**Practical No-08: Designing Test Suites | Software Testing**

**1. Aim of the Experiment**

To design and implement a test suite for the **IPCRA (Interview Preparation Chatbot with Resume Analyzer)** system that thoroughly verifies its functionality and performance, identifies defects, and ensures the system meets its specified requirements and quality standards.

**2. Introduction**

Software testing is a vital phase in the Software Development Life Cycle (SDLC) that ensures the quality, correctness, and reliability of a software product. For a system like IPCRA, which supports interview preparation and resume analysis, robust testing is essential to guarantee accurate feedback, secure data handling, and a seamless user experience. This practical focuses on understanding different testing types and designing effective test cases for IPCRA.

**3. Objectives**

After completing this experiment, you will be able to:

* Learn about different techniques of testing a software.
* Design unit test cases to verify the functionality and locate bugs, if any.

**4. Theory**

**a. Software Testing** Software testing is the process of executing a program with the intent of finding errors. It ensures that the software product works as expected, meets the user's requirements, and is free from defects.

**b. Verification, Validation**

* **Verification:** "Are we building the product right?" It's a static analysis method that checks if the software confirms to specifications and established standards
* **Validation:** "Are we building the right product?" It's a dynamic analysis method that checks if the software meets the user's actual needs and expectations

**c. Standards for Software Test Documentation** Effective testing requires clear documentation. Key documents include:

* **Test Plan: Scope, approach, resources, schedule.**
* **Test Cases: Steps, input, expected results.**
* **Test Scenarios: End-to-end system paths.**
* **Test Reports: Summary of execution and defects.**

**d. Testing Frameworks**

Frameworks used for IPCRA testing may include:

* **Pytest** (Python-based backend)
* **Selenium** (Web automation for UI)
* **Unittest** (Standard Python testing)

**e. Need for Software Testing**

* **Quality Assurance:** Ensures the product is high-quality.
* **Bug Detection:** Identifies defects early in the cycle.
* **Security:** Safeguards the system against vulnerabilities.
* **Performance:** Checks system responsiveness and stability under load.
* **Customer Satisfaction:** Delivers a reliable and functional product.
* **Cost Reduction:** Fixing bugs earlier is significantly cheaper.

**f. Test Cases and Test Suites**

* **Test Case:** A set of conditions under which a tester will determine whether a system under test satisfies requirements or works correctly. It includes test steps, input data, and expected output.
* **Test Suite:** A collection of test cases that are grouped together to test a particular functional area or an entire application.

**g. Types of Software Testing**

* **Unit Testing:** Tests individual components or modules in isolation.
* **Integration Testing:** Tests how different modules interact with each other.
* **System Testing:** Tests the complete and integrated software system.
* **Acceptance Testing:** Formal testing conducted to determine if the system satisfies acceptance criteria and enables users to perform their tasks.
* **Performance Testing:** Evaluates system responsiveness, stability, scalability, and resource usage under various workloads.
* **Security Testing:** Identifies vulnerabilities and weaknesses in the system.
* **Usability Testing:** Evaluates how easy and user-friendly the system is for end-users.

**h. Unit Testing, Integration Testing, System Testing** (Detailed)

* **Unit Testing: Example — testing analyzeResume() function for correct keyword extraction.**
* **Integration Testing: Example — testing interaction between chatbotResponse() and resumeFeedback() modules.**
* **System Testing: Example — full flow from user login, resume upload, chatbot interaction, and feedback generation.**

**5. Write Test Cases (Formal specified CASE study tab on V-Lab)**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **A. Unit Test Cases (Resume Analysis Module)** | | | | | | |
|  |
|  |
| **Test Case ID** | **Feature to Test** | **Test Steps** | **Input Data** | **Expected Output** | **Actual Output** | **Status** |  |
| UTC\_RES\_001 | analyzeResume() with valid resume | Call analyzeResume() | Resume with standard format | Extracted keywords and score | Extracted keywords and score | Pass |  |
| UTC\_RES\_002 | analyzeResume() with missing sections | Call analyzeResume() | Resume missing experience section | Warning or low score | Warning or low score | Pass |  |
| UTC\_RES\_003 | analyzeResume() with empty file | Call analyzeResume() | Empty document | Error message | Error message | Pass |  |
| **B.Integration Test Cases (Chatbot and Resume Modules)** | | | | | | |  |
|  |
|  |
|  |
| **Test Case ID** | **Feature to Test** | **Test Steps** | **Input Data** | **Expected Output** | **Actual Output** | **Status** |  |
| ITC\_CHAT\_RES\_001 | Resume upload and feedback generation | 1. Upload resume 2. Chatbot analyzes and responds | Valid resume file | Feedback with score and suggestions | Feedback with score and suggestions | Pass |  |
| ITC\_CHAT\_RES\_002 | Chatbot response to invalid resume | 1. Upload corrupted file 2. Chatbot processes | Corrupted resume file | Error response | Error response | Pass |  |
|  | | | | | | |  |
|  |
|  |
|  |
|  |
|  |
| **C. System Test Cases (End-to-End Flow)** | | | | | | |  |
|  |
|  |
| **Test Case ID** | **Feature to Test** | **Test Steps** | **Input Data** | **Expected Output** | **Actual Output** | **Status** |  |
| STC\_E2E\_001 | Full user flow | 1. User logs in 2. Uploads resume 3. Interacts with chatbot 4. Receives feedback | Valid credentials, resume file | Feedback generated and displayed | Feedback generated and displayed | Pass |  |
| STC\_SEC\_001 | Unauthorized access to resume analysis | 1. User not logged in 2. Accesses /analyze | No credentials | Redirect to login or access denied | Redirect to login or access denied | Pass |  |

**6. References**

1. **Pressman, R. S., & Maxim, B. R. (2020)** – *Software Engineering: A Practitioner's Approach* (9th ed.), McGraw-Hill Education
   * Covers SDLC, testing strategies, and documentation standards.
2. **Sommerville, I. (2016)** – *Software Engineering* (10th ed.), Pearson
   * Offers insights into verification, validation, and system-level testing.
3. **Myers, G. J., Sandler, C., & Badgett, T. (2011)** – *The Art of Software Testing* (3rd ed.), Wiley
   * Focuses on test case design, test planning, and defect detection.
4. **Jain, S. (2009)** – *Software Testing: Principles and Practices*, Oxford University Press

* Explains testing types, frameworks, and quality assurance techniques.