-prone areas. The system will integrate real-time weather forecasts and government notifications to automate holiday announcements. The system will use various communication channels, such as SMS, email, and web portals, to ensure timely dissemination of information.

### 1.5 REFERENCES

- National Weather Forecasting APIs
- Government Disaster Management Guidelines
- School Emergency Communication Plans

# 2. OVERALL DESCRIPTION

### 2.1 PRODUCT PERSPECTIVE

The system will act as a centralized platform that integrates weather alerts and government notifications with educational institutions. It will use automated decision-making algorithms to determine when to announce holidays based on predefined thresholds for flood risks.

## 2.2 PRODUCT FEATURES

- Real-time Flood Alert Integration: Connects with weather services to monitor flood risks in real-time.
- Automated Announcement: Automatically announces holidays based on set flood risk parameters.
- Multi-Channel Communication: Sends notifications through SMS, email, and web portals.
- Admin Dashboard: Allows manual overrides of automated announcements by authorized personnel.

## 2.3 USER CLASS and CHARACTERISTICS

- Institution Administrators: Manage and receive holiday announcements.
- Government Officials: Monitor and regulate flood alerts.
- IT Support Staff: Provide technical support for the system.
- Parents and Students: Receive notifications and alerts.

## 2.4 OPERATING ENVIRONMENT

The system will be cloud-based, accessible via standard web browsers and mobile devices. It will run on distributed server architecture to ensure high availability and reliability.

## 2.5 DESIGN and IMPLEMENTATION CONSTRAINTS

- **Data Privacy Compliance**: The system must comply with government regulations on data privacy.
- **API Call Limits**: Integration with third-party weather services may be constrained by API quotas.
- **Scalability**: The system must handle simultaneous notifications to multiple institutions without delays.

## 2.6 ASSUMPTION DEPENDENCIES

- Reliable Data Source: Assumes access to accurate and timely weather forecasts.
- Communication Infrastructure: Depends on the availability of internet and mobile networks to send notifications.
- Institutional Infrastructure: Assumes institutions are equipped to receive and respond to alerts.

# 3. SYSTEM FEATURES

### 3.1 FUNCTIONAL REQUIREMENTS

- **Flood Alert Monitoring**: Continuously monitors flood alerts.
- **Automated Decision-Making**: Uses algorithms to decide on holiday announcements.
- **Notification Dispatch**: Sends notifications via SMS, email, and web portals.
- **Manual Override**: Allows authorities to manually announce holidays.

# 4. EXTERNAL INTERFACE REQUIREMENTS

### 4.1 USER INTERFACES

- **Web Portal**: User-friendly interface for administrators and authorities to manage alerts.
- **Mobile Interface**: Optimized mobile interface for on-the-go access.
- **Notification System**: Interfaces with email and SMS gateways.

### 4.2 HARDWARE INTERFACES

- **Standard Devices**: Supports desktops, laptops, smartphones, and tablets.
- **Servers**: Utilizes cloud servers for database storage and processing.

### 4.3 SOFTWARE INTERFACES

- **Weather APIs**: Connects with third-party weather services.
- **Government Databases**: Interfaces with government systems for authentication and data access.
- **SMS/Email Gateways**: Sends notifications through external communication services.

### 4.4 COMMUNICATION INTERFACES

- **HTTPS**: Ensures secure web communication.
- **SMTP/HTTP**: Handles email and SMS notifications.

# 5. NONFUNCTIONAL REQUIREMENTS

### 5.1 PERFORMANCE REQUIREMENTS

- **Real-time Processing**: Alerts and notifications should be processed within 5 minutes of receiving flood data.
- **Scalability**: Supports up to 10,000 concurrent users and simultaneous notifications.

### 5.2 Safety Requirements

- **Data Integrity**: Ensures only verified and accurate flood data triggers announcements.
- **Fail-Safe Mechanisms**: Prevents unnecessary announcements during system failures.

### 5.3 Security Requirements

- **Role-Based Access Control**: Ensures only authorized users can manage and announce holidays.
- **Data Encryption**: Protects data transmission using TLS encryption.

### 5.2 SAFETY REQUIREMENTS

If there is extensive damage to a wide portion of the database due to catastrophic failure, such as a disk crash, the recovery method restores a past copy of the database that was backed up to archival storage (typically tape) and reconstructs a more current state by reapplying or redoing the operations of committed transactions from the backed up log, up to the time of failure.

### 5.3 SECURITY REQUIREMENTS

Security systems need database storage just like many other applications. However, the special requirements of the security market mean that vendors must choose their database partner carefully.

### 5.4 SOFTWARE QUALITY ATTRIBUTES

- **Reliability**: The system should maintain 99.9% uptime.
- **Usability**: Intuitive interfaces with minimal training required.
- **Scalability**: Designed to accommodate increasing data and users.
- **Maintainability**: Modular and well-documented codebase for easy maintenance and updates.

<table>
<tr><td></td><td>Group Details
<table>
<tr><td>Sr. No.</td><td>Roll No.</td><td>Name of the Batch</td><td>Name of the Student</td></tr>
<tr><td>1.</td><td>34</td><td>B</td><td>Aibal Biju</td></tr>
<tr><td>2.</td><td>35</td><td>B</td><td>Ramya Kulkarni</td></tr>
<tr><td>3.</td><td>40</td><td>B</td><td>Shaun Menezes</td></tr>
</table></td></tr>
<tr><td>REFERENCES</td><td>4. Ian Somerville, Software Engineering, 9th edition, Addison Wesley, 2011<br>5. http://vlssit.iitkgp.ernet.in/isad/isad/1/theory/<br>6. http://www.srmuniv.ac.in/sites/default/files/files/SOFTWARE%20Engineering<br>7. %20LAB-CS0411.pdf<br>8. https://web.cs.dal.ca/~hawkey/3130/srs_template-ieee.doc</td></tr>
</table>