Software Design Specification

for

Welcome Home

Smart Escrow Contract

Version 1.0

Prepared by:

Casey Munga and Sergio Prieto

Smart Escrow Management

April 26th 2022

Table of Contents

Revision History

Name	Date	Reason For Changes	Version
Casey Munga	04-26-2022	Initial Document	V1

1. Introduction

1.1 Purpose

This is a Smart Contract for rental escrows using block chain technology. The smart contract solution will seek to reduce the need of trusted mediators, arbitration and enforcement costs when transacting rental property deposits, mitigating the risk of fraud and financial losses, as well as the reduction of malicious and accidental exceptions.

1.2 Document Conventions

None

1.3 Intended Audience and Reading Suggestions

This document's target audience is for the angel investor who are more technically including; the development team and stakeholders.

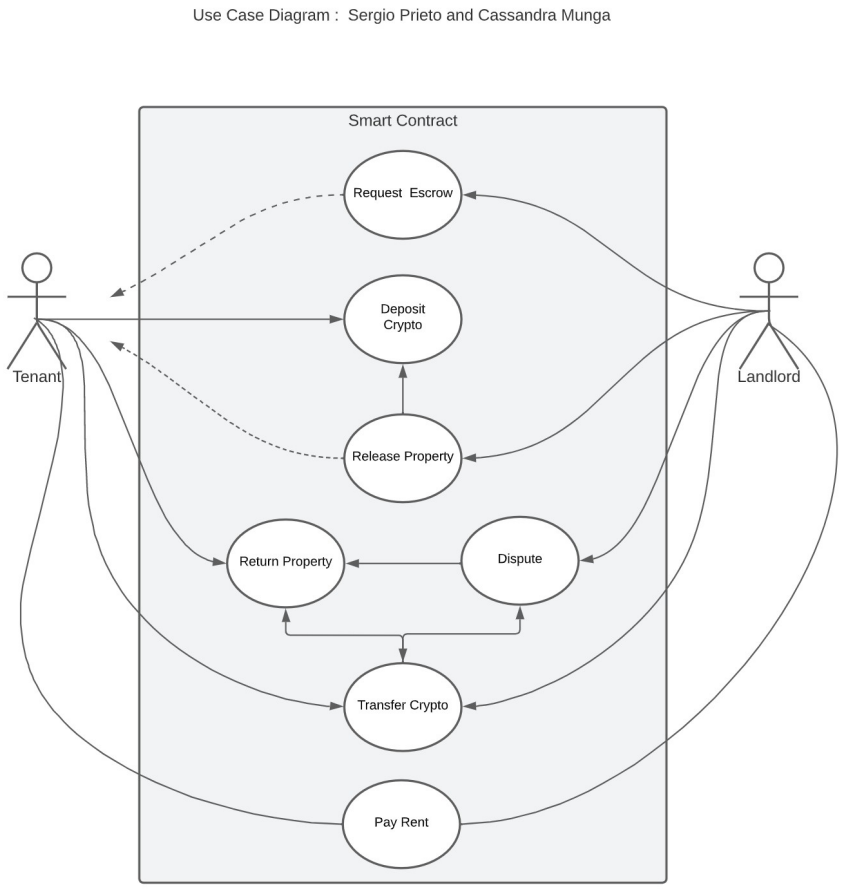
1.4 Product Scope

1.4.1 Purpose

When renting a property, the landlord and tenant are involved in a contract that requires a transaction in the form of a deposit. This money, in most cases, cannot be monitored by the tenant after it is given to the landlord.
In addition, the Deposit deductions are usually done arbitrarily by one the parties, without an efficient or cost effective means to dispute. By using an escrow smart contract, the risk of fraud can be reduced by ensuring proper escrow logic execution, along with the transparency of having the transaction accessible to all participants through an open distributed ledger or block-chain.

1.4.2 High Level Flow

This product is a new product that is designed to be scale-able web driven transactional app.



1.5 References

*A **gateway** to the rental payment portal is listed below.*

1.Welcome Home Rental Payment Portal Smart Escrow Contract

Use Cases:

2. Project Overview
3. Login
4. Initiate Contract
5. Create Contract
6.Create and Fill Wallet
7.Deploy Contract on Block Chain

2. Overall Description

2.1 Product Perspective

This product is a new product that is designed to be scale-able web driven transactional app .

2.2 Product Functions

Major features

- *Smart contract creation*
- *Crypto wallet creation*
- *Users views*
- *Accounts*
- *Dashboards*
- *Contracts*
- *Transaction*
- *Block Chain Deployment*
- *Arbitration module (Phase II)*

2.3 User Classes and Characteristics

The intended target market will be a class of real estate holders known as landlord and renters or tenants.

2.4 Operating Environment

The Software will be hosted within the cloud with interactions with external entities namely financial institutions and Block-chain and MetaMask. Software environment will be a front end Web app that can accessed on a mobile device, tablet and laptop independent of any specific O/S.

2.5 Design and Implementation Constraints

At this time developers will be limited to the web and software as a service. As the needs demands a downloadable app for different OS may be developed. However that sits outside the present scope. MongoDB or AWS will be the DB system. HTTP and latest security protocols must be adhered to. Block-chain and cryptocurrency will the official trading currency. HTML5 for client , php for server response. Simple and unfettered interface. Site must be responsive.

2.6 User Documentation

Search and user help will be available inside the app. In app chat will be available inside the app during phase 2 but out of scope in this iteration. Use cases are listed in the document SmartContract-SRS-Munga_Prieto.pdf

2.7 Assumptions and Dependencies

Selenium programming language, Truffle and block chain API s will be the main development and NoSQL., AWS Cloud

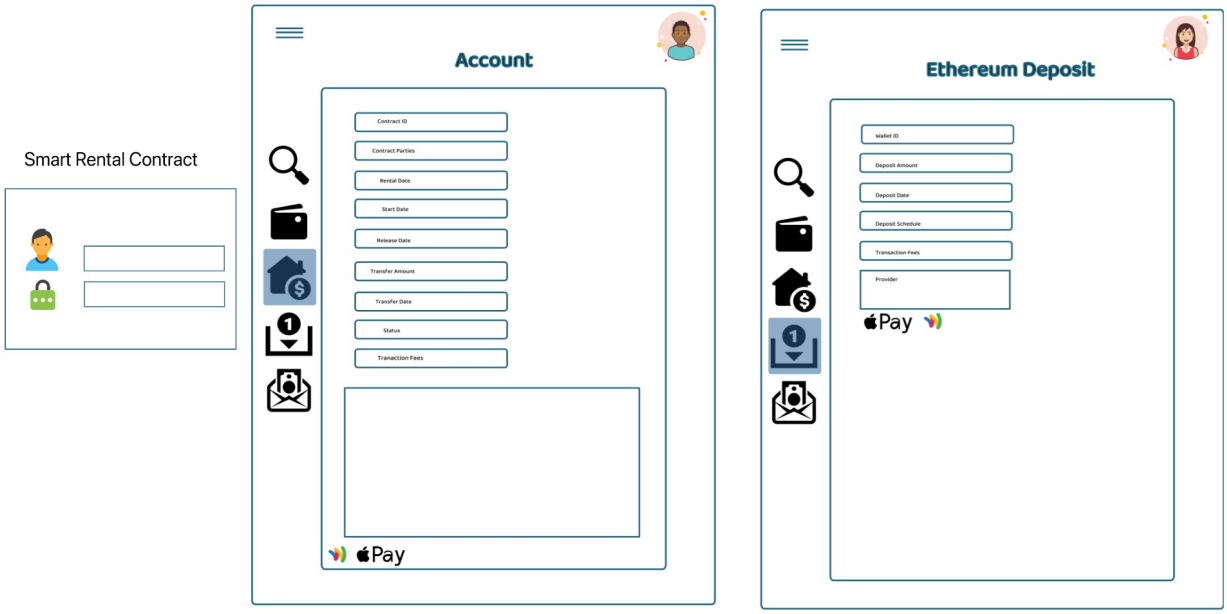
27 lines (27 sloc) | 1007 Bytes

```
[
  '{{repeat(5, 7)}}',
  {
     name: '{{firstName()}} {{surname()}}',
    _clientId: '{{objectId()}}',
    company: '{{company().toUpperCase()}}',
    email: '{{email()}}',
        password:'{{lorem(1, "words")}}',
    index:'{{index()}}',
    guid: '{{guid()}}',
    isActive: '{{bool()}}',
    contracts:[
      '{{repeat(3)}}',
      {
         guid: '{{guid()}}',
         rentalDate: '{{date(new Date(2014, 0, 1), new Date(), "YYYY-MM-ddThh:mm:ss Z")}}',
         startDate: '{{date(new Date(2014, 0, 1), new Date(), "YYYY-MM-ddThh:mm:ss Z")}}',
         releaseDate: '{{date(new Date(2014, 0, 1), new Date(), "YYYY-MM-ddThh:mm:ss Z")}}',
         balance: '{{floating(1000, 4000, 2, "$0,0.00")}}',
         transferDate: '{{date(new Date(2014, 0, 1), new Date(), "YYYY-MM-ddThh:mm:ss Z")}}',
         transactionFees: '{{floating(10, 100, 2, "$0,0.00")}}',
         days: '{{integer(1, 1000)}}',
        status: '{{random("paid","deployed","refunded")}}'
      }
    ]
  }
]
```

3. External Interface Requirements

3.1 User Interfaces Overview

GUI Model : Sergio Prieto and Cassandra Munga

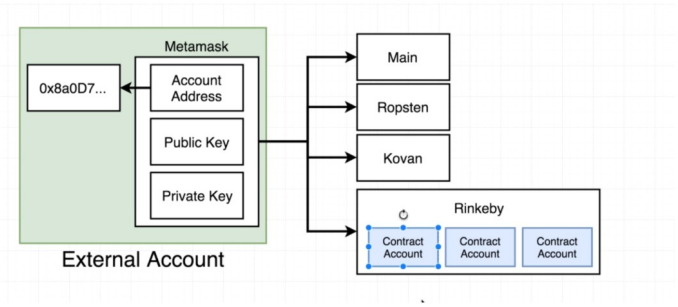


3.2 Hardware Interfaces

<Describe the logical and physical characteristics of each interface between the software product and the hardware components of the system. This may include the supported device types, the nature of the data and control interactions between the software and the hardware, and communication protocols to be used.

3.3 Software Interfaces

Contract Account	
Field	Description
balance	Amount of ether this account owns
storage	Data storage for this contract
code	Raw machine code for this contract



4. Design Considerations

4.1 Assumptions and Dependencies

The following assumptions are

- *smart and web capable device*
- *Will be cross platform tested*
- *End User do not have to have technical experience as the app will be simple*
- *Chat enabled in the future*

4.2 General Constraints

There are no major general constrains with the exception of a smart device such as a tablet, phone, or computer with the ability to receive network signals. The device must be able to access the web.

4.3 Development Methods

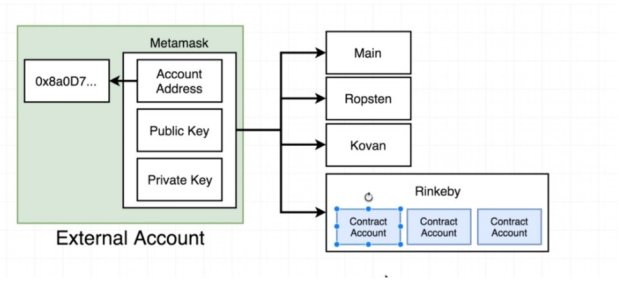
The code development will be Selenium, Web 3, MetaMask used to develop the block chain technology HTML5, JavaScript, Node.Js for the UI/UX, Java and object oriented coding will be used on the Server Side. The database will be NOSQL as MongoDB and Cassandra. Apps will be hosted and run in the cloud.

5. System Architecture

5.1 Architectural Strategies

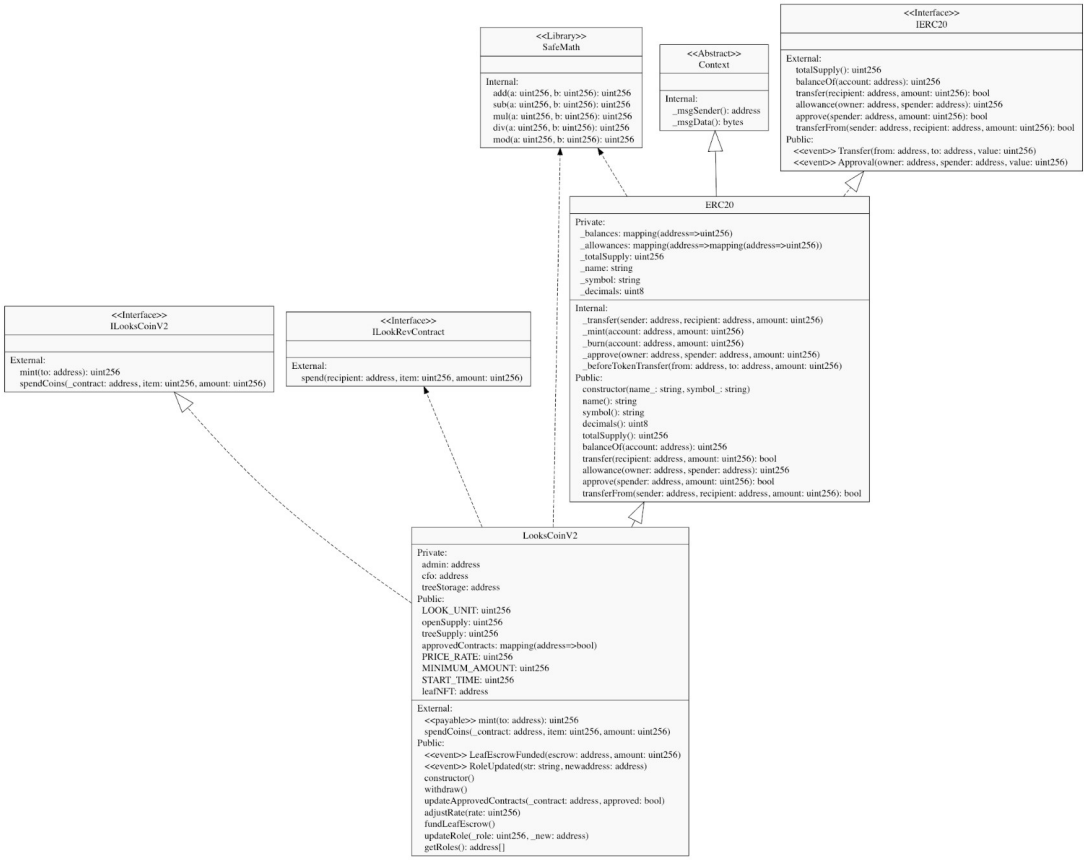
Block Chain will be our primary architecture. Over AWS Server

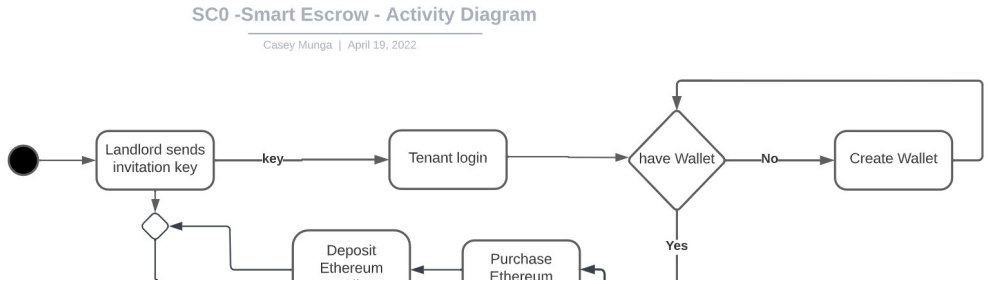
Contract Account	
Field	Description
balance	Amount of ether this account owns
storage	Data storage for this contract
code	Raw machine code for this contract



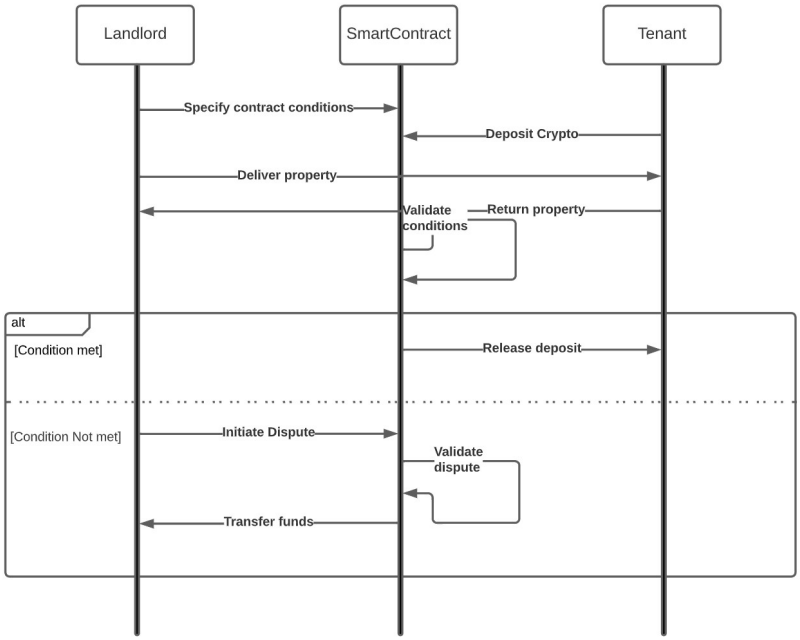
5.2 High level Overview of System Architecture

diagram showing the major subsystems and data repositories and their interconnections.





Sequence Diagram : Sergio Prieto and Cassandra Munga

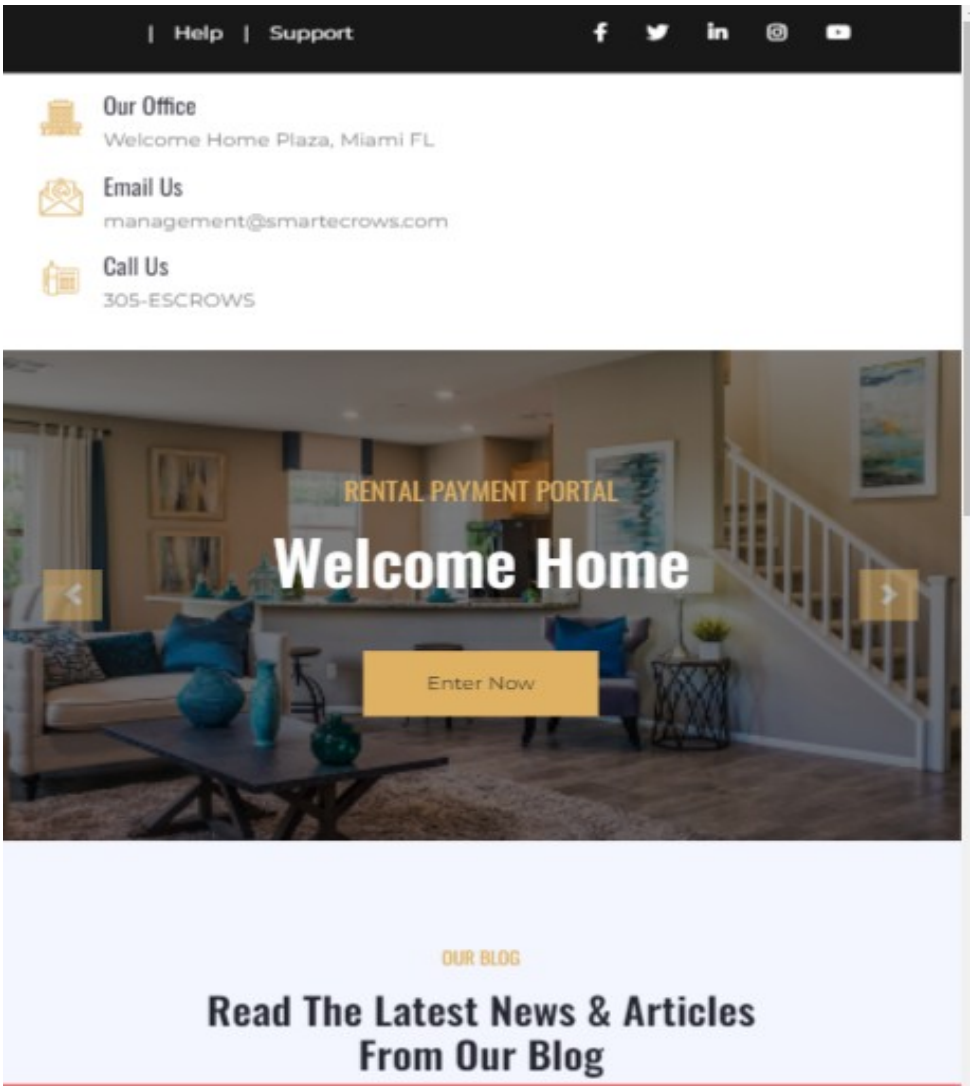


6. Human Interface Design

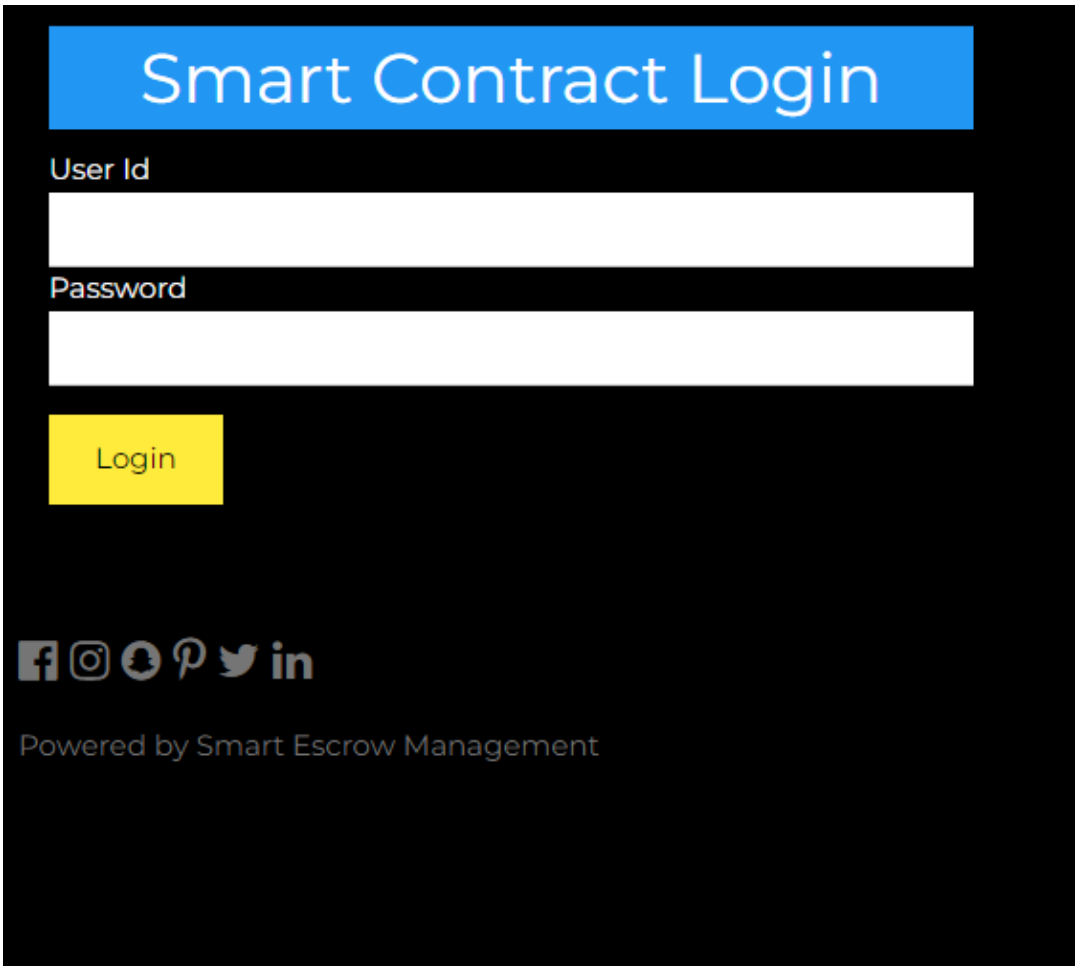
6.1 Screen Images

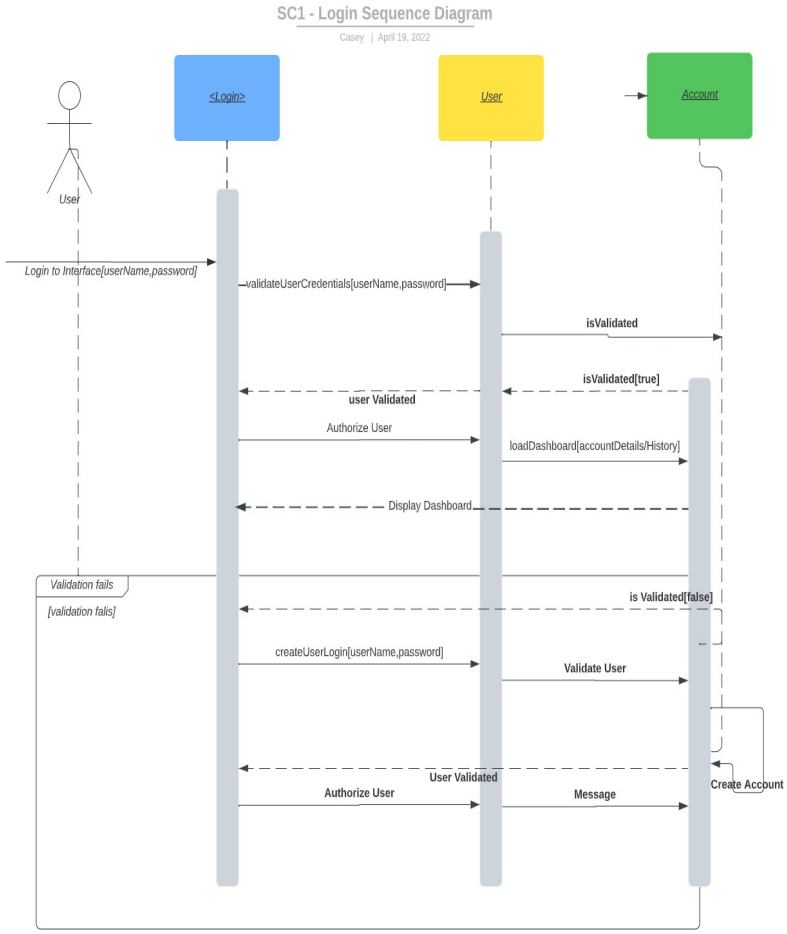
6.1.0 Payment Portal Gateway

Welcome Home Rental Portal Gateway



6.1.1 Login

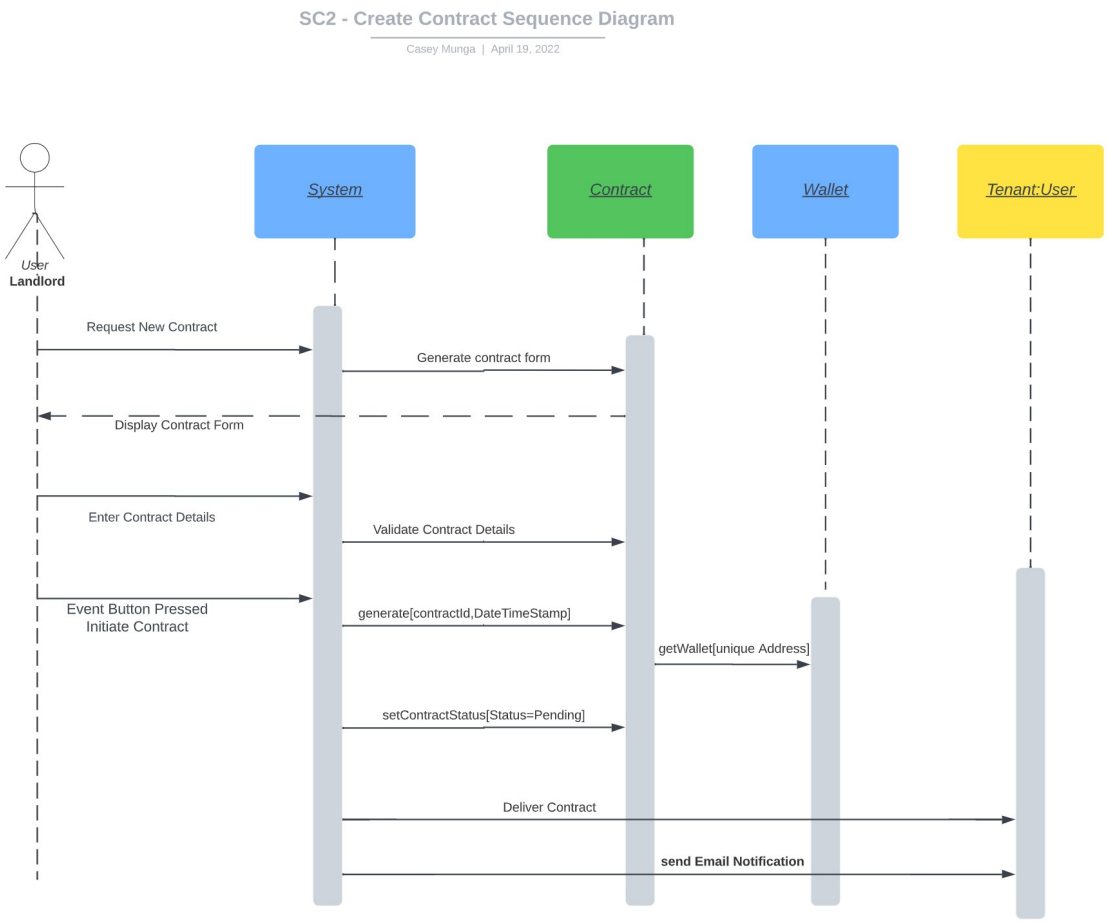

Smart Contract Login
User Id
Password
Login
Powered by Smart Escrow Management



[Login -full-size-image]

6.1.2 Initiate Contract

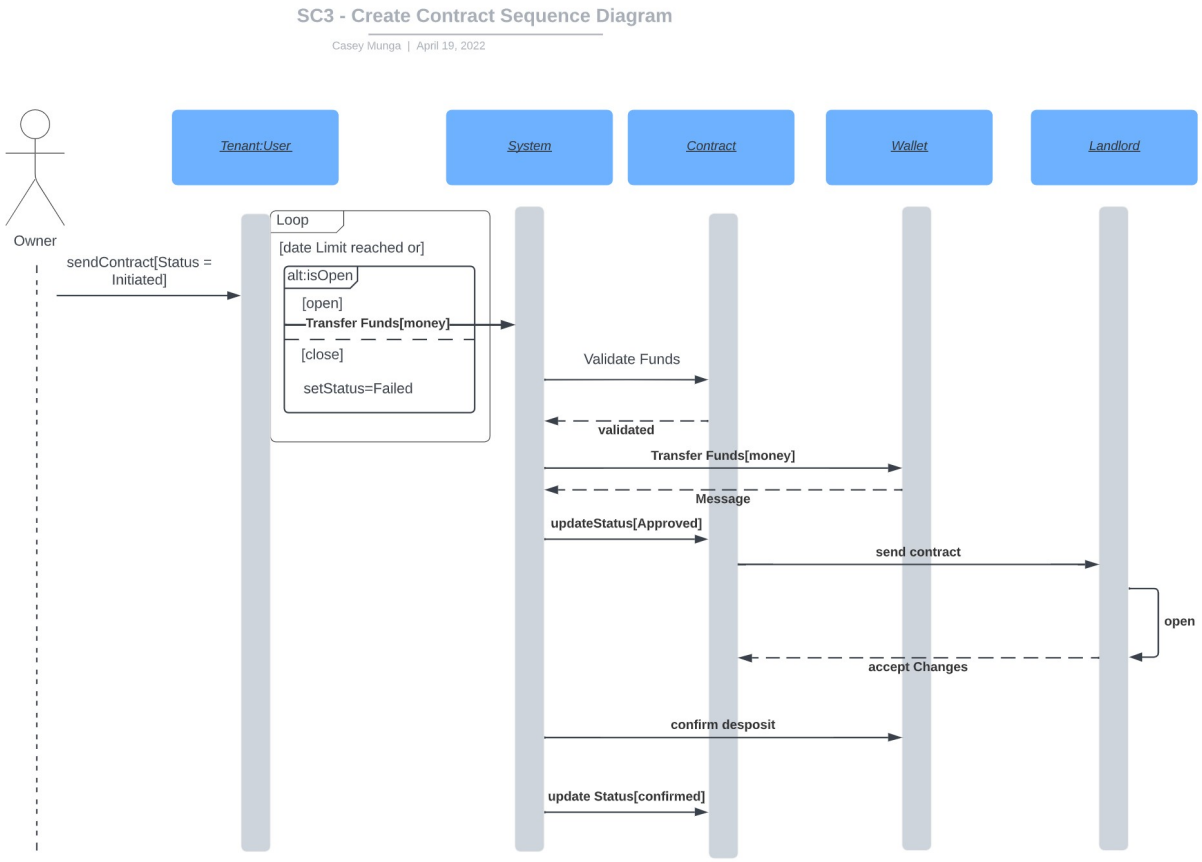
This form page is only available to the Owner/Landlord. The landlord will enter the information and select the initiate Contract button. A notification email will be sent to the tenant.



[Initiate Contract full-size-image]

6.1.3 Create Contracts

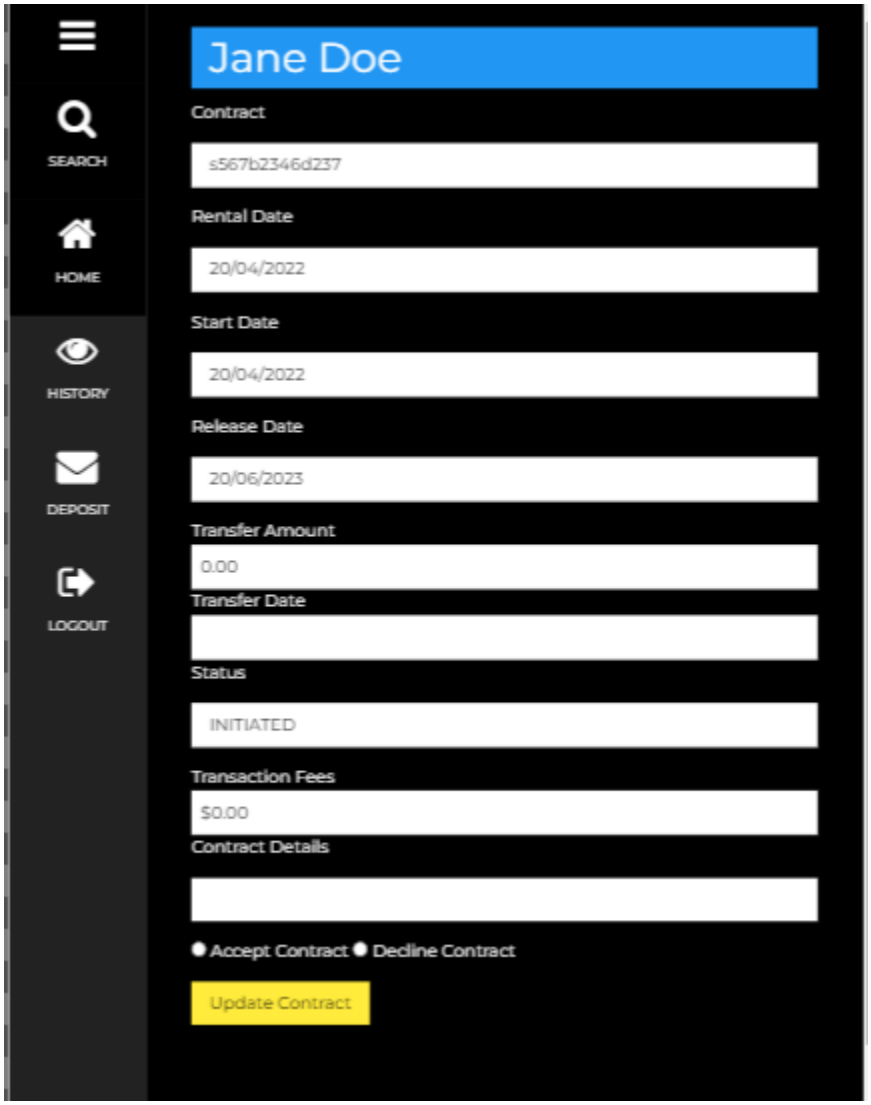
Owner receives an approved contract and will enter the transaction fees. The button changes to Confirm Contract.



[Create Contract-full-size-image]

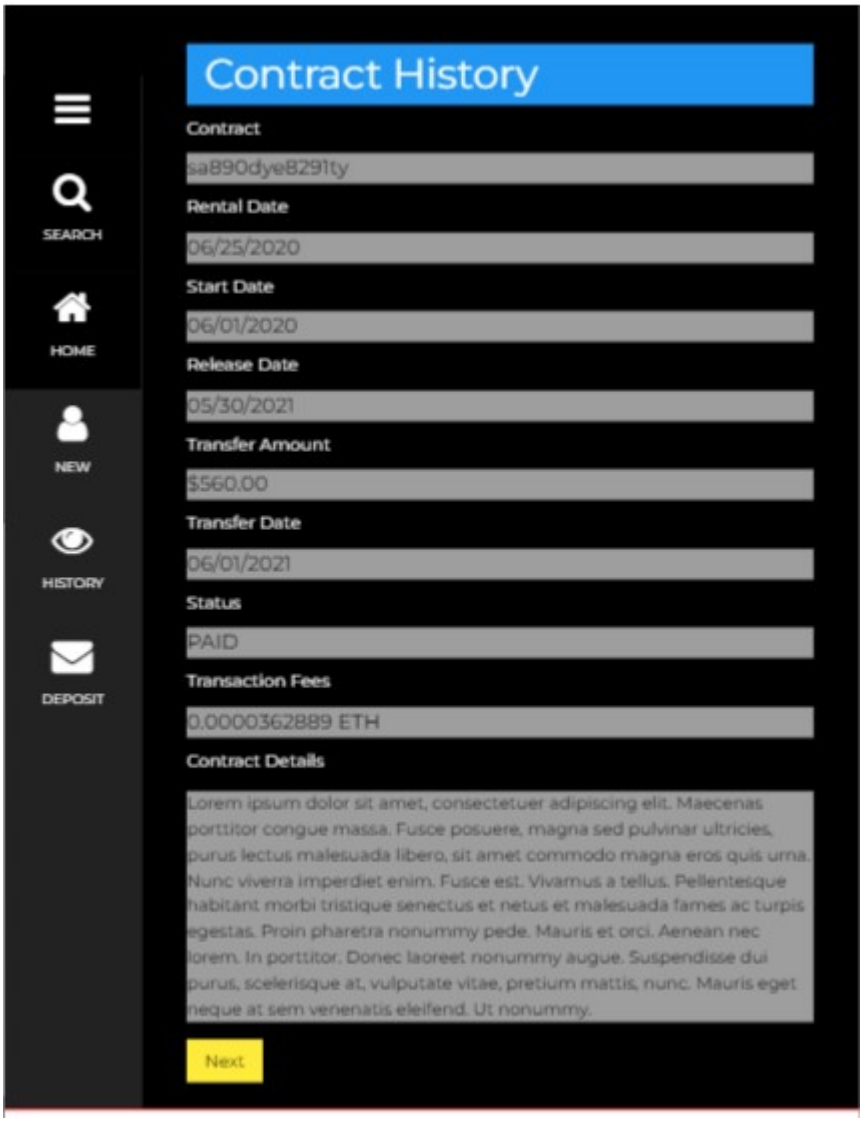
6.1.4 Account

View when tenant log on to account. Navigation Buttons will be present. Tenant is presented with view of any new Initiated contract or the current contract. See use case 1.5 Tenant selects Accept contract and is presented with deposit page to deposit money into wallet



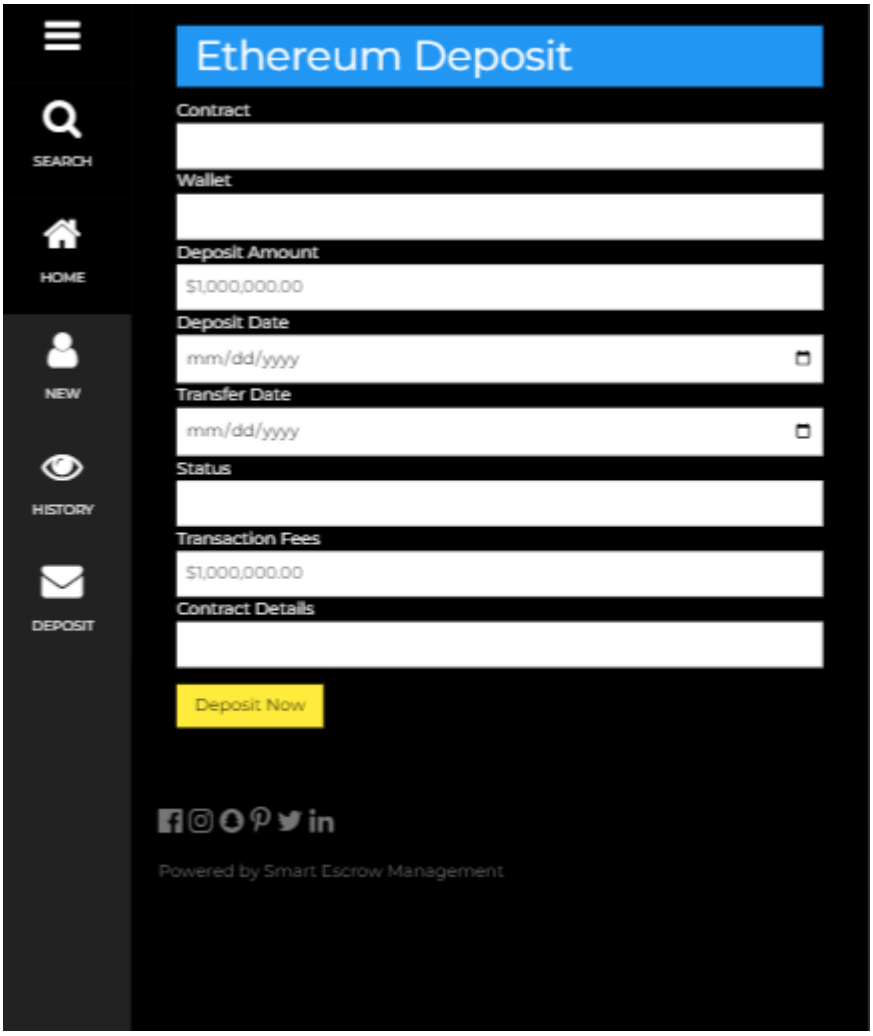
6.1.5 History

User is shown most recent contract transaction but can opt to see a list if all historical data of every transaction



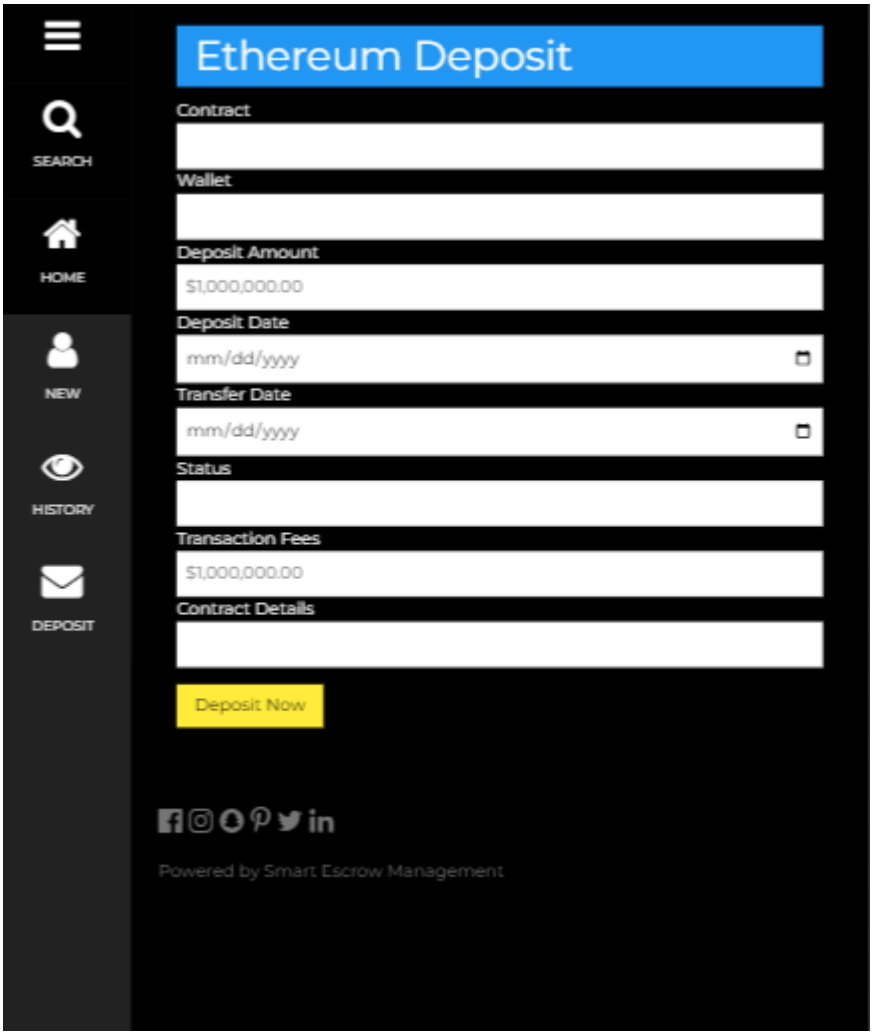
6.1.6 Deposit

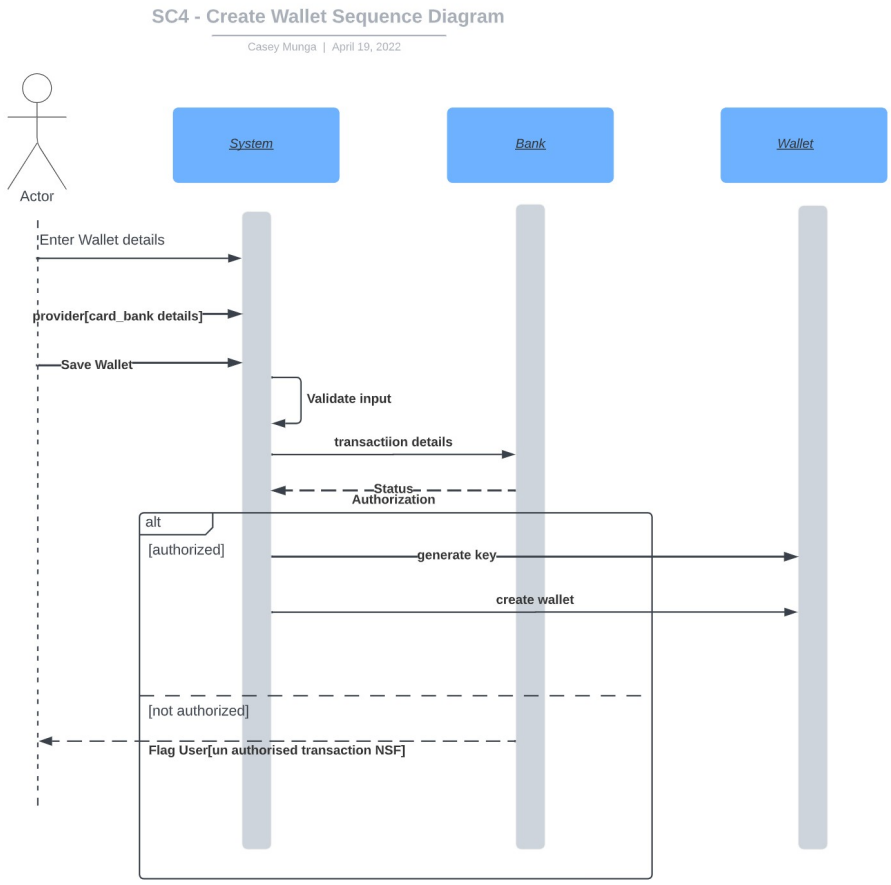
This page is served when the user wants to deposit money from a banking institution into his wallet and purchase ethereum.



6.1.7 Create Wallet

All currency must be kept in a wallet. New users must create a wallet and assign an external entity such as a credit card, bank account, paypal and must transfer / deposit money into their account at this stage. The wallet is a transactional component and will be used to uniquely identify the account holder. An account can only have one wallet, but a wallet can be used in more than one account. This form will be repurposed programmatically to create wallet when served up

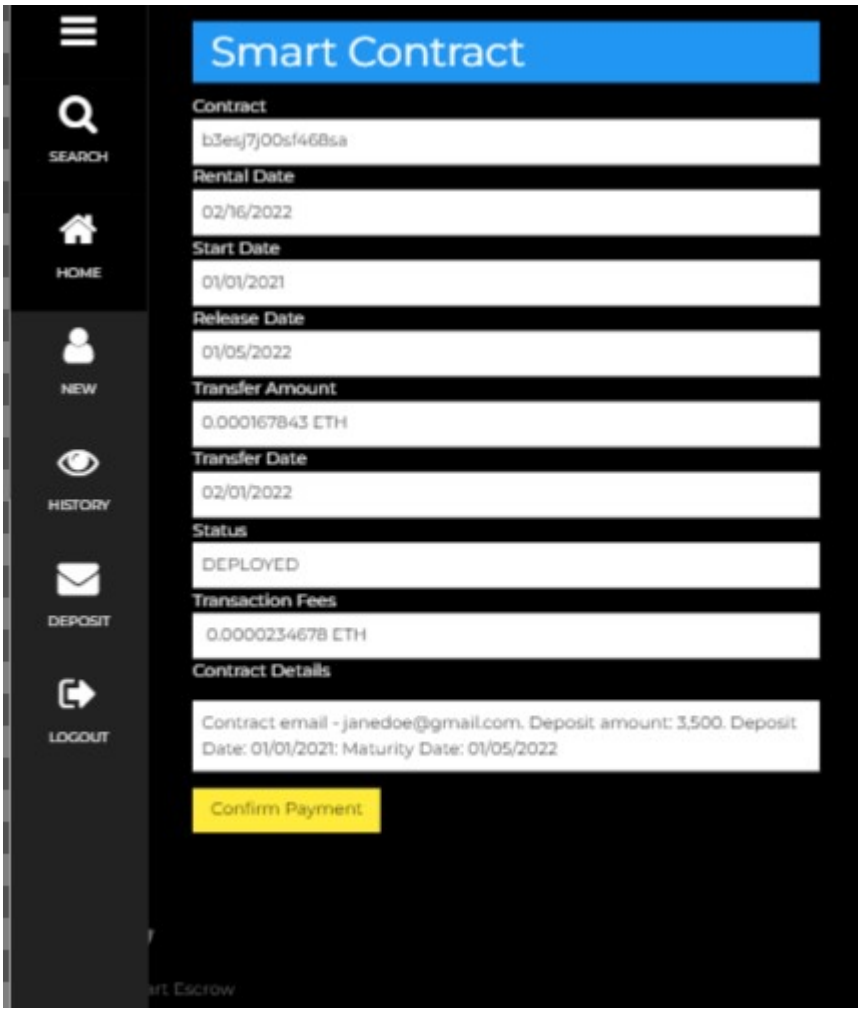




[Create Wallet full-size-image]

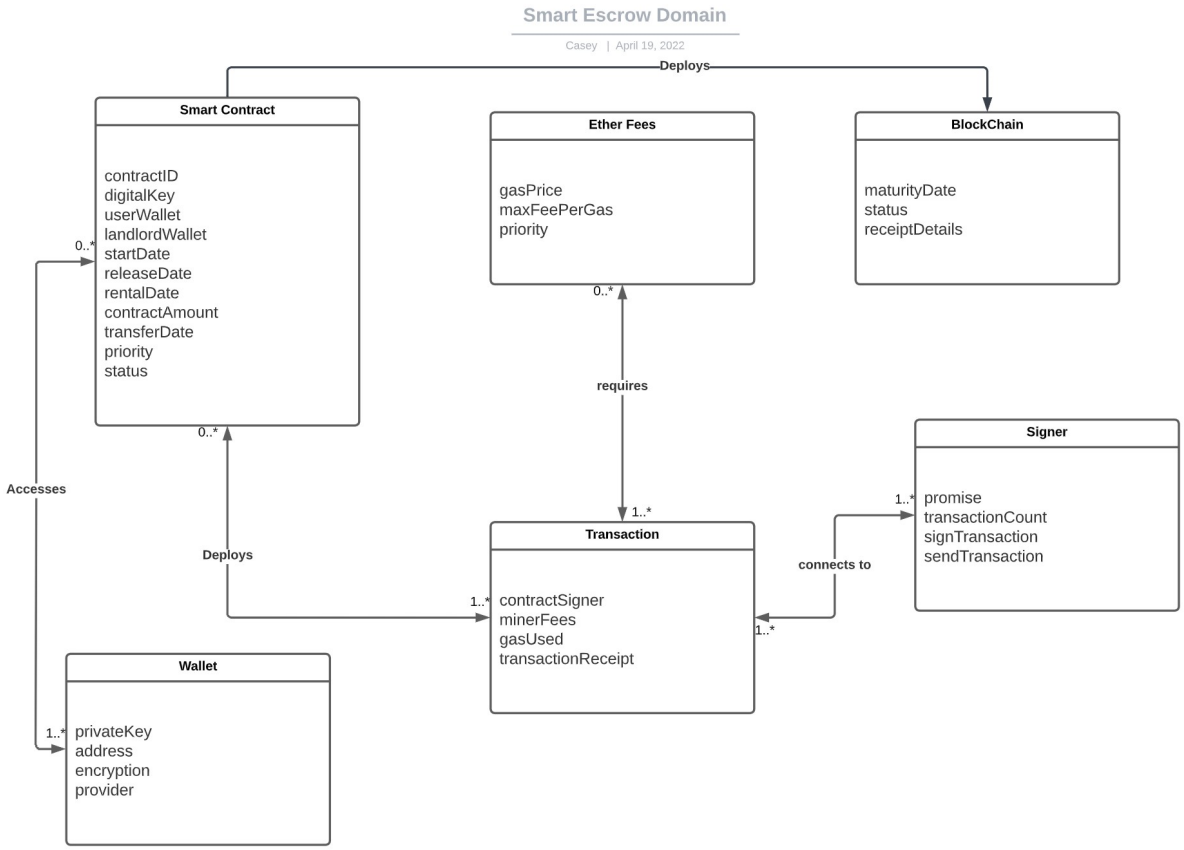
6.1.8 Deploy Contract

This page is available after the contract had been confirmed and all fees paid. The button's text property will be changed to Deployed. The owner can now deploy the account to the block chain. Please see use case section 1.5

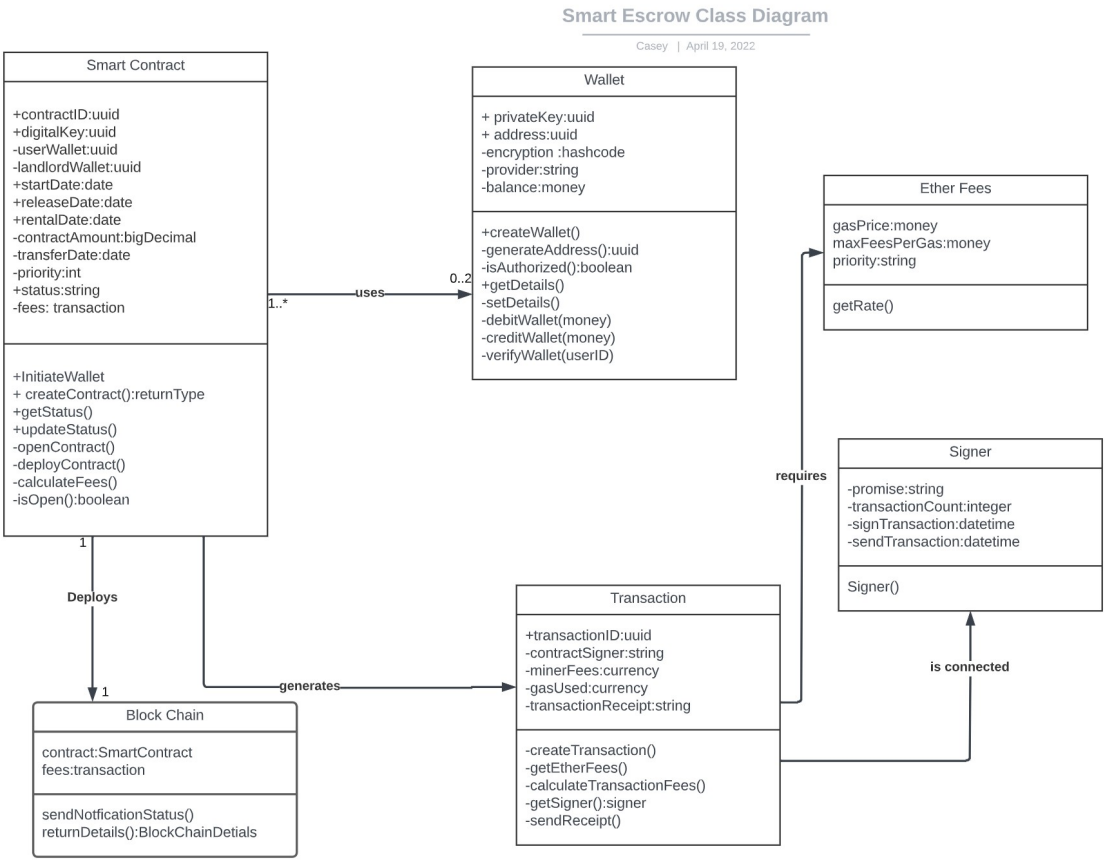


7. Detailed System Design

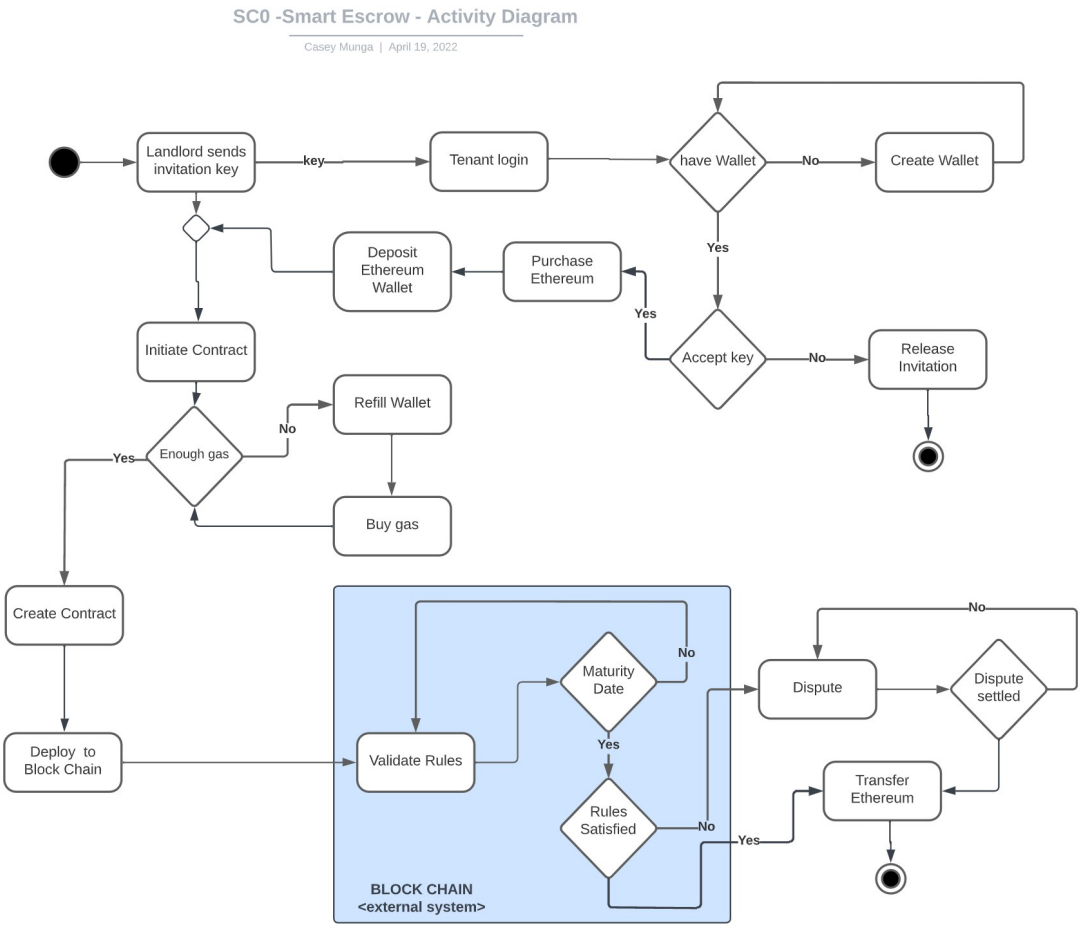
7.1.1 Smart Escrow Domain Diagram



7.1.2 Smart Escrow Class Diagram

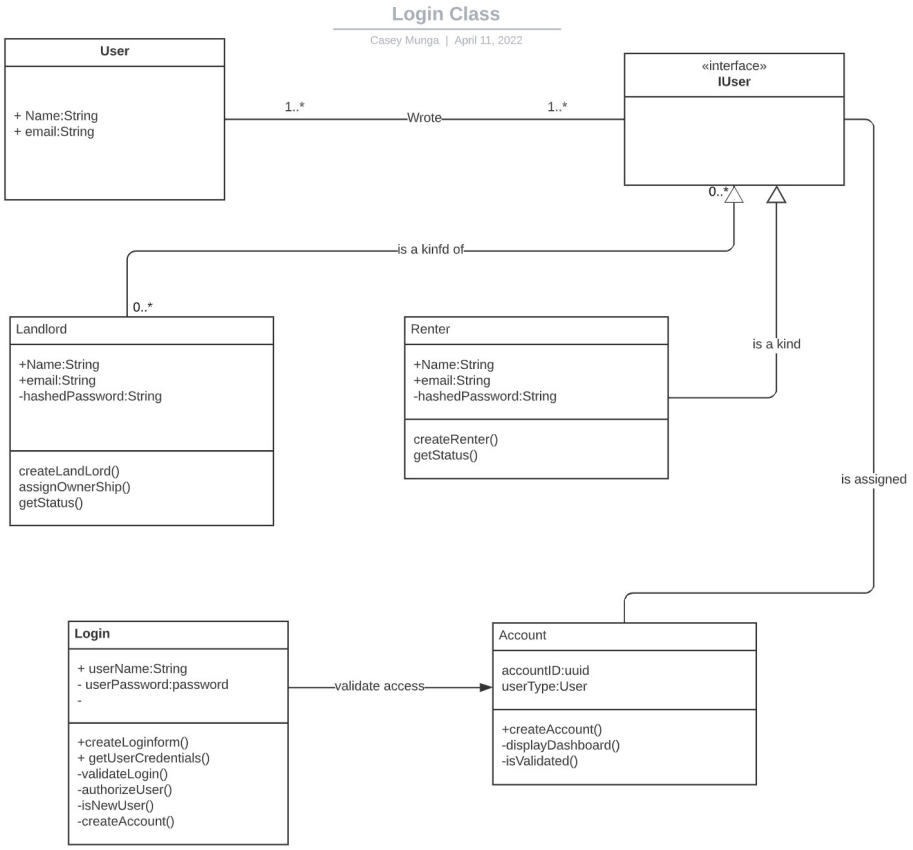


7.1.3 High Level View Of System

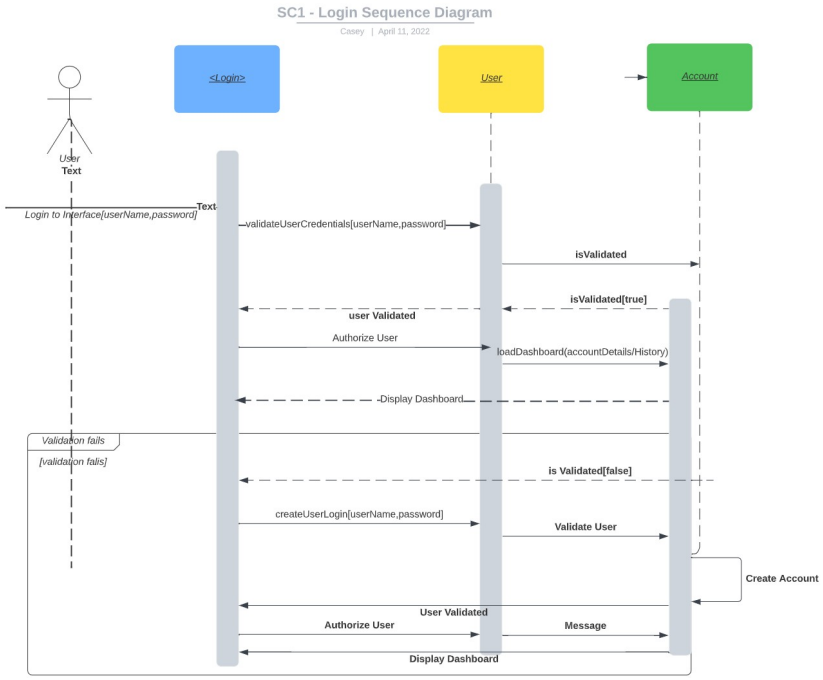


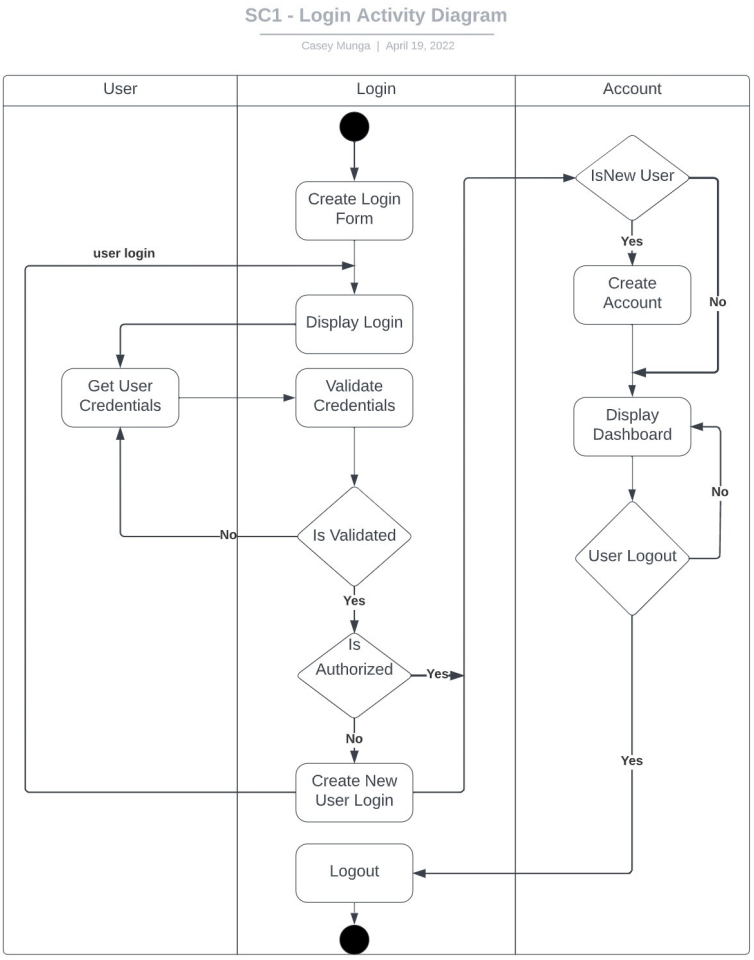
7.2 Login

7.2.1 Class Diagram



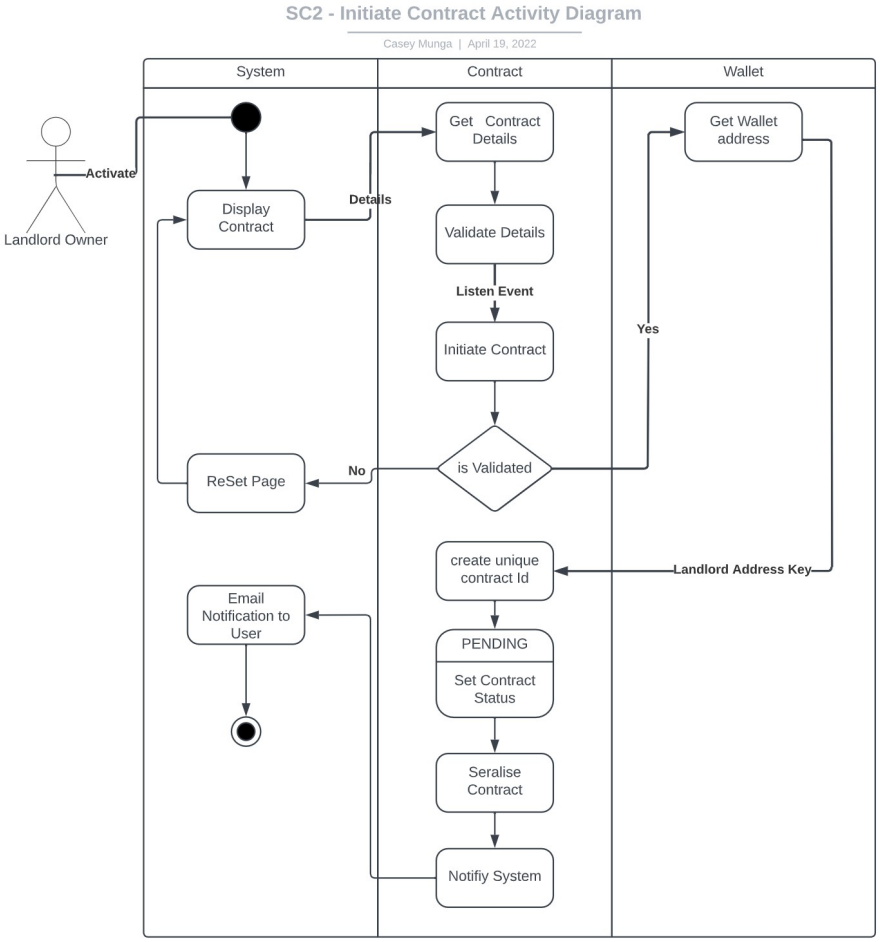
7.2.2 Activity Diagram



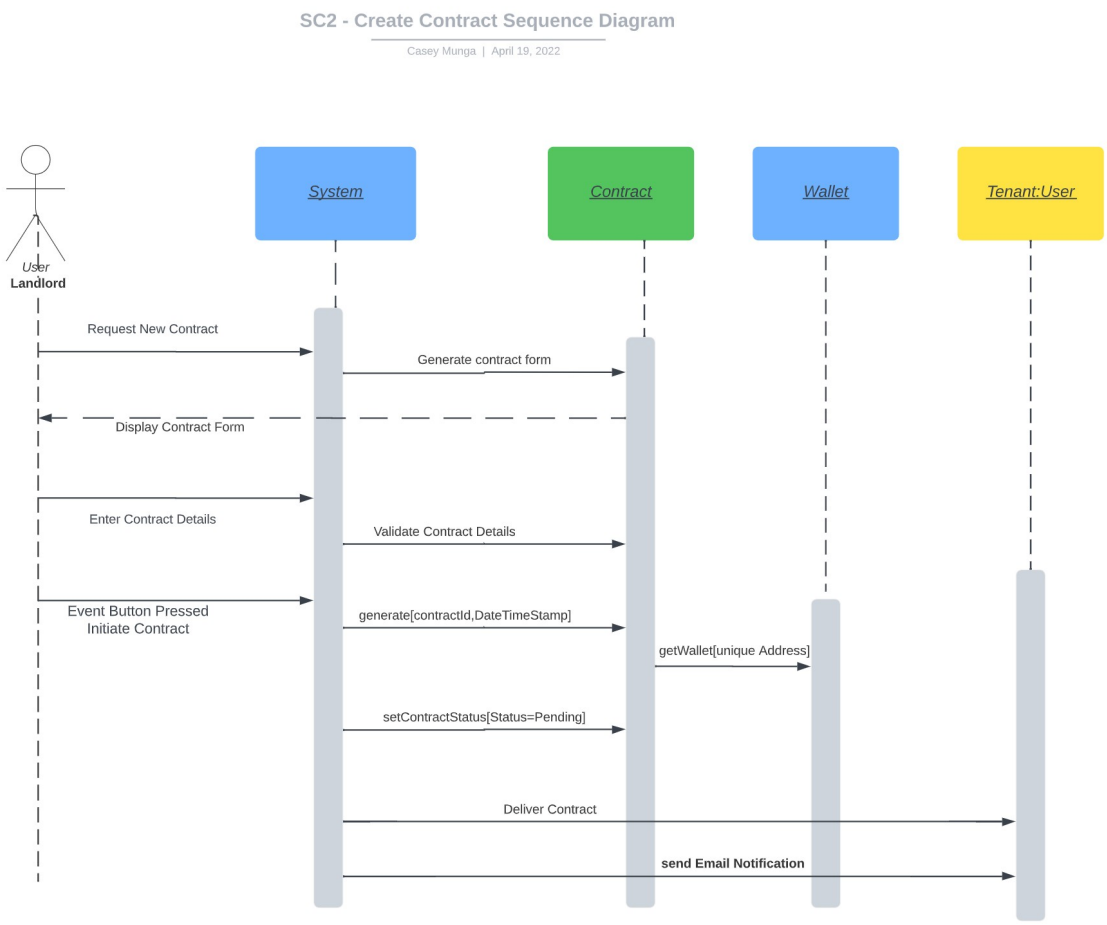

SC1 - Login Activity Diagram
Casey Munga | April 19, 2022
User
Login
Account
Create Login Form
user login
Display Login
Get User Credentials
Validate Credentials
Is Validated
No
Yes
Is Authorized
Yes
No
Create New User Login
Logout
IsNew User
Yes
Create Account
No
Display Dashboard
No
User Logout
Yes

7.3 Initiate Contract

7.3.1 Activity Diagram

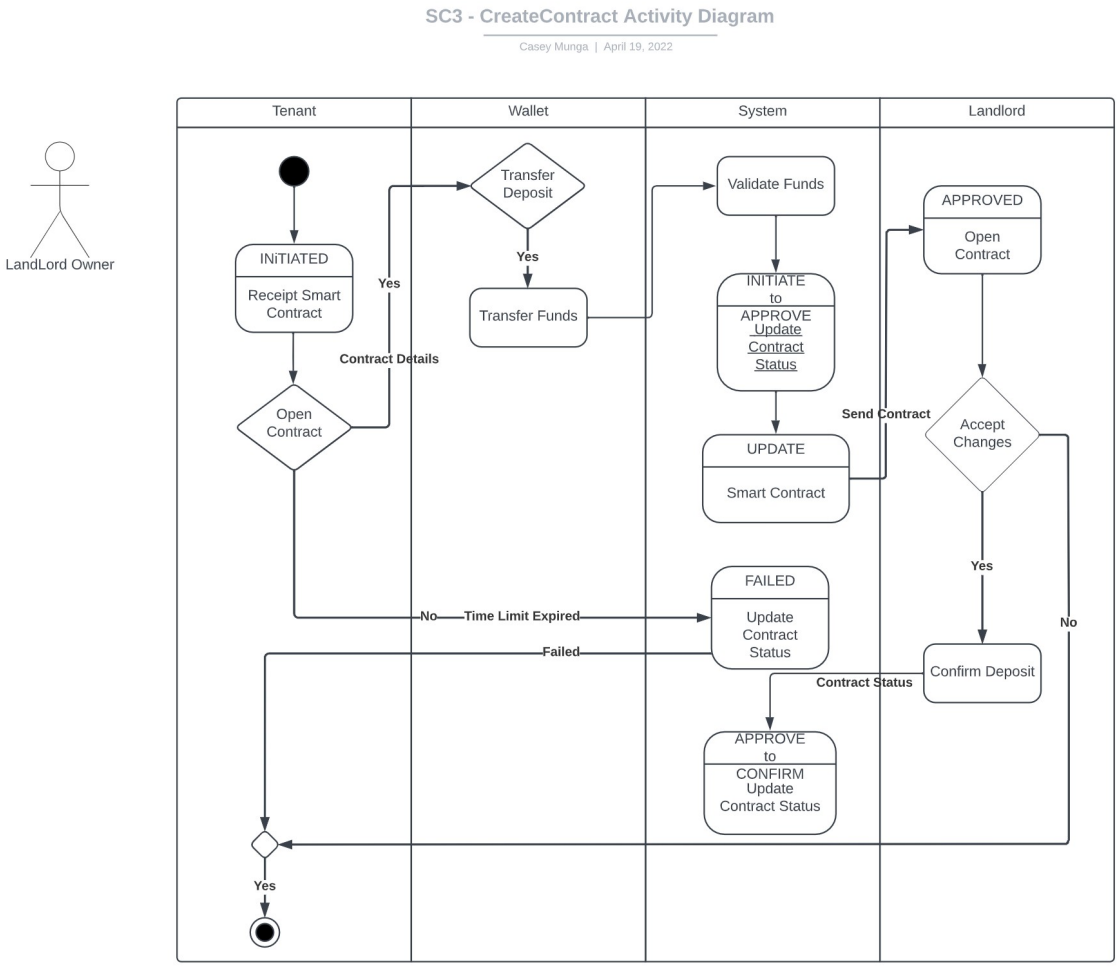


7.3.2 Sequence Diagram

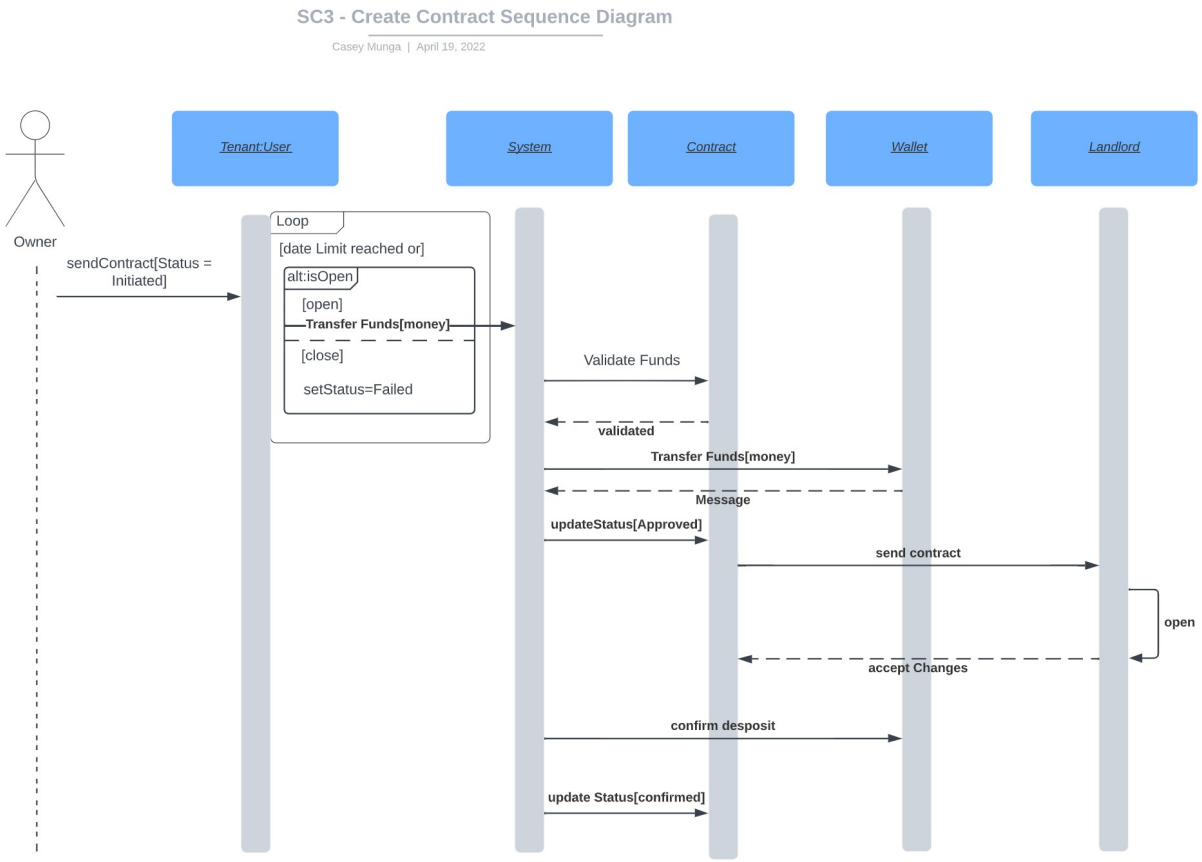


7.4 Create Contract

7.4.1 Activity Diagram

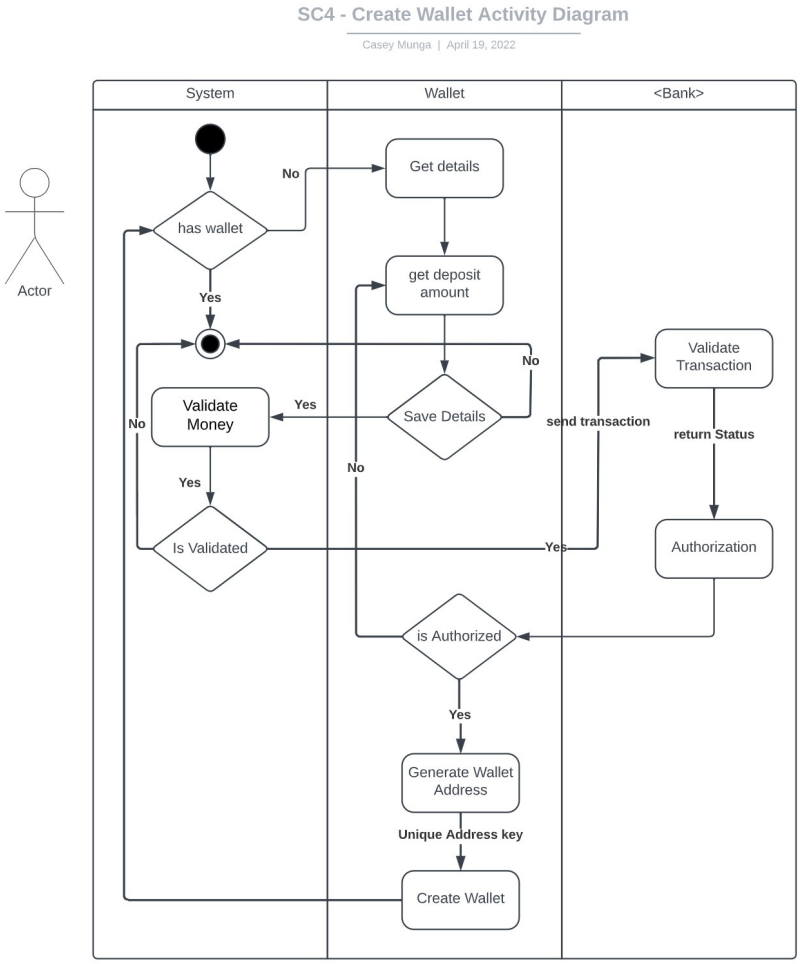


7.4.2 Sequence Diagram

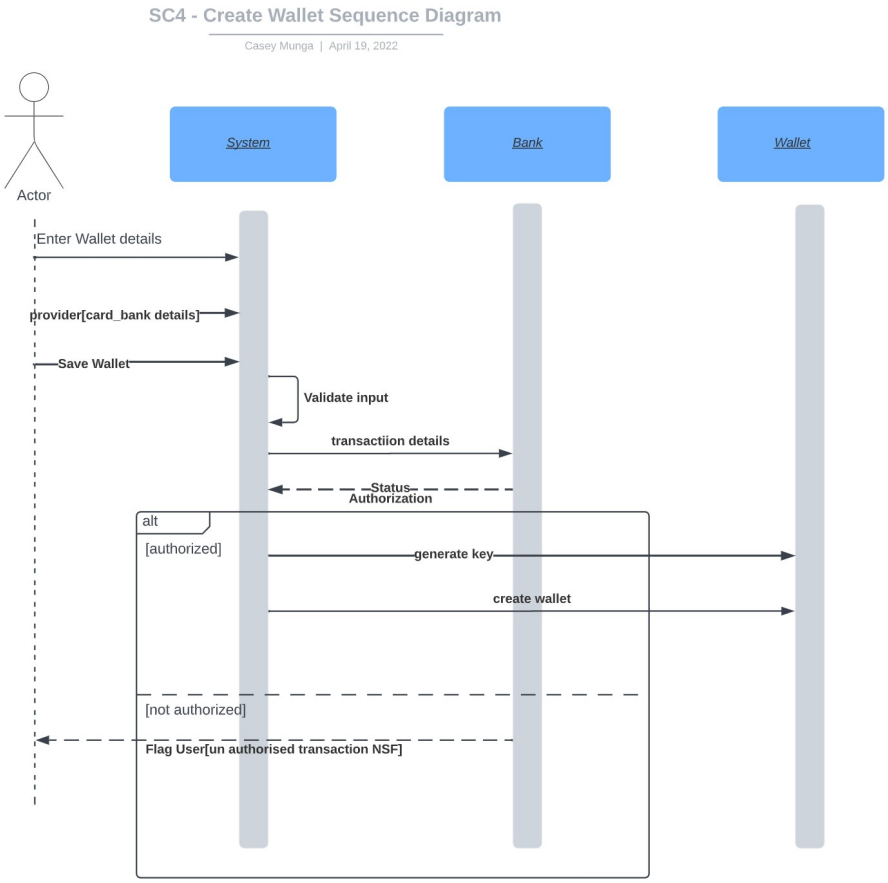


7.5 Create Wallet

7.5.1 Activity Diagram



7.5.2 Sequence Diagram



7.6 Deploy Contract

7.6.1 Activity Diagram