CS411 - Testing Documentation

LandLedger:

Blockchain-Based Land Registration System

Atesam Abdullah	2021114
Huzaifa Javed	2021234

Table of Contents

Code available at: https://github.com/ATESAM-ABDULLAH/LandLedger

1. Introduction

This document provides detailed testing results for the **LandLedger** system, a blockchain-based platform designed to streamline land registration and ownership transfer. By leveraging the Ethereum blockchain, this decentralized application ensures transparency, security, and efficiency.

The testing scope includes:

- Functional testing of core features.
- Performance testing under varied conditions.
- Security testing of smart contracts and access control.
- Integration testing with Metamask and Ethereum blockchain.
- UX validation across platforms.

Key Objectives

- Validate the functionality and security of the smart contracts.
- Ensure data immutability and transparency.
- Evaluate system performance and scalability.
- Confirm role-based access control and secure transactions.

2. Functional Testing

Table 1: Functional Requirements

Test Type	Test ID	Description	Test Steps	Expected Result
Property Addition	ADD-001	Add a new property with valid details	Submit valid property details as a user.	Property added successfully.
Property Verification	VER-001	Verify a property successfully	Submit a valid property ID as an inspector for verification.	Property state changes to Verified.
Ownership Transfer	TRF-001	Transfer property ownership after payment	Approve a valid buyer's payment and transfer ownership.	Ownership transferred successfully.
Property Management	PMG-001	Retrieve property details	Fetch details of an existing property using its ID.	Property details are displayed accurately.
User Management	USR-001	Register a new user	Submit unique CNIC and valid user details.	User registered successfully.
Sale and Purchase	SLP-001	List property for sale by owner	Owner lists property with a set price.	Property listed for sale successfully.
	SLP-002	Submit a purchase request	Buyer submits a request for a listed property with an offer.	Request submitted successfully.

	SLP-003	Accept a purchase request by owner	Owner accepts an offer from the purchase requests.	Ownership transfer initiated.

Figure 1: Contract Testing

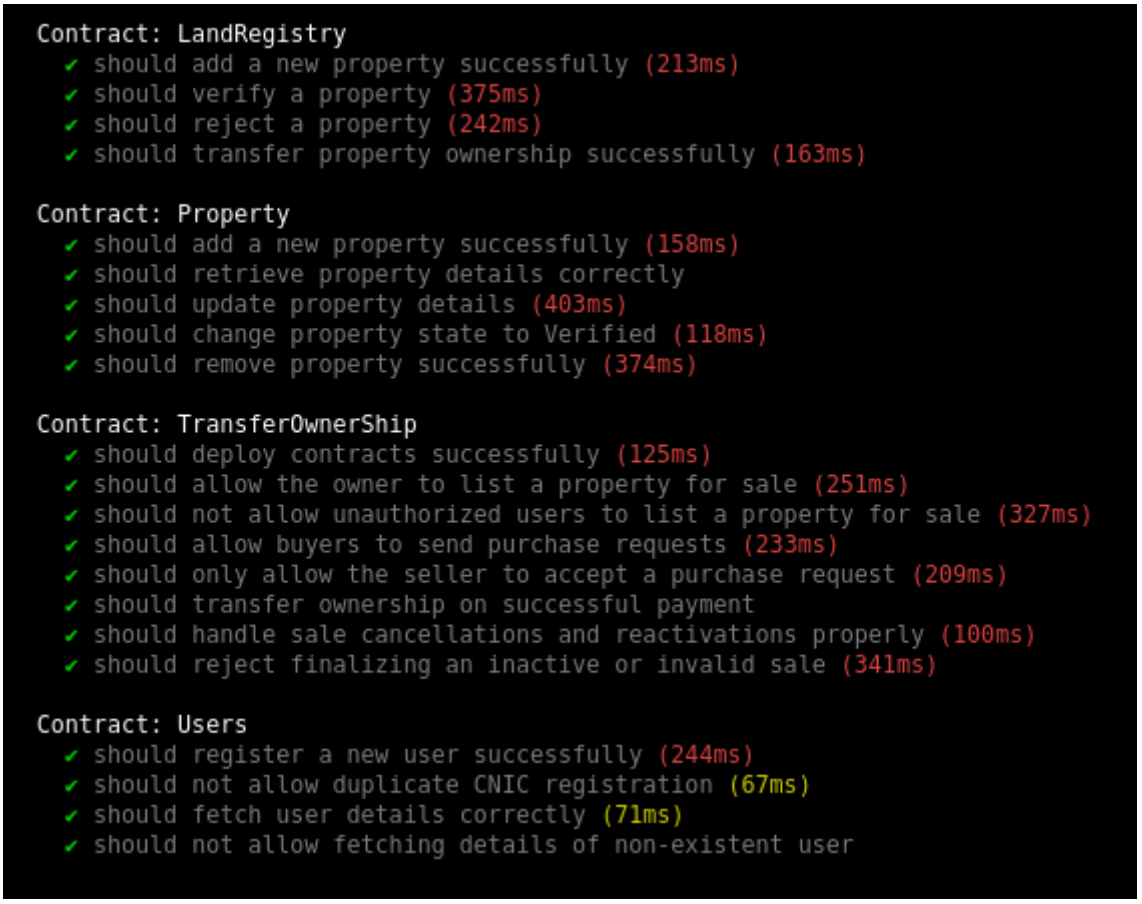

Table 2: Overall Performance Metrics

Metric	LandRegistry (Add)	LandRegistry (Verify)	TransferOfOwnership (Transfer)	User (Add)	User (Update)
Max Throughput (TPS)	25	32	19	70	41
Average Gas Cost	180,000	110,000	210,000	120,000	80,000
Failure Rate (%)	3%	6%	24%	0	0
CPU Utilization (Ganache + Mongodb)	~65%	~58%	~72%	~32%	~20%

5. Integration Testing

Table 3: Component Testing

Component	Test ID	Description	Test Steps	Result
Flask	FL-001	Test server connection	Start Flask server and access the home endpoint.	Server responds successfully with status 200.
	FL-002	Send data Flask -> MongoDB	Submit valid property details through the API.	Property logged in blockchain and MongoDB.
Metamask	MM-001	Test wallet connection	Connect Metamask wallet to the application.	Wallet connected successfully.
	MM-002	Verify transaction signing	Sign a property addition transaction through Metamask.	Transaction signed and confirmed in blockchain.
Truffle+Ganache	TG-001	Verify smart contract deployment	Deploy contracts to Ganache using Truffle.	Contracts deployed successfully.
	TG-002	Test property ownership transfer	Execute ownership transfer using deployed contracts.	Ownership updated successfully in blockchain.

Table 4: Property Documents (PDFs)

Role	Test ID	Description	Test Steps	Expected Result
Seller	MONGO-001	Upload property documents	Seller uploads documents to the system.	Documents stored in MongoDB.
Revenue Employee	MONGO-002	Review property documents	Retrieve and review uploaded documents.	Documents retrieved successfully.
Buyer	MONGO-004	Access approved documents	Request access to approved documents.	Documents retrieved for access.
	MONGO-005	Restrict unapproved document access	Attempt to access unapproved documents.	Access denied with error message.

6. Bug Log

Table 5: Bugs encountered

Bug ID	Description	Component	Trigger	Impact	Status
BUG-001	Incorrect status update after approval rejection	Revenue Employee Module	Revenue employee rejects a property, but status remains "Pending."	Confusion for seller and buyers.	Fixed
BUG-002	Unauthorized buyers can see property metadata	Buyer Access Control	Incorrect permission handling allows metadata visibility.	Potential information leakage.	Fixed
BUG-003	Review resets seller-uploaded documents	Document Approval System	Retry operation replaces uploaded documents with blank entry.	Seller must re-upload, causing delays.	Fixed
BUG-004	Gas consumption spikes during ownership transfers	Smart Contracts	Simultaneous transfers cause high gas usage.	Increased transaction costs and rejections.	Identified
BUG-005	Wallet disconnection doesn’t update UI state	Metamask Integration	Disconnecting wallet doesn’t reflect on UI.	Users think they’re connected but can’t operate.	Fixed

7. Recommendations

1. **Improve Status Management:**
   - 1.1. Implement stricter backend validations to ensure accurate property status updates after approval or rejection by revenue employees.
   - 1.2. Introduce a logging mechanism to trace any discrepancies in status changes for debugging.
2. **Optimize Document Retrieval:**
   - 2.1. Compress property documents before uploading them to MongoDB to reduce retrieval times.
   - 2.2. Use caching mechanisms for frequently accessed documents to enhance performance.
3. **Reduce Gas Consumption:**
   - 3.1. Optimize smart contract functions, especially for ownership transfers, to minimize gas usage.
   - 3.2. Batch similar operations whenever possible to improve gas efficiency.
4. **Enhance UI State Sync:**
   - 4.1. Synchronize Metamask wallet connection status with the application UI to provide users with real-time feedback on their connection state.
   - 4.2. Add alerts or prompts for users when wallet disconnection is detected.
5. **Prevent Concurrent Data Collisions:**
   - 5.1. Introduce unique locking mechanisms for property uploads to ensure simultaneous operations by multiple users don’t overwrite data.
   - 5.2. Add real-time upload progress tracking to prevent re-submissions during ongoing operations.

8. Conclusion

The **LandLedger** system has shown significant potential in revolutionizing land registration and ownership processes by leveraging blockchain technology. While the testing process has validated the core functionality, several areas require attention to ensure a seamless and reliable user experience. Addressing the identified bugs, such as status management inconsistencies, gas inefficiencies, and data collisions, will further enhance the platform's robustness.

By implementing the recommendations, the system can achieve improved performance, better scalability, and enhanced trustworthiness. With these optimizations, **LandLedger** will be well-positioned to deliver a transparent, secure, and efficient solution to modernize land registry systems.