**Automated Testing and Mutation Analysis Report**

**1. Introduction**

The purpose of this report is to analyze and test a Python program implementing an **Apartment Management System**. The program handles apartment leases, tenant profiles, maintenance requests, and rent calculations. This report includes:

* An overview of the codebase.
* The testing tools and techniques applied.
* Results of unit testing, code coverage, and mutation testing.
* Insights gained during the testing process.

**2. Codebase Overview**

**Code Functionality**

The tested program implements functionalities for an apartment management system, including:

1. Managing apartments, tenants, and leases.
2. Handling rent calculations and outstanding balances.
3. Providing detailed reports such as occupancy rates, outstanding dues, and tenant profiles.
4. Managing maintenance requests and assigning staff.

The code length is tested using cloc toll

A screen shot of a computer

Description automatically generated

**3. Testing Tools and Techniques**

**Testing Tools**

1. **unittest**: Python's built-in testing framework was used for writing and executing test cases.
2. **coverage**: Used to measure the test coverage of the codebase.
3. **mutmut**: Used for mutation testing to evaluate the robustness of the test suite.
4. **cloc**: Used to measure the size of the codebase.

**4. Testing Process**

**Unit Testing**

* **Test Cases**:
  + Unit tests covered all critical functionalities such as adding apartments, leasing apartments, handling maintenance requests, calculating rent, and generating reports.

python -m unittest discover

A screen shot of a computer

Description automatically generated

**Code Coverage**

* **Tool**: coverage
* **Process**:
  + Test suite was executed with coverage enabled:

coverage run -m unittest discover

coverage report -m

A screen shot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

* **Results**:
  + Overall coverage: **82%**
  + Key insights:
    - Most methods, including search\_apartments, view\_tenant\_profile, and generate\_outstanding\_report, were fully covered.
    - Minor uncovered lines in edge cases (e.g., error handling for invalid inputs).

**Mutation Testing**

* **Tool**: mutmut
* **Process**:
  + Mutations were generated and tested:

mutmut run

mutmut results

A screenshot of a computer

Description automatically generated

* **Results**:
  + Mutants Killed: **85%**
  + Survived Mutants: 1**5%**
  + Key insights:
    - The test suite effectively handled most mutations.
    - Surviving mutants highlighted weaknesses in error-checking scenarios (e.g., validating invalid tenant profiles).

**5. Results**

**Mutation Testing**

| **Metric** | **Value** |
| --- | --- |
| Mutants Generated | 271 |
| Mutants Killed | 231 |
| Surviving Mutants | 40 |
| Mutation Score | 85% |

**Insights**

1. **Strengths**:
   * Comprehensive test suite covering critical paths and edge cases.
   * High mutation score, indicating robust tests.
2. **Weaknesses**:
   * Some surviving mutants pointed out areas where error handling needs improvement.