LAB REPORT : TEST-DRIVEN DEVELOPMENT (TDD)

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

Test-Driven Development(TDD) is a key practice in agile methodologies that emphasizes creating tests prior to implementing functionality that is it encourages writing tests before actual implementation. Our team incorporated TDD during the second sprint of the **”JU Exam Office Management System”** project. My primary responsibility was developing the **”Approve Application for Physically Disabled and Sick Students”** feature. This report highlights my contributions, the tools chosen, and my reflections on utilizing TDD in this context.

2. Project Context

2.1 Agile Framework

- **Total Sprints:** 2
- **TDD Integration:** Second Sprint
- **Focus Feature:** To manage student applications for accommodations due to disability or illness.

2.2 TDD Purpose

The Test-Driven Development (TDD) approach was utilized to:

- Improve Code Quality and Reliability
- Catch Bugs Early

- Reducing the Complexity of Development
- Enhance Collaboration
- Simplifying the Debugging Process

2.3 Tools and Frameworks

To effectively implement TDD, the following tools and frameworks were utilized:

- **Django Framework:** Core backend development.
- **Pytest:** Primary testing framework for writing and running unit tests.
- **Django Test Client:** Simulate HTTP requests for functional and integration testing.

3. Test Cases Overview

3.1 Test Request Accommodation Success

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

3.2 Test Request Accommodation Missing Document

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

3.3 Test Review Accommodation Success

Objective: Tests the department successfully reviewing an accommodation request.

3.4 Test Review Accommodation Missing Comments

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

3.5 Test Accommodation Status Page

Objective: Tests that the student can view the status of their accommodation request.

3.6 Test Accommodation Status Page Not Found

Objective: Tests the scenario where the accommodation request doesn't exist (failure).

3.7 Test Review Accommodation Access By Non-Department User

Objective: Tests that a non-department user cannot access the review page.

4. Implementation Process

4.1 Sprint Activities

4.1.1 Preparation

- Set up the Django environment and Pytest for testing
- Created fixtures to simulate database objects

4.1.2 Test Writing

- Developed unit tests for to manage student applications for accommodations due to disability or illness
- Focused on edge cases and invalid scenarios

4.1.3 Implementation

- Wrote functional code to pass tests
- Incrementally refined the code based on test failures

4.1.4 Continuous Testing

- Automated testing with every code change
- Used CircleCI pipelines to ensure consistency and smooth integration across different environments

4.2 Benefits Realized

4.2.1 Improve Code Quality and Reliability

- TDD encourages writing clean, modular, and maintainable code by focusing on small, testable units

4.2.2 Catch Bugs Early

- By writing tests before implementation, potential defects are identified and resolved at an early stage

4.2.3 Simplified Refactoring

- Tests acted as a safety net during code changes

4.2.4 Enhanced Collaboration

- TDD encourages clear communication among team members about expected functionality, improving collaboration

4.2.5 Reducing the Complexity of Development

- TDD supports building software in small, manageable increments

4.2.6 Simplifying the Debugging Process

- Streamline Debugging that is failing tests pinpoint issues quickly

4.2.7 Better Understanding of Requirements

- Writing tests clarified ambiguities in requirements

5. Personal Reflection

Adopting TDD had a profound impact on my development process. Writing tests before implementation helped me to concentrate on functional requirements and create more modular, well-structured code. The iterative refinement approach ensured that the functionality was reliable, even when addressing edge cases. The collaborative aspects of TDD, combined with continuous integration through CircleCI, enhanced our workflow and deepened our understanding of the project's requirements. Although there were challenges initially, the experience proved invaluable, underscoring the significance of disciplined, test-driven development.

6. CONCLUSIONS

Implementing TDD during the second sprint of the **”JU Exam Office Management System”** project proved to be a highly rewarding experience. This approach not only improved the reliability of the attendance tracking feature but also fostered a strong culture of quality and teamwork within the team. Looking ahead, I plan to incorporate TDD into future projects to consistently leverage its advantages.