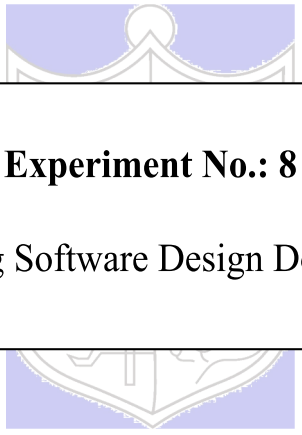

Experiment No.: 8

Title: Preparing Software Design Document (SDD)

Batch: B–1 **Roll No.: 16010422234** **Experiment No.: 8**

Aim: To prepare Software Design Document (SDD)

Resources needed: Internet Explorer, LaTex Editor

Theory

This system test document specifies the tests for the entire software system, defines the test schedule, and records the test results. This document does not cover unit testing (the testing of individual sub-systems or components of the system).

The system may consist of multiple items that are to be tested separately. The system may be tested in one or more increments of functionality; the system test document should cover each version of the system separately. When different versions of the system are tested, make sure to clearly identify the version of the software and the relevant test and test result. Also, extend the unique identifier scheme to include the version of the software system under test (SUT) – for example, use TC to identify test case for software system version.

Software Test Document (STD) Template:

1 INTRODUCTION

1.1 System Overview

Briefly detail the software system and items to be tested. Identify the version(s) of the software to be tested.

1.2 Test Approach

Describe the overall approach to testing. For each major group of features or feature combinations, specify the approach that will ensure that these feature groups are adequately tested. Specify the major activities, techniques, and tools that are used to test the designated groups of features. The approach should be described in sufficient detail to permit

identification of the major testing tasks and estimation of the time required to do each one. Identify significant constraints on testing, such as deadlines.

2 TEST PLAN

Describe the scope, approach, resources, and schedule of the testing activities. Identify the items being tested, the features to be tested, the testing tasks to be performed, the personnel responsible for each task in the case of a group project.

2.1 Features to be tested

Identify all software features and combinations of software features to be tested. Identify the test case(s) associated with each feature and
each combination of features. Identify the version of the software to be tested.
When multiple versions of the software are tested in a planned, incremental
manner, then use section numbers 2.1.n to identify the features to be tested for each version.

2.2 Features not to be tested

Identify all features and significant combinations of features that will not be tested and the reasons for not doing so.

2.3 Testing Tools and Environment

Specify test staffing needs. For an individual project, specify the time to be spent on testing. For a group project, specify the number of testers and the time needed.

Specify the requirements of the test environment: space, equipment, hardware, software, special test tools. Identify the source for all needs that are not currently available.

3 TEST CASES

A test case specification refines the test approach and identifies the features to be covered by the case. It also identifies the procedures required to

accomplish the testing and specifies the feature pass/fail criteria. It documents the actual values used for input along with the anticipated outputs.

If an automated test tool is to be used:

1. document each test case here as a specification for the test tool;

2. document the procedure that must be followed to use the test tool.

3.n Case-n (use a unique ID of the form TC-nnnn for this heading)

3.n.1 Purpose

Identify the version of the software and the test items, and describe the features and combinations of features that are the object of this test case. For each feature, or feature combination, a reference to its associated requirements in the software requirement specification (SRS) should be included.

3.n.2 Inputs

Specify each input required to execute the test case. Some of the inputs will be specified by value (with tolerances where appropriate), while others, such as files or URLs, will be specified by name. Specify all required relationships between inputs (e.g., ordering of the inputs).

3.n.3 Expected Outputs & Pass/Fail criteria

Specify all of the expected outputs and features (e.g., response time) required of the test items. Provide the exact value (with tolerances where appropriate) for each required output or feature. Specify the criteria to be used to determine whether each test item has

passed or failed testing. If an automated test tool is used, identify how the results of that tool are to be analysed.

3.n.4 Test Procedure

Detail the test procedure(s) needed to execute this test case. Describe any special constraints, such as: special set up, operator intervention, output determination procedures, and special wrap up.

4.Test Logs

A test log is used by the test team to record what occurred during test execution. A.n

Log for test-n (use a unique ID of the form TL-nnnn for this heading)

5.Test Results

For each execution, record the date/time and observed results (e.g., error messages generated). Also record the location of any output (e.g., window on the screen). Record the successful or unsuccessful execution of the test.

Incident Report (add a unique ID of the form TIR-nnnn to this heading) If the test failed, or passed with some unusual event, fill in this incident
report with the details. Summarize the incident, identifying the test items involved, and the anomaly in the results. Indicate what impact this
incident will have on the project.

Procedure:

1. Prepare Test plan
2. Give descriptions of a minimum five test cases.
3. Prepare Test case Table for test cases.

Results: STD document in given format

```
\documentclass[a4paper,12pt]{article}

\usepackage{geometry}

\usepackage{graphicx}

\usepackage{hyperref}

\geometry{margin=1in}

\title{\textbf{Software Test Document (STD)}}

\author{Generating VR Visualization from Textual Description of Numerical Data}

\date{\today}

\begin{document}

\maketitle

\section{Introduction}

\subsection{System Overview}

This document details the software testing plan for the system responsible for generating VR visualizations from textual descriptions of numerical data. The system takes numerical data as input, processes it through AI-driven text analysis, and generates interactive VR representations. The software version under test is v1.0.

\subsection{Test Approach}

The testing approach includes functional testing, integration testing, and system testing. Automated testing tools and manual validation will be used to verify the correctness of the VR visualizations. Performance metrics such as rendering time, accuracy of visual representation, and interaction responsiveness will be assessed.

\section{Test Plan}

\subsection{Features to be tested}

\begin{itemize}
```

```
    \item Data extraction from textual descriptions.

    \item AI-based data processing and categorization.

    \item VR visualization rendering.

    \item User interaction within the VR environment.

\end{itemize}

\subsection{Features not to be tested}

AI model training and external API response time will not be tested in
this phase.

\subsection{Testing Tools and Environment}

\begin{itemize}

    \item Hardware: High-performance GPU-enabled system.

    \item Software: Unity 3D, OpenAI API, Python.

    \item Tools: SonarQube, Selenium, JUnit.

\end{itemize}

\section{Test Cases}

\subsection{TC-0001: Data Processing Test}

\textbf{Purpose:} Validate the accuracy of text-to-data processing.

\textbf{Inputs:}

\begin{itemize}

    \item Textual description: "Sales data for Q1 2025 shows revenue
growth of 15\%."

\end{itemize}

\textbf{Expected Outputs \& Pass/Fail Criteria:}

\begin{itemize}

    \item Extracted numerical data: Q1 2025, 15\% revenue growth.

    \item If correctly extracted, the test passes.

\end{itemize}
```

```
\textbf{Test Procedure:}

\begin{enumerate}

    \item Input textual data.

    \item Run data extraction module.

    \item Compare extracted values with expected output.

\end{enumerate}

\subsection{TC-0002: VR Rendering Test}

\textbf{Purpose:} Ensure correct visualization of data in a VR
environment.

\textbf{Inputs:}

\begin{itemize}

    \item Processed numerical data from previous test.

\end{itemize}

\textbf{Expected Outputs \& Pass/Fail Criteria:}

\begin{itemize}

    \item 3D pie chart representation in VR.

    \item Proper scaling and color differentiation.

    \item If visualization matches expected format, the test passes.

\end{itemize}

\textbf{Test Procedure:}

\begin{enumerate}

    \item Load extracted data into the visualization module.

    \item Render visualization in VR.

    \item Verify correctness of the generated VR representation.

\end{enumerate}

\section{Test Logs}
```

```
\subsection{TL-0001: Log for TC-0001}

\textbf{Date:} \today \\

\textbf{Observed Results:} Extracted numerical data correctly. Test
Passed.

\subsection{TL-0002: Log for TC-0002}

\textbf{Date:} \today \\

\textbf{Observed Results:} VR visualization loaded correctly. Test Passed.

\section{Test Results}

All tests executed successfully. No critical failures detected.

\section{Conclusion}

The software system was successfully tested for core functionalities such
as data processing, VR visualization, and user interactions. All tests met
expected results, ensuring the system performs as intended.

\end{document}
```

Software Test Document (STD)

Generating VR Visualization from Textual Description of Numerical Data

March 24, 2025

1 Introduction

1.1 System Overview

This document details the software testing plan for the system responsible for generating VR visualizations from textual descriptions of numerical data. The system takes numerical data as input, processes it through AI-driven text analysis, and generates interactive VR representations. The software version under test is v1.0.

1.2 Test Approach

The testing approach includes functional testing, integration testing, and system testing. Automated testing tools and manual validation will be used to verify the correctness of the VR visualizations. Performance metrics such as rendering time, accuracy of visual representation, and interaction responsiveness will be assessed.

2 Test Plan

2.1 Features to be tested

- Data extraction from textual descriptions.
- AI-based data processing and categorization.
- VR visualization rendering.
- User interaction within the VR environment.

2.2 Features not to be tested

AI model training and external API response time will not be tested in this phase.

2.3 Testing Tools and Environment

- Hardware: High-performance GPU-enabled system.
- Software: Unity 3D, OpenAI API, Python.
- Tools: SonarQube, Selenium, JUnit.

3 Test Cases

3.1 TC-0001: Data Processing Test

Purpose: Validate the accuracy of text-to-data processing.

Inputs:

- Textual description: "Sales data for Q1 2025 shows revenue growth of 15%."

Expected Outputs & Pass/Fail Criteria:

- Extracted numerical data: Q1 2025, 15% revenue growth.
- If correctly extracted, the test passes.

Test Procedure:

1. Input textual data.
2. Run data extraction module.
3. Compare extracted values with expected output.

3.2 TC-0002: VR Rendering Test

Purpose: Ensure correct visualization of data in a VR environment.

Inputs:

- Processed numerical data from previous test.

Expected Outputs & Pass/Fail Criteria:

- 3D pie chart representation in VR.
- Proper scaling and color differentiation.
- If visualization matches expected format, the test passes.

Test Procedure:

1. Load extracted data into the visualization module.
2. Render visualization in VR.
3. Verify correctness of the generated VR representation.

4 Test Logs

4.1 TL-0001: Log for TC-0001

Date: March 24, 2025
Observed Results: Extracted numerical data correctly. Test Passed.

4.2 TL-0002: Log for TC-0002

Date: March 24, 2025
Observed Results: VR visualization loaded correctly. Test Passed.

5 Test Results

All tests executed successfully. No critical failures detected.

6 Conclusion

The software system was successfully tested for core functionalities such as data processing, VR visualization, and user interactions. All tests met expected results, ensuring the system performs as intended.

Questions:

1. **Differentiate between verification and validation.**
   - **Verification** ensures that the software meets the specified requirements before it is developed (static testing). It involves activities like reviews, walkthroughs, and inspections.
   - **Validation** ensures that the developed software meets the user's needs and expectations (dynamic testing). It involves activities like functional testing, system testing, and acceptance testing.

2. **List down all OO software testing strategies.**
   - Unit Testing
   - Integration Testing
   - System Testing
   - Regression Testing
   - Acceptance Testing
   - Mutation Testing
   - State-based Testing

Outcomes: CO4 — Demonstrate test case design

Conclusion:

The preparation of a Software Test Document (STD) helped in understanding the importance of systematic testing in software development. The test cases, test plan, and test logs ensure that the software meets the functional and non-functional requirements. By conducting rigorous testing, defects and anomalies can be identified early, improving the reliability and quality of the final product.

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. Ian Somerville, Software Engineering, 9th edition, Addison Wesley, 2011.
3. http://vlabs.iitkgp.ernet.in/se/
