## **1. Introduction**

The Blockchain-Based Land Management and Registration System is designed to enhance security, transparency, and efficiency in property registration and ownership management. By leveraging blockchain technology, the system ensures tamper-proof records, verifiable transactions, and reliable audit trails.

This course project simulates a real-world Agile software development process, where students work collaboratively to manage the project lifecycle-from requirement analysis to implementation and testing-using tools such as Trello for task planning and GitHub for version control. The objective is to demonstrate practical project management skills, stakeholder engagement, and structured documentation in a professional software development environment.

The project focuses on meeting the diverse needs of multiple stakeholders while adhering to both functional and non-functional requirements to ensure the system is secure, compliant, and scalable.

## **2. Stakeholder Identification**

The system involves multiple stakeholders with varying interests, roles, and levels of influence. Each stakeholder plays a critical role in the successful operation and acceptance of the system. Proper identification and analysis of stakeholders ensure that their needs are accurately captured and addressed in both system design and implementation.

## **3. Stakeholder Needs and Expectations**

Understanding the specific needs and expectations of stakeholders is essential to ensure that the system delivers value and aligns with legal, operational, and usability requirements. The stakeholders include property owners, government authorities, system administrators, developers, auditors, and other users who interact with the system in different capacities.

|  |  |
| --- | --- |
| **Stakeholder** | **Needs / Expectations** |
| Property Owners | Secure, tamper-proof ownership transfers; minimal transaction fees; easy-to-use interface; real-time transaction verification. |
| Government / Registrar Authority | Immutable ledger; audit trails for dispute resolution; integration with national property databases; legal compliance. |
| System Administrators | Redundant blockchain nodes; performance monitoring tools (TPS, latency); smart contract monitoring; quick recovery from failures. |
| Notaries / Legal Advisors | Verified document authenticity via blockchain hashing; identity verification; support in dispute resolution. |
| Financial Institutions | Automated transaction fee handling; secure banking API integration; audit-ready transaction logs. |
| End Users / Public | Transparent view-only access to property ownership; trust in blockchain verification; historical property data access. |
| Developers / QA | Clear technical documentation; smart contract templates; access to test environments; feedback on usability issues. |
| Auditors / Compliance Agencies | Transparent reports of all blockchain transactions; compliance with data protection, financial, and legal regulations. |

**Stakeholder Needs and Expectations Table**

Stakeholder expectations range from secure and fast property transactions for owners, compliance and auditability for government authorities, to system reliability and monitoring for IT teams. By addressing these expectations, the system ensures transparency, trust, and operational efficiency.

## **4. Data Collection and Techniques Used**

Comprehensive data collection is critical for capturing stakeholder requirements accurately. Multiple techniques were employed to gather information, including:

* **Online Forms:** Used for property owners and agents to submit ownership details, property documents, and feedback.
* **Structured Interviews & Workshops:** Conducted with government authorities, system administrators, and auditors to capture regulatory and operational requirements.
* **Surveys:** Distributed to end users to collect feedback on usability and transparency requirements.
* **Monitoring Tools & Logs:** Used by system administrators to track performance metrics, such as transactions per second (TPS) and latency logs.
* **Daily Stand-ups & Sprint Retrospectives:** Employed by developers and QA teams to capture implementation challenges, bugs, and feedback on usability.

|  |  |  |  |
| --- | --- | --- | --- |
| **Stakeholder** | **Data to be Collected** | **Collection Method** | **Purpose / Result** |
| Property Owners and Agents (Buyers/Sellers and Agents) | Ownership details, property documents, transfer requests, user experience feedback | Online forms, document uploads to blockchain portal, structured interviews, surveys | Verify property ownership, capture user requirements, identify usability improvements |
| Government / Registrar Authority | Legal requirements, property records, compliance guidelines | Policy documents, official databases, direct consultation workshops | Ensure compliance, validate ownership legitimacy, set legal framework for blockchain rules |
| System Administrators / IT Team | Technical requirements, system performance logs, uptime and security data | Monitoring tools (TPS, latency logs), admin interviews, system health reports | Ensure scalability, reliability, and security of blockchain infrastructure |
| Developers / QA Team | Technical feasibility, smart contract implementation challenges, bug reports | Daily stand-ups, sprint retrospectives, bug tracking tools (GitHub issues) | Improve development efficiency, refine blockchain features, ensure quality assurance |
| Auditors / Compliance Agencies | Audit requirements, reporting standards, data protection needs | Compliance reports, regulatory guidelines, structured interviews | Guarantee immutability, transparency, and adherence to financial/legal standards |

**Data Collection Table**

These techniques ensured that requirements were gathered systematically and verified through multiple sources, providing a reliable foundation for system design and development.

## **5. Requirements per Stakeholder**

### **5.1 Property Owners (Buyers/Sellers)**

**Functional Requirements**

* The system shall allow property owners to upload property documents securely to the blockchain portal.
* The system shall provide a digital interface for initiating property transfer requests.
* The system shall enable users to view and track the status of their property ownership and registration updates.

**Non-Functional Requirements**

* The system shall ensure ease of use with a simple and intuitive user interface.
* The system shall process property transfer requests within 5 seconds under normal load.
* The system shall provide multi-language support to improve accessibility.

### **5.2 Government / Registrar Authority**

**Functional Requirements**

* The system shall allow government officials to verify property records against official databases.
* The system shall enforce compliance with local and national real estate laws during property transactions.
* The system shall generate compliance reports for property ownership validation.

**Non-Functional Requirements**

* The system shall ensure that legal rules are configurable to adapt to future regulatory changes.
* The system shall maintain data accuracy of 99.9% for government property records.
* The system shall provide secure access through role-based authentication for registrar staff.

### **5.3 System Administrators / IT Team**

**Functional Requirements**

* The system shall provide administrators with dashboards for monitoring TPS (transactions per second) and latency logs.
* The system shall allow administrators to configure redundancy for blockchain nodes.
* The system shall notify administrators in case of downtime or performance degradation.

**Non-Functional Requirements**

* The system shall achieve 99.95% uptime availability.
* The system shall ensure encryption of sensitive monitoring data in transit and at rest.
* The system shall support horizontal scalability to handle up to 10,000 concurrent users.

### **5.4 Developers / QA Team**

**Functional Requirements**

* The system shall provide an environment for creating, deploying, and testing smart contracts.
* The system shall allow bug tracking integration with GitHub for efficient issue resolution.
* The system shall support automated test case execution for ownership verification.

**Non-Functional Requirements**

* The system shall provide development tools with a response time of less than 2 seconds for API calls.
* The system shall maintain detailed error logs for debugging and troubleshooting.
* The system shall ensure version control compatibility for continuous integration and deployment (CI/CD).

### **5.5 Auditors / Compliance Agencies**

**Functional Requirements**

* The system shall generate immutable audit trails for all property ownership transactions.
* The system shall provide hash-based verification for document validation.
* The system shall allow auditors to export compliance reports in standard formats (PDF, CSV).

**Non-Functional Requirements**

* The system shall guarantee immutability of records using blockchain consensus mechanisms.
* The system shall meet industry-standard data protection regulations (e.g., GDPR, ISO 27001).
* The system shall provide audit report generation within 10 seconds regardless of database size.

## **6. UI/UX Enhancements**

To improve usability and user satisfaction, the system includes:

* Enhanced user interface for document upload.
* Intuitive dashboards for administrators, auditors, and users.
* Efficient property search functionality for quick access to ownership records.

## **7. Test Cases**

Testing is a critical component of the system development lifecycle to ensure that all functional and non-functional requirements are met and the system operates as intended. For the Blockchain-Based Land Management and Registration System, test cases are designed to validate system behaviour under various conditions, including normal operation, boundary scenarios, and error conditions.

The test cases cover multiple testing levels, including:

* **Unit Testing:** Verifies individual modules or components of the system, such as smart contract functionality, property document uploads, and authentication mechanisms. Each unit is tested in isolation to confirm that it performs as expected.
* **Integration Testing:** Ensures that different system modules work together seamlessly. Examples include integration between the property registration portal, blockchain ledger, and auditor reporting module.
* **System Testing:** Evaluates the complete system in an environment that closely resembles production. This includes testing workflows such as property ownership transfer, audit trail generation, and compliance report exports.
* **User Acceptance Testing (UAT):** Confirms that the system meets the needs of stakeholders, including property owners, government authorities, auditors, and administrators. UAT focuses on usability, functionality, and adherence to defined requirements.

Each test case is structured with the following components:

* **Test Case ID:** A unique identifier for tracking purposes.
* **Description:** A brief summary of what the test case aims to validate.
* **Test Type:** Categorized as positive or negative to indicate whether it verifies expected behaviour or system response to invalid inputs.
* **Pre-Conditions:** Conditions that must be met before executing the test, such as user login or available data.
* **Remarks:** Notes on the purpose or special considerations for the test.
* **Expected Result:** The anticipated system response when the test is executed correctly.
* **Status:** Indicates whether the test passed, failed, or is pending.
* **Actual Output:** Documents the system's actual behavior during testing to facilitate validation and debugging.

The test case tables are prepared in **Excel format** to allow clear tracking, easy updates, and structured documentation of all testing activities. Each table corresponds to specific stakeholder requirements, ensuring comprehensive coverage of all functional and non-functional aspects.

## **8. Summary**

This document consolidates the key aspects of the Blockchain-Based Land Management and Registration System, including stakeholder analysis, functional and non-functional requirements, and data collection techniques. The system ensures secure, transparent, and efficient property registration while meeting the needs of property owners, government authorities, administrators, developers, and auditors.

Comprehensive test cases covering unit, integration, system, and user acceptance testing validate all functional and non-functional requirements, ensuring performance, security, and regulatory compliance. The project also demonstrates real-world Agile practices, including requirement analysis, backlog management, sprint planning, and version control using GitHub.

Overall, this report provides a complete blueprint for implementing a blockchain-based land management system while showcasing both technical and project management competencies in a professional software development environment.