LAB REPORT : UNIT TESTING REPORT

COURSE CODE: CSE 404
COURSE TITLE: SOFTWARE ENGINEERING AND ISD LABRATORY

Submitted by

Suraiya Mahmuda (364)

Submitted to

Dr. Md. Mushfique Anwar, Professor

Dr. Md. Humayun Kabir, Professor

Computer Science and Engineering

Jahangirnagar University

Dhaka, Bangladesh

January 07, 2025

Contents

1 Introduction **1**

2 The Testing Tool : Pytest **1**

3 What is Unit Testing **2**

4 Sample Unit Tests During Sprints **2**

4.0.1 Test Request Accommodation Success 3

4.0.2 Test Request Accommodation Missing Document 3

4.0.3 Test Review Accommodation Success 3

4.0.4 Test Review Accommodation Missing Comments 4

5 Pros and Cons of Unit Testing **4**

6 Recommendations for Effective Unit Testing **5**

1. Introduction

Unit testing is a software development practice aimed at validating the functionality of individual components or units of code in isolation. By incorporating unit testing, developers can confirm that each unit behaves as intended, reducing bugs and enhancing overall software quality. This report highlights the importance of unit testing in the development of the **"JU Exam Office Management System"** project, with a focus on its role during two pivotal sprints.

2. The Testing Tool : Pytest

Testing Framework: Pytest

For this project, we utilized **Pytest**, a powerful and versatile testing framework for Python. Pytest offers the following features:

- **Ease of Use:** Minimal setup and straightforward syntax for writing tests.
- **Fixture Support:** Simplifies test setup by reusing common configurations.
- **Advanced Assertions:** Provides expressive and detailed assertion capabilities.
- **Plugins and Extensions:** Allows integration with additional tools for enhanced functionality.

By leveraging Pytest, we streamlined the unit testing process, ensuring rapid develop-

ment and reliable test coverage across all project modules.

3. What is Unit Testing

Unit testing is a software testing technique that involves testing individual components or units of a program in isolation to ensure they function as expected. A ”unit” typically refers to the smallest testable part of an application, such as a function, method, or class.

In the **”JU Exam Office Management System”** project, we implemented unit testing to validate critical functionalities during both sprints.

4. Sample Unit Tests During Sprints

Sprint 1: Apply for Marksheet Feature

Objective

To enable students to apply for their certificates while ensuring eligibility criteria, such as library and hall clearance, are met.

Key Test Scenarios

- Verifying that authorized students can access the application form.
- Ensuring that the system blocks ineligible students due to missing clearances.
- Validating the behavior when mandatory fields are left blank.

Sprint 2: Approve Application for Physically Disabled and Sick Students

Objective

It has the workings to manage student applications for accommodations due to disability or illness.

Key Test Scenarios

4.0.1 Test Request Accommodation Success

Objective: Tests that a student can successfully submit an accommodation request.

4.0.2 Test Request Accommodation Missing Document

Objective: Tests submitting a request with a missing required document (failure).

4.0.3 Test Review Accommodation Success

Objective: Tests the department successfully reviewing an accommodation request.

4.0.4 Test Review Accommodation Missing Comments

Objective: Tests reviewing a request with missing required comments (failure).

5. Pros and Cons of Unit Testing

Pros

Early Bug Detection:

- Identifies issues in individual units early in the development process, reducing the cost and effort needed to fix them later.

Improved Code Quality:

- Encourages writing clean, modular, and maintainable code as units need to be easily testable.

Facilitates Refactoring:

- Unit tests act as a safety net, allowing developers to confidently refactor or enhance code without fear of breaking functionality.

Supports Continuous Integration:

- Automated unit tests ensure that new code integrates smoothly with existing functionality in CI pipelines.

Faster Debugging:

- When a unit test fails, it directly points to the part of the code that needs attention, simplifying the debugging process.

Documentation:

- Tests can serve as documentation, providing insights into how different parts of the system are expected to behave.

Reusable Test Cases:

- Unit tests can often be reused as part of integration or system testing, saving time and effort in the long run.

Cons

Time-Consuming Initial Setup:

- Writing comprehensive unit tests requires significant initial effort, especially for large or legacy systems.

Limited Scope:

- Unit tests only verify individual components in isolation, not their interactions with other parts of the system.

Maintenance Overhead:

- Tests required updates whenever functionality changed during Sprint 2.

6. Recommendations for Effective Unit Testing

- **Adopt a Clear Testing Strategy:** Define objectives, standards, and guidelines for unit testing to ensure consistency across the team.
- **Write Tests Before Code (TDD):** Embrace Test-Driven Development (TDD) by writing tests before implementing functionality to ensure code aligns with requirements.
- **Reuse Fixtures:** Use shared fixtures for test data to streamline the process.

- **Automate Test Execution:** Integrate tests into CI/CD pipelines for rapid feedback.