# **SMART ESTATE SPACE.**

Design Document Specification

1. **Introduction**

This document outlines the design specifications for Smart Estate Space, a decentralized web application (dApp) built on blockchain technology, aimed at revolutionizing secure and transparent rental transactions. It addresses the limitations of traditional methods, replacing paper contracts, manual processes, and centralized third-parties with smart contracts and on-chain automation.

1. **System Objectives**

* **Simplify Rent Payments**: Automate monthly rent deductions and secure deposits to landlord accounts using smart contracts.
* **Enhance Security Checks**: Utilize Web3 for location-based functionalities with user consent:
* **On-chain location verification**: Anonymously verify user location during security checks without revealing exact coordinates.
* **Decentralized geo-fencing**: Trigger actions based on user location relative to predefined areas for compliance monitoring.
* **Off-chain location data integration**: Integrate trusted off-chain location services with user consent for enhanced security.
* **Facilitate Escrow Management**: Securely hold security deposits in a blockchain-based escrow, automatically releasing them to tenants upon lease termination based on pre-defined conditions.
* **Streamline Maintenance Requests**: Enable tenants to submit maintenance requests through the platform, triggering automated notifications and facilitating efficient task management.
* **Target Audience**: Landlords and tenants seeking a secure, transparent, and efficient rental experience.

1. **Architectural Design**

3.1 Component Diagram

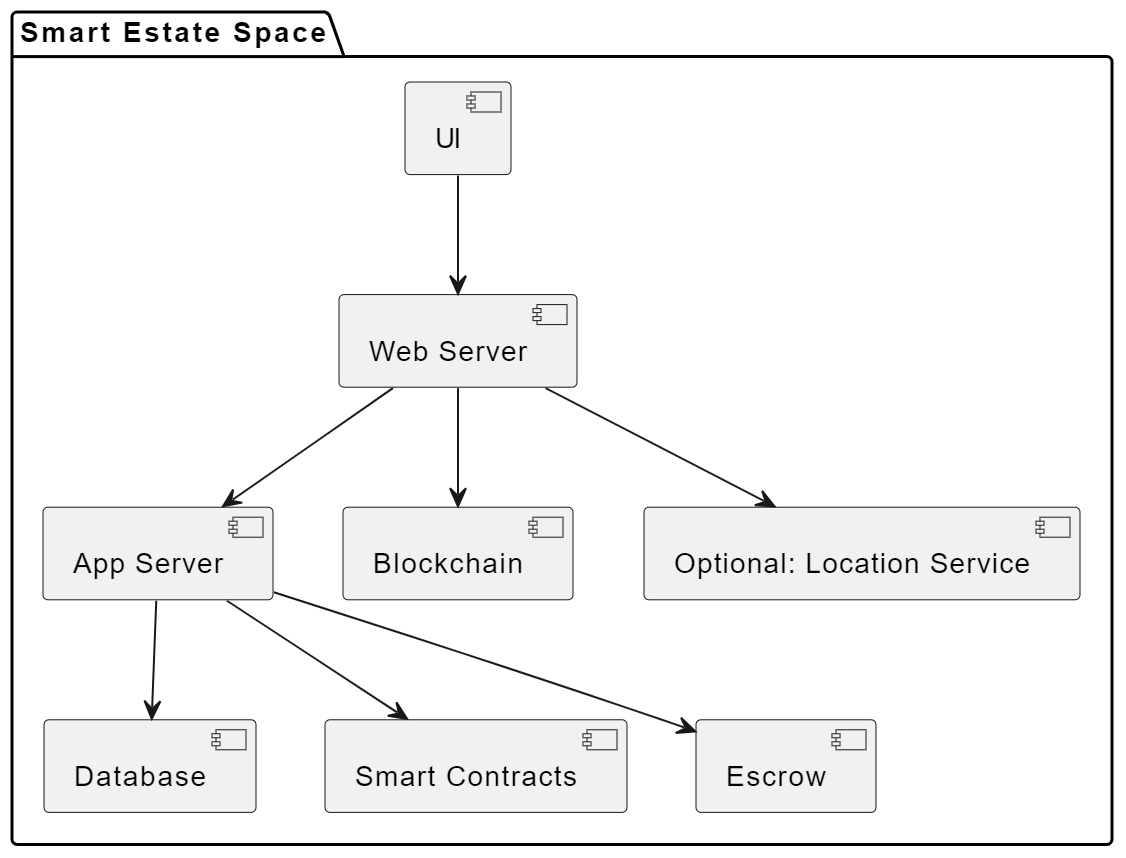


Diagram depicting the primary components and their interactions, including user interface, web server, application server, blockchain network, smart contracts, escrow service, and optional off-chain location service.

**3.2 Key Components**

**User Interface (UI)**: Web-based interface for user interactions, including:

* account management
* property search
* rental agreements
* secure transactions
* maintenance requests.

**Web Server**: Handles HTTP requests and responses, routes data to the application server, and implements API endpoints for communication with other components.

**Application Server**: Executes business logic, interacts with the database, triggers smart contract interactions, and processes data exchanges with the UI and web server.

**Blockchain Network**: Securely stores rental agreements, transaction history, and property data. (Specify chosen network: Ethereum, Hyperledger Fabric, etc.)

**Smart Contracts**: Automate rental agreements, payments, escrow management, and security checks based on pre-defined conditions.

**Escrow Service**: Manages the on-chain escrow functionality for security deposits.

**Off-chain Location Service (Optional)**: Integrate with a trusted off-chain location service with user consent to provide additional verification during security checks.

1. **Detailed Design**

**4.1 User Management**

- User registration and login with secure authentication mechanisms.

- User profile management with personal information, contact details, and rental history.

- Role-based access control for landlords and tenants with distinct functionalities.

**4.2 Property Management**

- Property listing with detailed descriptions, photos, amenities, and availability.

- Search and filter functionalities based on location, type, price, and availability.

- Secure rental agreement creation and execution using smart contracts.

**4.3 Transaction Management**

- Automated monthly rent payments from tenant wallets to landlord accounts.

- Secure on-chain escrow management for security deposits with automatic return conditions.

- Integration with a chosen payment gateway for cryptocurrency transactions.

**4.4 Security Checks**

- Implement on-chain location verification with user consent, ensuring validity without revealing exact coordinates.

- Define decentralized geo-fencing zones for specific actions based on user location.

- Integrate with optional off-chain location service for enhanced security when necessary.

**4.5 Maintenance Requests**

- Tenants submit maintenance requests through the platform.

- Automated notifications to landlords regarding requests.

- Task scheduling and tracking functionalities for efficient maintenance management.

1. **Technology Stack**

**Frontend**

* HTML, CSS, JavaScript
* React.js (or a similar framework for efficient UI development and state management)

**Backend**

* Node.js (for server-side JavaScript execution)
* Express.js (or a similar web framework for routing, middleware, and API development)

**Database**

* PostgreSQL (or a similar relational database optimized for ACID properties and complex data relationships)

**Blockchain Network**

* Kushite ICP (for decentralized data storage, smart contract execution, and integration with the Internet Computer)

**Smart Contract Language**:

* Motoko (the native language for smart contracts on the Internet Computer, offering security and scalability)

**Payment Gateway**

* Stripe (for seamless credit card and cryptocurrency payment processing, global reach, and developer-friendly integration)

**Off-chain Location Service (Optional)**

* Google Maps Platform (for features like geocoding, maps integration, and location-based services, with user consent for privacy considerations)

1. **Deployment**

- Cloud-based deployment on a secure platform (AWS, Azure, etc.).

- Load balancing and scaling strategies to handle user traffic.

- Regular maintenance and updates to ensure platform stability and security.

1. **Testing**

- Unit testing for individual components.

- Integration testing for component interactions.

- User acceptance testing to validate platform functionality and user experience.

- Security audits and penetration testing to identify and address vulnerabilities.

1. **Conclusion**

This design document specification outlines the architecture, components, and technical details for Smart Estate Space.