جـــامعـــة الـشـــارقـــة
UNIVERSITY OF SHARJAH

Phase 2
Syntax Solutions

College of Computing and Informatics
Fall Semester 2024/2025

Software Engineering - Phase 1
Dr. Manar Abu Talib
Section : 12

Name's & ID's :

Amr Ahmed	U20103240
El Mahdi Darouich	U20105997
Hamza Luai Ahmed	U22105870
Ali Adil Mashkoor	U21102681
Omar Ahmad Ateya	U21106598
Mahaz Ishtiaq khan	U22200217
Karam Mubadda Saleh	U22201141

Table of content

1 Introduction:

Our mobile application is designed to transform property transactions by using blockchain technology to enhance transparency, security, and trust. It tackles common issues and skepticism in real estate by securely recording every transaction detail such as ownership records, legal approvals, and transaction histories on an immutable blockchain ledger.

The app integrates all key stakeholders buyers, sellers, landlords, government authorities, and legal entities into a unified platform. This integration allows users to access verified property information, automate processes through smart contracts, and track the entire transaction history. By using blockchain, the app helps prevent fraud and tampering, providing a reliable and efficient environment for property transactions.

In addition to its core functionalities, the app aims to simplify and expedite the real estate process by reducing paperwork and manual checks. By automating routine tasks and ensuring all data is easily accessible and verifiable, the app not only improves the efficiency of property transactions but also enhances user confidence. This streamlined approach helps to minimize delays and errors, making the overall experience more seamless and user-friendly.

Our aim is to make the process of buying and selling property straightforward and reliable, thereby addressing the common challenges faced in real estate transactions and fostering a more trustworthy market environment.

1.1 Goals & Objectives:

The primary goal of our application is to build trust in property transactions through enhanced transparency at every step. By providing verified property information—including ownership records, legal status, and government approvals—all stored securely on the blockchain, the app ensures that all data is accurate and reliable.

The app also aims to simplify the buying process using smart contracts that automate key tasks such as payment transfers, ownership changes, and legal clearances. This automation reduces the need for intermediaries, minimizes delays, and lowers the risk of errors.

Reducing fraud and scams is another key objective. The immutable nature of blockchain guarantees that once property records and transaction data are recorded, they cannot be altered or tampered with, enhancing security and protecting users from fraud.

Moreover, the app promotes transparency by allowing users to track the entire transaction history, ensuring that all steps of the process are verifiable. By integrating all relevant parties including landlords, government authorities, and legal entities into a single platform, the app aims to create a secure and efficient property transaction ecosystem.

In summary, our app is designed to make property transactions more transparent, secure, and efficient, addressing common challenges and improving the overall buying and selling experience.

1.2 Statement of Scope

The project's scope is to create a mobile app that improves property transactions using blockchain technology. The app will ensure transparency, security, and trust by recording and verifying property details, ownership, legal approvals, and transaction histories on a secure blockchain ledger.

Accessible on Android, the app will require users, buyers, sellers, landlords, government authorities, and legal entities, to create accounts for streamlined communication. A real-time database will store property and user data, updated with each transaction. Users must provide accurate property details, legal documentation, and contact information.

Our property transaction app is designed to tackle the common challenges in real estate dealings, such as lack of transparency and security. The major functionalities of the app include:

Essential Requirements:

Feature	Description
Contracts	Automates property transactions by handling payment transfers, legal approvals, and ownership transfers. • Auto-execution: Executes contract terms automatically when conditions are met. • Dispute Resolution: Provides mechanisms for resolving contract disp
Record History of Lands	Maintains an immutable, transparent record of land ownership and transaction history using blockchain. • Searchable Database: Allows users to search transaction history by property or owner. • Document Upload: Supports uploading and linking relevant documents to the transaction history. • Audit Trails: Provides detailed audit trails for all changes and updates.
Easier Way to Meet Government Regulations	Streamlines compliance by integrating government regulations into the transaction process.
Secure Transactions	Ensures secure payment processing and transaction management via blockchain's cryptographic protocols. • Encryption: Uses advanced encryption for transaction data. (hashing maybe?) • Two-Factor Authentication: Requires additional verification for transaction approval. • Fraud Detection: Implements mechanisms to detect and prevent fraudulent activities.
Tenant Screening	Verifies tenants' identity and financial background for landlords using blockchain-based data. (ID, Credit Score, Financial History)
Proof of Income for Buyers	Requires buyers to provide verifiable proof of income before proceeding with property transactions.
Investment Opportunities	Offers users the chance to invest in real estate through secure, blockchain-backed contracts.
Ownership Validation	Verifies property ownership through blockchain, ensuring clear and tamper-proof validation.
Loan Management	Provides tools to manage loans and financing within the app, integrated with blockchain for transparency. • Loan Application Integration: Create a user-friendly application form with document upload capabilities. • Loan Approval Workflow: Implement both automated and manual loan approval processes.
User-Friendly Property Verification (QR Codes)	Enables easy property verification through QR codes linked to blockchain data.

Desirable Requirements:

Feature	Description
Past User Feedback of Landlords	Allows buyers and tenants to view feedback and reviews of landlords from previous users.
Predefined Regulations	Provides users with predefined legal and financial regulations to simplify compliance in property deals.
Buying Shares	Enables users to invest in fractional property ownership by buying shares in real estate.
Data Export	Allows users to export property transaction data for external analysis or legal documentation.
Easier Negotiations	Facilitates communication between buyers and sellers with features that help streamline negotiations.
Web Page Access	Offers access to property transactions and information via a dedicated web interface.

Future Requirements:

Feature	Description
Credit Score and Financial History from Banks	Integrates with banks to pull users' financial history and credit scores for better transaction assessment.
Integration with Government Entities	Expands the app's functionality to further integrate government services for enhanced compliance checks.
Advanced AI-based Property Recommendations	Uses AI to recommend properties based on user preferences, financial status, and market trends.
Software Worldwide Deployment	Plans to expand the software's reach worldwide, ensuring it can operate across different legal frameworks.
iOS Version	Develops a dedicated iOS version of the app to cater to Apple device users.
Multilingual Assistance	Adds support for multiple languages to cater to a global user base.
Multi-currency Support	Enables transactions in various currencies, facilitating global real estate investments.
User-Friendly Enhancements	Focuses on improving the overall user interface and user experience to make the app more accessible.

1.3 Software Context:

The property transaction app operates within the real estate sector, aiming to enhance the process of buying and selling property through blockchain technology. It serves as a platform for all stakeholders involved in property transactions, including buyers, sellers, landlords, government authorities, and legal entities. The app provides a secure, transparent, and efficient environment by utilizing blockchain for immutable record-keeping, smart contracts for automating transactions, and a comprehensive system for tracking and verifying property information.

1.4 Major constraints:

The app faces several constraints, including the need for a robust blockchain infrastructure and seamless integration with existing systems. It must also ensure data privacy and comply with legal standards across different jurisdictions. Widespread user adoption is crucial for effectiveness, and the app must maintain technical reliability and scalability to handle a high volume of transactions smoothly.

- **Blockchain Integration:** Requires reliable infrastructure and integration with current systems.
- **User Adoption:** Needs acceptance from various stakeholders for optimal use.

2 Data Design

The property transaction application leverages blockchain technology to maintain transparency, security, and efficiency. Its data design underpins all core functionalities, ensuring that data is structured, accessible, and interoperable. The app's data design includes internal, global, and temporary data structures to facilitate seamless operations across components and ensure compliance with legal and user requirements.

2.1 Internal Software Data Structure

Internal data structures are designed to handle operations specific to individual components. These structures facilitate tasks like contract creation, ownership validation, and tenant screening. For example, the **contract data structure** captures details such as `contractID`, `landID`, parties involved, and contract terms, enabling the creation and management of property contracts. The **property ownership data structure** records ownership details and histories, supporting ownership verification and transfer processes. Similarly, the **tenant screening data structure** includes attributes like `tenantID`, financial status, and credit scores to validate prospective tenants. Additionally, the **transaction data structure** logs all transaction details to maintain audit trails and ensure transparency.

2.2 Global Data Structure:

Global data structures provide a shared foundation for the app's architecture, ensuring consistency and interoperability across components. The **blockchain ledger** serves as the backbone of the system, storing immutable records of contracts, ownership transfers, and transactions. The **user profiles database** maintains essential user details such as `userID`, roles, and contact information, enabling seamless authentication and communication.

The **property records structure** stores comprehensive property details, including legal status, market value, and linked documents, ensuring accurate and accessible information for transactions. The **notification queue** manages alerts and updates system-wide, ensuring users stay informed. Finally, the **audit trail structure** logs every action performed within the app, promoting accountability and enabling traceability.

This combination of internal and global data structures ensures the app operates efficiently, securely, and transparently, aligning with its goal of revolutionizing property transactions using blockchain technology.

2.3 Temporary Data Structures:

Minimal temporary data is used, mainly for input validation and real-time processing. Temporary data is committed or discarded post-operation.

2.4 Database Description

Outlined below are the tables, their corresponding attributes, and brief descriptions for the property transaction App.

Table Name : LandLord

Attribute	Data Type	Constraint Type	Optional
ID	int	Primary key	No
Name	string(50)		
Email	string(100)	Unique	
Phone	string(15)		
Address	string(150)		Yes
RegistrationDate	Date		
IsVerified	bool	Default False	No

The Landlord table stores the details of property owners registered on the platform. This table includes basic contact information, registration date, and a verification flag to indicate whether the landlord's identity has been confirmed

Table Name : Customer

Attribute	Data Type	Constraint Type		Optional
Username	string(50)	Unique		No
Password	string(100)			No
ID	int	Primary key		
Email	String(50)	Unique		
IsVerified	bool	Default Fale		Yes
WalletBalance	float(10,2)			

The Customer table holds the profile information of users who interact with the platform as buyers, renters, or general consumers.
This includes login credentials, contact information, registration data, and financial details such as wallet balance.
The IsVerified attribute helps track if the customer has completed identity verification.

Table Name : Acquire Property

Attribute	Data Type	Constraint Type		Optional
Username	string(50)	Unique		No
Password	string(100)			No
ID	int	Primary key		
Email	String(50)	Unique		
IsVerified	bool	Default Fale		Yes
WalletBalance	float(10,2)			

The Acquire Property table records the transaction details when a property is acquired. It includes key attributes like acquisition ID, customer ID, and property ID to track who is involved in the acquisition. Other attributes, such as ContractHash,
ensure the uniqueness and integrity of the contract. The BarkOTP is used for secure user authentication during the acquisition process. The table can reference other entities such as Customer, Landlord, and Property for complete relational mapping.

Table Name : Checkout

Attribute	Data Type	Constraint Type	Optional
Checkout_ID	int	Primary key	No
Customer_ID	int	Foreign Key	
Tax_Amount	double(15, 2)	Unique	
Payment_Method	string(30)		
TransactionID	string(50)	Unique	Yes
LandID	string(50)	Foreign Key	
Customer	Customer		Yes

The Checkout table contains records of transactions at the final stage of a purchase process. Key attributes include Checkout_ID and TransactionID to uniquely identify each transaction.
The table references Customer_ID and LandID as foreign keys to associate with respective customers and properties.
Tax_Amount, Payment_Method, Amount, and Currency fields detail the financial specifics of the transaction.
Status and Timestamp help in tracking the state and time of the transaction. The Customer field is an object that can hold further details about the customer if needed

Table Name : Payment Method (Sensitive Information)

Attribute	Data Type	Constraint Type	Optional
paymentMethodId	int	Primary key	No
cardHolderName	string(50)		
CardNumber	string(20)	Unique	
expirationDate	string(10)		
securityCode	string(5)	Unique	No
userId	int	Foreign Key	
paymentType	String(20)		No

The Payment Method table holds sensitive financial details associated with user payment methods.
The paymentMethodId serves as the primary key, while userId references the user who owns the payment method.
Attributes such as cardNumber and securityCode are stored in an encrypted format for security. The isPrimary field indicates if the method is the default for transactions, and paymentType specifies the type of payment (e.g., credit card, debit card).
The encryptData field can be used to store any additional encrypted data related to the payment method.

Table Name : Apply for Loan

Attribute	Data Type	Constraint Type	Optional
applicationID	int	Primary key	No
loanAmount	Double(15,2)		
status	string(20)		NO
monthlyRepaymnt	double(15, 2)		
creditedToWallet	bool		No
interestRate	double(5, 2)		
propertyID	int	Foreign Key	No

The Apply for Loan table records the loan applications made by tenants for specific properties.Each loan application is identified by applicationID as the primary key.
The tenantID and propertyID serve as foreign keys linking to tenants and properties respectively. The loanAmount represents the amount requested, while interestRate and loanTerm indicate the terms of the loan.
The monthlyRepayment field is calculated based on the loan details. The status attribute tracks the current state of the application (e.g., pending, approved). The creditedToWallet field indicates whether the loan amount has been transferred to the tenant's wallet.

3 Architectural and component-level design

Suitable architecture style: Layered architecture style

Reasoning:
The layered architecture is selected for its capacity to distinctly separates blockchain components from traditional ones, This modularity allows for independent development, testing, and updates, ensuring flexibility and adaptability, uphold robust security boundaries, and enable continuous improvement, keeping the system current with technological.

Our layered architecture will consist of four layers: the *presentation layer*, the *business logic layer*, the *blockchain integration layer*, and the *data persistence layer*. Each layer has distinct functions and responsibilities, ensuring a well-organized and efficient system.

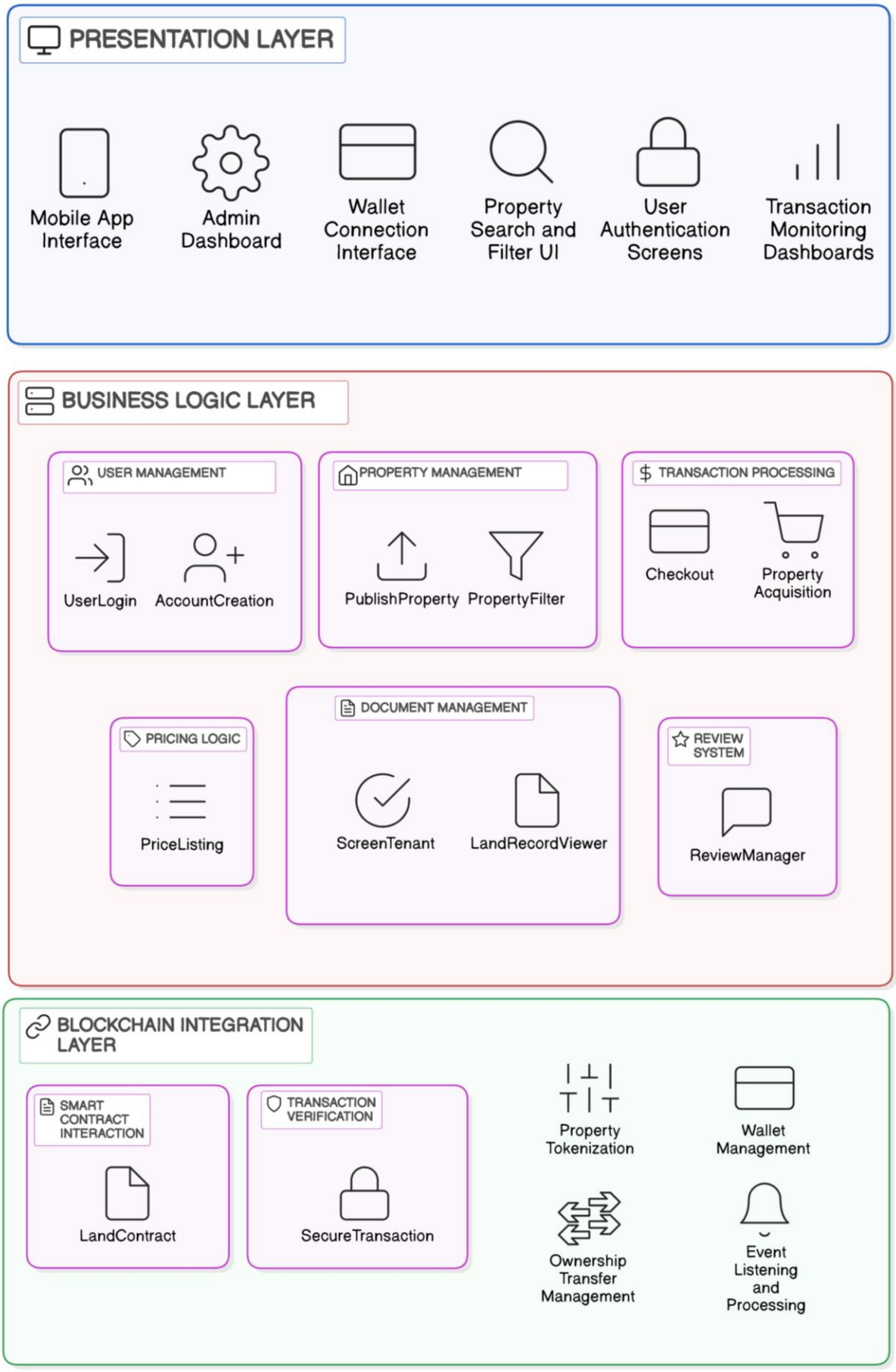
- **The Presentation Layer**: focuses on the user interface and experience, managing the interface, mobile applications, admin dashboards, wallet connections, property searches, authentication screens, and transaction monitoring. This layer ensures that users interact seamlessly with the system, providing a smooth and intuitive experience across various platforms and functionalities.

- **The Business Logic Layer:** is responsible for the core application logic, handling user operations, property listings, transactions, reviews, documents, pricing, and tenant screening. It achieves this through dedicated components such as UserLogin, PropertyFilter, and Checkout, ensuring that all essential functions are managed efficiently and effectively within the system.

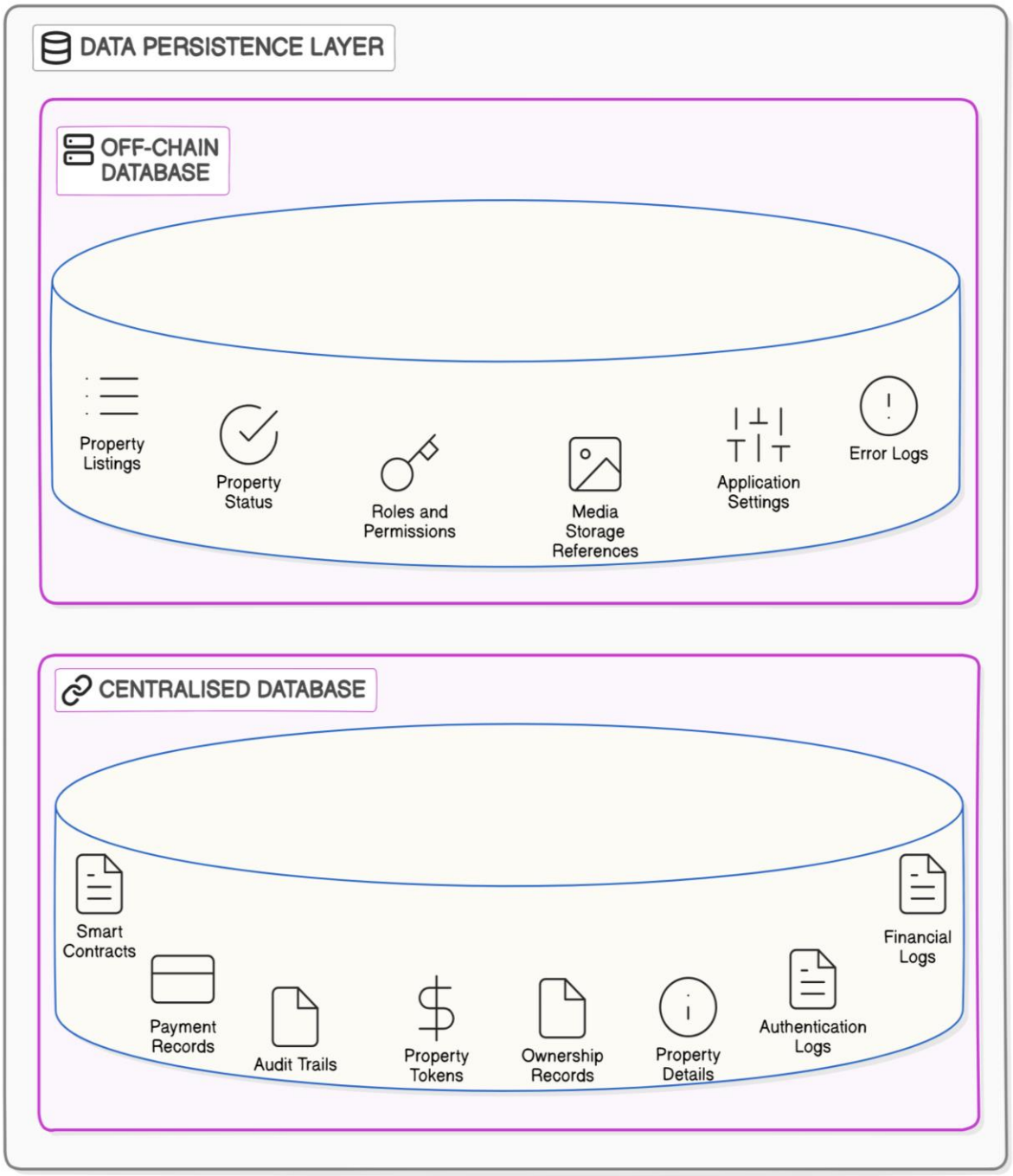
- **The Blockchain Integration Layer:** serves as a bridge between traditional and blockchain systems, managing smart contract interactions, ownership transfers, transaction verification, property tokenization, and wallet management. It utilizes specialized components like LandContract, OwnershipTransfer, and SecureTransaction to ensure seamless and secure integration of blockchain functionalities within the system.

- **The Data Persistence Layer:** consists of two parts: the Off-chain Database and Centralised Database. The Off-chain Database handles traditional data storage, including user profiles, roles, authentication, property listings, details, media, transaction payments, status, audits, and system configurations. The Centralised Database Blockchain Storage focuses on immutable transaction records, managing smart contracts, property tokens, ownership records, and transaction hashes, ensuring secure and permanent data storage.

Project Architecture



Project Architecture

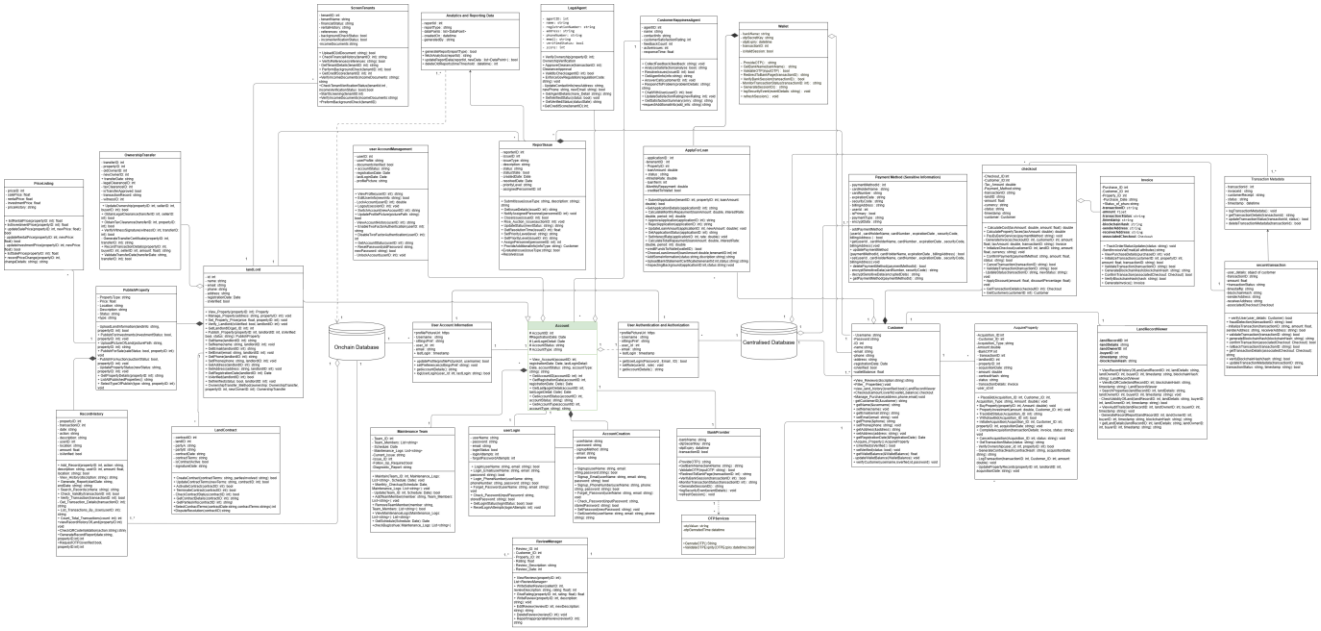


3.1 System Structure

A detailed description the system structure chosen for the application is presented.

3.1.1 Architecture diagram

A pictorial representation, using a UML detailed class diagram, of the architecture is presented.



3.2 Description for Component State Management

3.2.1 Processing narrative (PSPEC) for components State Management CustomerHappinessAgent and ReportIssue

This component uses classes like LandContract, LegalAgent, OwnershipTransfer, and ScreenTenants to handle specific responsibilities related to blockchain-enabled property transactions as creation of a land contract involves defining terms and validating parties before moving to verification or execution. Similarly, ownership transfer encompasses steps like obtaining legal and tax clearancesReportIssue, verifying ownership, and recording the transaction on the blockchain.The State Management component in the application supports the logical flow of various property transaction processes, ensuring smooth transitions between states.

3.2.2 Component State Management interface description.

The State Management interface is initiated when a user or system event requires a change in the operational flow, such as creating a new contract, verifying a property or finalizing ownership transfer.The interface begins by capturing the user input or event like selecting a property or entering contract terms. Based on the state and the associated event,the interface invokes methods like CreateContract() for contract initiation, VerifyOwnership() for validation, or UpdateOwnership() for ownership change.

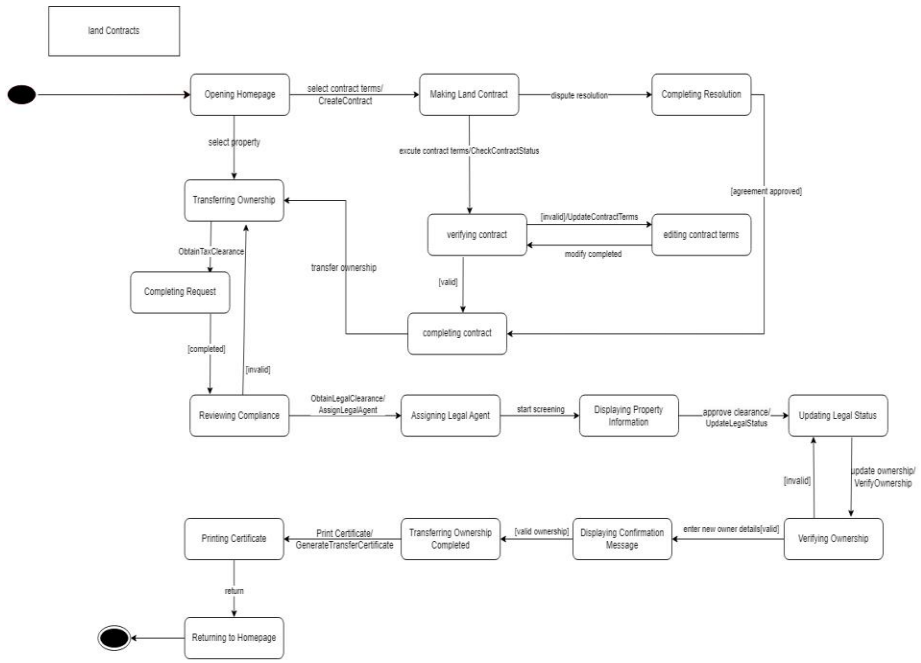
3.2.3 Components State Management processing detail

Start
User initiates a property transaction.
System identifies the type of transaction
activates the relevant process LandContract for creating contracts,OwnershipTransfer for ownership changes.

Process
For Creating Contract:
User provides contract details (parties involved, terms).
System verifies provided details.
Contract state transitions to "Active" upon successful verification.
For Ownership Transfer:
System requests legal and tax clearance
Ownership state transitions to "Pending Approval."
Once approvals are granted and ownership is verified, the transaction is recorded.

End
Process completes successfully when all validations pass, and actions (UpdateOwnership) are executed.
Errors or failed verifications lead to notifications and prompt corrective actions.

3.2.3.1 Design Class hierarchy for State Management



When a property transaction is initiated in the application, proper documentation and verification processes must be followed. The State Management component processes these requirements through multiple class interactions.LandContract: Handles all contract-related processes, including creation, updates, and validation, LegalAgent: Manages legal compliance by verifying ownership and approvals where OwnershipTransfer: Focuses on transferring property ownership with clearances and transaction recording and ScreenTenants: Supports tenant-related processes when applicable.

3.2.3.2 Restrictions/limitations for component State Management

Dependency on External Verifications:
The State Management component relies on external entities like legal agents for clearance, blockchain for record validation). Delays or failures in these entities could impact the process timeline.

Real-Time Blockchain Updates:
The immutable nature of blockchain ensures data security but may limit flexibility in correcting errors once a transaction is recorded.

3.2.3.3 Performance issues for component State Management

Not Available

3.2.3.4 Design constraints for component State Management

Compliance:
The design must ensure compliance with legal regulations and jurisdictional requirements for property transactions.

Blockchain latency:
The system's reliance on blockchain technology ensures transparency and immutability but adds a constraint for real-time updates due to inherent latency in block confirmations.

3.2.3.5 Processing detail for each operation of component State Management

The StateManager class coordinates operations across various classes the key processing details for its main operations are CreateContract(), VerifyOwnership(), UpdateOwnership(), ObtainClearance(), GenerateTransferCertificate().

3.2.3.5.1 Processing narrative (PSPEC) for each operation

CreateContract()
This method is invoked when a user initiates the creation of a land contract. It first validates the input terms and ensures the parties involved are registered in the system. If validation is successful, the contract is stored in the blockchain, and all parties are notified.

VerifyOwnership()
Invoked when the ownership of a property needs validation. The method interacts with the LegalAgent class to retrieve ownership details and cross-checks with blockchain records. If ownership is verified, the state transitions to "Ownership Verified."

UpdateOwnership()
This method processes ownership transfer requests. It ensures all clearances are obtained, verifies witness signatures if required, and updates the ownership details on the blockchain. Notifications are sent to the buyer and seller upon completion.

ObtainClearance()
Invoked during ownership transfer to request legal or tax clearance. The method communicates with the LegalAgent class, updating the system state based on clearance results. In case of rejection, the process retries or prompts the user for corrections.

GenerateTransferCertificate()
This method compiles transaction details to generate a transfer certificate. It ensures all necessary approvals are in place, securely records the certificate on the blockchain, and provides it to the relevant parties.

CreateContract()

Start
Validate contractTerms and partiesInvolved
If validation successful:
  Store contract in blockchain
  Notify all parties
Else:
  Prompt user to correct input
End

VerifyOwnership()

Start
Request ownership validation from LegalAgent
Cross-check with blockchain records
If ownership valid:
  Transition state to "Ownership Verified"
Else:
  Notify user of invalid ownership
End

UpdateOwnership()
Start Request legal and tax clearance Verify witness signatures (if required) Update ownership on blockchain Notify buyer and seller End
ObtainClearance()

Start
Request clearance from LegalAgent
If clearance granted:
  Update system state
Else:
  Retry or prompt user for additional data
End

GenerateTransferCertificate()
Start Compile transaction details Ensure all clearances and approvals are present Generate transfer certificate Record certificate on blockchain Provide certificate to relevant parties End

3.2 Description for Component Feedback and Issue Resolution System

3.2.1 Processing narrative (PSPEC) for components CustomerHappinessAgent and ReportIssue

The Feedback and Issue Resolution System comprises the classes **CustomerHappinessAgent** and **ReportIssue**. This component is responsible for managing user-submitted feedback and reported issues. CustomerHappinessAgent collects and processes feedback, evaluates satisfaction, and ensures user communication. ReportIssue identifies, prioritizes, and resolves issues flagged by users or system notifications. The primary role of this component is to maintain a high level of customer satisfaction and operational integrity by addressing user concerns, prioritizing maintenance, and ensuring efficient communication.

3.2.2 Component Feedback and Issue Resolution System interface description.

- The Feedback and Issue Resolution System interfaces include User Feedback Input for users to submit feedback, Issue Submission Interface for reporting issues, Notification Interface for updates on feedback or issues, Database Interface to store feedback and issue reports securely, and Maintenance Interface for automating maintenance tasks. User Feedback Input allows rating, comments, and suggestions submission. Issue Submission Interface includes fields for describing the problem, attaching files, and categorizing issues. Notification Interface sends updates via email, SMS, or in-app messages. Database Interface stores feedback and issue reports for analysis. Maintenance Interface triggers maintenance actions like repairs, updates, and system checks. These interfaces work together to facilitate feedback submission, issue reporting, and resolution in an organized and efficient manner.

3.2.3 Components Feedback and Issue Resolution System processing detail

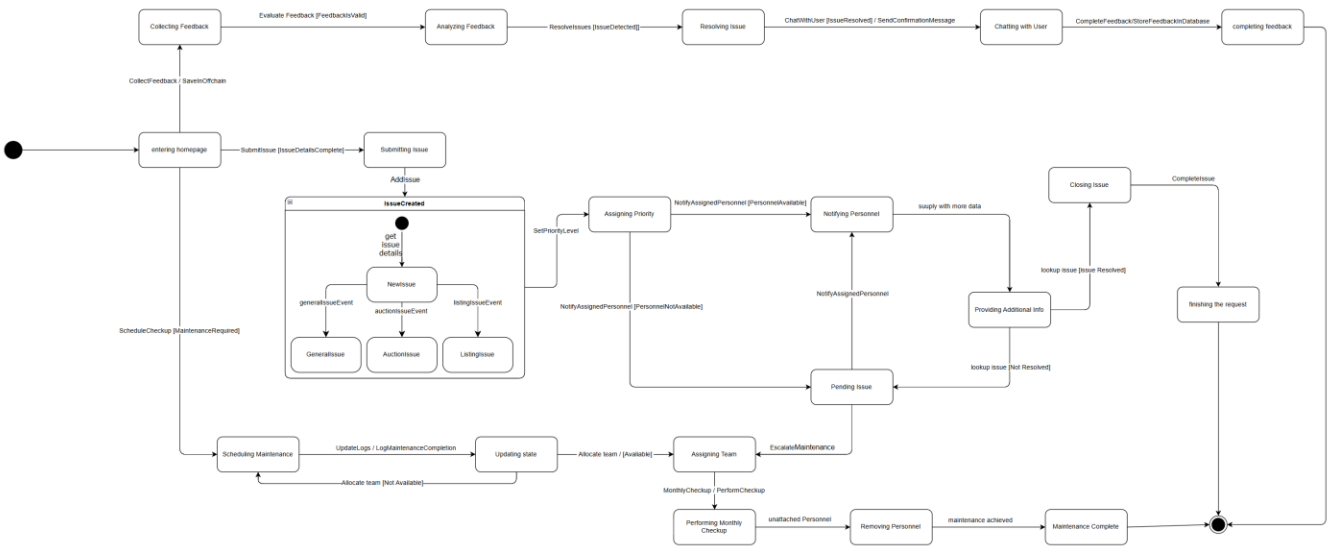
Start
User submits feedback or reports an issue.
System analyzes feedback and initiates issue resolution if needed.

Process
If feedback indicates an issue, the ReportIssue class categorizes it, assigns a priority, and triggers the resolution process.

CustomerHappinessAgent manages responses to user feedback, ensuring satisfaction. Notifications are sent to users upon resolution or request for additional information.

End
The feedback or issue is resolved, logged, and stored in the database.

3.2.3.1 Design Class hierarchy for Feedback and Issue Resolution System



When a user provides feedback or submits an issue, the CustomerHappinessAgent class is activated to gather and analyze the information. If the feedback indicates user dissatisfaction, the CustomerHappinessAgent assesses the feedback and, if necessary, transfers the issue to the ReportIssue class.

The ReportIssue class then categorizes the issue based on severity, assigns a priority level, and initiates the resolution process. Depending on the priority and type of issue, ReportIssue either handles the problem directly or escalates it to a specialized team. Each step in the resolution process is recorded, and upon completion, the system updates the issue status and sends a notification to the user.

Throughout this workflow, CustomerHappinessAgent maintains responsibility for ensuring user satisfaction by managing communication and providing status updates. Once the issue is resolved, the system reverts to a neutral state, ready to receive new feedback or issue reports. This loop allows the Feedback and Issue Resolution System to provide continuous, responsive support to users.

3.2.3.2 Restrictions/limitations for component Feedback and Issue Resolution System

Accuracy of User-Provided Feedback:
The system's effectiveness heavily relies on the quality of feedback provided by users. If the feedback is incomplete, vague, or unclear, the system may struggle to accurately assess and address the issues. This can lead to misclassification of issues, incorrect prioritization, and potentially ineffective resolutions.

Real-Time Responsiveness:
The system's ability to respond in real-time can be affected by several factors:
Network Speeds: Slow internet connections can delay the transmission of feedback and the system's responses, impacting the overall user experience.
System Load: During periods of high user traffic, the system may experience increased load, which can slow down processing times and response rates. This can be particularly challenging during peak times when many users are submitting feedback or issues simultaneously.

Scalability:
As the number of users and the volume of feedback grow, the system must scale accordingly. Without proper scalability measures, the system might become overwhelmed, leading to slower response times and potential failures in handling feedback efficiently.

Dependency on Automated Processes:
While automation can streamline the feedback and issue resolution process, it also introduces limitations. Automated systems may not always accurately interpret the nuances of user feedback, especially if the feedback includes complex or ambiguous language. This can result in misinterpretations and inadequate responses.

Security and Privacy Concerns:
Handling user feedback and issues involves processing potentially sensitive information. Ensuring the security and privacy of this data is crucial. Any breaches or lapses in data protection can undermine user trust and lead to significant repercussions.

Resource Allocation:
The system's performance is also dependent on the availability of resources, such as server capacity and technical support staff. Insufficient resources can hinder the system's ability to process feedback and resolve issues promptly.

3.2.3.3 Performance issues for component Feedback and Issue Resolution System

Concurrency: The system must efficiently manage multiple feedback and issue reports simultaneously, ensuring responsiveness during peak usage.

Notification Timeliness: Immediate notification upon issue identification is essential to prevent delays in resolution and enhance user satisfaction. Robust notification systems with redundancy can facilitate timely communication.

3.2.3.4 Design constraints for component Feedback and Issue Resolution System

Data Security: Protect user feedback data and sensitive issue information to follow "Personal Data Protection Law" in UAE .

Compliance: Ensure that all user feedback handling complies with data protection regulations.

3.2.3.5 Processing detail for each operation of component Feedback and Issue Resolution System

CustomerHappinessAgent operations:

- **CollectFeedback**: Collects feedback from users and stores it temporarily for analysis.
- **AnalyzeSatisfaction**: Analyzes feedback to assess user satisfaction and identifies areas for improvement.
- **NotifyUser**: Notifies the user regarding feedback status or resolution of issues.

ReportIssue operations:

- **SubmitIssue**: Submits and logs the issue in the system for resolution.
- **PrioritizeIssue**: Assigns a priority to the issue based on severity.
- **ResolveIssue**: Executes resolution steps or escalates for further action.
- **UpdateStatus**: Updates the issue status and sends notifications to the user.

3.2.3.5.1 Processing narrative (PSPEC) for each operation

CollectFeedback(): This method is designed to gather user feedback upon submission. It validates the feedback for completeness and clarity, ensuring it is ready for subsequent analysis.

AnalyzeSatisfaction(): This method assesses user satisfaction by analyzing the collected feedback based on predefined criteria. It identifies whether users are satisfied or dissatisfied, flagging any negative feedback for further action.

NotifyUser(): This method notifies users about the status of their feedback or issues. It provides updates including receipt acknowledgments, progress notifications, and resolution confirmations, fostering user engagement.

SubmitIssue(): When an issue arises, this method logs the relevant details into the system's database, capturing the issue description, user information, and supporting data for tracking.

PrioritizeIssue(): This method evaluates the severity and impact of identified issues to assign them appropriate priority levels, facilitating effective resource allocation and urgent response management.

ResolveIssue(): Depending on priority and nature, this method resolves issues either directly or through escalation to specialized teams, ensuring accurate diagnosis and solution implementation.

UpdateStatus(): This method updates the issue status in the system, detailing the current resolution stage and actions taken. Once resolved, it sends a final notification to the user, providing closure.

3.2.3.5.2 Algorithmic model (e.g., PDL) for each operation

CollectFeedback()
Start Capture user feedback input Store feedback temporarily for analysis End
AnalyzeSatisfaction()

Start
IF feedback indicates dissatisfaction THEN
  Flag feedback for issue analysis
ELSE
  Mark feedback as positive
ENDIF
End

NotifyUser()
Start Retrieve feedback or issue status Send notification to user End
SubmitIssue()
Start Capture issue details from user input Log issue in database End
UpdateStatus()
Start Log resolution steps Update issue status in database Notify user of resolution status End

3.2.3.5.2 Algorithmic model (e.g., PDL) for each operation

PrioritizeIssue()
Start Assess issue severity Assign priority level End
ResolveIssue()

Start
IF issue resolution is possible THEN
  Apply resolution steps
ELSE
  Escalate issue
ENDIF
End

3.2 Description for Component View Land Record

3.2.1 Processing narrative (PSPEC) for components View Land Record

The View Land Record component involves the **Customer** and **LandRecordViewer** classes. This component provides secure access to land records and includes authentication, record validation, and property display functionalities.

The **Customer** class manages user verification with attributes like **username**, **password**, **ID**, and **isVerified** status, which controls access. It includes the view_**land_history()** and **verifyCustomer()** methods to validate the user's identity. The **LandRecordViewer** class manages the display and verification of land records, with attributes such as **landRecordID**, **landDetails**, **landOwnerID**, **buyerID**, and blockchainHash for securely tracking property information.

Key responsibilities include verifying customer credentials, checking record validity, and securely displaying or restricting access to land records based on the verification results. This component ensures that only verified customers can view land records, leveraging blockchain hashing for data integrity.

3.2.2 Component View Land Record interface description.

The **View Land Record** interface is triggered when a customer initiates a land record view. The process is as follows:

- **User Verification**: The Customer interface checks isVerified and calls verifyCustomer() with credentials (username, password, ID). If verification fails, an OTP prompt initiates as a secondary measure.
- **Accessing Land Records**: Upon successful verification, the view_land_history() method calls LandRecordViewer.ViewRecordHistoryOfLand() to retrieve and display land records, verifying the record against the blockchain hash.
- **QR Code and Validity Checks**: For additional security, LandRecordViewer.ViewByQRCode() verifies land details using a QR code scan. LandRecordViewer.CheckValidityOfLand() confirms the record's authenticity before proceeding.
- **Displaying and Generating Record Reports**: Upon successful checks, LandRecordViewer.GenerateRecordReport() produces a detailed record report. If the record is invalid, the interface displays an error message, prompting the user to retry or return to previous options.

3.2.3 Components View Land Record processing detail

Start

Verification and Validation Check
Once the customer's verification is confirmed (Customer.isVerified = true), they can proceed to the Authentication stage.

If the system is not verified, it initiates the OTP verification process to grant access.

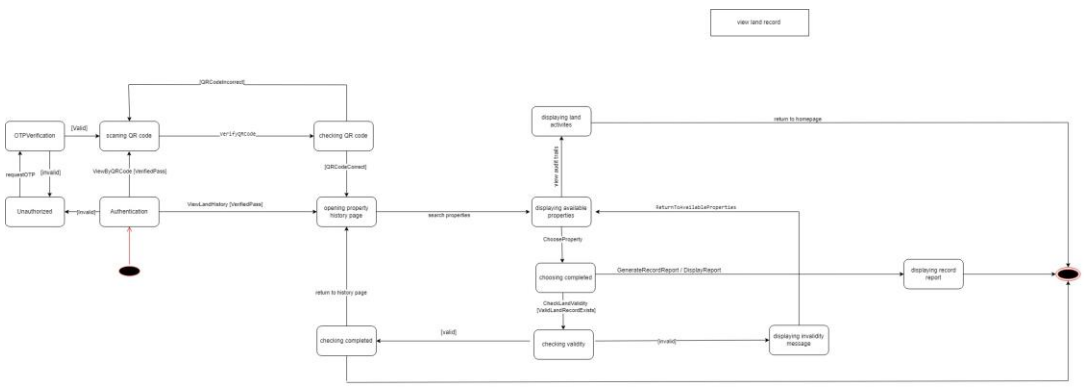
Verification through OTP and QR Code Scanning
After OTP verification, the customer uses LandRecordViewer.ViewByQRCode() to scan a QR code and locate the desired land record.

Check Record Validity
LandRecordViewer.CheckValidityOfLand() verifies the record's validity. If successful, the system displays the record; otherwise, it returns an invalidity message.

Showing Record and Creating Report
Upon confirming the record's validity, the system uses LandRecordViewer.GenerateRecordReport() to create a comprehensive report for the user.

END

3.2.3.1 Design Class hierarchy for View Land Record



The View Land Record component involves verifying a customer's access rights using verifyCustomer() based on their isVerified status. Upon successful verification, the Customer class calls view_land_history() to display the land record. LandRecordViewer then manages access, including methods like ViewByQRCode() for QR code retrieval, CheckValidityOfLand() for authenticity verification, and GenerateRecordReport() for comprehensive reporting. This hierarchy ensures secure access, authentic records, and report generation for authorized customers.

3.2.3.5.1 Processing narrative (PSPEC) for each operation (continue)

view_land_history()
Retrieves the history of land records associated with a customer. First, it validates the customer's access by checking the isVerified attribute. Then, it retrieves the land record history for the specified landRecordID and displays it securely to the customer.

isVerified()
Checks if the customer is verified. This function ensures that only authorized and verified customers can access sensitive land records and operations. It validates the customer's credentials and status before granting access.

verifyCustomer()
Verifies the customer's credentials and status. This function ensures that the customer is authorized to perform specific operations related to land records. It checks the customer's verification status and grants or restricts access accordingly.

3.2.3.5.2 Algorithmic model (e.g., PDL) for each operation

ViewRecordHistoryOfLand()

```
Start

IF isVerified(customerID) THEN
   Retrieve land record history for landRecordID
   Display land record history securely to the customer
ELSE
   Display error message indicating access is denied
ENDIF

End
```

ViewByQRCode()

```
Start

Retrieve landRecordID from QRCode
Verify land record details against blockchain data

IF isVerified(customerID) AND QR code matches THEN
   Retrieve and display associated land details
ELSE
   Display error message indicating access is denied
ENDIF

End
```

3.2.3.5.2 Algorithmic model (e.g., PDL) for each operation

CheckValidityOfLand()

```
Start

IF verifyCustomer(customerID) THEN
   Retrieve stored blockchain data for landRecordID
   Compare landRecordID, landOwnerID, and blockchainHash with stored
data

   IF details match THEN
      Flag record as valid
      Return true
   ELSE
      Return false
      Display error message indicating access is restricted
   ENDIF
ELSE
   Display error message indicating access is denied
ENDIF

End
```

GenerateRecordReport()

```
Start

IF isVerified(customerID) THEN
   Collect details for landRecordID (owner's information, transaction
history, blockchain hash)
   Format details into a structured report
   Make report available for viewing or download
ELSE
   Display error message indicating access is denied
ENDIF

End
```

verifyCustomer()

```
Start

Retrieve stored credentials for customerID from database
Compare provided credentials with stored credentials

IF credentials match THEN
   Update customer's status to verified in database
   Return true
ELSE
   Return false
ENDIF

End
```

isVerified()

```
Start

Retrieve customer's verification status from database

IF customer's status is verified THEN
   Return true
ELSE
   Return false
ENDIF

End
```

view_land_history()

```
Start

IF isVerified(customerID) THEN
   Retrieve land record history for landRecordID
   Display land record history securely to the customer
ELSE
   Display error message indicating access is denied
ENDIF

End
```

3.2 Description for Component: Acquiring Property System

3.2.1 Processing Narrative (PSPEC) for Components: PropertyAcquisition and Checkout

The Property and Checkout System is designed to handle the end-to-end process of property transactions and payment processing, incorporating user interactions, payment handling, tax calculations, and order management. The key classes involved are PropertyAcquisition and Checkout, which focus on managing property listings and handling secure payment transactions efficiently.

PropertyAcquisition:

- Manages property listing validation and verification
- Handles property details input and storage
- Processes investment calculations and returns
- Updates property records and ownership status
- Manages property checking and validation processes
- Generates investment reports and analysis Checkout:
- Facilitates the complete payment processing workflow
- Manages payment method validation and verification
- Handles tax calculations and applications
- Processes order status updates and confirmations
- Generates invoices and payment receipts
- Controls payment session management and security
- Manages transaction rollbacks and error handling
- Coordinates order completion and fulfillment

This component aims to ensure secure and efficient property transactions through robust payment processing, accurate tax calculations, and reliable order management. The system prioritizes data consistency, transaction security, and proper state management throughout the checkout process while maintaining clear communication with users through notifications and status updates.

3.2.2 Component Acquiring Property System Interface Description

The Acquiring Property System incorporates several interfaces to facilitate the property purchase and bidding processes:

Property Request Interface:

-Enables users to initiate property acquisition requests.

-Users can choose to either buy properties outright or place bids.

Ownership Verification Interface:

-Verifies property ownership and legal status before processing acquisitions.

-Ensures compliance with property regulations and secure transfer of ownership.

Bidding Management Interface:

-Allows users to place bids, adjust existing bids, and track current bidding status.

-Automates notifications for higher bids, bid acceptance, or rejection.

Payment Processing Interface:

-Supports payment via multiple methods (bank services, escrow accounts, etc.).

-Ensures secure transaction validation and payment confirmation.

Notification and Update Interface:

-Provides real-time updates on bid statuses, property acquisition, and payment confirmation.

-Notifies users via email or in-app messages about the progress of their property acquisition journey.

3.2.3 Acquiring Property System Processing Detail (Aligned with State Chart)

Processing Flow: Start:

-User Actions: The user starts by either checking the property request or proceeding directly to the purchase or bid interface.

Process:

-Ownership Verification:

- The system verifies ownership approval for the requested property.

-Bidding:

- If a user chooses to bid, they can place or adjust their bids.
- The system tracks bids, comparing new entries with existing bidding amounts.
- Higher bids are escalated, and notifications are sent to outbid users.
- Lower bids result in withdrawal or rejection notifications.

-Property Purchase:

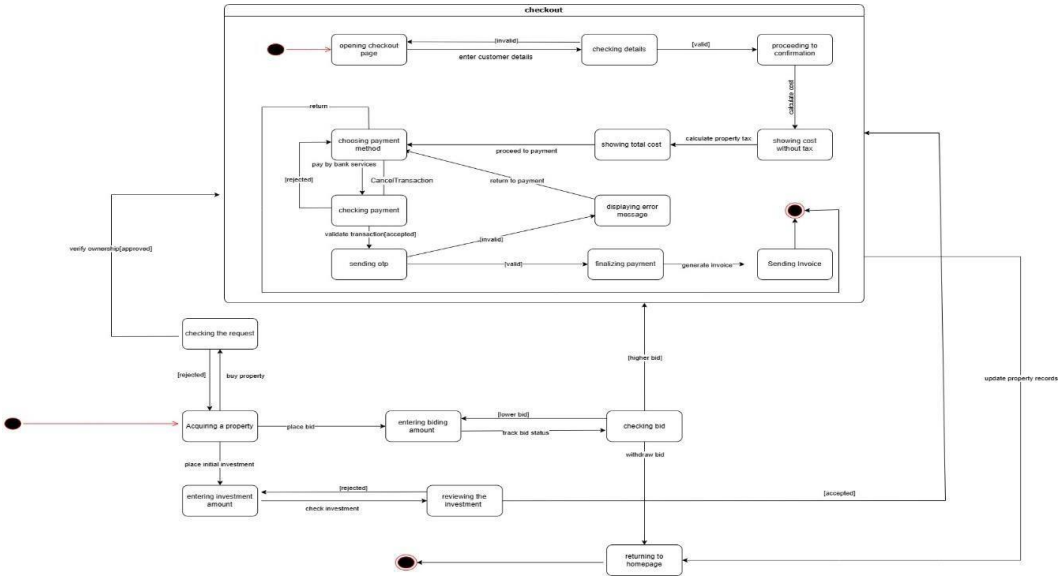
- If the property is acquired without a bid, the system processes payment.
- Valid transactions trigger property record updates.

End:

-Once a property is acquired or bid outcomes are finalized:

- The user is notified of successful acquisition.
- The system logs the transaction, updates the property database, and returns users to the homepage.

3.2.3.1 Design Class hierarchy for property acquiring and checkout



The statechart illustrates the Checkout and Property Acquisition System, which allows users to either purchase properties directly or participate in a bidding process. It begins with the checkout process, where users enter their details, confirm information, calculate costs (including property tax), and proceed to payment. Users select a payment method, with options to validate the transaction via OTP, finalize the payment, and receive an invoice. If a user opts to acquire property through bidding, they place a bid, which the system tracks and compares against other bids. The system enables users to monitor bid status, place higher bids if necessary, or withdraw. Upon successful purchase or winning a bid, property records are updated to reflect new ownership. Throughout, users receive notifications, and error handling ensures a smooth transaction experience.

3.2.3.2 Restrictions/limitations for component property acquiring and checkout

Accuracy of User Flow

The system's effectiveness heavily relies on the quality of user interactions during checkout. If the user input is incomplete, invalid, or unclear during any stage, the system may struggle to accurately process the order and payment. This can lead to abandoned carts, failed payments, and incomplete transactions, ultimately affecting business revenue.

Real-Time Responsiveness

Network Speeds: Slow internet connections may cause delays in processing payments, checking details, and loading checkout pages, impacting the overall user experience.

System Load: During peak periods, such as property bid times or sales events, the system may experience heavy load, potentially slowing down the response time for actions like payment processing or bid status updates.

Scalability

As the number of users and transactions increase, the system must scale to handle greater loads. Without proper scalability measures, checkout processing times could lengthen, and system overload could impact critical activities like verifying ownership or updating property records.

3.2.3.3 Performance Issues for Component: Checkout and Property Acquisition System

Concurrency Management

Efficient management of multiple, simultaneous activities—like handling numerous bids and checkout processes—is crucial to ensure the system remains responsive, especially during high-demand periods.

Notification Timeliness

Timely notifications on bid status or payment confirmation are essential to keep users informed and engaged, helping to minimize any uncertainties or delays in the purchase or bidding process.

3.2.3.4 Design Constraints for Component: Checkout and Property Acquisition System Data

Security

The system must ensure secure handling of sensitive data, including payment information and personal details, in compliance with the UAE's Personal Data Protection Law.

Compliance

All transactions, payment processes, and property ownership verifications must follow applicable regulations to ensure user data protection and secure transaction handling.

3.2.3.5 Processing Detail for Each Operation of Component: Checkout and Property Acquisition System

Checkout Operations

OpeningCheckoutPage: Opens the checkout page, allowing the user to start the transaction process.

EnterCustomerDetails: Captures customer information required for checkout, such as contact and payment details.

CheckingDetails: Verifies the accuracy and completeness of entered customer information.

ProceedToConfirmation: Directs the user to confirm their details before moving to the payment phase.

CalculatePropertyTax: Calculates applicable property tax based on transaction details.

ShowCostWithoutTax: Displays the total cost of the property without the tax included.

ProceedToPayment: Initiates the payment process once the total cost is confirmed.

CheckingPaymentMethod: Allows the user to select their preferred payment

method. PayByBankServices: Processes payment through the selected bank services.

CancelTransaction: Provides an option to cancel the payment process if required.

CheckingPayment: Verifies if the payment was successful and completed.

ValidateTransactionReceipt: Validates the payment receipt for authenticity and completeness.

SendingOtp: Sends an OTP (One-Time Password) for verification of the transaction.

FinalizingPayment: Finalizes the payment and prepares for invoice generation.

GenerateInvoice: Generates an invoice for the transaction.

SendingInvoice: Sends the invoice to the user for record-keeping.

Property Acquisition Operations

VerifyOwnershipApproved: Verifies that property ownership approval is granted.

CheckingTheRequest: Reviews and validates the user's request to buy or bid on a

property. BuyProperty: Completes the property acquisition process for the user.

AcquiringAProperty: Assigns ownership of the property to the user after successful payment.

PlaceBid: Allows the user to place a bid on a property. ExistingBiddingAmount: Compares

the user's bid with the current highest bid. TrackBidStatus: Continuously tracks the status of

the placed bid.

CheckBidStatus: Checks if the user's bid is the highest or if a higher bid has been placed.

WithdrawBid: Allows the user to withdraw their bid if they choose to do so.

ReturningToHomepage: Redirects the user to the homepage upon completion of transactions.

UpdatePropertyRecords: Updates records to reflect new ownership details after successful purchase.

3.2.3.5.1 Processing Narrative (PSPEC) for Each Operation

OpeningCheckoutPage(): This method opens the checkout page, initializing the transaction process. It sets up the necessary fields and options for the user to input details.

EnterCustomerDetails(): This method captures the user’s details required for the transaction, ensuring essential information is collected for verification and payment.

CheckingDetails(): This method verifies the entered customer details for completeness and accuracy, preventing errors during checkout.

ProceedToConfirmation(): This method moves the user to the confirmation page to review their details before proceeding to payment, allowing them to make corrections if necessary.

CalculatePropertyTax(): This method calculates the property tax based on the transaction details and prepares it to be added to the total cost.

ShowCostWithoutTax(): This method displays the property’s cost before including tax, providing transparency on the amount being charged.

ProceedToPayment(): This method initiates the payment process, prompting the user to confirm and make the payment.

CheckingPaymentMethod(): This method allows the user to select a payment method and ensures the selected method is valid for the transaction.

PayByBankServices(): This method processes the payment through bank services, ensuring secure transfer of funds.

CancelTransaction(): This method gives the user the option to cancel the transaction at any point before payment is confirmed.

CheckingPayment(): This method checks if the payment was successfully processed, allowing the transaction to proceed only if payment is confirmed.

ValidateTransactionReceipt(): This method validates the authenticity of the payment receipt, ensuring the payment details are legitimate.

SendingOtp(): This method sends an OTP (One-Time Password) to the user for additional transaction security.

FinalizingPayment(): This method completes the payment process, ensuring all funds are transferred, and transaction records are updated.

GenerateInvoice(): This method generates an invoice after successful payment, capturing all transaction details for the user's records.

SendingInvoice(): This method sends the generated invoice to the user via their preferred contact method.

VerifyOwnershipApproved(): This method verifies that the user has received ownership approval for the property, allowing them to proceed with the transaction.

CheckingTheRequest(): This method reviews the user's request to buy or bid on a property, ensuring all requirements are met.

BuyProperty(): This method finalizes the purchase process, transferring ownership of the property to the user.

AcquiringAProperty(): This method officially assigns the property to the user following a successful purchase.

PlaceBid(): This method allows the user to place a bid, capturing the amount and updating bid records.

ExistingBiddingAmount(): This method compares the user's bid with the current highest bid, ensuring the new bid is higher if required.

TrackBidStatus(): This method tracks the status of the user's bid, continuously checking if it remains the highest.

CheckBidStatus(): This method checks if there is a higher bid on the property, updating the user on their bid status.

WithdrawBid(): This method allows the user to withdraw their bid if they no longer wish to compete for the property.

ReturningToHomepage(): This method redirects the user back to the homepage after completing transactions or actions in the checkout and property acquisition system.

UpdatePropertyRecords(): This method updates the property records to reflect new ownership or changes after a successful transaction.

3.2.3.5.2 Algorithmic model (e.g., PDL) for each operation

OpeningCheckout()
Start Capture user checkout input Store checkout temporarily for processing End
CheckingOrder()
Start IF order details are valid THEN Proceed to payment processing ELSE Return to checkout page ENDIF End
ProcessingPayment()

Start

Capture payment details from user input Validate payment information

Process payment through gateway IF payment successful THEN

Generate invoice ELSE

Show error message ENDIF

End

CalculatingTotal()

Start

Retrieve order items Calculate subtotal

Apply taxes and shipping Calculate final amount End

CheckingTax()

Start

Retrieve billing location

Determine applicable tax rates Calculate tax amount

End

SubmittingOrder()

SubmittingOrder()
Start Verify final order details Update inventory Create order record Generate order confirmation End
SendingInvoice()
Start Compile order details Generate invoice document Send to customer email End
AcceptingProperty()
Start Capture property details Validate listing information Store in database End
CheckingInvestment()
Start Review investment criteria Calculate potential returns Generate investment report End

3.2.1 Processing Narrative (PSPEC) for Components: Customer System

The Customer System is designed to handle the end-to-end process of customer interactions, property reviews, and purchase management, incorporating user authentication, property exploration, and review management. The key components involved are CustomerAuthentication and ReviewManagement, which focus on managing user sessions and handling property review workflows efficiently.

CustomerAuthentication:

-Manages user login and session verification

-Handles account detail validation

-Controls user access and permissions

-Updates customer profile information

-Maintains wallet balance and transactions

-Processes contact information updates

ReviewManagement:

-Facilitates complete review lifecycle

-Manages review editing and updates

-Handles review status tracking

-Processes review moderation and approvals

-Generates review ratings and calculations

-Coordinates review reporting and resolution

-Controls review deletion and archiving

-Manages review display and visibility

3.2.2 Customer System Interface Description

The Customer System incorporates several interfaces to facilitate user interaction and review management.

Login Interface:

- Enables users to authenticate and access the system

- Validates user credentials and maintains session security

Property Review Interface:

- Allows users to view and submit property reviews
- Manages the editing and updating of existing reviews
- Facilitates review moderation and status updates

Account Management Interface:

- Supports profile and contact information updates
- Manages wallet balance and transaction history
- Handles customer preferences and settings

Property Exploration Interface:

- Enables browsing and searching available properties
- Provides detailed property information and history
- Supports property comparison and selection

Purchase Processing Interface:

- Manages property purchase transactions
- Handles wallet balance verification and updates
- Processes transaction confirmations and records

3.2.3 Customer System Processing Detail (Aligned with State Chart) Processing Flow:

3.2.3 Customer System Processing Detail (Aligned with State Chart)

Processing Flow:

Start:

- User Actions: The user begins by logging in to the system through authentication

Process:

- Authentication:
  - System verifies user credentials and account details
  - Establishes user session and access permissions

- Property Exploration:
  - Users can browse available properties
  - System displays property details and review history
  - Enables property search and filtering

- Review Management:
  - Users can create, edit, and delete reviews
  - System moderates and processes review submissions
  - Handles review reporting and resolution
  - Manages review ratings and calculations

- Purchase Processing:
  - Verifies wallet balance and purchase eligibility
  - Updates contact information as needed
  - Processes property acquisition transactions

Start:

- User Actions: The user begins by logging in to the system through authentication

Process:

- Authentication:
  - System verifies user credentials and account details
  - Establishes user session and access permissions

- Property Exploration:
  - Users can browse available properties
  - System displays property details and review history
  - Enables property search and filtering

- Review Management:
  - Users can create, edit, and delete reviews
  - System moderates and processes review submissions
  - Handles review reporting and resolution
  - Manages review ratings and calculations
- Purchase Processing:
  - Verifies wallet balance and purchase eligibility
  - Updates contact information as needed
  - Processes property acquisition transactions

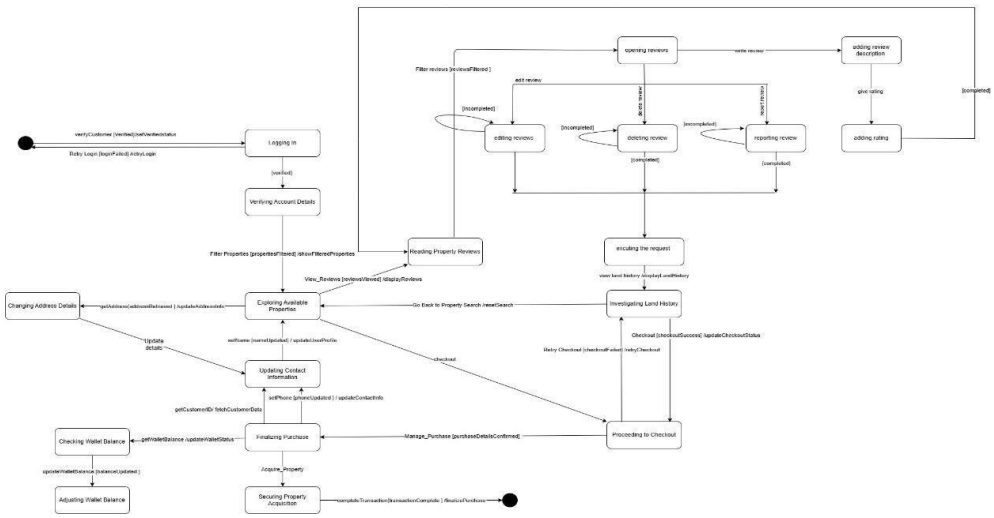
End:

- Once a purchase is completed or review process is finalized:
  - The system updates transaction records
  - Confirms purchase completion or review submission

- Returns users to appropriate interface based on action

- Updates wallet balance and property status accordingly

- **3.2.3.1 Design Class hierarchy for customer**



The statechart depicts a user journey through an application for managing property-related activities, starting with user login and account verification. Once logged in, the user can explore available properties, update their contact or address information, and investigate land history for selected properties. The chart also includes processes for managing reviews, allowing users to open, write, edit, delete, or report property reviews, as well as add ratings. After finalizing a property, users proceed to checkout and manage wallet balances, ensuring secure property acquisition. The chart concludes with completing the transaction, offering a comprehensive overview of system states and transitions.

3.2.3.2 Restrictions/limitations for component acquiring property and checkout System

Authentication and Access Limitations

The system's authentication process creates potential bottlenecks during peak usage periods. Users must complete the entire verification sequence before accessing features, which can impact experience during system slowdowns. Strict session timeouts and security lockouts after failed attempts may disrupt user activities, especially during complex property transactions or reviews.

Property Review System Constraints

The review workflow enforces strict sequential progression from editing to deleting to reporting reviews, preventing parallel processing. This linear approach can create processing queues during high-volume periods. The requirement for complete review cycle completion before moving to new reviews can create bottlenecks, especially with multiple properties or reviewers.

Transaction Processing Limitations

The checkout process requires rigid, linear progression through each step, from wallet verification to purchase confirmation. The system cannot process multiple transactions simultaneously, and its dependency on real-time wallet balance verification can cause delays. Once started, transaction details cannot be modified without restarting the entire process.

Data Management and Synchronization Restrictions

The system must carefully control property information updates to prevent conflicts during simultaneous access. Real-time synchronization between listings and availability can be delayed due to processing constraints. During high-volume periods, maintaining data integrity becomes challenging, especially with multiple property updates occurring simultaneously.

Technical Infrastructure Constraints

Network dependency creates vulnerabilities during connectivity issues, potentially disrupting critical transactions. The system's concurrent user capacity is limited by server resources, leading to degraded performance during peak usage. Resource allocation must be carefully managed to prevent bottlenecks during complex operations like property searches
or reviews.

User Experience Limitations

Users must follow predefined paths when exploring properties or completing transactions, limiting flexibility. Submitted information or reviews cannot be easily modified without formal processes. The linear progression through various stages doesn't accommodate simultaneous work on multiple properties, potentially reducing efficiency in managing multiple listings.

3.2.3.3 Performance issues for component acquiring property and checkout System

State Transition Responsiveness The system's ability to smoothly transition between states (from property exploration to checkout, or through the review cycle) becomes critical during high traffic. Each state transition shown in the statechart must be processed efficiently to maintain system responsiveness. Delays in state transitions, particularly during crucial processes like wallet verification or property acquisition, can significantly impact user experience.

Notification Synchronization Real-time notification delivery across various system states presents performance challenges. The system must efficiently notify users about review updates, transaction status changes, and property availability while maintaining state consistency. The interconnected nature of the statechart components requires careful management of notification timing and delivery to prevent information delays or conflicts.

3.2.3.4Design constraints for component acquiring property and checkout System

Data Security Requirements Following UAE's Personal Data Protection Law, the system must implement robust security measures:

- User authentication and authorization must be strictly enforced at each state transition
- Property transaction data and user financial information must be encrypted
- Personal information in property reviews and user profiles must be protected
- Access logs and audit trails must be maintained for all sensitive operations
- Data retention and deletion policies must align with UAE regulations

Compliance Architecture The system's design must incorporate specific compliance measures:

- User consent management for data collection and processing
- Clear data privacy notices at key interaction points (login, review submission, checkout)
- Secure data storage and transmission protocols
- Mechanisms for users to access and modify their personal information
- Audit capabilities for compliance verification
- Regular security assessment and update procedures

Regulatory Documentation The system must maintain comprehensive documentation:

- Data flow diagrams showing personal information handling

- Security protocols for each state transition
- User consent records and privacy preference management
- Incident response procedures
- Regular compliance audit reports
- Data protection impact assessments

5. Processing detail for each operation of component acquiring property and checkout System

Login and Authentication Operations: verifyCustomer Operation

- Validates user credentials against stored data
- Checks account status and permissions
- Creates authenticated session
- Logs login attempt and timestamp
- Redirects to property exploration on success
- Handles failed login attempts with appropriate error messages Property

Exploration Operations:

exploreAvailableProperties Operation

- Retrieves current property listings
- Filters based on user preferences
- Updates property status in real-time
- Tracks user viewing history
- Manages property search results
- Integrates with review system for property ratings
- Handles property detail requests Review Management Operations:

editingReview Operation

- Validates review content
- Processes review submission
- Updates review database
- Checks for review guidelines compliance
- Manages review attachments
- Triggers notification system
- Integrates with property rating system

deletingReview Operation

- Verifies deletion authorization
- Processes review removal
- Updates property rating calculations
- Maintains review history log
- Sends deletion confirmations
- Updates review counts
- Manages associated review data

reportingReview Operation

- Processes review reports
- Categorizes reported issues
- Assigns review investigation priority
- Updates review status
- Notifies relevant parties
- Tracks report resolution
- Maintains reporting history

Transaction Processing Operations:

checkingWalletBalance Operation

- Verifies available funds
- Updates wallet balance

- Validates transaction limits
- Checks for pending transactions
- Processes balance inquiries
- Maintains transaction logs
- Ensures data consistency

proceedingToCheckout Operation

- Initializes checkout process
- Validates property availability
- Reserves property temporarily
- Processes payment details
- Updates transaction status
- Generates checkout records
- Manages transaction workflow

Property Acquisition Operations:

acquireProperty Operation

- Finalizes property purchase
- Updates property status
- Generates ownership documents
- Records transaction completion
- Updates system inventory
- Triggers notification system
- Maintains acquisition logs

Contact Management Operations:

updatingContactInformation Operation

- Validates contact details
- Updates user profile
- Ensures data format compliance
- Maintains contact history

- Processes change notifications
- Verifies contact authenticity
- Updates associated records

3.2.3.5.2 Algorithmic model (e.g., PDL) for each operation

Login Operation
Start Verify customer credentials Create session if valid Redirect to property exploration End
ReviewManagement Operation
Start IF action is edit THEN Process review edits ELSE IF action is delete THEN Remove review

ELSE IF action is report THEN Submit review report

ENDIF

Update review status End

ExploreProperties Operation

Start

Fetch available properties Apply search filters Display property listings

End

CheckoutProcess Operation

Start

Verify wallet balance Check property availability Process transaction Update

property status

End

UpdateContactInfo Operation

Start

Validate new contact details Update user profile

Save changes End

ProcessPurchase Operation()
Start Lock property for transaction Complete payment process Transfer property ownership Generate documentation End

3.2.1 Processing Narrative (PSPEC) for Components: Property Publishing System

The Property Publishing System is designed to handle the end-to-end process of property listing and management, incorporating user authentication, property submission, and publication workflow. The key components involved are UserAuthentication and PropertyManagement, which focus on managing user sessions and handling property listing workflows efficiently.

UserAuthentication:

- Manages user login and session verification
- Handles account detail validation
- Controls user access and permissions
- Updates publisher profile information
- Maintains account status and verification
- Processes contact information updates PropertyManagement:
- Facilitates complete property listing lifecycle
- Manages property listing creation and updates

- Handles property status tracking
- Processes property moderation and approvals
- Manages property type selection and details
- Coordinates document upload and verification
- Controls property listing visibility
- Handles photo and media management
- Maintains property information accuracy
- Processes compliance and final approval

3.2.2 Property System Interface Description

The Property System incorporates several interfaces to facilitate user interaction and property management.

Validation Interface: Ensures the accuracy and completeness of uploaded property details. Invalid or incomplete entries prompt users to make corrections before proceeding.

Published Properties Interface: Displays all published properties in an organized view. Users can update property status, modify details, and retrieve information.

Notification and Confirmation Interface: Sends confirmation messages upon successful publication of a property and updates users about property-related activities, such as bids received or investment returns.

Database Interface: Safely stores all property details, user inputs, and system-generated updates in a structured format, ensuring easy access for analysis and audits.

3.2.3 Component Publish Property System Processing Detail Start

-The user logs in through the User Login Interface and accesses the homepage.

-The user selects an option to publish a property or manage previously published properties.

Process

-If the user chooses to publish a property, they upload details (land information and pictures) via the Property Publishing Interface.

-The system validates the data through the Validation Interface.

- If valid, the property is published for auction, sale, or investment, and a confirmation message is sent.

- If invalid, the system prompts the user to correct the details before proceeding.

-If the user chooses to manage properties, the Published Properties Interface displays all listed properties.

- Users can update property statuses, edit details, or retrieve specific property information.

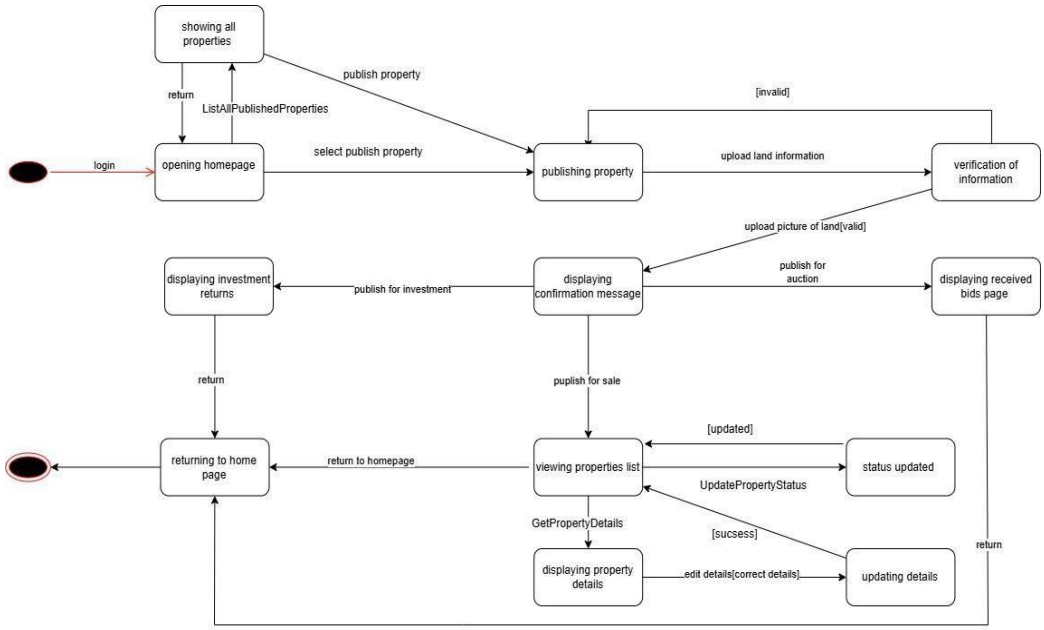
-Notifications are sent to users regarding successful updates, bids, or investment activity.

End

-The system logs all actions, including published properties and updates, in the database for future reference.

-Users can return to the homepage to perform additional tasks.

3.2.3.1 Design Class hierarchy for Feedback and Issue Resolution System



The Property Publishing and Management statechart illustrates a comprehensive real estate system that begins with user login and flows through various property management functionalities. After login, users enter the opening homepage where they can view all published properties or initiate new property publishing. The publishing process involves uploading land information and property pictures, followed by a verification stage to ensure data validity. Once verified, properties can be published for different purposes including investment, auction, or sale, with each path leading to specific display pages (investment returns, received bids, etc.). The system maintains property details through a viewing properties list state, where users can update property status and edit details as needed. Throughout the workflow, users can return to the homepage, and the system includes status
update confirmations and detail correction capabilities. The statechart emphasizes a structured approach to property management while maintaining user navigation flexibility and data integrity through various states and transitions.

3.2.3.2 Restrictions/limitations for component Feedback and Issue Resolution System

Authentication and Access

The system has strict login dependencies and access control restrictions. Users must authenticate before accessing any property publishing features. The sequential nature of the publishing workflow prevents parallel property submissions or simultaneous updates.

Publication Process

The property publication process follows a rigid sequential flow from information upload to verification. Each property must complete all verification steps before becoming visible. Invalid submissions must restart the process, potentially causing delays.

Property Status Management

Property status updates can only be processed one at a time, limiting the system's ability to handle bulk updates. The synchronization between display pages and status updates may cause temporary inconsistencies during high traffic.

3.2.3.3 Performance issues for component Feedback and Issue Resolution System

Concurrency Handling

- Multiple users attempting to publish properties simultaneously
- Concurrent access to property listing views
- Simultaneous status update requests
- Multiple bid processing for auction properties Response Time
- Property image upload and verification delays
- Listing refresh latency during status updates
- Bid display synchronization delays
- Homepage loading time with multiple properties

3.2.3.4 Design constraints for component Feedback and Issue Resolution

System Data Security

- Secure storage of property information
- Protection of bid data and user details
- Access control for property editing
- Secure verification process for property documents Compliance

- UAE property listing regulations
- Data protection requirements for user information
- Transaction recording requirements
- Property verification documentation standards

3.2.3.5 Processing detail for each operation of component Feedback and Issue Resolution System

Authentication & Home Page Operations

- Login: Authenticates user credentials and establishes session
- OpeningHomepage: Initializes and displays the main dashboard interface
- ListAllPublishedProperties: Retrieves and displays all published property listings

Property Publishing Operations

- PublishProperty: Initiates the property publication workflow
- SelectPublishProperty: Allows selection of specific property for publication
- UploadLandInformation: Processes and validates property/land details
- UploadPictureOfLand: Handles property image upload and validation

Verification Operations

- VerificationOfInformation: Validates submitted property information
- DisplayingConfirmationMessage: Shows verification status and next steps

Publication Type Operations

- PublishForInvestment: Processes investment-focused property listings
- PublishForAuction: Handles auction-based property listings
- PublishForSale: Manages direct sale property listings

Property Management Operations

- ViewingPropertiesList: Displays comprehensive property inventory
- DisplayingPropertyDetails: Shows detailed property information
- UpdatePropertyStatus: Manages property status changes
- UpdatingDetails: Handles property information modifications

Processing Narrative (PSPEC)

Authentication & Homepage Operations

Login()

- Validates user credentials against stored data
- Establishes secure session
- Redirects to homepage upon successful authentication
- Handles failed login attempts with appropriate error messages

OpeningHomepage()

- Initializes user dashboard
- Loads relevant property listings
- Displays user-specific notifications and alerts
- Provides access to primary system functions

Property Publishing Operations

PublishProperty()

- Initiates new property listing workflow
- Validates user permissions
- Creates preliminary listing record

- Prepares upload interface for property details

UploadLandInformation()

- Processes property specifications
- Validates required fields
- Performs data format checking
- Stores property information securely

VerificationOfInformation()

-Reviews submitted property data
-Validates against system requirements
-Checks for completeness and accuracy
-Flags any discrepancies for correction

Publication Type Operations

PublishForInvestment()

-Processes investment-specific details
-Calculates relevant financial metrics
-Generates investment prospectus
-Sets up investment tracking parameters

PublishForAuction()

-Initializes auction parameters
-Sets bidding rules and timeframes
-Creates auction listing
-Prepares bid tracking system

- Validates modified data
- Maintains change history
- Updates associated records

State Transitions

The system follows a logical flow:

1.User authentication → Homepage

2.Property listing creation → Information verification

3.Publication type selection → Specific workflow execution

4.Status updates → Property list refresh

5.Return to homepage for new operations

Each transition maintains data integrity and user session context throughout the process flow.

3.2.3.5.2 Algorithmic model (e.g., PDL) for each operation

Login Operation
Start Validate input credentials Create user session Redirect to homepage End
Opening Homepage Operation
Start Load user dashboard Retrieve active listings Display system options End
Publishing Property Operation
Start Validate property details Upload property information Set listing parameters Initialize publication status End

Verification of Information Operation
Start Check document authenticity Validate property details Verify ownership records Process verification status End
Displaying Confirmation Message Operation
Start Generate status message Display verification results Show next action steps End
DisplayingReceivedBids Operation
Start Retrieve active bids Update bid rankings Display bid status End
ViewingPropertiesList Operation
Start Load property inventory Apply list filters Display property status Update listing view End

DisplayingPropertyDetails Operation
Start Load property information Display property images Show property specifications Present transaction history End
UpdatingDetails Operation
Start Retrieve current details Process updates Validate changes Save modifications End
Status Updated Operation
Start Process status change Update system records Generate notifications End
Returning to Home Page Operation
Start Save current state Clear session data Load homepage content End

3.2 Description for Component: apply for loan

3.2.1 Processing Narrative (PSPEC) for Component Apply for loan

The *Apply for Loan* component outlines the sequential flow triggered by specific events during the loan application process. Users initiate the process by providing credentials, selecting a loan type, and submitting required documents. The system processes each step, performing validation and background checks before reaching a final decision. Triggering events like Add some information(), Request acceptance(), or Finalize process() dictate transitions between states, enabling smooth progression or error handling.

3.2.2 Apply for Loan Interface Description

The Apply for Loan interface is initiated when a user starts the loan application process. It begins with user identity verification (verify user()), where credentials are provided and validated. Once verified, the user selects a loan type (choosing loan()) and fills in the required details (Add some information()).

Next, users upload supporting documents like proof of income and identification (upload Bank statement certificate()), which are validated for authenticity. The interface then conducts background checks (inspecting background()) and, based on the results, either proceeds to final approval (Request acceptance()) or flags the application for review.

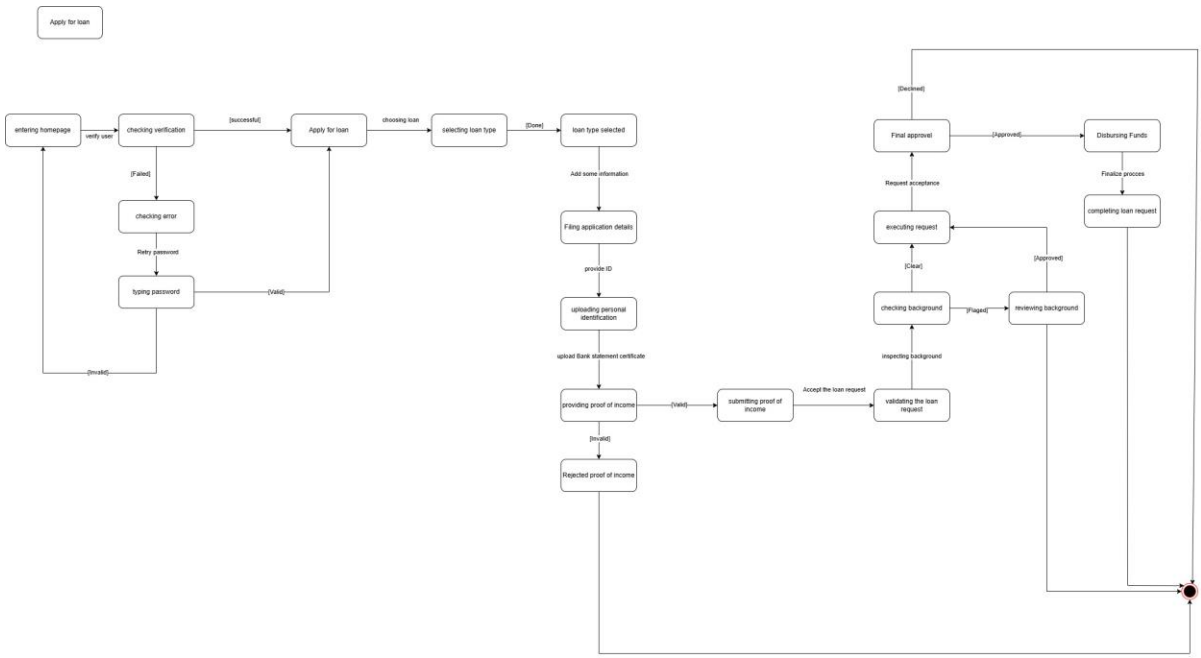
If approved, the funds are disbursed (Finalize process()), completing the process. Invalid inputs, failed validations, or flagged checks may loop the user back for corrections or terminate the application. The interface provides feedback at every step for a seamless user experience.

3.2.3 Processing Detail for apply for loan

- **Initial Verification:** Validates user credentials.
- **Loan Type Selection:** Captures the chosen loan type and ensures its validity.
- **Application Submission:** Processes personal information and uploaded documents.
- **Background Verification:** Uses external systems to verify the user's eligibility.
- **Final Approval:** Combines all input and validation results to determine the outcome.
- **Loan Disbursement:** Completes the process and transfers funds.

Each triggered event drives specific transitions and executes associated actions.

3.2.3.1 Design Class Hierarchy for Component apply for loan



- The *Apply for* Loan state chart begins at the **Entering Homepage** state, where the user initiates the loan application process. The system moves to **Checking Verification**, triggered by verify user(). Here, the user provides credentials (email and password) for validation. If valid, the process proceeds; otherwise, the user is prompted to retry or exit after repeated failures.
- Upon successful verification, the process transitions to **Apply for Loan**, where the user selects a loan type (choosing loan()), leading to **Selecting Loan Type**. After choosing a valid loan, the system moves to **Filing Application Details**, where users input personal and loan-specific information. Next, in **Uploading Personal Identification**, users submit identification documents, followed by **Providing Proof of Income**, where financial documents like bank statements are uploaded and validated. Invalid submissions prompt users to resubmit; valid submissions move to **Submitting Proof of Income**.
- The system then enters **Validating the Loan Request**, ensuring all information meets eligibility criteria. Once validated, the process transitions to **Checking Background**, where external systems perform a credit and history check. If flagged, the application enters **Reviewing Background** for manual inspection. Clear results move to **Final Approval**, triggered by Request acceptance(). Approved applications proceed to **Disbursing Funds**, where the loan is finalized and funds are transferred, completing the process in **Completing Loan Request**. Invalid or failed steps at any stage cause the system to loop back or terminate the process.

3.2.3.2 Restrictions/Limitations for apply for loan

All triggered events require valid data inputs, Background checks depend on external services, which may introduce delays, incorrect document submissions or user errors can cause application rejections.

3.2.3.3 Performance Issues for apply for loan

Slow response times during high traffic could delay triggering events like Request acceptance() or inspecting background().
Heavy reliance on external systems for document validation and background checks may result in latency.
Large file uploads (e.g., documents) could cause processing delays.

3.2.3.4 Design Constraints for apply for loan

Must securely handle sensitive user data (e.g., personal identification, financial records).
Should integrate seamlessly with external verification and financial systems.
Needs scalability to support multiple concurrent loan applications.

3.2.3.5 Processing Detail for Each Operation of Component in apply for loan

verify user(),choosing loan(),Add some information(),upload Bank statement certificate(),inspecting background(),Request acceptance(),Finalize process()

3.2.3.5.1 Processing Narrative (PSPEC) for Each Operation in apply for loan

verify user()

The system receives email and password inputs, validates them against the database, and transitions to the next state if successful. If failed, the system prompts for retries or terminates the process after maximum attempts.

choosing loan()

The user selects a loan type, which the system validates to ensure compatibility with user eligibility. The system saves the selection and transitions to the application submission state.

Add some information()

The user provides additional details for the loan application. The system verifies completeness and accuracy before storing the data and moving to the document submission state.

upload Bank statement certificate()

The user uploads proof of income, and the system validates the documents. Invalid submissions are rejected, prompting the user to correct and resubmit

inspecting background()

The system sends user data to external verification services for background checks. If flagged, the application enters manual review; otherwise, it transitions to final approval.

Request acceptance()

The system aggregates all application data, evaluates it against eligibility criteria, and determines final approval or rejection.

Finalize process()

For approved applications, the system initiates fund disbursement and notifies the user that the loan request is complete.

3.2.3.5.2 Algorithmic Model (PDL) for Each Operation in apply for loan

verify user()

Start
Request email and password
Validate credentials
If valid, proceed to the next state
If invalid, prompt to re-enter credentials
End

choosing loan()

Start
Display available loan options
Select a loan type
Confirm selection
End

Add some information()
Start
Request user to fill in loan details
Validate the entered information
If valid, proceed to the next state
If invalid, prompt for corrections
End
upload Bank statement certificate()
Start
Request proof of income and bank statement
Upload the document
Validate the uploaded document
If valid, proceed to the next state
If invalid, prompt for re-upload
End
inspecting background()
Start
Initiate background checks via external systems
Review financial and personal history
If clear, proceed to approval
If flagged, transition to manual review
End
Request acceptance()
Start
Aggregate all submitted information
Verify application completeness and correctness
Make approval or rejection decision
If approved, proceed to disbursing funds
If rejected, terminate the process
End
Finalize process()
Start
Prepare funds for disbursement
Transfer funds to the applicant
Confirm transaction
End

3.2 Description for Component: Log in & user account management

3.2.1 Processing Narrative (PSPEC) for Component log in & user account management.

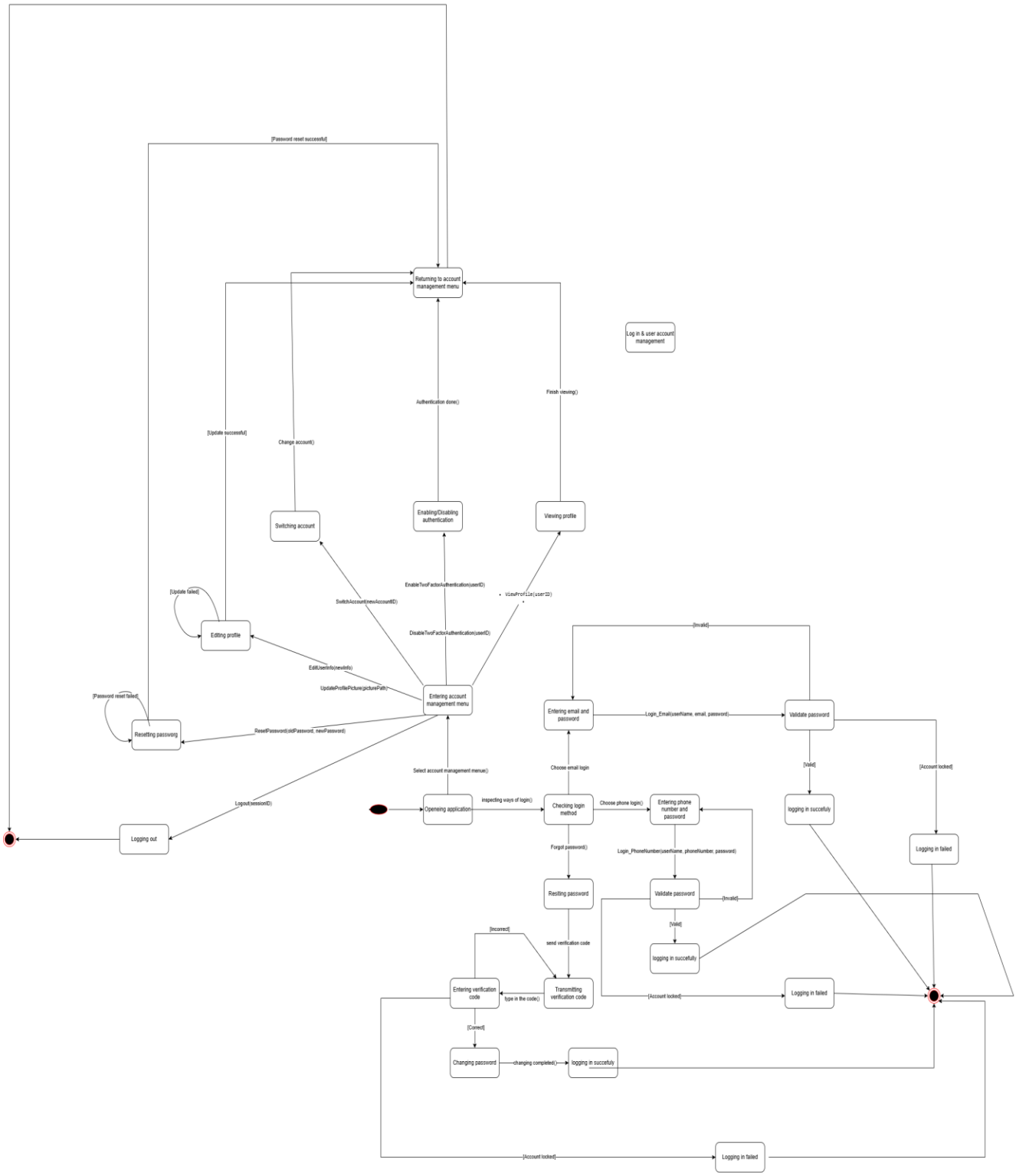
The Login & user account management Component manages user authentication through various methods, including email, phone number, and username. It includes attributes for user credentials (e.g., username, email, password) and functions such as Login Email, Login PhoneNumber, and Forget Password for handling login, password recovery, and session management. This component ensures secure validation of user data, tracks login attempts, and provides mechanisms to reset passwords and lock accounts when necessary.

3.2.2 log in & user account management interface description

The Login and User Account Management Component serves as the central interface between the User Interface (UI), the Authentication System, and the User Database. The UI collects input from the user, such as credentials (email, phone number, password) or profile updates, and communicates with this component to process requests. The component interacts with the Authentication System to validate login attempts, manage password recovery, and enforce security measures like account locking or enabling two-factor authentication. Simultaneously, it interfaces with the User Database to retrieve, update, and manage user profiles, account histories, and security settings. This component ensures seamless interaction between the user and the system for secure access and account management.

- **3.2.3 Processing Detail for Login & User Account Management**
- The user opens the application and selects the login or user account management menu.
- If the user chooses to log in:
- The system verifies the login method selected (email, phone number, or username).
- The user enters their credentials, which are validated against the stored records.
- If the credentials are incorrect, the system increments the login attempt counter and displays an error.
- If login attempts exceed the threshold, the account is locked.
- If the credentials are correct, the system grants access to the homepage.
- If the user forgets their password:
- The system prompts for the registered email or phone number.
- A verification code is sent to the user for confirmation.
- Upon successful verification, the user can reset their password.
- If the user accesses the user account management menu:
- The system retrieves and displays the user profile from the database.
- The user can view their account history, update profile details, or switch to a different account.
- Profile updates (e.g., name, email, or profile picture) are validated and saved to the database.
- For security settings:
- The user can enable or disable two-factor authentication, requiring verification before saving changes.
- Password reset requests require the old password to be verified before accepting the new password.
- **End**
- Once the user completes their tasks (e.g., login, profile updates, or account switching), the session can end by logging out.
- The system resets login attempts upon successful login or password recovery.

3.2.3.1 Design Class Hierarchy for Component log in & user account management



The state chart diagram illustrates the flow of various account management and authentication processes. It begins with core functionalities like logging in, logging out, and profile management. The login process includes entering email and password, validating credentials, and determining outcomes such as successful login or failure. For password management, users can reset their passwords by providing an email, validating a reset token, and setting a new password. The diagram also covers profile-related actions like viewing, editing, and updating profiles, as well as enabling or disabling two-factor authentication. Account management features include switching between accounts and recovering accounts through email verification or other recovery methods. Additionally, the chart handles error scenarios like invalid credentials, expired tokens, or unsuccessful actions, ensuring robust error-handling mechanisms. Overall, the diagram provides a comprehensive view of the system's state transitions for managing user accounts and authentication.

3.2.3.2 Restrictions/Limitations for Component log in & user account management

Limited to one active session per user and Requires internet connectivity for verification.

OTP-based recovery depends on email or phone services.

3.2.3.3 Performance Issues for Component log in & user account management

Password hashing can be computationally intensive, and High traffic may cause delays in OTP delivery.

Locking mechanisms could introduce latency in high-concurrency scenarios.

3.2.3.4 Design Constraints for Component log in and user account management

Must adhere to security standards for password storage (e.g., hashing with salt), Ensure support for international phone formats and Enforce limits on login attempts.

3.2.3.5 Processing Detail for Each Operation of Component in log in & user account management

The classes log in & user account management have functions called: Login_Verification(email, password), Validate password(),Choose login option(),Change password(),Switching account(),Editing profile(),Resetting password(), Viewing profile(),Logging out()

3.2.3.5.1 Processing Narrative (PSPEC) for Each Operation log in & user account management

Login_Verification(email, password)

The system requests the user's email and password, validates them against the stored database, and determines if the credentials are correct. A successful login transitions the user to their account dashboard, while invalid credentials result in a failure message prompting a retry.

Validate password()

The user-provided password is compared with the stored hash in the database. If matched, the user is granted access; otherwise, they are requested to retry entering the password.

Choose login option()

The system presents multiple login methods to the user (e.g., standard login, third-party OAuth). Based on the user's selection, the system initiates the corresponding verification process.

Change password()

This process begins by verifying the user's current password. If verified, the system allows the user to enter a new password, checks its validity (e.g., length, complexity), and saves the updated password in the database.

Switching account()

This event logs out the current user, clears the session, and transitions the interface back to the login screen, allowing the user to log in with another account.

Editing profile()

The system fetches and displays the user's profile information, allowing specific fields to be updated. After validation, the new data is saved, and the profile is updated.

Resetting password()

When a user forgets their password, the system requests their email and sends a recovery link or OTP. After validating the OTP or link, the user is allowed to set a new password.

Viewing profile()

The system retrieves the user's profile data from the database and displays it in a read-only format, ensuring no accidental edits.

Logging out()

The system ends the user session by clearing all session data and redirecting the user to the login page for security purposes.

3.2.3.5.2 Algorithmic Model (PDL) for Each Operation in log in & user account management

Login_Verification(email, password)
Start
Request email and password from the user
Validate credentials with the database
If valid, proceed to the "Logging in successful" state
If invalid, proceed to the "Logging in failed" state
End

Validate password()
Start
Compare the entered password with stored data
If valid, allow login
If invalid, request the user to retry
End

Choose login option()
Start
Display login options (e.g., standard login, third-party login)
Accept user selection
Transition to the relevant login verification process
End

Change password()
Start
Request the current password for authentication
Allow the user to input a new password
Validate the new password format
Save the updated password
End

Switching account()
Start
Log out of the current account
Transition to the login interface
Allow user to select another account to log in
End

Editing profile()
Start
Display the user's current profile details
Allow the user to edit desired fields
Validate the changes and save them
End

Resetting password()
Start
Request the user's email for account recovery
Send a verification email or OTP
Validate the provided OTP or email link
Allow the user to set a new password
End

Viewing profile()
Start
Retrieve and display user profile details
Allow read-only access to the information
End

Logging out()
Start
End the user session
Redirect to the login page
Clear session data for security
End

CheckValidityOfLand()

```
Start

IF verifyCustomer(customerID) THEN
   Retrieve stored blockchain data for landRecordID
   Compare landRecordID, landOwnerID, and blockchainHash with stored
data

   IF details match THEN
      Flag record as valid
      Return true
   ELSE
      Return false
      Display error message indicating access is restricted
   ENDIF
ELSE
   Display error message indicating access is denied
ENDIF

End
```

GenerateRecordReport()

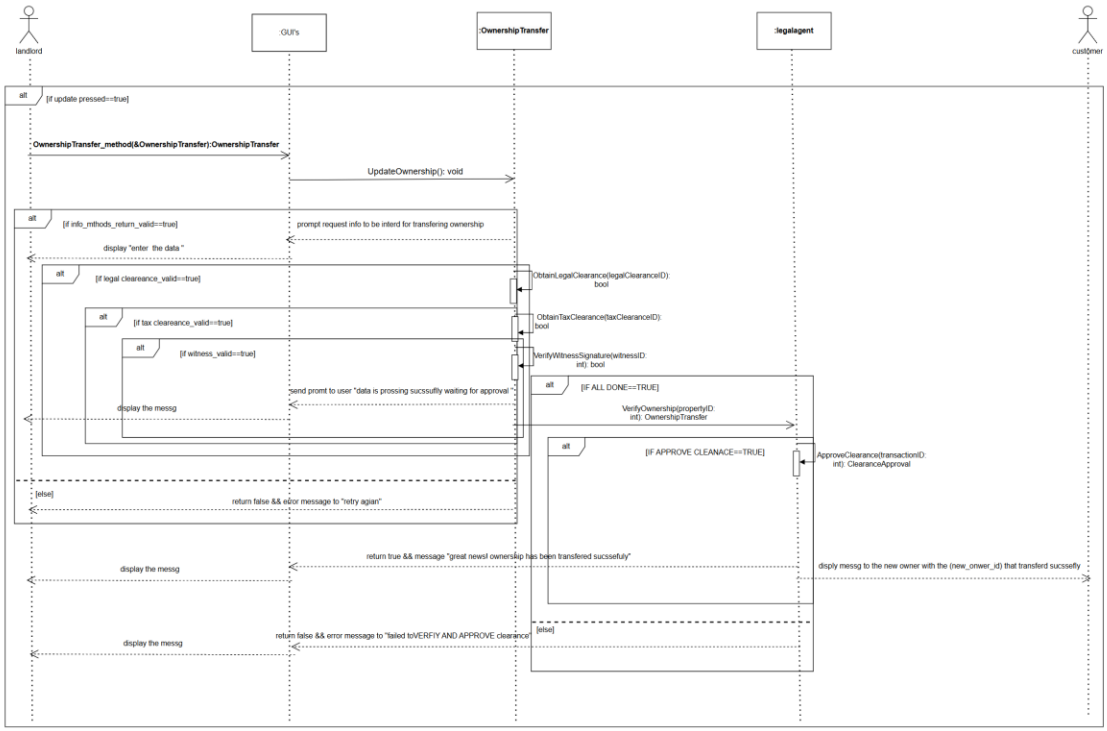
```
Start

IF isVerified(customerID) THEN
   Collect details for landRecordID (owner's information, transaction
history, blockchain hash)
   Format details into a structured report
   Make report available for viewing or download
ELSE
   Display error message indicating access is denied
ENDIF

End
```

3.3 Dynamic Behavior for Components Interaction Diagrams

3.3.1 ownership transfer



The sequence diagram illustrates the process of ownership transfer within a blockchain-based system involving multiple entities, such as the Customer, OwnershipTransfer, and Landlord.

Ownership Transfer Request:

The process begins with the Customer initiating a request to transfer ownership by sending a message to the `OwnershipTransfer` object.

The system validates this request to ensure it meets all necessary criteria.

Authorization Check:

Once the request is validated, `OwnershipTransfer` initiates an authorization check to confirm the Customer's eligibility for transferring ownership.

This may include verifying if the Customer has the authority to initiate the transfer and any required permissions.

Payment and Address Validation:

If the Customer passes the authorization check, `OwnershipTransfer` interacts with the `Customer` entity to process the payment.

Once payment is confirmed, the system checks the necessary information, such as land records, and retrieves the property address details.

Confirmation with Landlord:

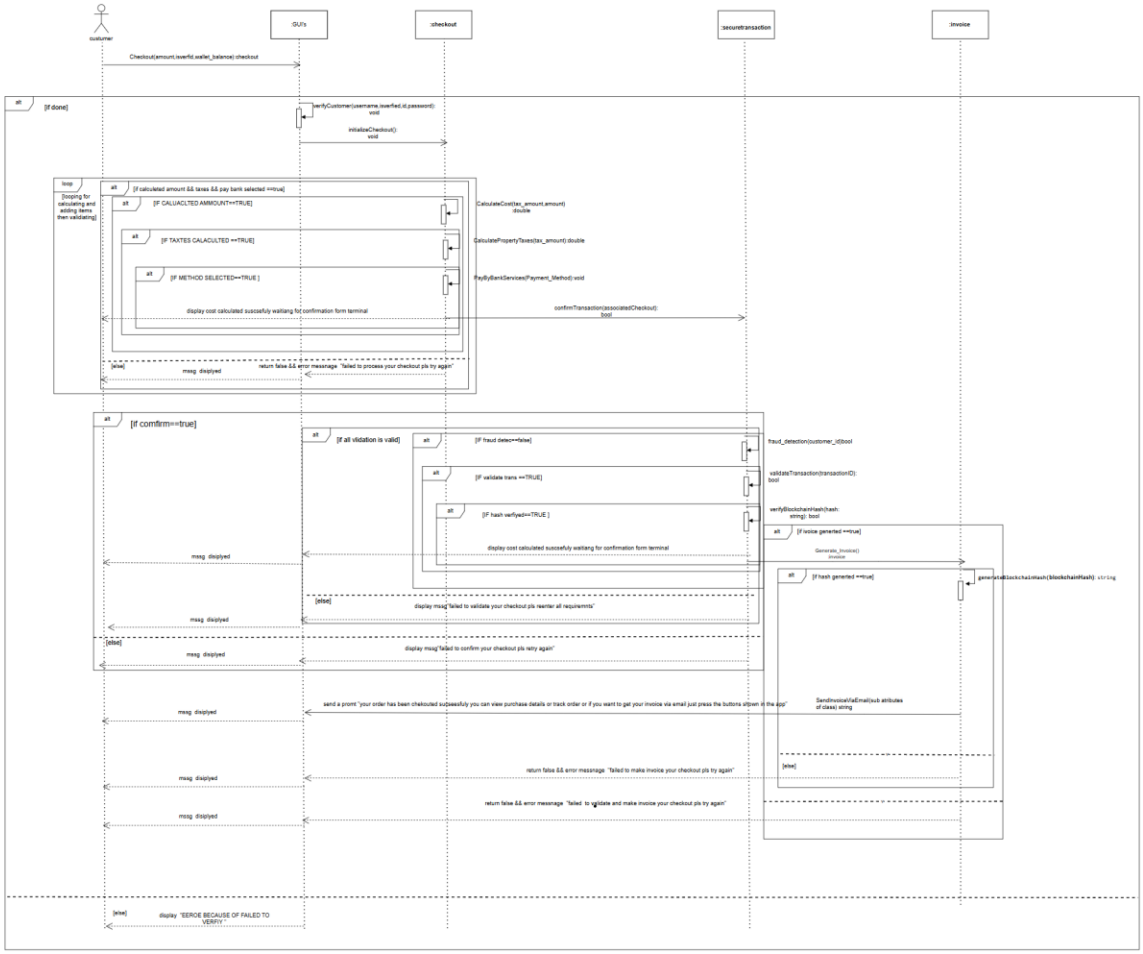
After the information is gathered, `OwnershipTransfer` contacts the `Landlord` entity to notify them of the transfer request.

The Landlord reviews the details and provides confirmation of the transfer, allowing the process to continue.

Ownership Record Update:

With both payment and authorization verified, the `OwnershipTransfer` object updates the blockchain's ownership records, transferring the title from the previous owner to the new Customer.

3.3.2 checkout



This sequence diagram illustrates the process of a Customer performing a checkout within a system that involves several entities: Customer, Checkout, Payment, Invoice, and Admin.

Key Steps:

Checkout Initialization:

The Customer initiates the checkout process by sending a request to the `Checkout` object.

The system checks if the checkout is "done" or ready to proceed.

Payment Processing:

Once the checkout is initialized, the `Checkout` object communicates with the `Payment` entity to handle payment details.

Payment verification and processing occur, ensuring the Customer's payment method is valid and funds are available.

Invoice Generation:

Following a successful payment, the `Checkout` object interacts with the `Invoice` entity to generate an invoice for the transaction.

The invoice creation confirms the details of the checkout and stores them for record-keeping.

- **Admin Notification**:

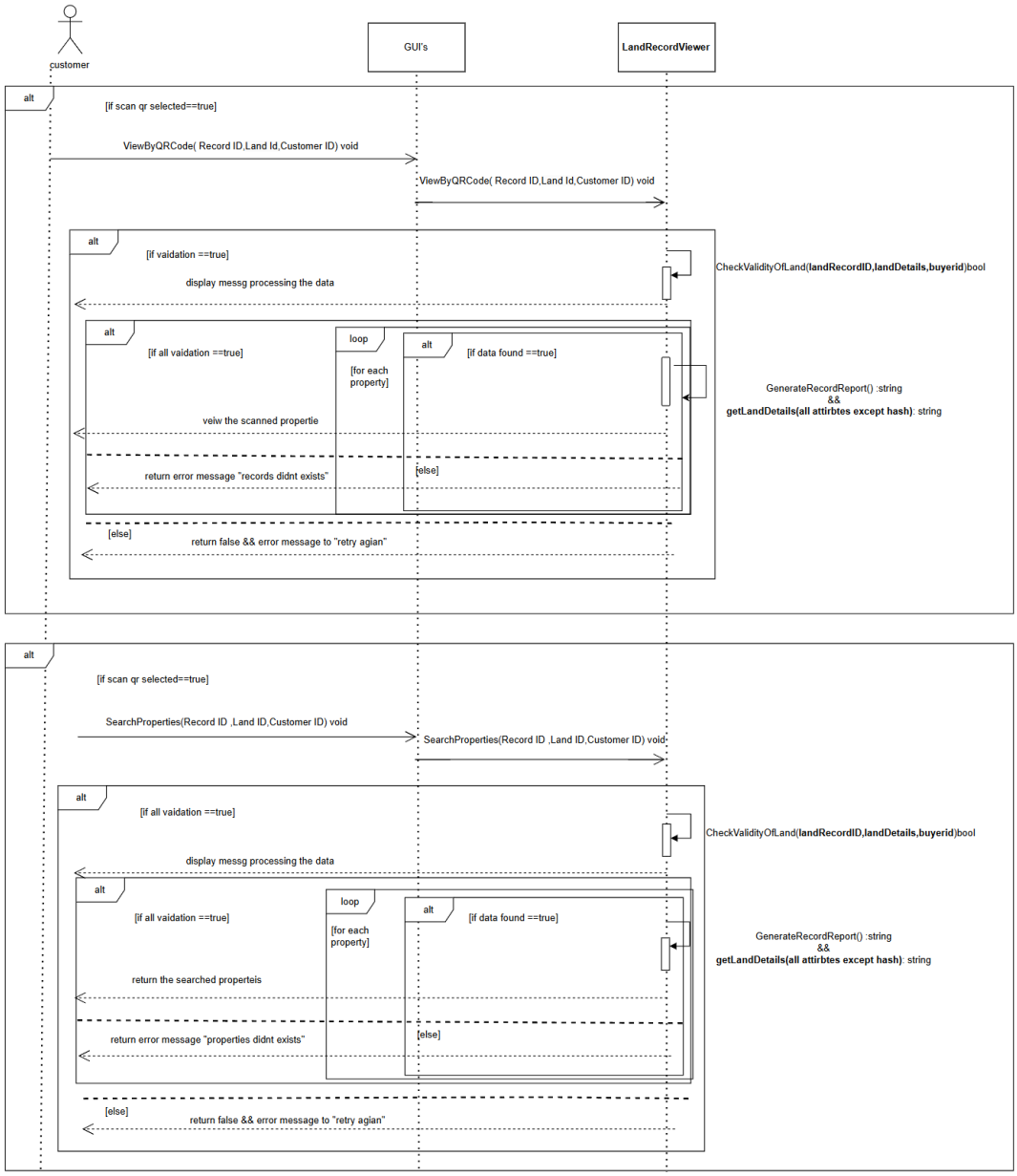
After payment and invoice generation, the system notifies the `Admin` entity to review or validate the completed checkout, if necessary.

- **Completion**:

The system sends a final confirmation back to the Customer, completing the checkout process.

This diagram provides a high-level overview of the checkout process, focusing on interactions between the Customer and system entities to validate payment, generate an invoice, and confirm the transaction.

3.3.3 view land records



This sequence diagram illustrates the "Land Record View" use case, allowing customers to view land records either by scanning a QR code or searching for records manually.

View by QR Code:

The customer initiates a request to view a record by QR code.

The system validates the data through `CheckValidityOfLand()`.

If validation succeeds, it retrieves land details using `getLandDetails()` and generates a record report through `GenerateRecordReport()`.

If data is found, the scanned property is displayed; otherwise, an error message is returned.

Search for Records:

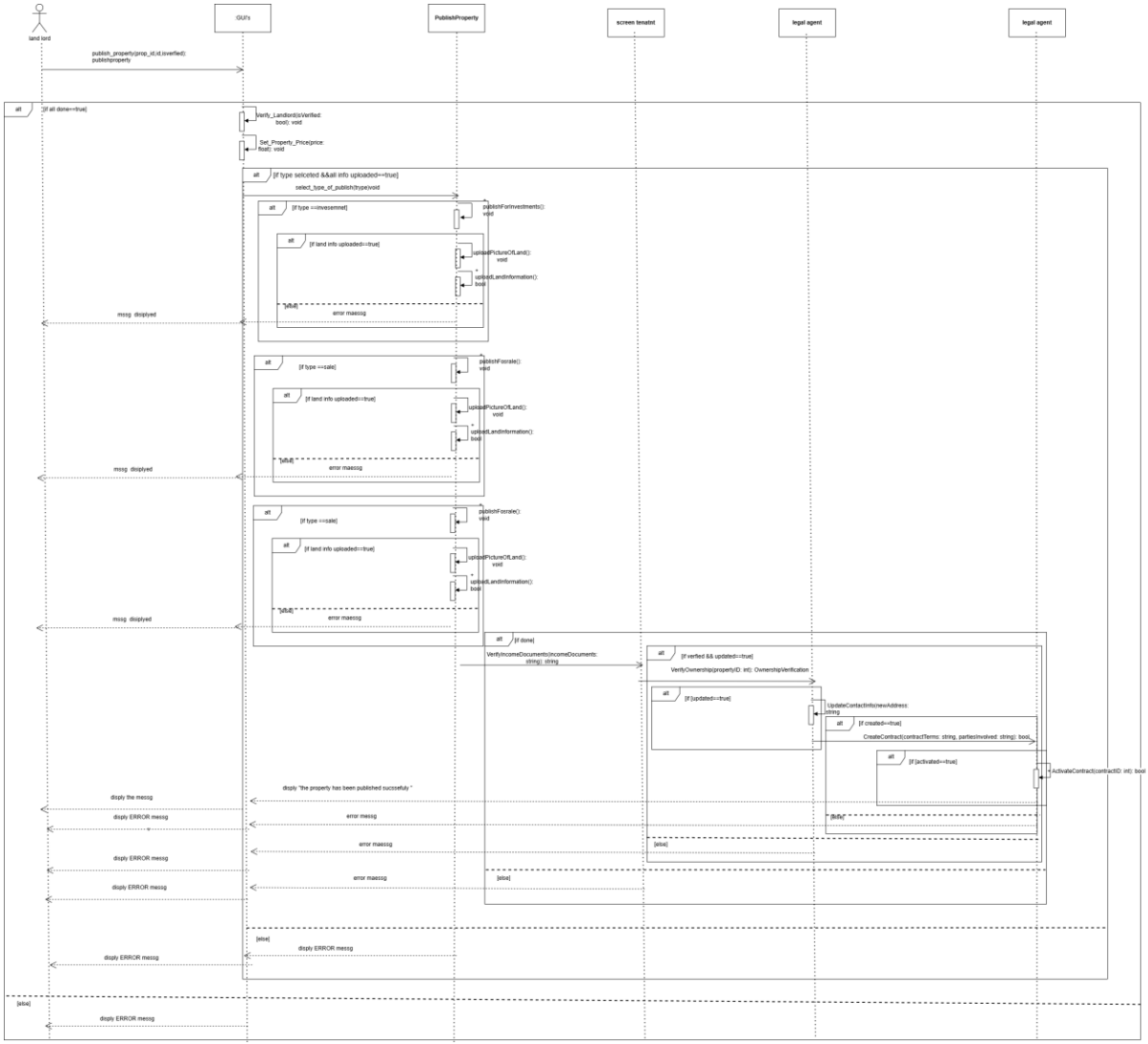
The customer initiates a search with the record ID, land ID, and customer ID.

The process similarly involves validation through `CheckValidityOfLand()`.

Upon successful validation, the system loops through each property, displaying found records. If no data is found, an error message is shown.

In both cases, if validation fails, an error message prompts the customer to retry.

3.3.4 Publish property



This sequence diagram illustrates the "Publish Property" use case, where an agent publishes a property through the system.

Property Publishing Request:

The agent initiates the process by sending a `publish_property_selected()` request to the GUI.

The system verifies the property details by interacting with `Public Property`, `Customer Details`, `Legal Agent`, and other related components.

Validation and Submission:

The system checks if the required fields are completed and if the user has selected the desired property.

For each validation step, the system interacts with various components like `Legal Agent` and `Owner Verification` to ensure compliance.

If all validations pass, the property details are saved and the property is published.

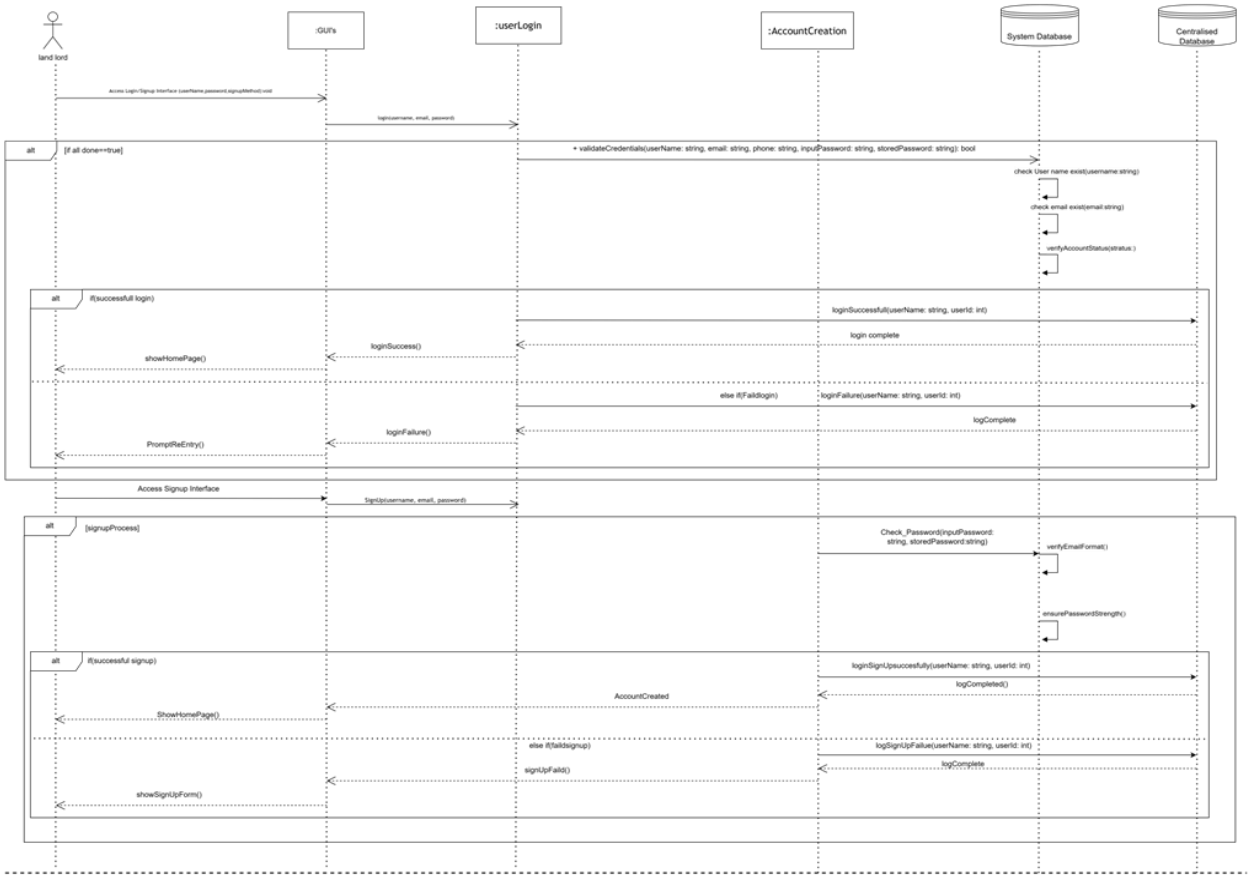
Error Handling:

If any validation fails, the system returns an error message to the agent, prompting correction and resubmission.

Different error messages are displayed depending on the specific validation

\This flow ensures that only valid and verified property details are published, with checks at each step to handle potential errors or missing information.

3.3.5 User Login



The sequence diagram depicts the process of a user logging into the system, as well as the account creation process if the user does not have an account.

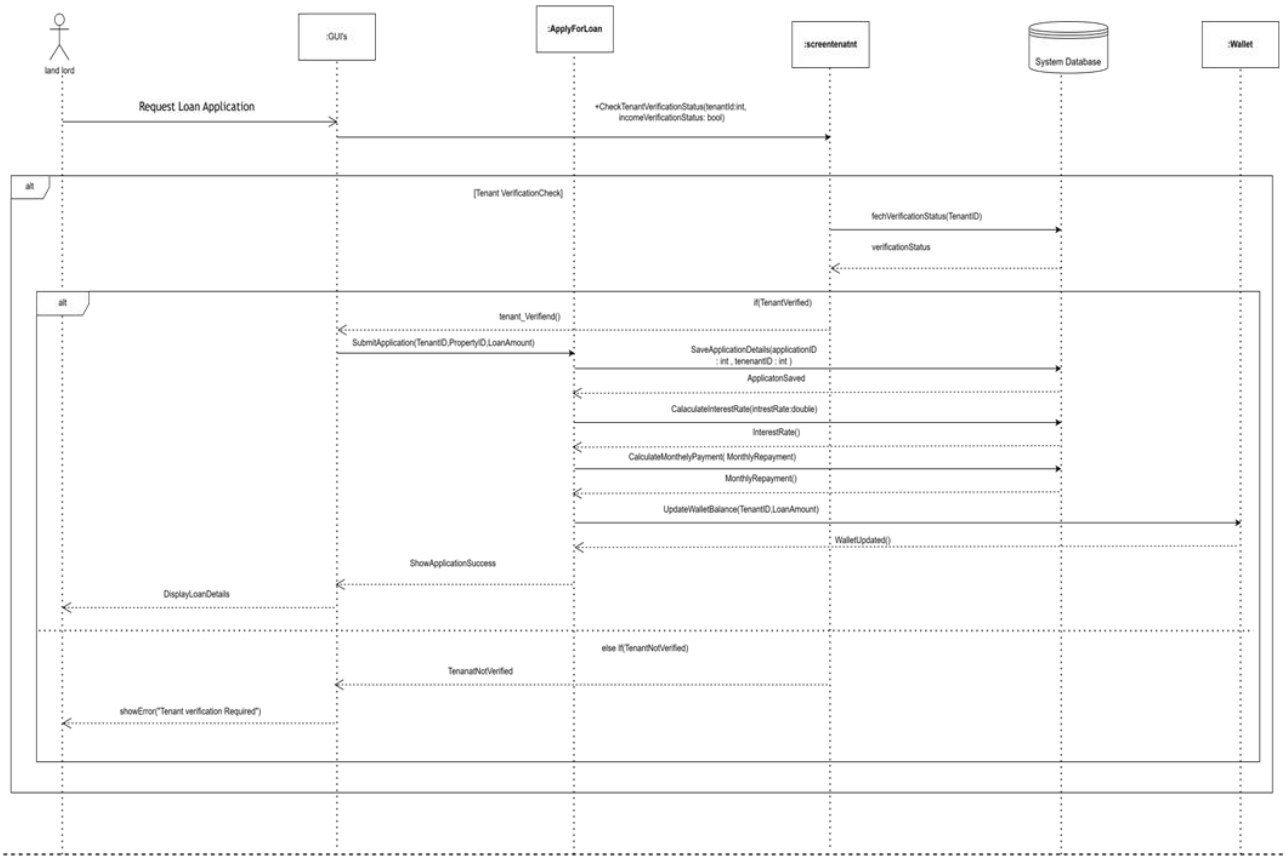
The user initiates the login process by accessing the GUI and entering their username and password. The GUI then sends the login request to the userLogin component, passing the provided credentials.
The userLogin component validates the credentials by calling the AccountCreation component, which checks the user's name, email, and password against the stored information in the system database. If the user's credentials are not found in the database, it means the user does not have an account.

In this case, the AccountCreation component will create a new account for the user. It verifies the user's information, such as name, email, and password, and then stores this new account information in the System Database.

Once the account is created, or if the user's credentials are successfully validated, the AccountCreation component informs the userLogin component that the login was successful. The userLogin component then completes the login process and notifies the GUI that the login was successful. Finally, the GUI displays the user's homepage, completing the login sequence.

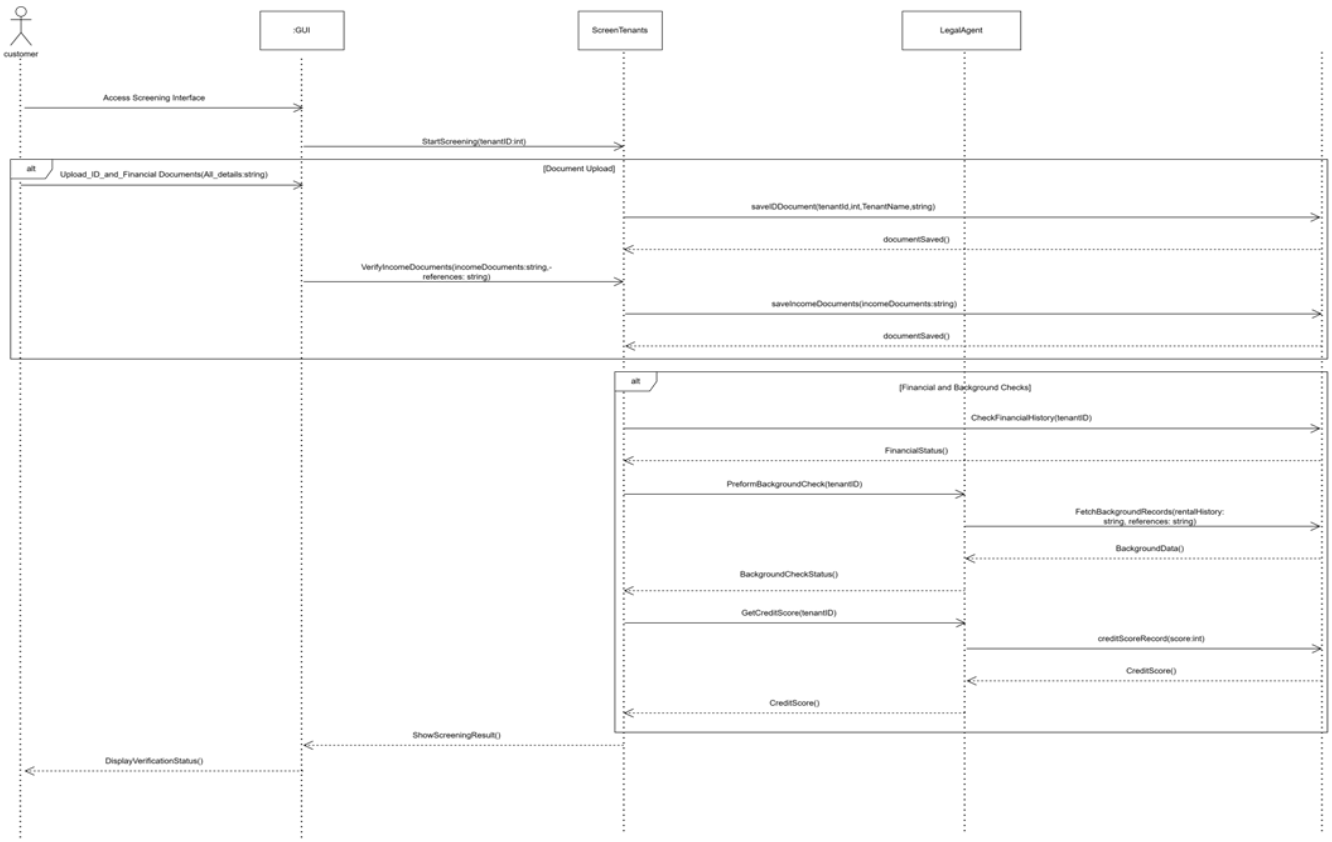
The diagram shows the interactions between the various system components, including the GUI, userLogin, AccountCreation, and the System Database, to facilitate both the user login and account creation processes.

3.3.6 Apply for Loan



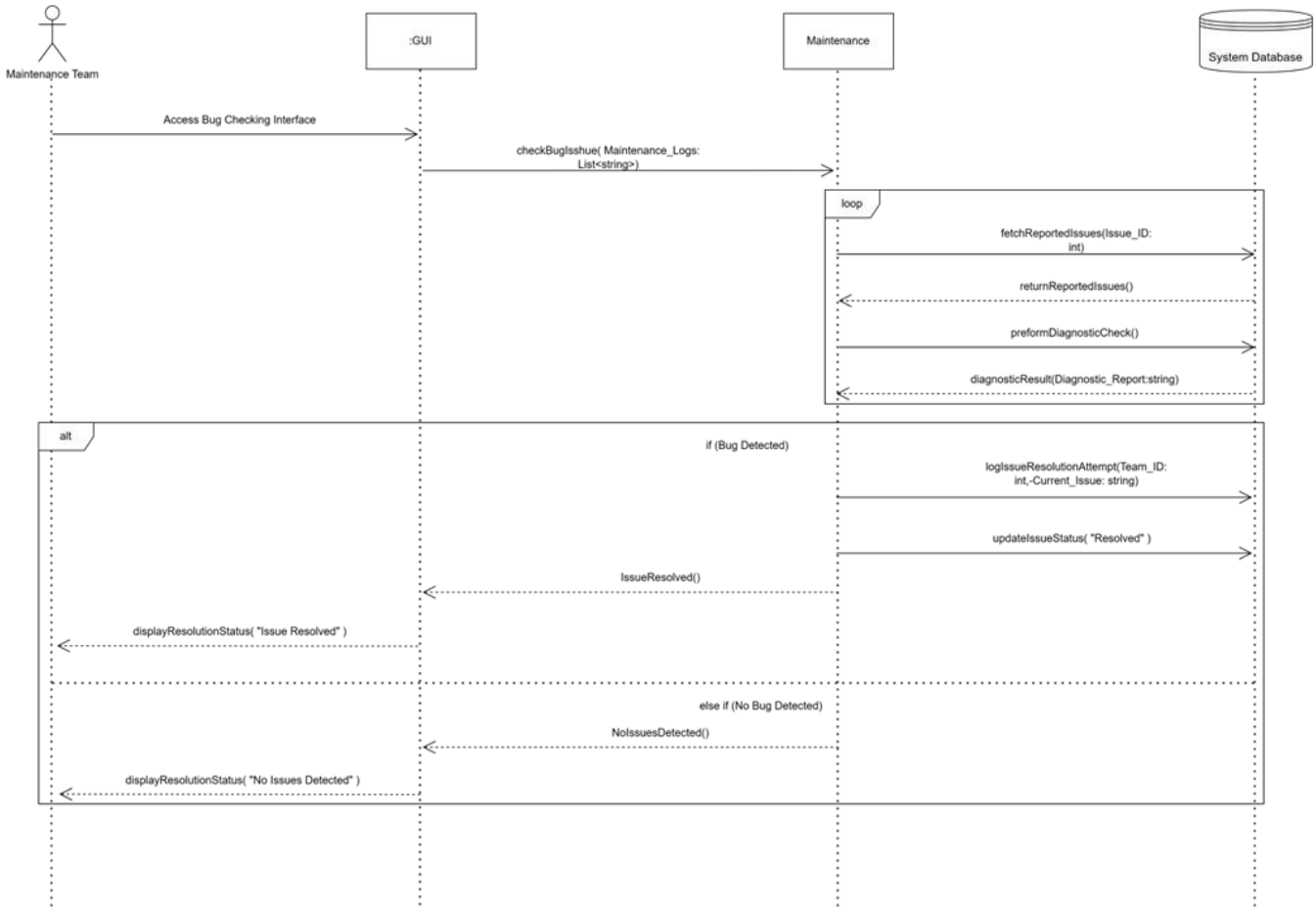
The sequence diagram depicts the process of a user applying for a loan. The user initiates the loan application by accessing the GUI and providing the necessary loan details, such as the loan amount, purpose, and personal information.
The GUI then sends the loan application request to the LoanApplication component, passing along the user's input data. The LoanApplication component interacts with the AccountCreation component to verify the user's identity and account information.
If the user's account information is not found or cannot be verified by the AccountCreation component, the LoanApplication component will notify the GUI that the tenant's account is not verified. In this case, the GUI will display an error message to the user, indicating that the account verification failed and the loan application cannot proceed.
However, if the user's account information is successfully verified by the AccountCreation component, the LoanApplication component will then process the loan application, potentially involving a CreditCheck component to assess the user's creditworthiness. The CreditCheck component provides the assessment results back to the LoanApplication component, which then analyzes all the relevant factors to make a decision on the loan application, whether to approve or reject it.
The LoanApplication component communicates the outcome, whether approved or rejected, back to the GUI. The GUI then displays the loan application result to the user, completing the process.

3.3.7 Screen Tenants



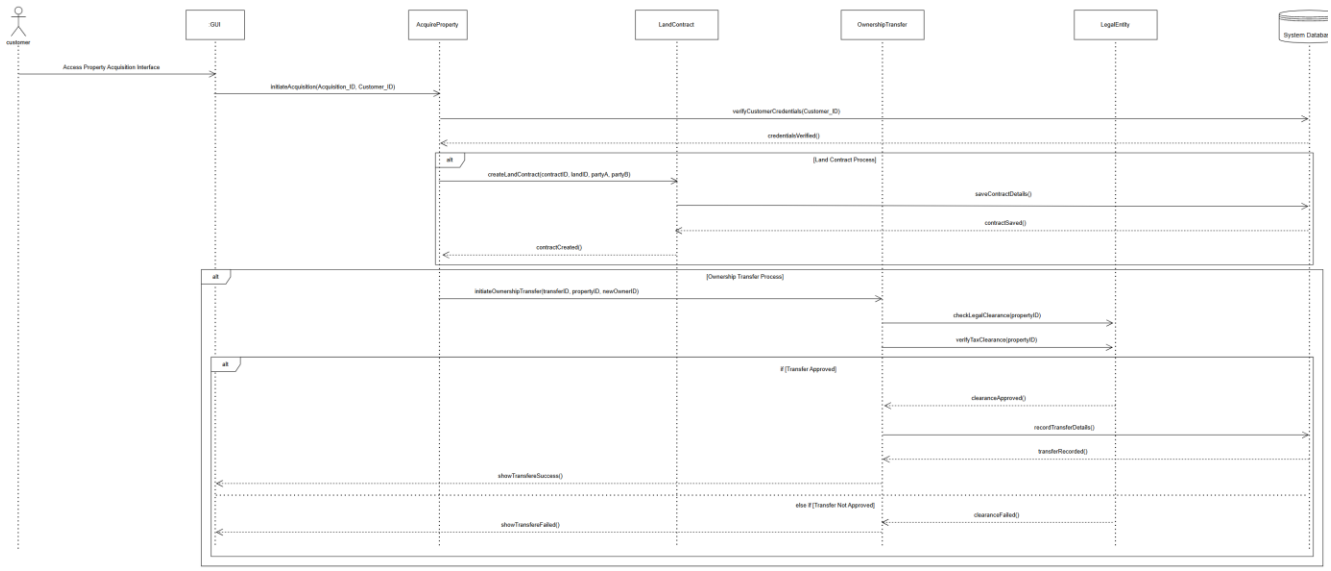
The sequence diagram depicts the process for a tenant interacting with the "ScreenTenants" component. The tenant initiates the screening process by accessing the GUI and providing the necessary information, such as uploading ID and financial documents.
The GUI sends the tenant's information, including the uploaded documents, to the ScreenTenants component. The ScreenTenants component then verifies the income documents, saves the verified income documents and the uploaded documents.
After the initial document verification and saving, the ScreenTenants component performs a series of financial and background checks on the tenant, including checking the tenant's financial history, performing a background check, retrieving the tenant's background records, and checking the status of the background check.
The ScreenTenants component then retrieves the tenant's credit score and compiles the screening results, which are displayed to the tenant.

3.3.8 Maintenance



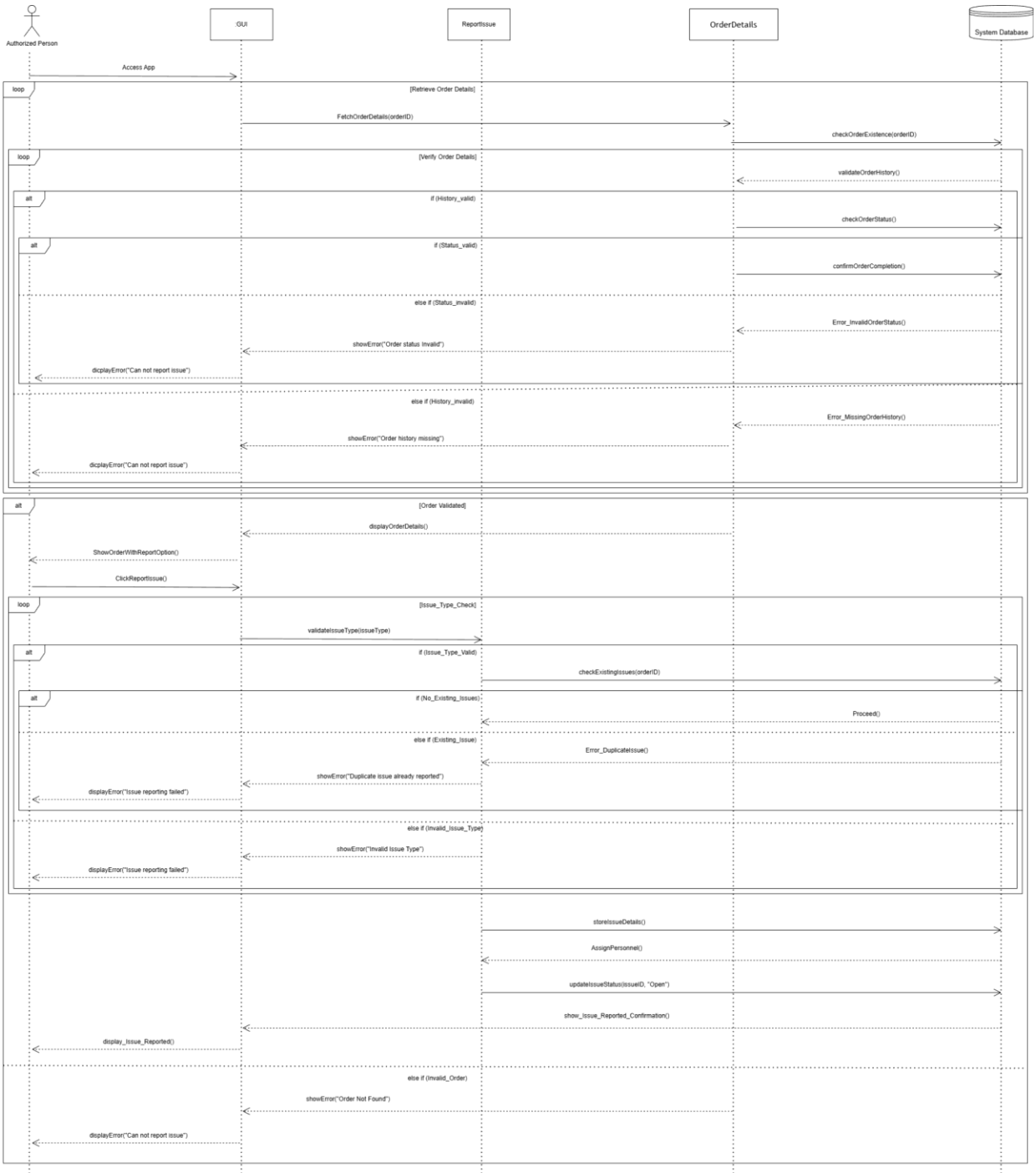
The sequence diagram depicts the process for the Maintenance team to handle bug checking and resolution. Here's a brief description:
The Maintenance team accesses the Bug Checking Interface, which is the entry point for the bug reporting and resolution process.
The GUI sends a request to the Maintenance component to check the bug issues. The Maintenance component then fetches the reported issues from the System Database, using the fetchReportedIssues method and passing the Issue ID.
The Maintenance component analyzes the reported issues by calling the performDiagnosticCheck method. This generates a diagnostic report on the issue.
If a bug is detected, the Maintenance component logs the issue resolution attempt by calling the logIssueResolutionAttempt method, passing the Team ID and the current issue details.
The Maintenance component then updates the issue status to "Resolved" by calling the updateIssueStatus method.
If no bug is detected, the Maintenance component simply logs that no issues were detected by calling the NoIssuesDetected method.
Finally, the Maintenance component displays the resolution status, either "Issue Resolved" or "No Issues Detected", to the GUI.
This sequence diagram outlines the key steps involved in the bug checking and resolution process, including fetching reported issues, performing diagnostics, logging resolution attempts, and updating issue status in the System Database.

3.3.9 Acquire Property



This sequence diagram depicts the acquisition process of a property. The customer sends an acquisition request via the GUI; this verifies his credentials and processes the creation of a land contract. Once the contract is finalized, the transfer of ownership starts by verifying the legal and tax clearances. On acceptance, it confirms the change of ownership; otherwise, it ends with a failure message.

3.3.10 Acquire Property



The sequence diagram below outlines the process flow of an authorized person accessing the application to retrieve and validate the order details. In this, there are interactions between the user, GUI, repository, order details, and the system database. Most importantly, the main actions in this are related to fetching order data, validating it against the database, and error handling due to missing history or invalid statuses. It also contains issue reporting, where issues are checked for duplicates, validated, and assigned to be resolved. The system then updates the status of the issues accordingly.

4.0 User Interface Design

A description of the user interface design of the software is presented.

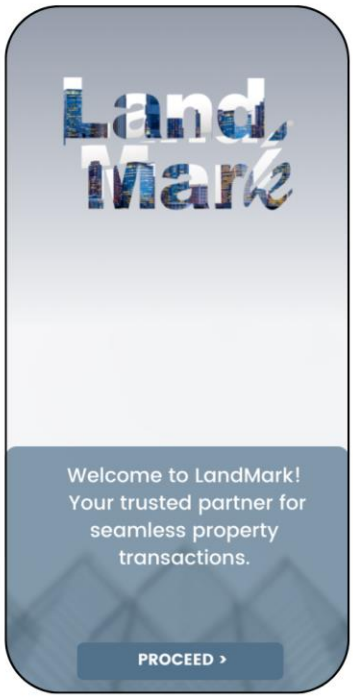
4.1 Description of the user interface

4.1.1 Screen Images

The property transaction application features a straightforward and user-friendly graphical interface, designed for easy navigation without the need for any specialized training. The interface provides a clear overview of property details, transaction progress, and user interactions, making it accessible for users of all experience levels.

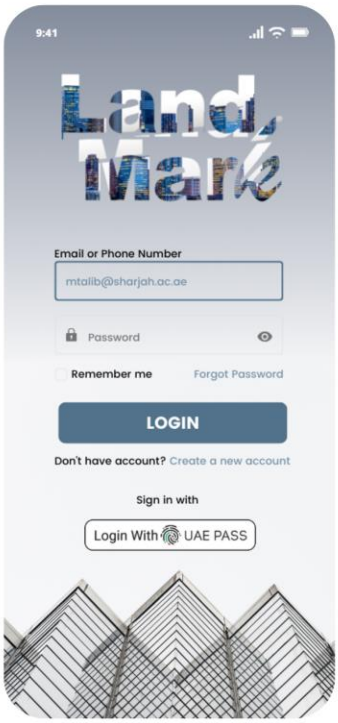
With intuitive icons and a minimalistic design, users can seamlessly browse property listings, verify ownership, track transaction histories, and communicate with relevant stakeholders. Notifications and alerts ensure users stay informed about their property dealings and deadlines. The application guarantees a seamless experience, empowering users to make confident and informed decisions regarding property transactions with security and transparency at its core.

Below are detailed images and descriptions of the GUI. The objects and actions performed when interacting with them are also identified.



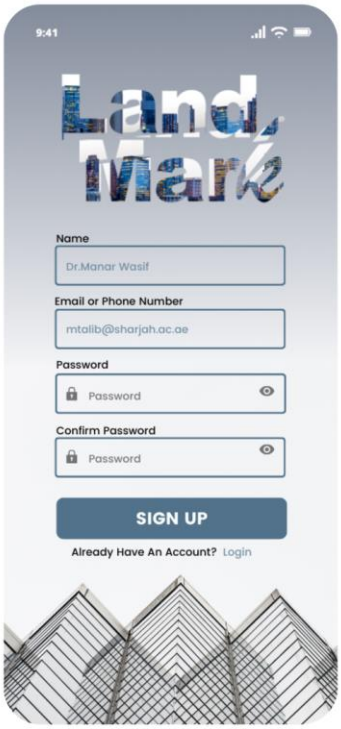
Splash Screen page

The "LandMark" welcome screen shows the logo at the top, a tagline inviting trust in property transactions, and a clear **PROCEED** button at the bottom for navigation.



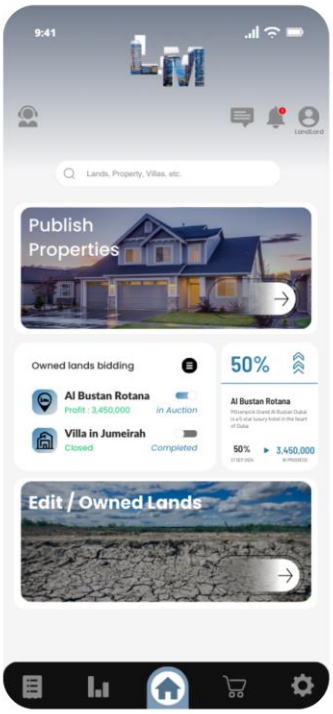
Login Page

The login screen features input fields for email/phone number and password, along with options to remember credentials and reset the password. A prominent **LOGIN** button and a **Login With UAE PASS** option provide easy access. Links to create a new account are also included for new users.



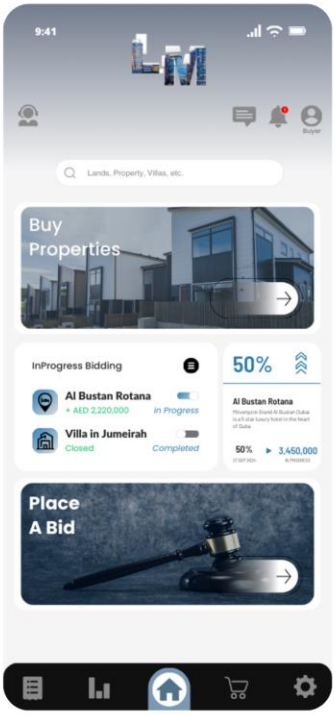
Sign-Up Page

The sign-up screen offers fields for name, email/phone number, password, and password confirmation. A clear **SIGN UP** button is placed below the fields, with a link to switch to the login page for existing users.



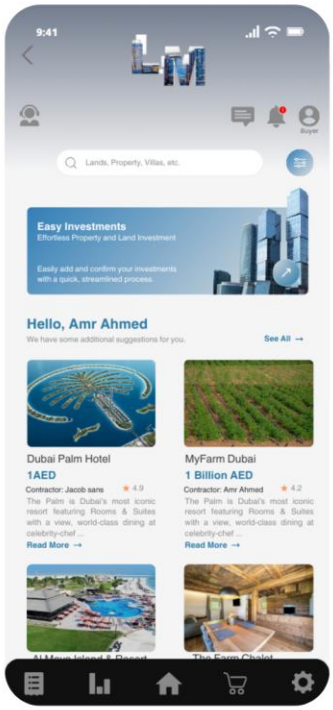
Landlord Dashboard

The landlord dashboard features a search bar for properties, quick navigation cards for publishing properties, and managing owned lands. It also displays active and completed bids with clear status indicators. A bottom navigation bar provides access to analytics, home, cart, and settings.



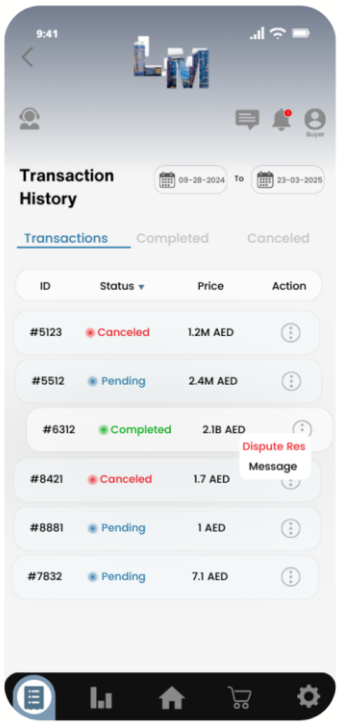
Buyer Dashboard

The buyer dashboard includes a search bar for properties, options to buy properties, and a section for placing bids. It highlights ongoing and completed bids with status updates. The bottom navigation bar ensures easy access to analytics, home, cart, and settings.



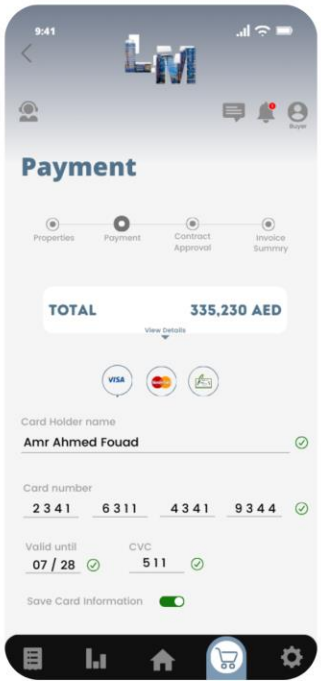
Investment Suggestions Page

This page highlights investment opportunities with a featured banner for "Easy Investments" and personalized suggestions. Users can explore property options with details like price, contractor, and reviews. A bottom navigation bar provides quick access to key sections.



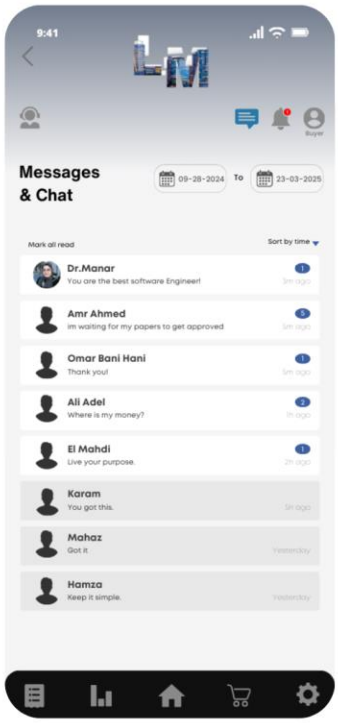
Transaction History Page

The transaction history page displays a categorized list of transactions (pending, completed, and canceled). Each entry includes transaction ID, status, price, and available actions like dispute resolution or messaging. Date filters and a clean layout enhance usability.



Payment Page

The payment page displays the total amount and provides a step-by-step progress tracker for the transaction process (properties, payment, contract approval, and invoice summary). Users can securely enter card details, with options to save the card for future use. The bottom navigation bar includes quick access to key sections.



Messages & Chat Page

The messages page allows users to view and manage conversations with other users or stakeholders. It includes filters for date ranges, options to sort by time, and a "mark all read" feature. A clean interface lists messages with timestamps for quick navigation.

4.2 User Interface Design

The GUI for our property transaction app consists of the following components:

- **Frames:** Display key information and labels.
- **Drop-Down Lists:** Used in sections like sign-up for quick selection.
- **Pop-Ups:** Provide alerts and messages for user actions.
- **Menus:** Offer options for navigation and access to features.
- **Command Buttons:** Execute tasks like submitting forms or navigating pages.
- **Scroll Bars:** Allow movement through lengthy pages or lists.
- **Chat Box:** Enables communication between users and stakeholders.
- **Toolbars or Key Buttons:** Perform multiple actions efficiently.
- **Text Fields:** Allow input and editing of user data

4.2.1 Interface Design Rules

To ensure usability and a seamless user experience, the following design principles are applied:

1.Continuity:
Consistent layouts, colors, and fonts across the app help users navigate effortlessly by creating a familiar experience.
2.Streamline:
The design focuses on essential features, avoiding unnecessary complexity to ensure a clean and intuitive interface.
3.Feedback:
Immediate responses like animations or confirmation messages reassure users that their actions have been acknowledged.
4.Error Control and Prevention:
Clear instructions and validation checks guide users, reducing errors and offering solutions when issues arise.
5.Visual Appeal:
A modern color scheme, clean typography, and consistent design elements create an engaging and professional look.
6.Mobile Responsiveness:
The app adapts to various devices and screen sizes, ensuring a seamless experience across platforms.
7.Clarity of Navigation:
Logical organization, clear labels, and tools like search bars and breadcrumbs make navigation straightforward.
8.Readability:
Text and elements are designed with proper hierarchy, spacing, and contrast, ensuring information is easy to read.
9.User Command:
Options like undo, redo, and customization give users control and flexibility over their actions.

4.3 Components Available

For the GUI, we utilized **Canva** during the initial design phase to create layouts and visual elements. In the next phase, we will transition to **Figma** for more robust and interactive design implementations.

4.4 UIDS Description

No specific User Interface Development System (UIDS) has been implemented for this phase of the project. The focus remains on conceptual and graphical design.

5.0 Restrictions, Limitations, and Constraints

Special design issues that impact the design or implementation of the blockchain property transaction software:

- The app must process transactions quickly and efficiently to meet user expectations.
- The software should have fully functional versions for web, iOS, and Android platforms.
- Changes or outages in blockchain networks or APIs may affect the application's functionality.
- Budget constraints can affect development speed and feature implementation.
- Ensuring robust security for user payment data and transaction details is essential.
- Developing and maintaining secure and efficient smart contracts can be technically challenging.
- Integrating with government and legal databases to verify ownership can be complex and subject to regulation.
- Adhering to various data privacy laws (e.g., GDPR) when handling user information.
- Implementing effective measures to detect and prevent fraudulent transactions.
- Building a user base that trusts the blockchain platform and understands how to use it effectively.
- Finding skilled developers with expertise in blockchain technology may be difficult.
- Blockchain network fees may fluctuate, impacting the cost of transactions for users.
- Coordinating with banks and financial entities for seamless payment processing and loan management.
- Ensuring a reliable system for resolving disputes without central authority.
- Ensuring that users have access to the necessary technology and internet bandwidth to use the app effectively.

6.0 Validation Criteria

This section outlines the testing techniques, expected software responses, performance boundaries, and identification of critical components to ensure the system's reliability, functionality, and security.

6.1 Classes Of Tests

The test cases are designed using the black-box testing method, incorporating its techniques such as boundary value analysis, equivalence partitioning, and decision table testing. It is important to note that white-box testing was not applied, as it requires access to the complete code, which is not yet available.

6.2 Expected Software Response

Name of Technique for BB Testing	Restrictions	User	Result	Valid Cases	Invalid Cases
Boundary Value Analysis	Password must be between 8 to 12 characters	Landlord / Customer / Authorized Person	Allows password of valid length	8: "Landlord123" 9: "Cust0m3r!" 11: "Authorized1"	7: "Land12" 13: "ExceedinglyLongPass"
Equivalence Partitioning	ID must be between 6 to 9 characters	Customer	Allows only IDs with valid length	7: "user123" 8: "cust1221"	5: "user1" 11: "customer9875"
Boundary Value Analysis	Customer cannot access Landlord's section	Customer	Allows access only to Customer-restricted areas	Customer views own details	Customer tries accessing Landlord data
Boundary Value Analysis	Authorized Person cannot access Landlord's data	Authorized Person	Authorized access limited to their domain	Authorized Person oversees relevant info	Authorized Person modifies Landlord data
Equivalence Partitioning	Landlord can only modify their own properties	Landlord	Landlord can edit only their property	Landlord edits own property	Landlord tries editing another Landlord's property
Boundary Value Analysis	Customer cannot access Authorized Person's data	Customer	Access restricted to Customer areas	Customer views own account info	Customer attempts access to Authorized Person data
Boundary Value Analysis	Landlord cannot access Customer's section	Landlord	Access restricted to Landlord domain	Landlord views only their own account	Landlord tries accessing Customer data

Decision Table Testing:

Condition	Rule 1	Rule 2	Rule3
Landlord	1	0	0
Customer	0	1	0
Authorized Person	0	0	1

Action	Landlord	Customer	Authorized Person
Access to Landlord account	Valid access	No access	No access
Access to Customer account	No access	Valid access	No access
Access to Authorized Person tasks	No access	No access	Valid access
Property listing modification	Yes	Yes	Yes
Approval for critical changes	Yes	No	No
Updating records in the system	Yes	Yes	Yes

6.3 Performance Bounds

1. **Transaction Processing Speed**: The application must handle transactions efficiently, ensuring that property transactions are completed within a specified time frame. Delays in transaction verification could negatively impact user experience and trust. The performance bound should ensure that each transaction is verified within a few seconds.
2. **Data Integrity Bound**: To maintain data accuracy, the system must prevent data corruption and unauthorized alterations. Any modifications to transaction data should be validated against security protocols to ensure accuracy and consistency across all transactions.
3. **Scalability Bound**: The application should support an increasing number of users and transactions without a decrease in performance. As the number of users and transactions grows, the system must remain responsive and stable.