A
PROJECT REPORT
ON
“Land Registration System Using Blockchain”

Submitted By

Name of Students	**PRN No.**
Rohan Sunil Mane	2167971242062
Raghav Rajendra Pai	2167971242024
Siddharth Deepak Shinde	2167971242046
Sanket Rajendra Jagtap	2167971242060

Under the
guidance ofProf.
G.H. Deo

In partial fulfillment of
the requirements for the award of the degree of

Bachelor of Technology in
Computer Science and Engineering

RAOSAHEB WANGDE MASTER CHARITABLE TRUST’SDNYANSHREE
INSTITUTE OF ENGINEERING AND
TECHNOLOGY,Sonawadi-Gajawadi, Satara, Pin: 415 013
Dr. Babasaheb Ambedkar Technological University, Lonere,
MaharashtraAcademic Year 2024-25

Certificate

This is to certify that the project entitled, **“ Land Registration System Using Blockchain”**is

Submitted by,

Name of Students	**PRN No.**
Rohan Sunil Mane	2167971242062
Raghav Rajendra Pai	2167971242024
Siddharth Deepak Shinde	2167971242046
Sanket Rajendra Jagtap	2167971242060

It is a bonafide work carried out by these students under guidance of Prof. G.H.Deo It has been accepted and approved for the partial fulfillment ofthe requirement of Dr. Babasaheb Ambedkar Technological University, Lonere, for the award of the degree of Bachelor of Technology (Computer Science and Engineering). This work and project report has not been earlier submitted to any other Institute or University for the award ofany degree or diploma.

Prof. G.H. Deo	Dr. C. S. Shinde	Prof. P.M. Pondkule	Dr.A.D. Jadhav
Project Guide	Project coordinator	Head of Department	Principal

Signature:
Prof. (External Examiner)

Place: -
Satara
Date:-

Acknowledgments

The satisfaction and euphoria that accompany the successful completion of any task would be incomplete without the mention of people who made it possible. So, we acknowledged all those whose guidance and encouragement served as a beacon of lightand crowned our efforts with success.

We have immense pleasure in expressing thanks to the Principal **Dr. A. D.Jadhav** , for providing all the facilities for the successful completion of the project.

Our profound sense of gratitude is due to our Head of the department **Dr. C.S. Shinde** for constant encouragement, valuable guidance and also for theinspiration to pursue the project.

We also thanks to **Dr. C.S. Shinde** Project Coordinator, for their guidance andfor being a constant source of support and guided us in this Endeavour.

Last but not the least we are thankful to all faculty members and lab instructors without whose support at various stages, this project would not have materialized.

”Thanking You”

Mr. Rohan Sunil Mane
Mr. Raghav Rajendra Pai
Mr. Siddharth Deepak Shinde
Mr. Sanket Rajendra Jagtap

Date:
Place:

List of Figures

Abstract

The Blockchain-Based Land Registration System is an innovative solution aimed at addressing inefficiencies and challenges in traditional land registry processes. This system leverages blockchain technology to ensure secure, transparent, and tamper-proof land transactions. The proposed system includes a web-based interface for users to initiate Land transactions, validate ownership, and access land records. User authentication is reinforced through Aadhaar or KYC integration, enhancing trust and security.

At its core, smart contracts automate critical processes such as Land ownership validation and Land transfers, reducing human intervention and minimizing errors. Blockchain ensures that all transactions are immutable and transparent, fostering trust among stakeholders. Additionally, the system employs off-chain storage for large Land documents, linked seamlessly to blockchain records for accessibility and verification. The Blockchain-Based Land Registration System is designed to mitigate issues like fraudulent activities, data manipulation, and delays in land registry operations.

It introduces a robust, scalable, and user-friendly platform that can revolutionize Land registration by providing a secure and efficient alternative to conventional methods. The system features a web-based interface that allows users to register properties, verify ownership, and initiate Land transfers. Blockchain technology guarantees that all transactions are recorded in a tamper-proof and transparent manner.

Keywords: Smart Contracts, Immutable Records, KYC(Know Your Customer).

Contents

CHAPTER 1: Introduction

The land registration system is a cornerstone of Land law and is essential for establishing and transferring ownership rights. It serves as the official mechanism by which ownership of land and Land is documented, providing legal proof of title and offering protection against claims from third parties. Traditionally, these systems have been managed by government agencies or legal institutions, which maintain physical or digital records. However, the conventional systems are often fraught with inefficiencies, errors, and vulnerabilities, leading to costly legal disputes, fraudulent activities, and delays in transactions.

Land registration is a fundamental process in Land management, yet traditional systems face several challenges, including inefficiencies, fraudulent activities, and a lack of transparency. These issues often result in disputes, delays, and increased costs for stakeholders. This innovative approach aims to enhance both customer and staff experience, setting a new standard in the hospitality industry.

the Blockchain-Based Land Registration System leverages blockchain technology to create a secure, transparent, and efficient solution. Blockchain technology offers a decentralized and tamper-proof framework for managing land records and Land transactions. By using smart contracts, the system automates critical processes such as ownership verification, Land transfers, and record updates. User authentication is enhanced through Aadhaar and KYC integration, ensuring that only verified users can engage in Land transactions. Additionally, the system incorporates off-chain storage for Land documents, which are securely linked to the blockchain for accessibility and verification.

Land registration systems form the backbone of Land management in any society, acting as a repository of ownership records and a medium for transferring Land rights.

Land registration systems form the backbone of Land management in any society, acting as a repository of ownership records and a medium for transferring Land rights. The Blockchain-Based Land Registration System aims to address these issues by leveraging blockchain technology—a decentralized, secure, and transparent digital ledger system. Blockchain's inherent features, such as immutability, transparency, and decentralized control, make it a perfect fit for land registration, where integrity and trust are of paramount importance.

The implementation of this system also aligns with broader goals of digital transformation and e-governance. Governments worldwide are increasingly adopting digital solutions to improve service delivery, enhance efficiency, and combat corruption.

CHAPTER 2: Literature Review

Blockchain for Land Registration: A Comprehensive Review by R. R. Ali et al. (2021) This review examines the various implementations of blockchain technology in land registration systems, focusing on security, transparency, and efficiency. The authors highlight several case studies where blockchain has been used to digitize Land records and reduce fraud. They analyze the technical aspects of blockchain, such as consensus mechanisms and smart contracts, and explore the challenges of adopting blockchain in traditional land registration frameworks. The paper also discusses the legal and regulatory hurdles that need to be addressed for successful implementation. Furthermore, it highlights the potential for blockchain to streamline real estate transactions and improve access to Land data globally.

Blockchain-Based Land Registration and Real Estate Transactions" by M. I. Shahraki et al. (2020) This paper delves into how blockchain can address inefficiencies in land registration and real estate transactions, particularly in countries with weak Land rights. It emphasizes how blockchain can reduce land disputes and improve data integrity by providing a tamper-proof and transparent record of ownership. The study also explores the role of smart contracts in automating Land transactions, including title transfers and payments, making the process faster and more cost-effective. Challenges such as technological infrastructure, regulatory frameworks, and data privacy concerns are discussed in depth. The authors conclude that blockchain can be a transformative solution to existing land registration problems.

Smart Contracts and Blockchain for Land Transactions by J. C. Smith and L. L. Roberts (2019) This paper focuses on the integration of smart contracts with blockchain technology for Land transactions and land registration. It explains how smart contracts can automate the verification, transfer, and recording of land titles, improving efficiency and reducing human error. The authors discuss how blockchain's decentralized nature eliminates the need for intermediaries such as notaries and banks, lowering costs and time delays. They also explore legal issues related to the enforcement of smart contracts in land transactions. The paper provides insights into how this integration could modernize land registration systems, especially in developing economies. the queue until their turn comes.

Blockchain Technology for Secure and Transparent Land Title Management" by L. Zhang and W. Xie (2021) This literature review provides an in-depth analysis of the role of blockchain in ensuring secure and transparent land title management. It outlines the potential of blockchain to create an immutable digital ledger for land ownership records, offering a solution to common problems such as title fraud and disputes. The authors analyze several international case studies where blockchain has been successfully implemented in land registration systems. They discuss the advantages of blockchain, such as real-time updates and public accessibility, which improve transparency. However, the review also highlights challenges such as data interoperability, regulatory frameworks, and the need for public awareness in adopting the technology.

Blockchain for Real Estate: Challenges and Opportunities" by A. L. Molnar T. F. Kowalski (2022) This review investigates the challenges and opportunities of integrating blockchain into real estate systems, with a focus on land registration. The authors explore various blockchain models, including public and private blockchains, and analyze their suitability for different jurisdictions.

The review also addresses the technical barriers to widespread adoption, such as scalability, regulatory acceptance, and the need for stakeholder collaboration. Finally, the paper discusses pilot projects and how they can pave the way for large-scale adoption.

CHAPTER 3: Design, Development and Drawing

3.1 Proposed Work

3.1.1 Problem Definition

The dining industry faces increasing pressure to provide a seamless and efficient customer experience in a fast-paced, technology-driven world. Traditional reservation andordering processes often involve manual steps, leading to potential delays, errors, and customer dissatisfaction. Customers today expect instant access to services, personalized recommendations, and quick responses to their queries, yet many existingsystems fall short of these expectations. This project aims to address these issues by developing a smart restaurant reservation and ordering system that leverages AI technology to automate and personalize the customer experience. By integrating an AI chatbot to handle reservations, offer tailored menu suggestions, and facilitate order placement, this project seeks to create a more engaging, efficient, and error-free diningexperience that aligns with modern customer demands.

3.1.2 Project Objective

1. To automate reservations and ordering efficiently.
2. To integrate AI chatbot for personalized assistance
3. To optimize customer data management securely.

3.2 Design

3.2.1 System Architecture

1. **User Interface (UI)**

The architecture of the Smart Restaurant Reservation and Ordering Systemwith AI Chatbot Integration is designed to deliver a seamless, efficient, and responsive user experience while ensuring scalability, security, and reliability in operation. The architecture consists of several layers, each playing a crucial role in the system's overall functionality. Below is a detailed explanation of the key components and their interact

The User Interface (UI) is the front-end layer where customers interact with the system. It provides an intuitive platform for making reservations, ordering food, and interacting with the AI-powered chatbot. The key components of the UI are:

- **Customer Reservation Form:** Enables users to book a table, view available slots,and confirm reservations.
- **Order Placement Section:** Allows customers to browse the menu, add items to their cart, and submit their orders.
- **Menu Display:** Shows detailed information about food items, including descriptions, prices, and images.
- **Real-time Chat Assistance:** A chatbot interface that helps customers with reservations, order inquiries, and menu suggestions. The chatbot is powered by AI and provides an interactive, conversational experience.

The UI is built with a responsive design, ensuring compatibility across different devices (desktop, tablets, and smartphones). It uses frameworks like **React.js** and **Next.js** to provide dynamic, fast, and scalable front-end functionalities.

2. AI Chatbot

The AI Chatbot acts as the virtual assistant for customers, enabling natural language interactions. It assists users in real-time with various tasks, such as:

- **Reservations:** Helps customers book tables by checking available slots and confirming bookings.
- **Personalized Recommendations:** Analyzes customer preferences, order history,and dietary restrictions to provide tailored suggestions.
- **Order Assistance:** Provides detailed information about menu items, helps modify orders, and answers any queries related to the menu.

The chatbot leverages **Natural Language Processing (NLP)** technologies, such as

Dialogflow, **OpenAI's GPT**, or **Rasa**, to interpret user inputs and generate context-awareresponses.

3. **Backend Server**

   The backend server is responsible for processing requests from the front end (UI and chatbot), handling the business logic, and ensuring seamless communication between various components. The key functions of the backend server include:

- **API Layer:** Manages communication between the user interface, AI chatbot, and database.
- **Business Logic:** Processes restaurant-related operations, such as managing reservations, processing orders, and updating the menu.
- **Data Processing:** Handles real-time data updates (e.g., order status, reservation status) and synchronizes information across all system components.

  The backend is typically developed using **Node.js**, **Python (Flask/Django)**, or similar technologies that provide scalability and performance.

4. **Database**

   The database is the central repository for storing and managing all data related to customers, orders, reservations, and menu items. Depending on the design, a **SQL (MySQL)** or **NoSQL (MongoDB)** database can be used. The database handles:

- **Customer Data:** Information such as user profiles, preferences, and order history.
- **Order Data:** Tracks the details of each customer’s order, including items,quantities, and status.
- **Reservation Data:** Stores table reservations, including customer details, date, and time.
- **Menu Data:** Contains information about available menu items, prices,ingredients, and special offers.

  The database ensures fast and reliable access to data, with secure storage and efficientretrieval mechanisms to support real-time customer interactions.

5. **Restaurant Admin Panel**

   The Admin Panel is designed for restaurant staff to manage day-to-day operations, trackorders, handle reservations, and interact with customers. The key functions of the Admin

Panel include:

- **Reservation Management:** View, add, and update customer reservations.
- **Order Management:** Monitor the status of orders (pending, preparing, ready to serve), and update order status.
- **Customer Interaction:** Manage customer feedback, handle special requests, and resolve issues.

The Admin Panel connects to the backend server to ensure real-time updates on orders, reservations, and customer interactions. It is typically accessed through a web-based interface.

6. **Integration Layer**

The integration layer facilitates communication between different modules (UI, backend,AI chatbot, and database). It ensures that data flows smoothly between the front end, backend, and database, allowing for a seamless user experience. This layer also integratesthird-party services such as payment gateways (e.g., **Stripe**) for secure transactionprocessing and cloud services (e.g., **Cloudflare Pages and Workers**) for hosting the platform.

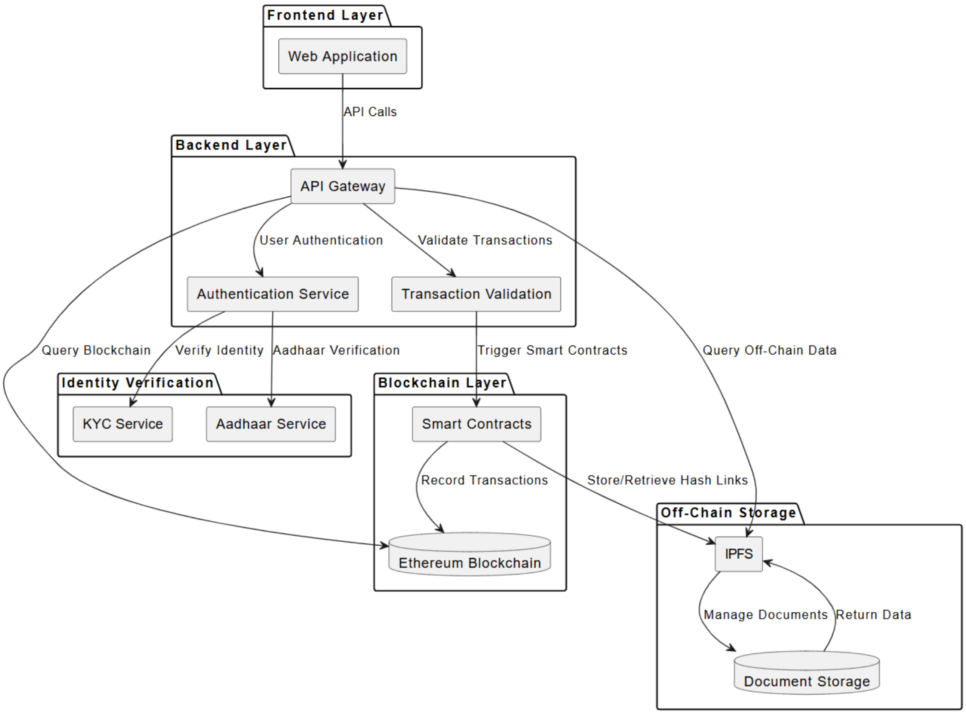


Figure 1: System Architecture

3.3 Development

3.3.1 Software Requirements Specifications

1. **Frontend Development**:

- React.js: For building an interactive and dynamic user interface.
- HTML5, CSS3, JavaScript: To ensure compatibility and modern webstandards.

2. **Backend Development:**

- Node.js: For efficient server-side logic and API development.
- Express.js: To simplify backend routing and middleware setup.

3. **Database Management:**

- MongoDB: A scalable NoSQL database for managing reservations,orders, and user data.

4. **AI Integration:**

- Dialogflow: For chatbot development and AI-driveninteractions.
- Rasa: For building custom, open-source conversational AI chatbots with advanced natural language understanding (NLU).

5. **Design Tools:**

- Figma: For UI/UX design and prototyping a visually appealinginterface.

6. **Version Control:**

- Git: For managing code changes.
- GitHub: For collaborative development and version control.

7. **Deployment:**

- Vercel: For deploying and hosting the web application with optimizedperformance for frontend frameworks like React.js or Next.js.
- Heroku or AWS: For deploying the application and ensuring

8. Additonal Tools:

- Postman: For API testing.
- Visual Studio Code: For code development.

3.3.2 Hardware Requirements Specifications

1. **Server-Side Requirements**

- **Minimum Configuration**:
  - 4-core CPU
  - 4GB RAM
  - 200 GB SSD
- **Recommended Configuration**:
  - 16-core CPU
  - 8 GB RAM
  - 500 GB SSD

2. **Client-Side Requirements**

- Hardware:
  - Dual-core processor
  - 4 GB RAM (minimum)
  - Modern web browsers (Chrome, Firefox, Safari, Edge)

3. **Development Environment**

- Hardware:
  - Quad-core processor
  - 4 GB RAM
  - 256 GB SSD
- Optional:
  - High-speed internet connection for seamless third-party API integrationand Docker container management

3.4 Drawing

3.4.1 Data Flow Diagram Level 0

This is a Zero Level Data Flow Diagram (DFD), also known as a Context Diagram, which provides a high-level overview of a system and its interactions with external entities. Here's an explanation of the diagram:

1. Entities:
   - Admin: Represents an external entity responsible for managing or maintaining the system. The Admin might perform tasks such as managing users, handling data, or monitoring the system.
   - User: Represents individuals or entities that interact with the system to perform tasks such as submitting requests, retrieving information, or using the system's services.
2. System:
   - This central process represents the main application or platform that facilitates the interaction between the Admin and the User. It processes inputs and provides outputs based on interactions with the external entities.
3. Data Flows:
   - The arrows between the entities and the system represent the flow of data. For example:
     - The Admin sends data or instructions to the system, such as updates, configurations, or management actions.
     - The system responds to the Admin with feedback or status updates.
     - Similarly, the User interacts with the system by providing input (e.g., login details, queries, or requests).
     - The system provides the User with responses or results based on their interaction.

This DFD does not go into the details of internal processes or data stores within the system but focuses on how the system interacts with the external entities. It's a starting point for understanding the boundaries and primary interactions of the system.

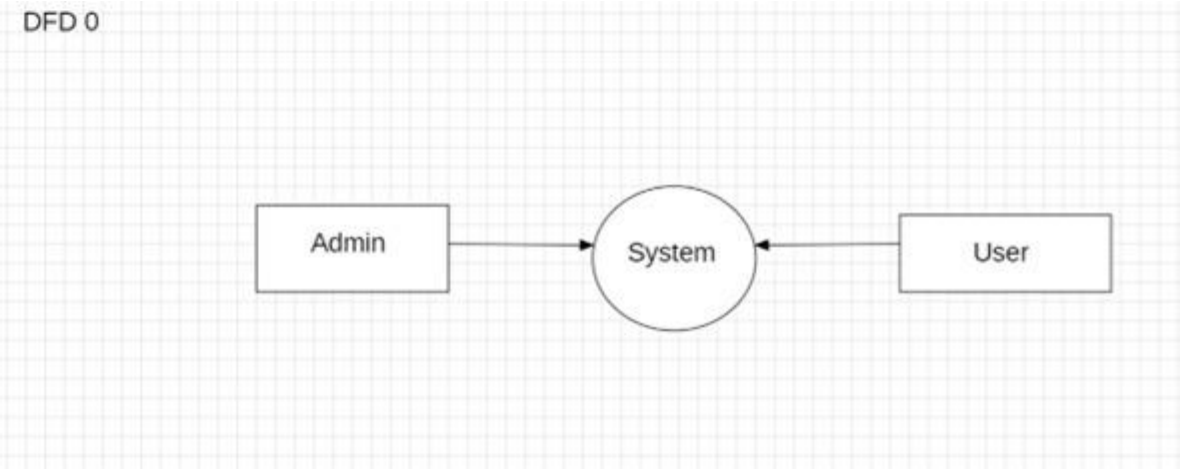


Figure 2: Data Flow Diagram level 0

3.4.2 Data Flow Diagram Level 1

This is a First-Level Data Flow Diagram (DFD) for a Land Registration System. A first-level DFD expands on the high-level processes and data flow in the system, providing more details compared to a context diagram.

1. **Central Process:**

- The Land Registration System is the central process that handles interactions between multiple modules and data flows.

2. **Processes:**

- Approval Management: Handles the approval of requests or actions within the system.
- Buyer Management: Manages the data and activities related to buyers.
- Seller Management: Manages data and actions related to sellers.
- Check Approval Request: Verifies and processes approval requests submitted by users.
- Report Generation: Includes generating various types of reports.
- Generate Property Report
- Generate Buyer Report
- Generate Seller Report
- Generate Land Type Report

3. **Data Flow:**

- The system receives input from management modules (e.g., Buyer Management, Seller Management, Approval Management).
- It processes these inputs to produce outputs such as reports or to handle approval requests.

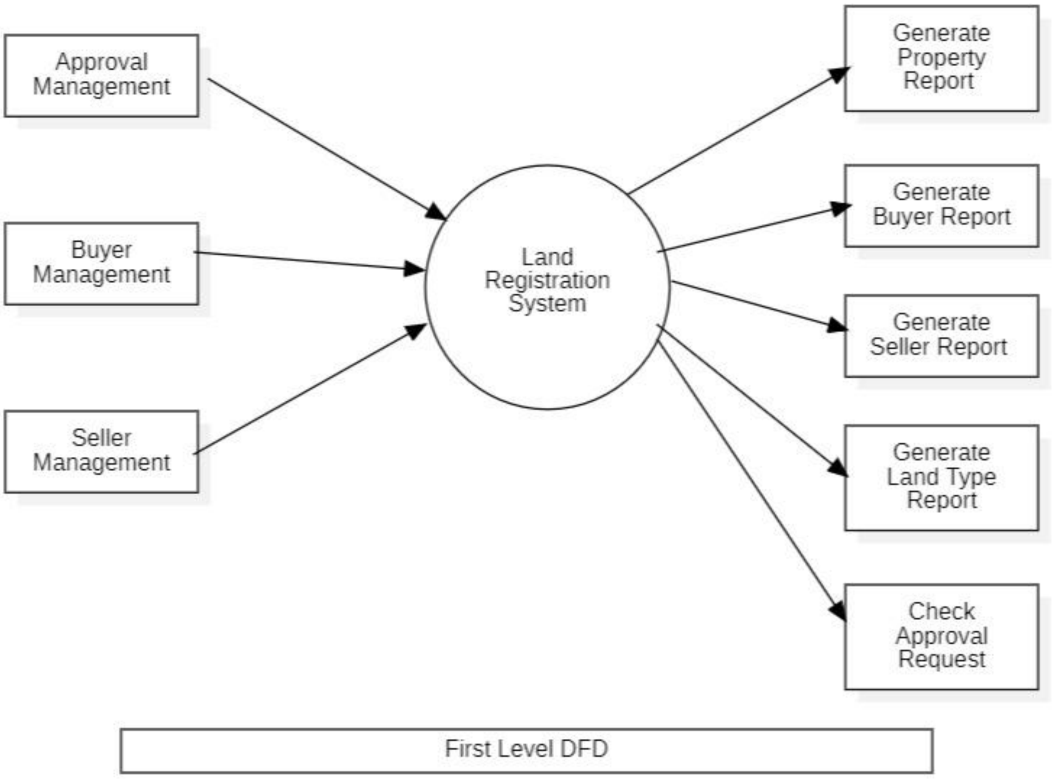


Figure 3: Data Flow Diagram level 1

3.4.3 Data Flow Diagram Level 2

This is a Second-Level Data Flow Diagram (DFD) for a Land Registration System. A DFD visually represents how data flows within a system and focuses on the processes, data stores, and the interactions between them.

1. **Processes (Circles):**

- Login to System: Represents the user authentication process.
- Check Credentials: Verifies user-provided credentials.
- Check Role of Access: Determines the permissions and access level of the user (e.g., admin, buyer, seller).
- Manage Modules: Central process managing core functionalities like land details, buyer details, seller details, etc.
- Forgot Password: A recovery process triggered when users forget their credentials.
- Send Email to User: Sends password recovery or system-related emails.
- Manage Roles of User & User Permissions: Handles user roles and defines their respective access rights.

2. **Data Stores (Rectangles):**

- Represent areas where information is stored or organized, such as:
- Manage Land Details
- Manage Buyer Details
- Manage Seller Details
- Manage Land Type
- Approval Management
- Registration Profile Management
- Cost Report

3. **Interactions:**

- Admin Interaction: Admin has primary control over processes like login, managing modules, handling approvals, and managing roles and permissions.
- The processes are interconnected with data stores and user roles, ensuring smooth data flow across modules.

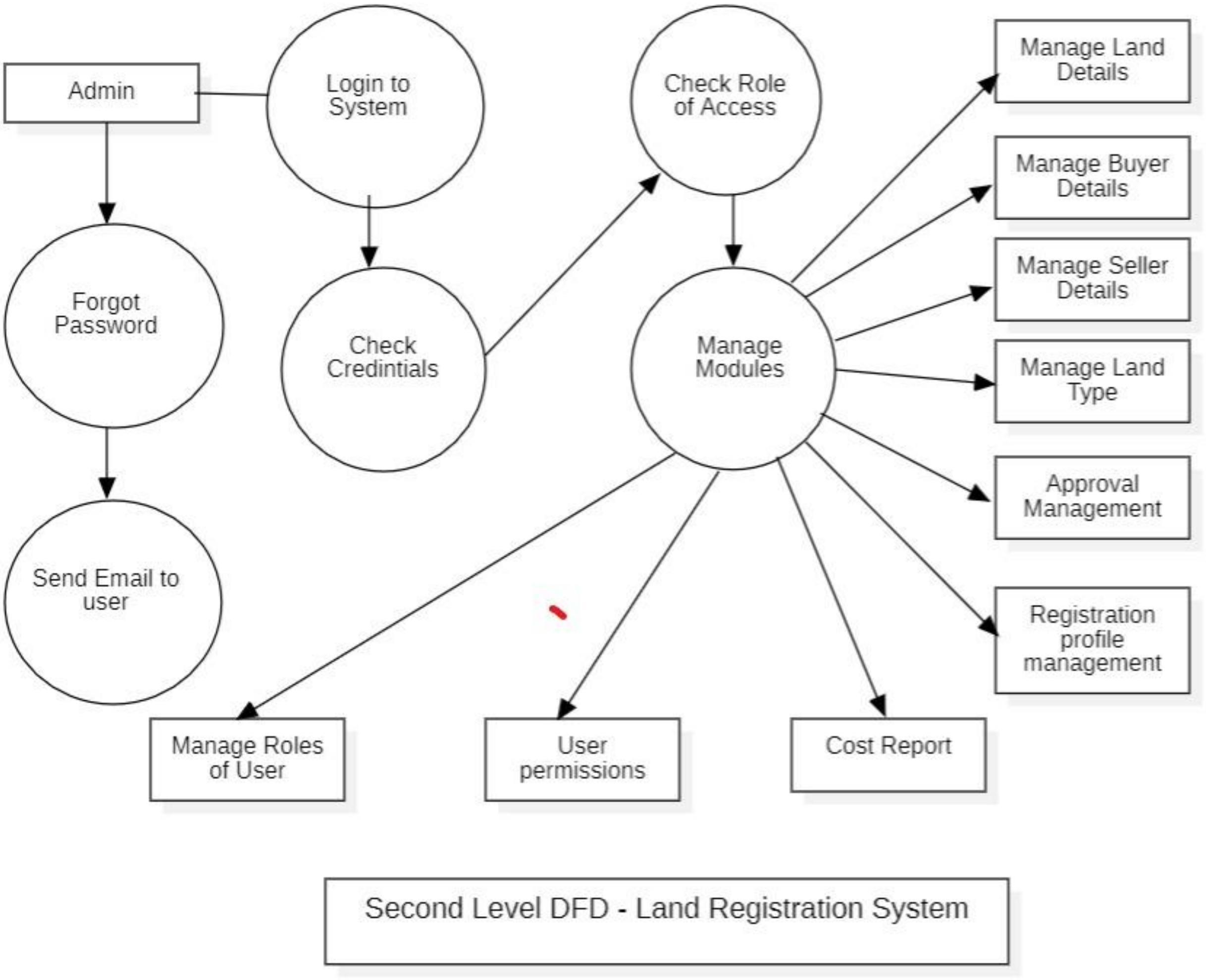


Figure 4: Data Flow Diagram level 2

3.5 UML Diagram

3.5.1 Use Case Diagram

This is a use case diagram for a Land Registration System. Use case diagrams are part of Unified Modeling Language (UML) and are used to visualize the functionalities of a system and the interactions between actors and the system.

1. **Actors:**

- System Admin**:** Responsible for administrative functionalities like managing users, buyers, sellers, and applications.
- System User: Likely a staff member or user with specific privileges to manage system-related data like property types, registration, and costs.
- Seller: A user who interacts with the system to create property-related activities like creating quotations or sending emails.
- Buyer: A user who uses the system to search for properties, view details, or request quotations.

2. **Use Cases:** These represent the functionalities of the system, such as:

- Manage Users & Applications: Allows the admin to control system access and handle applications.
- Login & Logout Profile, Update Profile, Change Account Password: Common functionalities for all users to manage their account and sessions.
- Search Property, View Details, Request for Quotations: Functionalities specific to buyers.
- Manage Buyers, Sellers, Property Type, Registration, and Cost: Administrative tasks likely for both the system admin and system user.
- Create Quotations, Check Enquiry, Send Email: Seller-specific functionalities for interacting with buyers.

3. **Relationships:**

- Each actor is connected to the relevant use cases they interact with.
- Lines between the actors and use cases indicate direct interaction.

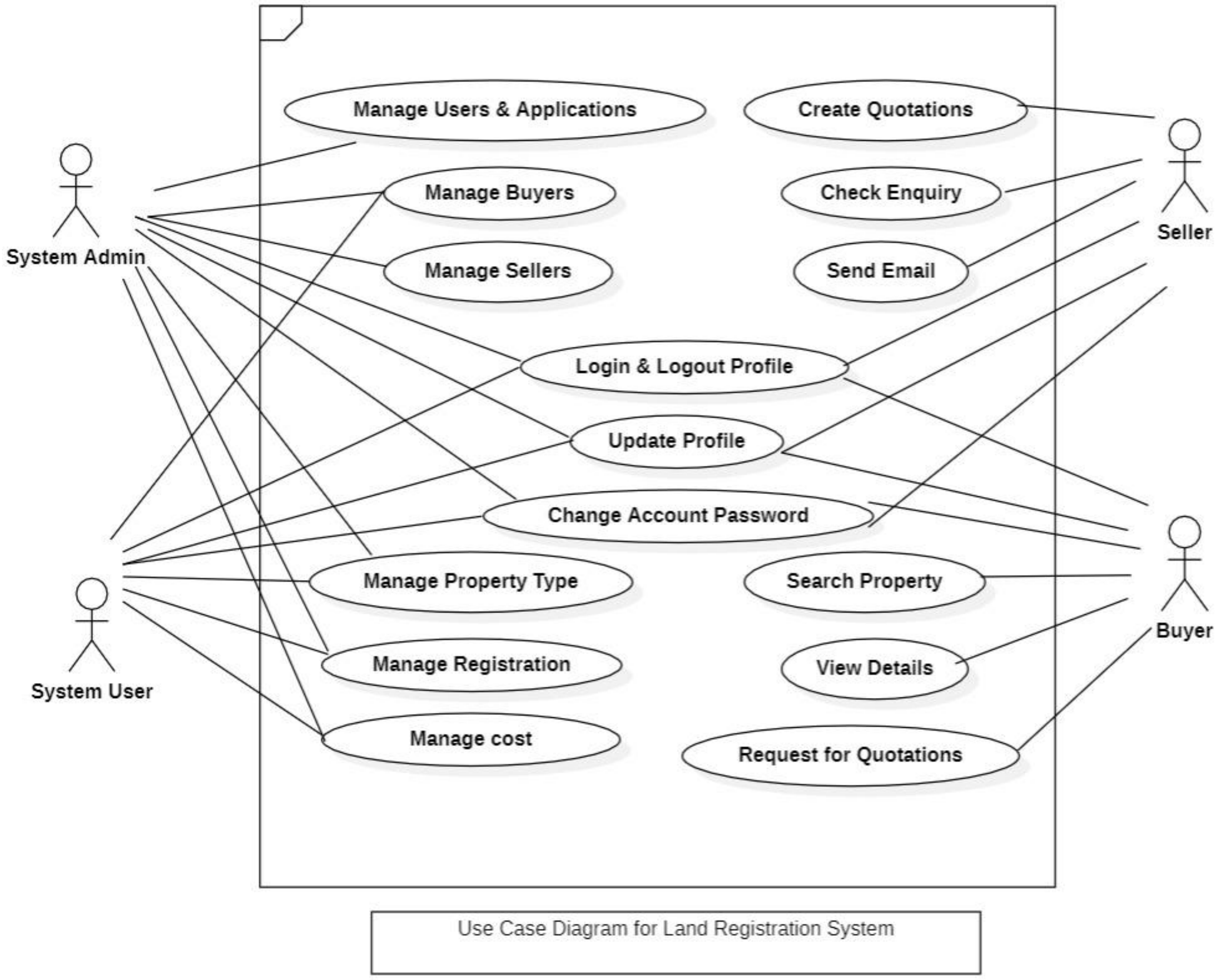


Figure 5: Use Case Diagram

3.5.2 Sequence Diagram

The image you've uploaded is a Sequence Diagram for a Blockchain-based Land Registration System. This diagram models the interactions between different system components over time. The diagram breaks down the process into five main entities or "lifelines" (shown as vertical dashed lines), each representing a key component of the system:

1. **User**: The individual submitting the land registration request.
2. **Backend**: The server-side component that handles requests and communicates with other system components.
3. **Admin**: The person responsible for verifying the ownership and documents associated with the land.
4. **SmartContract**: The component where the smart contract is executed to verify and store the transaction on the blockchain.
5. **Blockchain**: The decentralized ledger where transactions are stored securely.
6. **Frontend**: The user interface that communicates the results (confirmation or failure) back to the user.

Sequence of Events:

1. **User** initiates the process by submitting a land registration request. This is the first action that triggers the sequence.
2. **Backend** receives the request from the user and forwards it to the Admin for verification. This includes verifying the ownership and any associated documents.
3. **Admin** verifies the request, and once confirmed, the Backend sends the data to the **Smart Contract** for processing.
4. **SmartContract** executes the contract based on the received data. This step involves validating the transaction and ensuring that the land registration meets all the required criteria.
5. Once the Smart Contract has been executed, it sends the status of the transaction (either success or failure) back to the Blockchain.

6. **Blockchain** stores the transaction on the ledger. If the registration is valid and the smart contract executed successfully, the transaction is confirmed and stored securely. Finally, the Blockchain sends the transaction status back to the Frontend, which d is plays either a confirmation or failure message to the User

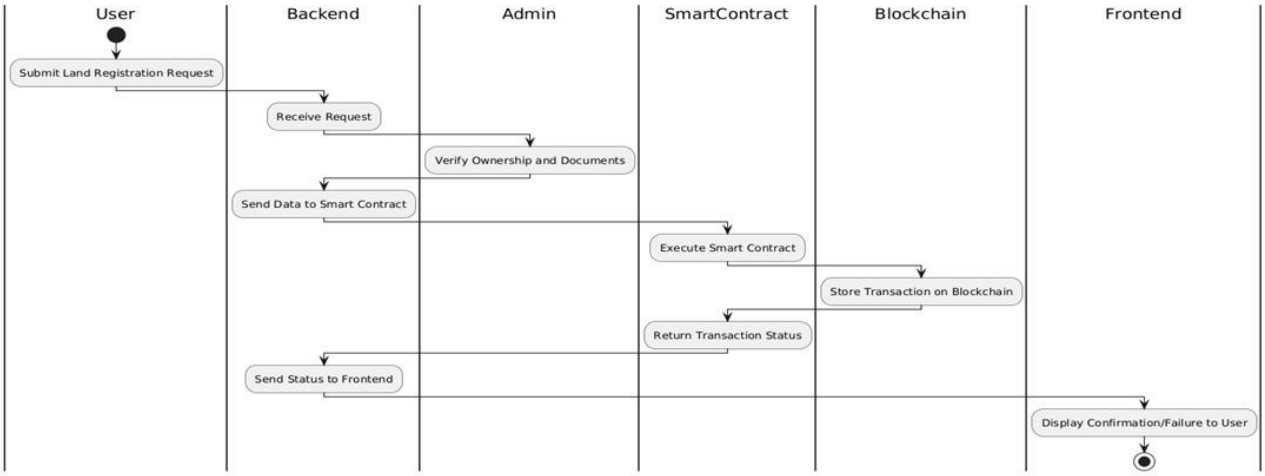

User
Backend
Admin
SmartContract
Blockchain
Frontend
Submit Land Registration Request
Receive Request
Verify Ownership and Documents
Send Data to Smart Contract
Execute Smart Contract
Store Transaction on Blockchain
Return Transaction Status
Send Status to Frontend
Display Confirmation/Failure to User

Figure 6: Sequence Diagram

CHAPTER 3. DESIGN, DEVELOPMENT AND DRAWING

3.5.3 Deployment Diagram

This is a Deployment Diagram that represents the physical architecture of a blockchain-based system. Deployment diagrams are used to model the distribution of software components across hardware nodes. Here’s an analysis of this diagram:

This deployment diagram showcases the interaction between various hardware and software components in a blockchain-based system. It provides an overview of how services are deployed, how data flows between components, and how user requests are handled.

Components Overview

1. **User Device**:
   - Represents the end-user's hardware (e.g., desktop, laptop, or mobile device).
   - Runs a Web Browser to interact with the system, making HTTP requests to the Frontend Server.
   - Acts as the entry point for the system.
2. **Frontend Server**:
   - Hosts the Web Application that provides the user interface for accessing the system.
   - Handles HTTP requests from the user device and communicates with the Backend Server through API calls.
   - Acts as a bridge between the user device and the backend services.
3. **Backend Server**:
   - The central node that processes and routes requests. Includes:
     - API Gateway: Manages and directs API calls to specific backend services.
     - Authentication Service: Authenticates users to ensure secure access.
     - Transaction Validator: Validates user transactions to ensure compliance before sending them to the blockchain.
   - Acts as the core processing unit of the system.
4. **Identity Verification Service**:
   - Ensures that users are legitimate by verifying their identity.
   - Includes:
     - KYC Service: Performs Know Your Customer checks.

- Aadhaar Service: Integrates Aadhaar verification for identity authentication (specific to India).

5. **Blockchain Network**:

- Handles decentralized, immutable record-keeping.
- Components:
- Smart Contracts: Automate the execution of business rules and agreements.
- Ethereum Blockchain: Provides the decentralized ledger for storing transactions securely.
- Acts as the core technology ensuring transparency, security, and immutability.

6. **Off-Chain Storage**:

- Manages the storage of large or sensitive documents that are impractical to store directly on the blockchain.
- Components:
- IPFS (InterPlanetary File System): Decentralized storage for documents, providing hash-based references.
- Document Storage: The physical or logical repository for documents.
- Links data stored off-chain to the blockchain using hash references for integrity.

Workflow Summary

1. User Interaction:

- The user interacts with the system via their Web Browser.
- Requests are sent to the Frontend Server.

2. Processing Requests:

- The Frontend Server forwards requests to the Backend Server via API calls.
- The Authentication Service verifies the user.
- The Transaction Validator processes the request.

3. Identity Verification:

- If necessary, the system uses the Identity Verification Service to validate the user’s identity via KYC or Aadhaar.

4. Blockchain Interaction:

- Valid transactions trigger Smart Contracts on the Blockchain Network.
- Transactions are immutably recorded in the Ethereum Blockchain.

5. Off-Chain Storage:

- Large documents or files are stored in IPFS and linked to the blockchain via cryptographic hashes.
- Hashes ensure document integrity and allow retrieval when needed.

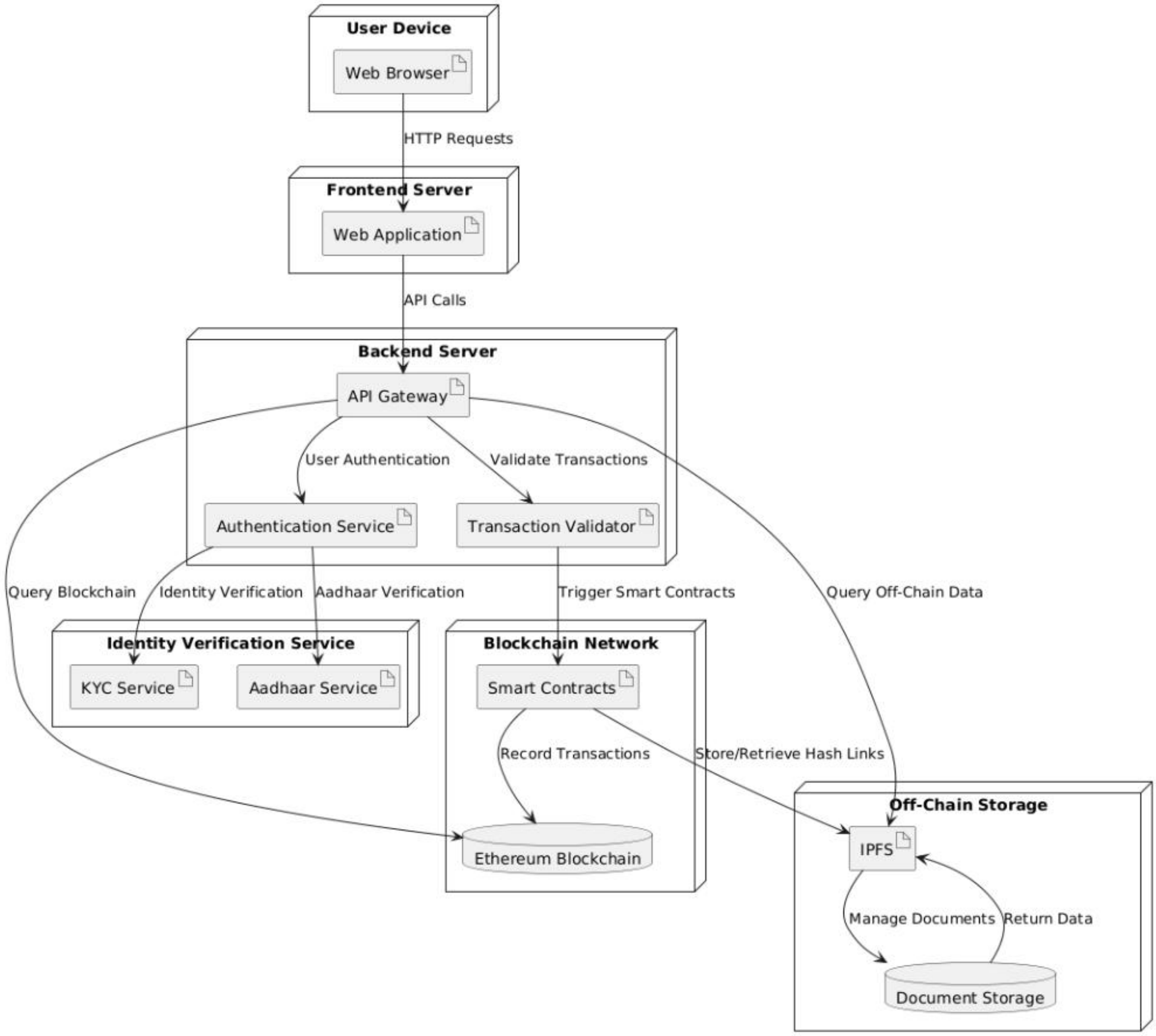


Figure 7: Deployment Diagram

3.5.4 Activity Diagram

This Activity Diagram represents the workflow of a Blockchain-based Land Registration System and shows the sequence of activities involved in the registration process. It illustrates how the system progresses through different stages, from submitting the registration request to confirming or failing the registration, with specific actions and decisions along the way. Here's an overview of the diagram and the key steps in the process:

Activity Diagram Overview:

1. **Idle State** (Starting Point):

o The system begins in an idle state, waiting for a user to initiate a registration request.

2. **Submit Registration Request**:

o The user submits a land registration request to the system.

o This action transitions the system from the Idle state to the RegistrationSubmitted state.

3. **Admin Verification**:

o Once the registration is submitted, the system enters the AdminVerification state.

o The admin is required to review the registration request and decide whether to approve or reject it.

4. **Admin Decision**:

o The admin's decision results in one of two possible paths:

▪ If the admin approves the registration, the process moves to the VerificationApproved state.

▪ If the **admin rejects** the registration, it moves to the VerificationFailed state, indicating the registration process is unsuccessful.

5. **Smart Contract Execution** (If Approved):

o If the admin approves the registration, the process moves to Smart Contract Execution.

o In this state, the system interacts with the blockchain to execute a smart contract, which is responsible for ensuring that the transaction is processed according to predefined rules.

6. **Transaction Status**:

- After executing the smart contract, the system checks whether the transaction is successful:
  - If the transaction is completed successfully, the registration moves to the RegistrationConfirmed state, meaning the land is officially registered.
  - If the transaction fails, the process transitions to the RegistrationFailed state, signaling that the registration process was unsuccessful.

7. **End States**:

- The diagram ends in either RegistrationConfirmed (successful registration) or RegistrationFailed (unsuccessful registration), depending on the outcome of the admin's approval and the smart contract execution.

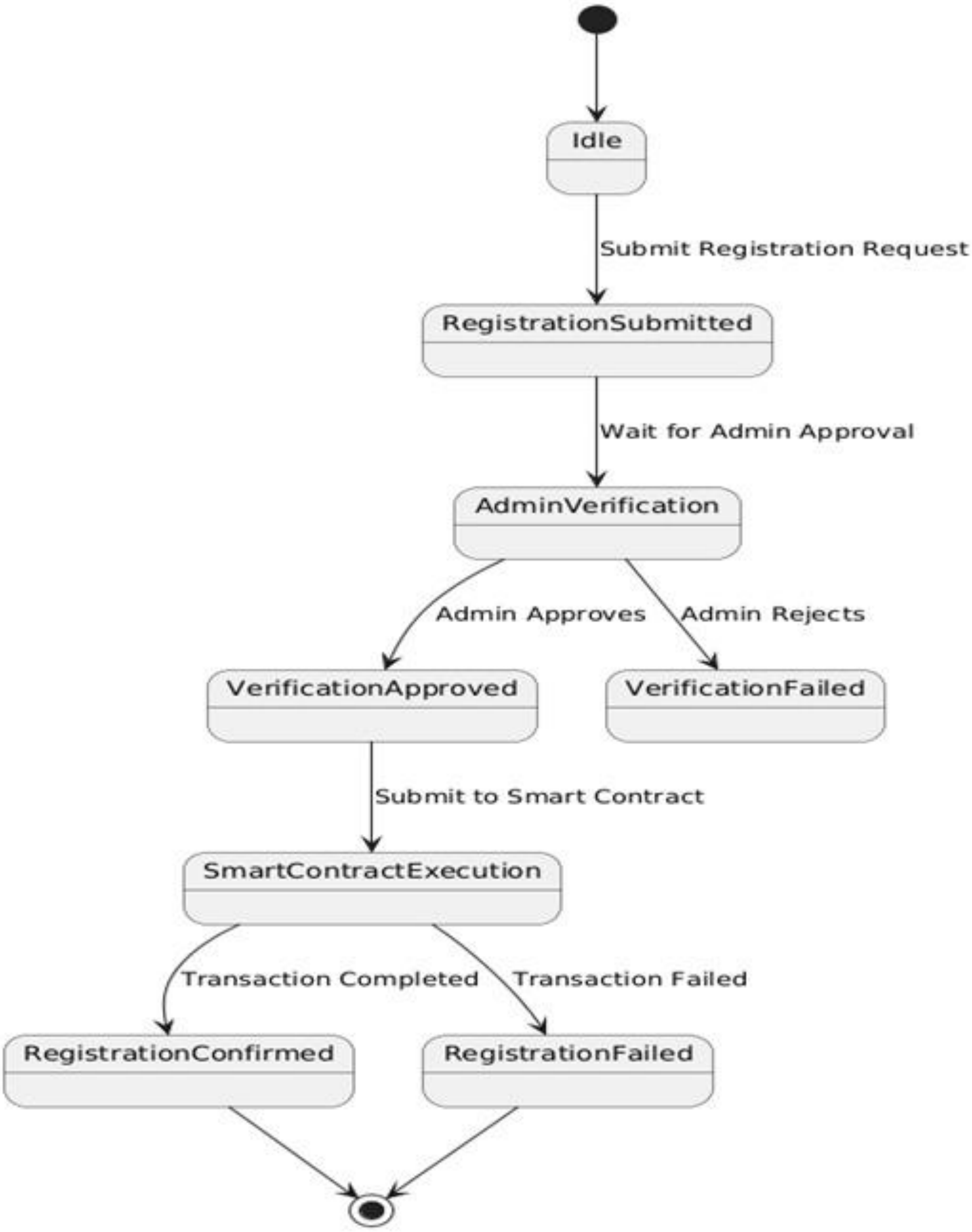


Figure 8: Activity Diagram

3.5.5 Class Diagram

This class diagram represents a Land Registration System, which is designed to manage and organize various aspects of Land transactions, including user roles, Land details, registrations, and buyer/seller information. Here's an overview of the components and their roles:

Classes and Their Roles:

1. **Role Class**:
   - This class manages the roles of users within the system (e.g., admin, buyer, seller).
   - Attributes include role details such as role_id, role_title, and role_description.
   - Operations include adding, editing, and deleting roles (addRole(), editRole(), deleteRole()).
2. **Permission Class**:
   - Defines permissions for each role (e.g., viewing properties, managing users).
   - Attributes include permission_id and permission_description.
   - The methods allow adding, editing, and deleting permissions for roles.
3. **User Class**:
   - Represents a user within the system, which could be an admin, buyer, or seller.
   - Attributes include user_id, user_name, user_email, and user_address.
   - Operations manage user details such as adding, editing, or deleting a user.
4. **Buyer Class**:
   - Represents a buyer who is looking to purchase properties.
   - Attributes include buyer_id, buyer_name, buyer_mobile, buyer_email, etc.
   - Methods allow adding, deleting, and searching buyer information.
5. **Seller Class**:
   - Represents the seller of properties.
   - Similar to the Buyer class, it contains attributes such as seller_id, seller_name, and seller_email.
   - The operations are focused on managing seller details like adding, deleting, and searching.
6. **Land Class**:
   - Manages the properties available for sale, with attributes such as Land_id,

Land_name, Land_description, and Land ownership details (Land_buyer_id, Land_seller_id).

- It also includes methods to add, edit, delete, and search properties.

7. **Land Type Class**:

- Defines different types of properties (e.g., residential, commercial).
- Attributes include Land_type_id, Land_type_name, and Land_type_description.
- Operations allow adding, editing, and searching Land types.

8. **Registration Class**:

- Manages the registration of properties within the system, storing details such as registration type, description, and date.
- Attributes like registration_id and registration_number are used for tracking the registration status of properties.
- It includes methods for adding, deleting, and searching registrations.

Connections Between Classes:

- Role Class is linked to Permission Class, as permissions are associated with specific roles.
- User Class interacts with the Buyer Class, Seller Class, and Registration Class to manage user-related operations in the system.
- Land Class is related to both Buyer Class and Seller Class, as properties are bought and sold by these users.
- Land Type Class categorizes the properties within the system.

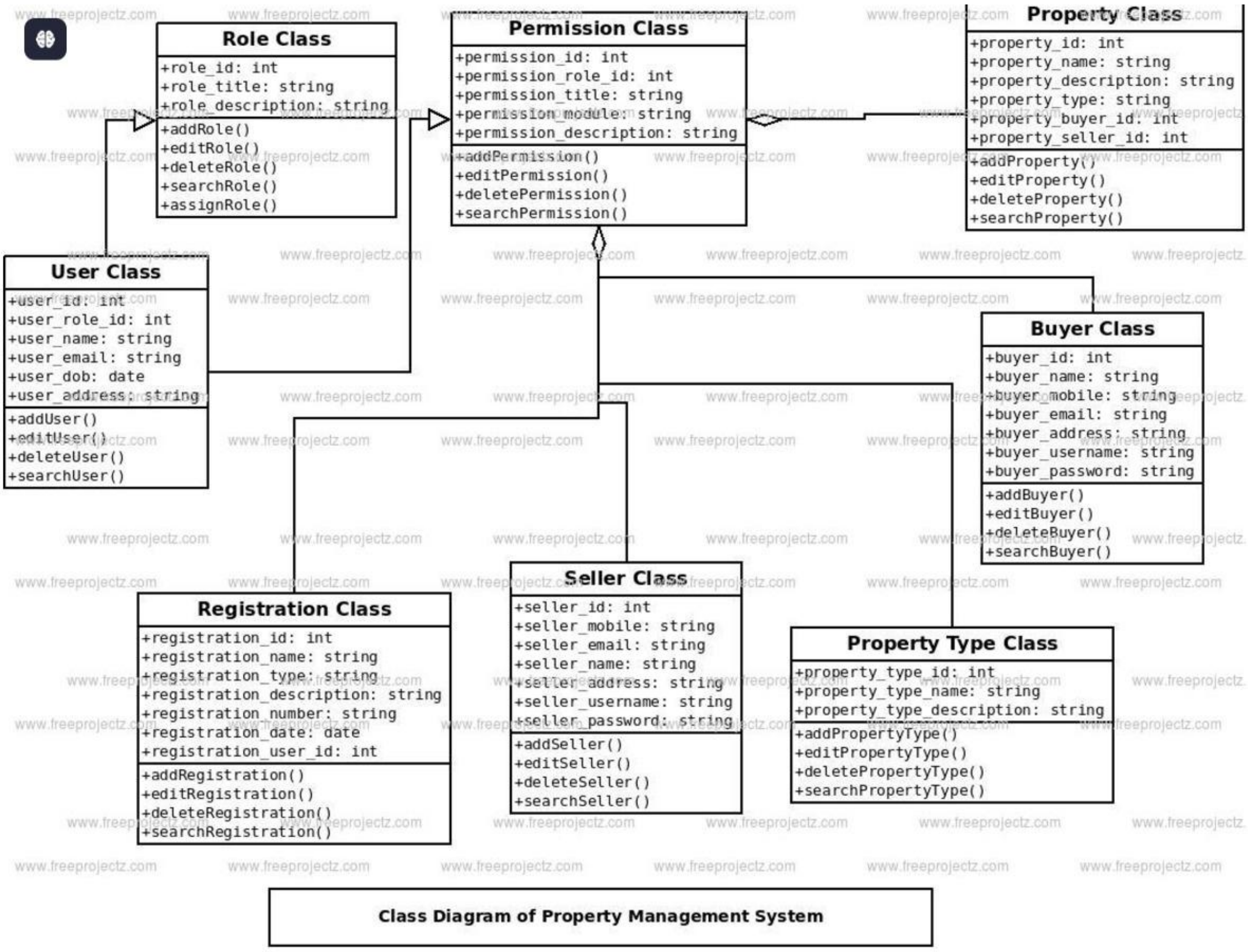


Figure 9: Class Diagram

3.5.6 Object Diagram

This object diagram represents a system for managing property transactions using blockchain technology. It integrates several components to provide a secure and transparent process for handling property ownership, transactions, and associated documents.

At the center of the system is the User, who represents individuals interacting with the system, such as buyers or sellers of property. Each user is uniquely identified and authenticated through the AuthenticationService, which verifies their credentials and ensures they are authorized to perform actions within the system.

When a user initiates a transaction, the Transaction object comes into play. This object captures key details such as the property being transacted, the parties involved (buyer and seller), the transaction status, and a timestamp. The transaction also references a Property object, which stores information about a specific property, including its unique identifier, owner details, location, size, and associated documents (e.g., deeds or legal papers).

The system employs SmartContracts to define and enforce the terms of property transactions. These contracts automate processes, ensuring that all conditions are met before finalizing the transaction. Once validated, relevant documents are stored securely using OffChainStorage. This component is crucial for managing large files, as it stores them externally while maintaining a reference to their integrity via cryptographic hashes.

Finally, the Blockchain serves as the backbone of the system, recording all critical data, such as transaction details and document hashes, in a tamper-proof manner. Each block in the blockchain links to the previous one through a cryptographic hash, ensuring data integrity and preventing unauthorized changes.

The system integrates these components to enable seamless property transactions, ensuring security, transparency, and trust. It is particularly suited for applications requiring robust record-keeping and automated enforcement of contractual terms.

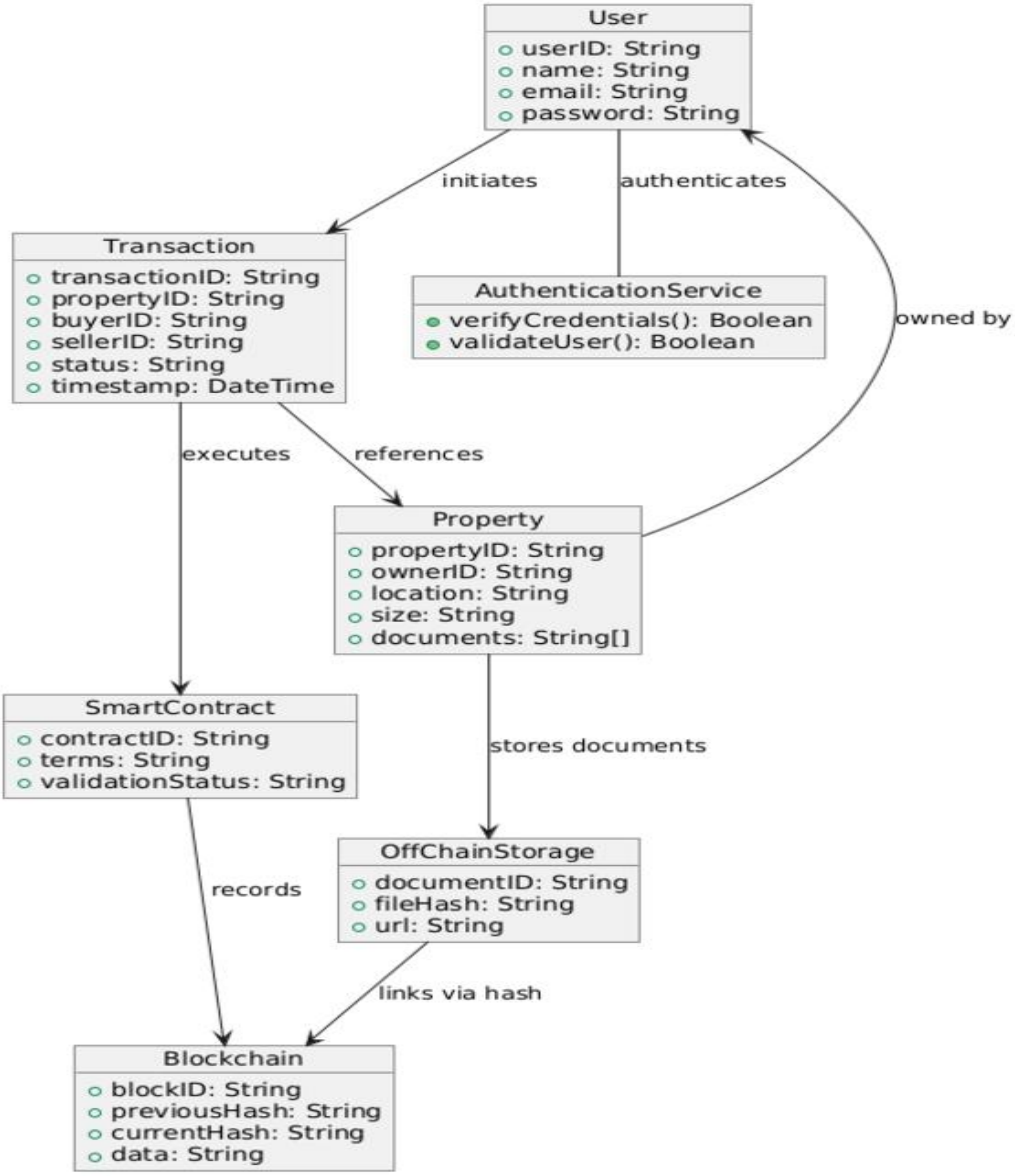
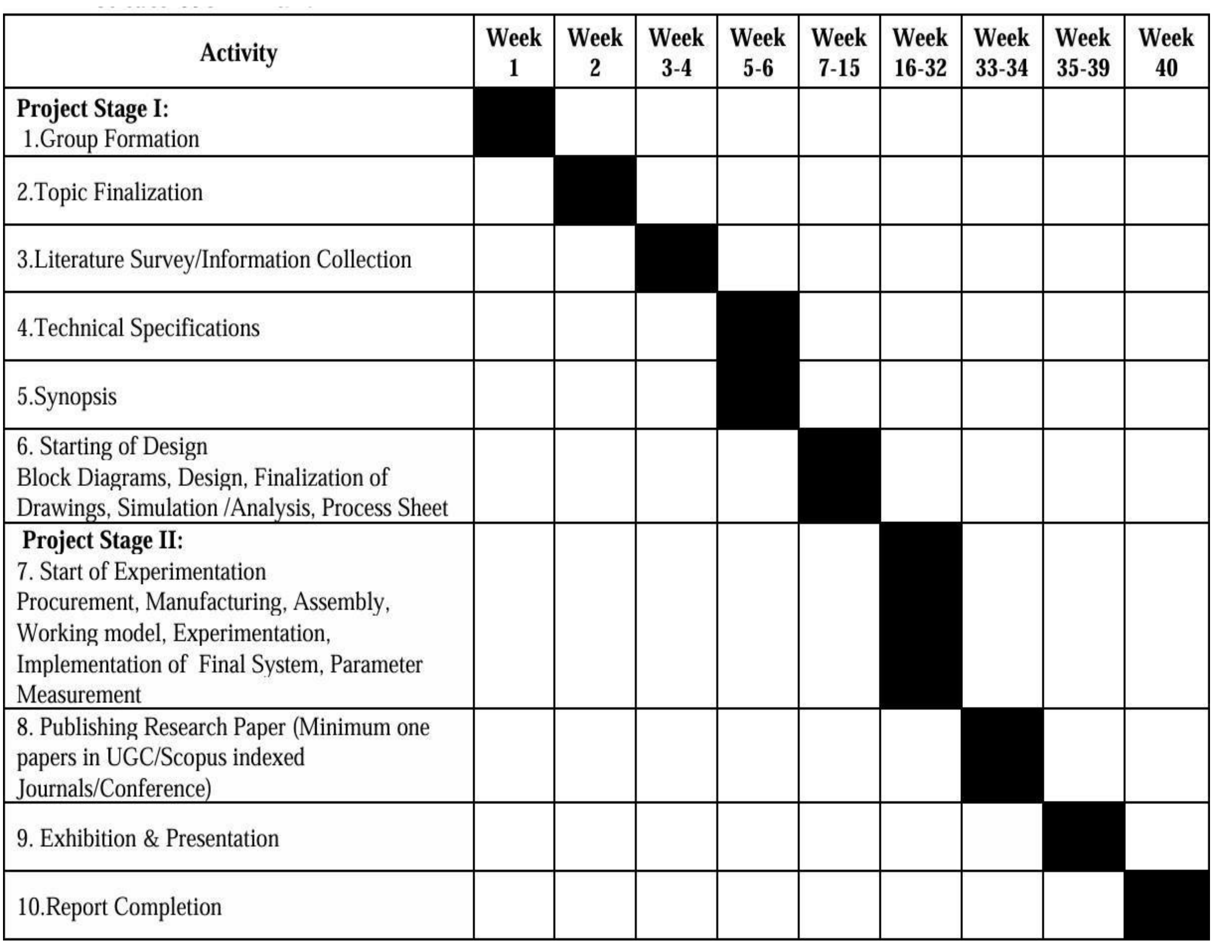


Figure 10: Object Diagram

3.5.7 TIMELINE OF PROJECT



Activity	Week 1	Week 2	Week 3-4	Week 5-6	Week 7-15	Week 16-32	Week 33-34	Week 35-39	Week 40
Project Stage I: 1.Group Formation	■								
2.Topic Finalization		■							
3.Literature Survey/Information Collection			■						
4.Technical Specifications				■					
5.Synopsis				■					
6. Starting of Design Block Diagrams, Design, Finalization of Drawings, Simulation /Analysis, Process Sheet					■				
Project Stage II: 7. Start of Experimentation Procurement, Manufacturing, Assembly, Working model, Experimentation, Implementation of Final System, Parameter Measurement						■			
8. Publishing Research Paper (Minimum one papers in UGC/Scopus indexed Journals/Conference)							■		
9. Exhibition & Presentation								■	
10.Report Completion									■

Figure 11: Timeline

CHAPTER 4: Conclusion And Future Scope

4.1 Conclusion

A blockchain-based land registration system offers a transformative approach to addressing the long-standing challenges of traditional land management. By leveraging blockchain's core attributes—decentralization, transparency, and immutability—the system can prevent fraud, reduce disputes, and streamline processes like Land registration and ownership transfer. Its ability to integrate advanced technologies such as AI for market valuation, GIS for precise land mapping, and automated smart contracts for transactions ensures a comprehensive and efficient solution. Privacy and security are further enhanced through tools like zero-knowledge proofs and biometric authentication, safeguarding user data while maintaining trust and transparency. Moreover, the system can adapt to various legal frameworks, enabling seamless adoption across national and international jurisdictions. Scalable infrastructure and interoperability with other networks make it capable of handling increasing user demands and cross-border transactions. Public awareness and education are vital to overcoming resistance and promoting widespread acceptance. As it continues to evolve, this blockchain solution holds the promise of revolutionizing the real estate sector, ensuring equitable access, secure transactions, and efficient land management, ultimately benefiting governments, institutions, and citizens alike. With its vast potential, the system is poised to become a cornerstone in the future of Land registration and management.

4.2 Future Scope

The future of a blockchain-based land registration system is transformative, with potential to enhance scalability, features, and adoption. Integrating the system with national and international databases will enable seamless operations across jurisdictions, while interoperability with other blockchain networks will facilitate global collaboration. Scalability can be improved using Layer-2 solutions like Polygon, ensuring faster transactions and reduced costs. Advanced features will further elevate its utility. Real-time market valuation powered by AI can provide accurate Land pricing based on location and demand. Mortgage and loan management tools will streamline collateral verification, and Geographic Information System (GIS) integration will enable precise land parcel mapping. Automated Land tax calculations through smart contracts can simplify taxation processes. Legal compliance and security are also critical. Smart contracts aligned with Land laws will secure and automate transactions, while dispute resolution mechanisms will ensure transparency. Proof-of-authenticity protocols will enable tamper-proof verification of documents and deeds. Enhanced privacy using technologies like zero-knowledge proofs (zk-SNARKs) and biometric authentication will safeguard user data and improve trust. To drive adoption, public awareness and education on blockchain's benefits in land registration are essential.

References

Land Registry System Using Blockchain Technology Authors: Sarfraz Nawaz Khan, Shayan Saleem, Muhammad Adeel, and Anam Saba.

Blockchain Technology in Land Registration: A Systematic Literature Review Authors: Reyan M. Zein and Hossana Twinomurinzi.

Blockchain-Based Land Registration Framework"
Authors: Multiple contributors from the International Journal of Creative Research Thoughts (IJCRT). Specific names not listed but available in the journal's records.

Blockchain-Based Land Management for Sustainable Development
Authors: Ivana Racetin, Jelena Kilić Pamuković, Mladen Zrinjski, Marina Peko
This paper discusses blockchain's application in land administration, cadastres, and management for sustainable development.

Land Registry System Using Blockchain Technology
Authors: G. T. R. Boaventura, A. Almeida, L. C. T. S. L. de Oliveira, et al.
This paper explores a blockchain-based system for land registry, focusing on reducing fraud and enhancing security.

A Secure Land Record Management System Using Blockchain Technology
Authors: Muhammad Usama, Wasim Ahmed, and Nasir Mehmood
This work presents a blockchain-based land record management system aimed at enhancing security and eliminating fraud.

Blockchain-Enabled Land Management Systems
Authors: Various scholars in land administration and technology fields, though specific author names are not provided in the snippet. This paper examines how blockchain addresses challenges in land management and registration.

Blockchain-Enabled Framework for Transparent Land Lease and Mortgage Management
Authors: Various authors in land administration and blockchain technology. This study proposes a transparent framework for land lease and mortgage management through blockchain.

These papers provide diverse perspectives on how blockchain can improve land management and governance.