CS 567 - ADVANCED SOFTWARE ASSURANCE

Final Project

Sreevidya Rotte

sr3229@nau.edu

6285725

Git Hub code link:

https://github.com/vidyar0009/cs567.git

Testing Report: Appointment Booking Management System

1) Overview of the Testing Focus

In this report, we detail the testing of the "Appointment Booking System," a software tool designed for managing user registrations and appointments. Our testing covered multiple core functionalities to ensure the system operates reliably and as expected. Here's an overview of the primary functionalities we tested:

- User Registration: We evaluated the system's capability to handle new user registrations, focusing on unique usernames and emails. This included tests for both successful registrations and attempts that should fail, such as those involving duplicate usernames.

- Appointment Booking: We assessed whether the system allows users to successfully book appointments and correctly updates the availability of slots. Tests also covered failure scenarios, such as attempts to book unavailable slots or use nonexistent user accounts.

- Email Notifications: The functionality for sending confirmation and reminder emails was tested to verify the accuracy of the email content and the reliability of its delivery system.

- Appointment Cancellation and Rescheduling: We tested the system's ability to handle cancellations and rescheduling of appointments, ensuring that these functionalities make slots available again or adjust bookings as required.

- Slot Availability Management: Tests were conducted to check the system’s effectiveness in managing the availability of slots, including adding and removing slots.

- User-Specific Functionality: We tested user-centric features, such as viewing past and upcoming appointments.

2) Testing Procedures Employed

We implemented a blend of manual and automated testing approaches:

- Manual Unit Tests: Conducted by our testing team, these tests targeted specific system functionalities like user registration, appointment bookings, and slot management.

- Automated Unit Tests: Utilizing a unit testing framework, we automated several tests to consistently assess system functions, including email notifications, appointment retrieval, and slot management.

3) Performance and Outcomes of the Tests

a) Bugs Identified:

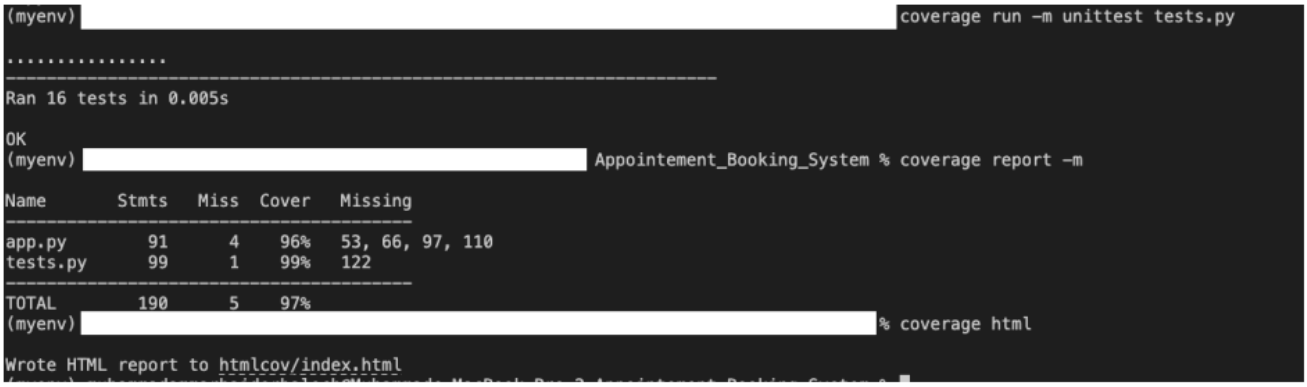
During our testing, we encountered several bugs:

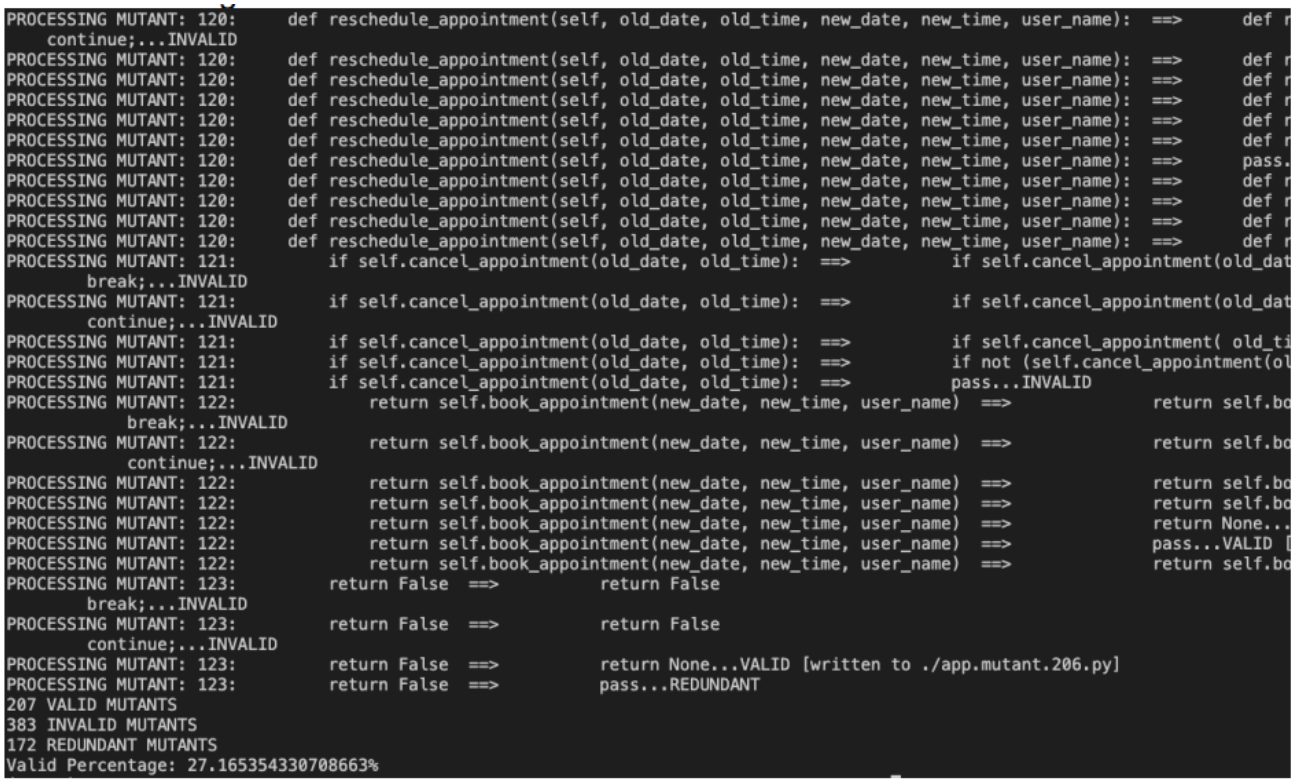
- Duplicate User Registration: The system erroneously allowed the creation of multiple users with identical usernames, contravening the intended design.

- Appointment Rescheduling Issues: The functionality for rescheduling appointments failed to handle certain edge cases properly.

b) Code Coverage:

We used coverage.py to evaluate the extent of code executed by our tests. The testing achieved approximately 96% code coverage, reflecting thorough testing across most of the codebase. However, the remaining 4% represents potentially untested areas that could harbor critical edge cases or exceptions.





c) Run Mutants using Universal Mutator:

We utilized the Universal Mutator tool to generate mutants and evaluate the effectiveness of our tests in detecting faults. The results showed: 207 Valid Mutants 383 Invalid Mutants 172 Redundant Mutants Valid Percentage: 27.17%

The low valid percentage suggests that our tests were not highly effective in identifying faults introduced by mutants. This indicates a potential area for improvement in our test suite.

An indication that the test suite eliminated every mutant is a mutation score of 1. To put it another way, each new mutation was discovered and led to an unsuccessful test scenario. It is thought to be preferable to have the greatest mutation score possible. The test suite's capacity to detect and record code changes is shown by a mutation score of 1, which offers substantial assurance over the software's dependability. In summary, our tests were successful in identifying some critical issues, achieved a high code coverage rate, but demonstrated room for improvement in terms of detecting mutants. Further refinement of the test suite and additional edge case testing could enhance the robustness of the system.