Rakshit Girish PES1UG22CS465
Rahul Pandith PES1UG22CS462

UE22CS341A: Software Engineering

Real Estate Management Implementation Test Plan

1. Introduction

This test plan defines the testing strategy, scope, objectives, and methodologies for
the
Real Estate Database Management System. The primary goal is to ensure that each feature functions as intended, and that the system meets quality and security standards for end users, including agents, property managers, buyers, and sellers.

1.1 Scope

In Scope:

Property listing and management, transaction handling, user authentication, and analytics features.

Out of Scope:

External API integrations and scalability testing beyond a certain user load

1.2 Quality Objective

1.2.1 Primary Objective

Ensure the application meets functional requirements, enabling efficient property listing, secure transactions, and user management.

1.2.2 Secondary Objective

Identify and mitigate bugs or risks that could affect functionality or user experience before deployment.

1.3 Roles and Responsibilities

1.3.1 Developer

- Develops the application and supports use cases and requirements.
- Conducts unit, system, regression, and integration testing.
- Supports user acceptance testing.

1.3.2 Adopter

- Contributes to requirement development and testing review.
- Conducts User Acceptance, regression, and end-to-end testing.

1.3.3 Testing Process Management Team

- Manages testing integrity and supports testing activities.
- Coordinates activities and dependencies across development and adoption teams.

1.4 Assumptions for Test Execution

- User Acceptance Testing (UAT) is conducted by end-users.
- Test results will be reported after each iterative development and feature introduction.
- Approved use cases will be available before testing.
- Test scripts, test environment, and dependencies are addressed at the kick-off meeting.

1.5 Constraints for Test Execution

- Developers cannot execute UATs.
- Adopters must identify and request dependencies required for testing.

1.6 Definitions

- **Bug:** Any error causing a malfunction that does not meet required specifications.
- **Enhancement:** An improvement or a defect fix not part of the original requirements.

2. Test Methodology

2.1 Purpose

The purpose of the Test Plan is to:

- Define strategies to cover functional and quality requirements.
- Divide the Design Specification into testable areas.
- Track bugs and identify testing risks.
- Provide resource information and testing schedules.

2.1.2 Usability Testing

Usability testing ensures the UI functions as expected for the end user, providing feedback for enhancements or modifications if needed.

2.1.3 Unit Testing

Unit Testing is conducted by developers to ensure functionality and code coverage. Areas to be tested include:

- Database interactions, stored procedures, triggers, tables, and indexes.
- User authentication and authorization.

2.1.4 Regression Testing

Regression Testing is conducted during each iteration cycle, focusing on verifying bug fixes and ensuring stability. Two to three cycles are expected, with a focus on critical functionality and integration.

2.1.5 Final Release Testing

Final release testing ensures readiness for distribution. Minor and trivial bugs are resolved, with critical bugs fixed before beta release.

2.2 Test Levels

2.2.1 Build Tests

These tests verify that the application builds and installs successfully and that all major components are functional. If any test fails, the build is returned for debugging.

All bugs resolved in the previous build undergo re-testing to confirm fixes.

2.2.2 Milestone Tests

2.2.2.1 Level 3 - Critical Path Tests

Critical Path tests cover core features and functionalities users interact with daily. These are executed during milestone and final release phases.

2.2.3 Release Tests

These include end-to-end tests verifying all components and interfaces in the application. This phase ensures the application's readiness for end users.

---

1. Test Approach Document

Purpose:
This document defines the testing approach to ensure that the property listings and transactions Computation application is thoroughly tested. The approach will involve:

1. **Test Plan:** Outline of the testing strategy covering data generation, property and transaction listing, MySQL database integration, and buyer query functionality.
2. **Test Coverage:** Ensuring every critical function is validated, including data generation, database storage, transaction calculation, property assignment, and output retrieval.
3. **Test Tools:** MySQL database, Python for scripting and data processing, YAML for configuration, and a CSV generator for output files.

Reviewers:
Project Supervisor, Development Lead, QA Coordinator

---

2. Deliverables Matrix

Deliverable	Type	Details
Test Plan	**Document**	**Specifies the approach, detailed feature breakdown, test case areas, and sign-off procedures.**
Test Schedule	**Document**	**Timeline for executing test cases aligned with project milestones.**
Test Specifications	**Document**	**Outlines test scenarios and objectives for data generation, calculation accuracy, and output generation.**
Requirements Traceability	**Document/Matrix**	**Maps requirements to test cases, ensuring that all functionalities are verified.**
Test Cases	**Test Case/Bug Write-Up**	**Detailed steps for test execution, covering data generation, database operations, property listing, and transactions.**
Bug Tracker	**Bug Report**	**Logs defects, including priority, severity, and reproduction steps.**
Test Final Report	**Report**	**A summary of results, including payment complete/not counts, coverage reports, unresolved issues, and metrics.**

---

3. Test Plan

Objective:
Verify each feature, functionality, and integration of the transaction and property listing application to ensure compliance with quality standards. Key focus areas include data generation, property listing, transactions, and query processing.

3.1 Functional Coverage

- Data Generation: Confirm that seller and property data is generated accurately with unique IDs and registration info.
- Database Storage and Retrieval: Ensure data is properly stored and retrievable in/from the MySQL database.
- Property Listing : Validate that properties for each buyer and overall data is computed accurately.
- Transactioning: Verify transactions based on property listings.
- Query Processing: Test the functionality for retrieving specific property data and top listings.

3.2 Test Environment

- **Hardware:** Standard PCs for local testing, deployment server for high-volume data testing.
- **Software:** Python 3.8 for data generation and processing, MySQL 8.0.23, YAML for configuration, and CSV module for output generation.

3.3 Risk and Contingency Plans

- **Risk:** Database integrity issues.
  - **Mitigation:** Regular data backups and consistency checks.
- **Risk:** High volume data causing slowdowns.
  - **Mitigation:** Optimize data queries and processing steps for efficiency.

---

5. Defect Tracking & Debugging

Bug Tracker:
Use a defect-tracking tool to log issues such as data inconsistencies, calculation errors, and system performance problems. Each defect will include details on severity, priority, and evidence (e.g., screenshots, error logs).

---

6. Reports

- **Testing Status Report:** A weekly summary covering test case progress, bug counts, and identified risks.
- **Phase Completion Report:** Documented at the end of each testing phase, detailing test execution, open issues, and overall progress.
- **Final Test Report:** A comprehensive report summarizing the testing outcome, covering test completion, quality metrics, unresolved issues, and overall release readiness.

7. Responsibility Matrix

Role	Name	Responsibilities
Test Lead	Rakshit, Rahul	Coordinate testing
Developer	Rakshit, Rahul	Fix defects
Product Manager	Rakshit, Rahul	Validate testing goals align with project objectives.
QA Engineer	Rakshit, Rahul	Execute test cases and report bugs.

---

8. Resource & Environment Needs

8.1 Test Environment

- **Hardware:**
  - Model: MacBook Pro
  - Processor: Intel Core i5 7th Gen
  - Memory: 24GB RAM
  - Graphics: Intel Iris Graphics
- **Software:**
  - **Database:** MySQL 8.0.23 for data storage
  - **Programming Language:** Python 3.8
  - **IDE:** VS Code
  - **Operating System:** MacOS 15.0

---

9. Severity and Priority Definition

Severity Level

- **Critical:** Application crashes or causes significant data loss.
- **High:** Major issues that impact functionality with limited or no workarounds.
- **Medium:** Minor functionality issues with available workarounds.
- **Low:** Cosmetic or documentation-related errors.

Priority Level

- **Must Fix:** Immediate resolution required for release.
- **Should Fix:** Important; should be addressed before release if possible.
- **Fix When Have Time:** Low impact; can be resolved in future updates.
- **Low Priority:** Non-essential; can be deferred if necessary.