Test Strategy Document

For Apartment Complex Account Management System

1. Introduction

1.1 Purpose

This document outlines the testing strategy for the **Apartment Complex Account Management System**. The goal is to ensure the system meets all functional and non-functional requirements while optimizing testing efforts.

1.2 Scope

In-Scope Features:

- Account Management (Create, Update, View)
- Transaction Management (Record, Prevent Negative Balances)
- Balance Tracking (Opening/Closing Balances, Profit/Loss Calculation)
- Reporting (Account Status, Fund Status, Transaction History)

Out-of-Scope Features:

- Multi-Currency Support
- Automated Bank Sync
- User Role Management (Future Enhancement)

1.3 References

- **Software Requirements Document (SRD)** – Chapter 5
- **User Stories & RTM** – Chapter 5
- **System Architecture** (Python/Django, SQLite/PostgreSQL, React)

---

2. Test Approach

2.1 Testing Levels

Level	Focus Area	Tools/Methods
Unit	Individual functions (e.g., balance calculation)	PyTest (Python), Jest (JS)
Integration	Interaction between modules (e.g., transaction updates balances)	Postman, Django Test Client

Level	Focus Area	Tools/Methods
System	End-to-end workflows (e.g., creating an account → recording a transaction)	Selenium, Cypress
UAT	Business validation by end-users (Accountants/Admins)	Manual Execution

2.2 Testing Types

Type	Coverage	Tools
Functional	Validate user stories (e.g., US-001 to US-007)	Selenium, Manual
Regression	Ensure new changes don't break existing features	Jenkins, GitLab CI
Performance	Response time (<2s), 1000 transactions/month	JMeter, Locust
Security	Authentication, audit logs, GDPR compliance	OWASP ZAP, SonarQube
Usability	UI clarity, error messages	Manual Testing

2.3 Automation Strategy

- **Automate:** High-priority regression tests (e.g., transaction validation, negative balance checks).
- **Manual:** Exploratory testing, UAT, edge cases (e.g., future-dated transactions).
- **Tools:** Selenium (UI), PyTest (API), JMeter (Load).

2.4 Risk-Based Testing

Priority	Risk Area	Mitigation
High	Incorrect balance calculations	Extensive unit + integration tests
High	Negative balance bypass	Boundary testing
Medium	Report generation delays	Performance benchmarks
Low	Optional field validations	Low-priority test cases

---

3. Test Environment

3.1 Hardware/Software

- **Backend:** Python/Django, Node.js
- **Database:** SQLite (Dev), PostgreSQL (Staging)
- **Frontend:** React, Chrome/Firefox browsers
- **Test Data:** Pre-populated accounts (Income, Expense, Asset, Liability)

3.2 Environments

Environment	Purpose
Dev	Unit/Integration Testing
Staging	System/Performance Testing
Production-like	UAT

---

4. Test Deliverables

- **Test Plans:** Per module (Account, Transaction, Reporting)
- **Test Cases:** Traceable to User Stories (e.g., TC-201 for US-001)
- **Defect Reports:** Logged in JIRA with severity (Critical/High/Medium/Low)
- **Traceability Matrix:** Links tests to requirements (BR-001 → TC-201)

---

5. Roles & Responsibilities

Role	Responsibility
Test Lead	Strategy, coordination
QA Engineers	Test execution, defect logging
Developers	Unit tests, bug fixes
Business Analysts	UAT validation

---

6. Schedule & Effort Estimation

Phase	Duration	Resources
Planning	1 week	Test Lead, QA
Execution	3 weeks	QA Team (3 members)
UAT	1 week	Business Stakeholders

---

7. Risks & Mitigations

Risk	Mitigation
Incomplete test data	Use synthetic data generators
Delayed environment setup	Early coordination with DevOps
Critical defects late in cycle	Prioritize high-impact tests early

---

8. Exit Criteria

- **95% test pass rate** for all priority test cases.
- **Zero Critical/High defects** unresolved.
- **UAT sign-off** from business stakeholders.

Tools & Compliance

- **Automation:** Selenium, PyTest, JMeter
- **Tracking:** JIRA, TestRail
- **Compliance:** GDPR (audit logs), data immutability checks.

Requirement Traceability Matrix (RTM) with Test Strategy Mapping

Req ID	User Story ID	Requirement Description	Test Level	Test Type	Test Tools	Test Case ID	Automation (Y/N)	Risk Priority
BR-001	US-001	Create financial accounts	Unit, System	Functional, Regression	PyTest, Selenium	TC-201	Y	High
BR-002	US-002	Update financial accounts	Unit, Integration	Functional, Negative	PyTest, Postman	TC-202	Y	Medium
BR-003	US-003	View accounts with filters	System, UAT	Functional, Usability	Selenium, Manual	TC-203	N	Medium
BR-004	US-004	Record transactions	Integration, System	Functional, Boundary	Postman, Selenium	TC-204	Y	High
BR-005	US-005	Prevent negative balances	Unit, System	Functional, Security	PyTest, OWASP ZAP	TC-205	Y	High
BR-006	US-006	Generate account status report	System, Performance	Functional, Performance	Selenium, JMeter	TC-206	Y	Medium
BR-007	US-007	Monthly profit/loss report	System, UAT	Functional, Accuracy	Selenium, Manual	TC-207	N	High

-

Approval:

Test Lead: _____

Date: _____