BSCS FINAL PROJECT

CRYPact – Smart Contracts

Advisor: Kamran Shabbir

Presented by:
Group ID: S18BS006

L1F14BSCS0405	Muhammad Samee Hassan
L1S15BSCS0024	Muhammad Talha
L1F14BSCS0530	Arif Mehmood Dar

Faculty of Information Technology

University of Central Punjab

CRYPact – Smart Contracts

By

Muhammad Samee Hassan
Muhammad Talha
Arif Mehmood Dar

Project submitted to

Faculty of Information Technology,

University of Central Punjab,

Lahore, Pakistan.

in partial fulfillment of the requirements for the degree of

BACHELOR OF SCIENCE
IN
COMPUTER SCIENCE

Project Advisor	Manager Projects

Abstract

In overall history, real estate has undoubtedly served as the most stable and valuable store. Real Estate by far is one of the most popular aspect in everyone's life. While so important for many of the people, the process of buying and selling and transferring the ownership of the land/property has never been so easy. This slow and hassle process can be eliminated using our system CRYPact – Smart Contracts. The core of the problem is the lack of automated system, the need of 3rd party (like real estate agents), and the most important, a central database for buying and selling properties which can be modified any time to make false records. That is why CRYPACT's vision is to radically **transform** the ecosystem of real estate (in this case we specifically designed for LDA, Lahore, Pakistan), by making transactions more: Transparent, secure and reliable. Resulting not only in improving the overall system, but more importantly helping to **protect** communities from real estate: Fraud, corruption and waste of time. So with the app like "CRYPact – Smart Contracts" being implemented, we are transforming the way we process and see real estate. We are eliminating 3rd parties, minimizing fraudulent activities and making all the property ownership/transfer process fast, secure and efficient. We can modify CRYPact according to the design and structure of whatever real estate flow we need to buy and sell property but the core idea and base technology will remain the same.

Dedication

We dedicate this work to everyone which has done so much to further the cause of CRYPact – Smart Contracts, in the hope that this paper and project will bring its aim forward which is to improve the eco-system of humans to make the world a better place (specifically in Real Estate system using Blockchain Tech as per this project idea). And to our God Almighty and beloved parents. We would also like to convey special thanks to our project advisor and mentor respected Sir Kamran Shabbir, who has done his utmost best to help us in every mean possible throughout the journey; even after his long work hours he was always there to guide us and motivate us to reach the end destination. Thank you to everyone who helped us along the way.

Acknowledgements

It goes without saying that we dedicate this work purely to the God Almighty who always helps us in achieving our goals and pass us through think and thins. We can never do a thing without Him.

We present our appreciation and gratitude for all the motivation and help given by our project advisor Professor Kamran Shabbir without whose continuous support our project would not be able to complete within time. He provided us with all the resources we needed.

Finally, would like to thank everyone who helped us and guided us in this completion of project.

Regards,

Muhammad Samee Hassan

Muhammad Talha

Arif Mehmood Dar

List of Figures

List of Tables

Table of Contents

Revision History

Name	Date	Reason For Changes	Version

Chapter 1. Introduction

In overall history, real estate has undoubtedly served as the most stable and valuable store. Real Estate by far is one of the most popular aspect in everyone's life. While so important for many of the people, the process of buying and selling and transferring the ownership of the land/property has never been so easy. Suppose you have to buy or sell a property that is under any scheme by LDA, Lahore, Pakistan. After you have gone through the hideous process of getting different kinds of documents such as recovery memo, possession letter, sanction letter etc., you have to write an application to LDA for NOC which takes around two months due to slow and manual process. After NOC is approved, buyer and seller agrees on amount to be paid for sold property on stamp paper (which includes 3rd party). Seller sends a transfer request application to LDA, LDA provides a date for both parties to come at their office so that they can be verified and property ownership can be changed. After that, LDA mentions a specific date on which buyer gets his "Transfer Letter". In all this process, there is lot of physical exercise where one party has to physically be appeared before LDA.
This slow and hassle process can be eliminated using our system CRYPact – Smart Contracts. This automated system is here to handle all this hideous and manual process. Buyers and sellers will be able to get transfer letter from LDA through our fast, more secure and reliable system. Here, we are digitalizing the way people obtain and store their property transfer letters, where we exempt the need of physical stamp paper and saving the records on Blockchain for security and transparency.
After the creation of contract/transfer letter, it will be immutable and non forgeable which is the core benefit of CRYPact – Smart Contracts.

1.1 Product (Problem Statement)

The core of the problem is the lack of automated system, the need of 3rd party, and the most important, a central database for buying and selling properties which can be modified any time to make false records. The users of our system will get a secure, fast and totally automated system for the transfer of ownership of property. People will be able to change the ownership of property without the need of third party vendors. Additionally, the informality of procedures prevalent in certain jurisdictions makes them particularly susceptible to fraud or tampering. This system allows to reliably secure and record ownership of properties on blockchain.

1.2 Background

Real estate is a fundamental asset. It meets our need as a place for a home, to do business, and for many it is their main investment vehicle for retirement. Yet, why is it that we have to accept the uneasy and hideous process to change the ownership of property and go through all manual process which takes months and months. Important information about property assets can be accessed and modified anytime as the database is central. The physical stamp papers can be forged and mutate.

Observing all this background and hassle procedure, there are many flaws which can be improved using some sophisticated automated system based on blockchain technology.

That is why CRYPACT's vision is to radically **transform** the ecosystem of real estate (specifically proposed to LDA, Lahore, Pakistan), by making transactions more:

- Transparent
- Fast
- Secure

Resulting not only in improving the overall system, but more importantly helping to **protect** communities from real estate:

- Fraud
- Corruption
- Waste of time

1.3 Objective(s)/Aim(s)/Target(s)

Following are the objectives that we will achieved after completion of project:

- The main and most important aim of CRYPact – Smart Contracts is to show how we can make manual, fraudulent and slow process of ownership transfer for any plot into a better, fast, automated and transparent process which is cost effective and efficient in terms of saving time.
- People will be able to transfer ownership of property titles using CRYPact.
- People will be able to produce e-Stamp using CRYPact.
- People will be able to obtain required documents from LDA through CRYPact system which is required to show the proper ownership of the plot.
- Once record is saved on blockchain, it is immutable and non-forgeable.

1.4 Scope

CRYPact system is made for Government of Pakistan (specifically for Lahore Development Authority, Lahore) and according to its legal land registry/property rules which is covering a specific category of plots under LDA called “Allottee plots”.

CRYPact will be a web based system provides the following functionalities:

- User (first owner of plot) will be able to obtain all the necessary documents for plot (such as Allotment letter, Recovery Memo etc.) from LDA through CRYPact

- User/seller/owner will be able to send request for “Transfer letter” to LDA through CRYPact.

- User will be able to change the ownership from LDA through CRYPact

- The system is just to demonstrate the how we can automate and secure Real Estate system

1.5 Business Goals

The system is specifically made for Govt. of Pakistan (specifically for LDA, Lahore) which can also be sold to any such entity like private societies for their property records and transfer processes after making this system complete according to the specific needs and requirements but the underlying technology and roadmap for creating new one will remain same.

1.6 Challenges

Following challenges are expected on the road of the project:

- Extensive research on different Blockchains supporting smart contracts.
- Learning all consensus algorithms.
- Learning distributed computing.
- Professional web app development.
- Backend development to create smart contracts on Hyperledger.
- Learning new programming languages.

1.7 Learning Outcomes

Following outcomes are expected:

- Deep understanding of blockchain.
- Experience in building complex and professional website.
- Understanding of smart contracts and development on Hyperledger.
- New programming languages.
- Working in a team under timelines.
- Integrating 3rd party APIs.

1.8 Nature of End Product

The end product will be a web portal software based on Blockchain technology as backend running smart contracts and will be demonstrating on abstract level the automate way of how we can change the ownership of property and save the record on Blockchain.

1.9 Related Work/ Literature Survey/ Literature Review

First and the foremost thing to understand the underlying technology behind CRYPact – Smart Contracts is to get familiar with two terms that are “Blockchain” and literal meaning of “Smart Contracts”.

NOTE: The content mentioned below is intentionally copied from different sources on Internet and is for better understanding of overall system. All the references mentioned in “Reference” section and we have completely studied following content diving deep into it to understand different Blockchains and is part of our research.
Also, you might see some red or blue line errors on MS Word. It may be due to the fact that most of the terms mentioned are technical and is not recognized by “MS Word”.

BLOCKCHAIN

Blockchain is a distributed database existing on multiple computers at the same time. It is constantly growing as new sets of recordings, or ‘blocks’, are added to it. Each block contains a timestamp and a link to the previous block, so they actually form a chain. The database is not managed by any particular body; (that’s what makes blockchain safe) instead, everyone in the network gets a copy of the whole database. Even if one party runs the system/database, the authority will be needing private keys of other participants (which they don’t own) to change any kind of data on blockchain. Old blocks

are preserved forever and new blocks are added to the ledger irreversibly, making it impossible to manipulate by faking documents, transactions and other information.
This is completely one-way process and once the transaction is made, it is almost irreversible. We can only create new one and linked it to the previous one.

SMART CONTRACT

Smart contracts are self-executing contracts with the terms of the agreement between buyer and seller being directly written into lines of code. The code and the agreements contained therein exist across a distributed, decentralized Blockchain network. Smart contracts permit trusted transactions and agreements to be carried out among disparate, anonymous parties without the need for a central authority, legal system, or external enforcement mechanism. They render transactions traceable, transparent, and irreversible.

There are very few such systems available in the market. Most of them are not based on the Blockchain which is the major flaw. Those start-ups who are using Blockchains are yet be developed properly and their system design is not finalized where one of the reasons are they are trying to introduce their own currency or trying to develop their own new blockchain design for Real Estate.

We are mentioning detailed research work on different Blockchains that support “Smart Contract”.

ETHEREUM

Introduction:

Ethereum is a platform and a better alternative of Bitcoin blockchain specifically build to create ***Dapps*** and running ***smart contracts***. It is blockchain allowing anyone to write smart contracts and decentralized applications where they can create their own arbitrary rules for ownership, transaction formats and state transition functions.

Ethereum Accounts:

State is made with what is called accounts and which is not present in bitcoin blockchain.
Contains:

- Nonce
- Accounts Eth balance (Currency)
- Contract Code (if there is any)
- Accounts storage (empty by default)

- **Types of accounts:**
- Externally owned accounts - controlled by private keys.
- Contract accounts - controlled by their contract code.

Contracts in Ethereum are not like entities there are to be filled up but rather like “**autonomous agents**” that live inside of the Ethereum execution environment, always executing a specific piece of code when "poked" by a message or transaction, and having direct control over their own ether balance and their own key/value store to keep track of persistent variables.

Transactions:

The term "**transaction**" is used in Ethereum to refer to the signed data package that stores a message to be sent from an externally owned account. Transactions contain:

- The recipient of the message
- A signature identifying the sender
- The amount of ether to transfer from the sender to the recipient

- An optional data field (use case e.g.: domain registration service – name + IP)
- A ***STARTGAS*** value, representing the maximum number of computational steps the transaction execution is allowed to take
- A ***GASPRICE*** value, representing the fee the sender pays per computational step (Ethereum's Anti-Denial service model)

Messages:

Contracts have the ability to send "messages" to other contracts. Messages are virtual objects that are never serialized and exist only in the Ethereum execution environment. A message contains:

- The sender of the message (implicit)
- The recipient of the message
- The amount of ether to transfer alongside the message
- An optional data field
- A STARTGAS value

Messages are just like transactions and contracts can have relationships just like external actors to while sending messages.

State Transition Function:

The Ethereum state transition function, can be defined as follows:

1. Check if the transaction is well-formed (i.e. has the right number of values), the signature is valid, and the nonce matches the nonce in the sender's account. If not, return an error.
2. Calculate the transaction fee as STARTGAS * GASPRICE, and determine the sending address from the signature. Subtract the fee from the sender's account balance and increment the sender's nonce. If there is not enough balance to spend, return an error.
3. Initialize GAS = STARTGAS, and take off a certain quantity of gas per byte to pay for the bytes in the transaction.
4. Transfer the transaction value from the sender's account to the receiving account. If the receiving account does not yet exist, create it. If the receiving account is a contract, run the contract's code either to completion or until the execution runs out of gas.
5. If the value transfer failed because the sender did not have enough money, or the code execution ran out of gas, revert all state changes except the payment of the fees, and add the fees to the miner's account.

Code Execution:

The code in Ethereum contracts is written in a low-level, stack-based bytecode language, referred to as "Ethereum virtual machine code" or "EVM code". In general, code execution is an infinite loop that consists of repeatedly carrying out the operation at the current program counter (which begins at zero) and then incrementing the program counter by one, until the end of the code is reached or an error or STOP or RETURN instruction is detected. The operations have access to three types of space in which to store data:

- The **stack**.
- **Memory**. (an infinitely expandable byte array)
- The contract's long-term **storage**, a key/value store. Unlike stack and memory, which reset after computation ends, storage persists for the long term.

The code can also access the value, sender and data of the incoming message, as well as block header data, and the code can also return a byte array of data as an output.

The formal execution model of EVM code is surprisingly simple. While the Ethereum virtual machine is running, its full computational state can be defined by the tuple (block_state, transaction, message, code, memory, stack, pc, gas), where block_state is the global state containing all accounts and includes balances and storage. At the start of every round of execution, the current instruction is found by taking the pcth byte of code (or 0 if pc >= len(code)), and each instruction has its own definition in terms of how it affects the tuple.

Blockchain and Mining:

The basic block validation algorithm in Ethereum is as follows:

1. Check if the previous block referenced exists and is valid.
2. Check that the timestamp of the block is greater than that of the referenced previous block and less than 15 minutes into the future
3. Check that the block number, difficulty, transaction root, uncle root and gas limit (various low-level Ethereum-specific concepts) are valid.
4. Check that the proof of work on the block is valid.
5. Let S[0] be the state at the end of the previous block.
6. Let TX be the block's transaction list, with n transactions. For all i in 0...n-1, set S[i+1] = APPLY(S[i],TX[i]). If any application returns an error, or if the total gas consumed in the block up until this point exceeds the GASLIMIT, return an error.
7. Let S_FINAL be S[n], but adding the block reward paid to the miner.
8. Check if the Merkle tree root of the state S_FINAL is equal to the final state root provided in the block header. If it is, the block is valid; otherwise, it is not valid.

Main difference b/w Bitcoin blockchain and Ethereum blockchain:

Ethereum blockchain also stores the current/recent state which is help in getting to know name registry, asset holding etc.

MORE…

- Architecture explained in terms of layers to build Dapps:

Layer 5: Dapps
Layer 4: Browsers
Layer 3: Interoperability
Layer 2: Blockchain services – *Codes that actually do things – Timestamps - Contracts*
Layer 1: Economic – *Token to compute things for nodes – Incentives - Cryptocurrency*
Layer 0: Consensus – *Meta Protocol – Independent Chain – Vote of Node*

- **IPFS – InterPlantary File System:**

Protocol network design – P2P2 method of storing and sharing.

We can't store everything on blockchain (like if there are movies) so we need some p2p storage mechanism. This storage uses "**Distributed Hash Table"** mechanism which provides hashes for content and which is saved on Blockchains. On Ethereum blockchain we can use this mechanism to store things while building Dapps.

- **Applications:**

1. Online voting
2. Decentralized governance
3. Financial apps like sub-currencies or hedging contract
4. Gambling Platform
5. Crowdfunding platform (**Example**: Wiefund.io)
6. Supply chain (**Example**: Provenance.org)
7. E-commerce (**Example**: Openbazaar.com)

- **More Use cases**

1. Degree attestation system (**Idea**: Stack overflow, April, 2017)
2. Digital identity or Self-sovereign identity (**Example**: civic.com or uport.me)
3. Vehicle registration or real estate property registries/contracts
4. Games (**Example**: Cryptokitties.io) – open source and not solving any problem actually just for fun.
5. ICO's
6. Auction process

Advantages of the Ethereum blockchain

- The data written on the blockchain cannot be changed. It can be updated, but the previous versions are still kept. In addition, the details of the transaction (sender, receiver approximate time) are always recorded.
- What is stored on the blockchain is safe. No one has ever stolen from a blockchain wallet, without having the passwords.
- The Ethereum Network is distributed amongst thousands of individual computers in a variety of countries. No individual, country or organization can control the blockchain in any manner.
- Public: All transactions are view able by everyone. This is both up and down.

Disadvantages of the Ethereum Blockchain

- While a major development goal for Ethereum is speed, it will always be slower than traditional computer processes. It's distributed architecture drives this.
- Every transaction takes gas, because it is requiring work from a number of computers.

CARDANO

Introduction:

Cardano is the new crypto currency launched in September 2017 after almost 2 years of development. It is different from other crypto currency projects as it is built around peer reviewed paper which means that instead of writing a white paper and implementing it, the cardano team makes sure that experts from around the world read their papers and agree upon them which makes it a very different approach.
Cardano claims to be the third generation of crypto currency. The three generations can be divided as follows:

First Generation:
The first generation of crypto currency is Bitcoin considered as digital gold which is used to transfer and store money but has some issues related to scalability.

Second Generation:

Smart contract help you exchange money, property, shares, or anything of value in a transparent, conflict-free way while avoiding the services of a middleman.

Vitalik Buterin's Ethereum is easily the stalwart of this generation. They showed the world how the blockchain can evolve from a simple payment mechanism to something far more meaningful and powerful.

Third Generation:

Cardano is considered to be the third generation of block chain technology and it wants to solve three main ploblems faced by the previous generations which are:

- Scalability.
- Interoperability.
- Sustainability.

let's look at the three elements in detail that Cardano aims to solve.

Element #1: Scalability

When we talk about scalability, people think of it just as the transactions per second but according to Charles Hoskinson, it is actually divided into three categories.

- Transactions per second/ Throughput
- Network.
- Data Scaling.

Throughput

Bitcoin manages 7 transactions per second and Ethereum manages 15-20. This is absolutely not acceptable for a financial system.

Cardano hopes to solve this problem with their consensus mechanism, Ouroboros. It is a provably secure proof-of-stake algorithm.

Now before talking about Ouroborus, we need to know what proof of work is.

Proof of stake:

Proof of stake will make the entire mining process virtual and replace miners with validators.

This is how the process will work:

- The validators will have to lock up some of their coins as stake.
- After that, they will start validating the blocks. Meaning, when they discover a block which they think can be added to the chain, they will validate it by placing a bet on it.
- If the block gets appended, then the validators will get a reward proportionate to their bets.

Ouroborus:

In this algorithm the network elects a few nodes to mine the next block.

These are called the slot leaders. To make this happen, Cardano divides the time into several epochs. An epoch is then divided into slots. A slot is a short period of time in which exactly one block can be created.

The network the elects a slot leader for each slot and this is the only person that can mine a block in that slot. The slot leaders listen for new transactions, verify them and then put them inside a block.

If a slot leader doesn't complete is task in time or doesn't show up, he looses the right to produce a block and has to wait until it is re-elected by the block.

Network:

As The transactions carry data. So as the number of transactions increases so does the requirement for network resources.

The notion is pretty straightforward: If a system is to scale up to millions of users, the network will need 100s of terabytes or exabytes of resources to sustain itself.

To solve this issue, Cardano is looking at a new type of technology called RINA, Recursive Inter-Network Architecture created by John Day. It is a new type of structuring networks using policies and ingenious engineering principles.

Data Scaling:

Blockchains store things for eternity. Every little piece of data, relevant or not gets stored in the blockchain for eternity. As, the system scales up and more and more people come in, with the sheer influx of data the blockchain gets more and bulkier.

Now, remember that a blockchain runs because it comprises of Nodes. Each node is a user who stores a copy of the blockchain in their system.

As the blockchain gets bulkier, it will demand more space, and that is unreasonable for a normal user with a normal computer.

The way Cardano wants to solve this problem is by implementing a simple philosophy, "Not everyone needs all the data."

The techniques that Cardano is looking into are:

- Pruning
- Subscriptions
- Compression.

Element#2: Interoperability

Let's look at the current ecosystem. In the cryptosphere, we have different crypto coins such as Bitcoin, Ethereum, Litecoin etc. Similarly, in the legacy financial world, we have systems like the traditional Banks which use SWIFT, ACH etc.

The problem lies in the fact that it is extremely difficult for these individual entities to communicate with one another. It is tough for bitcoin to know what is going on in Ethereum and vice-versa. This becomes doubly difficult when banks try to communicate with the cryptos.

This is why, the crypto exchanges, which provide a portal between cryptos and banks become so powerful and important. However, there in itself lies a problem. Exchanges are not a decentralized entity and are extremely vulnerable.

- They can get hacked.
- They can blackout for long periods for system up gradation

A third-generation crypto coin must provide an ecosystem where each individual blockchain can communicate with another blockchain and with external legacy financial systems.

So, let's looks at how Cardano plans to do increase interoperability in both the crypto world and the legacy world.

Cardano's vision is to create an eco system in which different coins can seamlessly flow in each other without the need to go through centralized exchanges.

1.10 The Legacy World: Bridging the Gap

When it comes to increasing the interoperability with the legacy world, Cardano wants to focus on the three obstacles that make the crypto world incompatible with the legacy world:

- Metadata.
- Attribution.
- Compliance.

Metadata:

Metadata means the story behind the transaction.

If Alice were to spend 50 USD, the metadata of that could be as follows:

- What did Alice spend the money on?
- Who did Alice give that money to?
- Where did she spend the money?

1.10.1 Attribution

Similar to metadata, via attribution the names of the people involved in the transactions gets known. Basically, who all are a particular transaction attributed to?

If the blockchain permanently fixes attribution to itself, it will greatly compromise on the privacy of the individuals involved.

Hence, Cardano plans to empower their users to hand out attribution as and when it is required.

1.10.2 Compliance

The third obstacle is “Compliance”.

Compliance includes factors like: KYC (Know Your Customer), AML (Anti Money Laundering), ATF (Anti Terrorist Financing) etc.

Compliance is used to check the legitimacy of a transaction. Basically, if Alice pays Bob $50, compliance is used to make sure that the transaction is not done for any nefarious purposes.

While the crypto world hasn’t really done much on this front, it is extremely critical in the banking world where the history and legitimacy of each transaction must be known.

Element #3: Sustainability

Right now there are a lot of people who want to build companies around crypto currencies. To raise money for there company, the launch an initial coin offering. After and ICO, the team ends up with a lot of capital that they can use to start there company. But what happens if after some time, this money runs out? Should they create a new coin and hold another ICO just to get some cash?

Cardano intends to solve this problem by creating a treasury. The idea behind a treasury is that it will receive a small percentage of a transaction that happens on the network. The treasury is a special wallet that isn’t controlled by anyone. It is a sort of smart contract that can release some funds to developers who intend to improve the Cardano system.

To do this, the developers have to submit a proposal to the community saying what they want to change and how much money they want to do that. The community then vote on the ideas that they think is most important. The treasury chooses the idea with the most number of votes after some time and releases funds for it.

In this way the treasury will keep Cardano sustainable by giving a contentious stream of money that can be used to do research and to improve the system.

Advantages:

One of the things that make Cardano unique is the fact that it has a fully open source. The team is developing a smart contract platform which will give users of the Cryptocurrency advanced features that aren’t available on other Cryptocurrencies. Cardano is the first blockchain platform that was formed solely through scientific philosophy and a research-driven approach. They can do this because the team is made up of expert engineers and researchers.

Regarding security, it is one of the most secure Blockchains. The development team are looking to create a platform for decentralized applications and smart contracts which can be processed using a technique called formal verification. Formal verification is a computational activity that leaves no details out. It ensures that all transactions and activities are fully secured while functions will be carried out without any changes in the cryptocurrency’s protocol.

Cardano’s (ADA) uses a special blockchain that is designed with the Ouroboros algorithm. The development team argue that this algorithm is one of the most secure systems ever developed. Thus investors can be at ease because their private information will never be leaked or stolen on the platform.

Cardano has been designed to work perfectly in regulated industries and protecting individual privacy at the same time. The applications developed on the Cardano system can be customized individually

to comply with the regulatory requirement. Individuals though are given the choice of protecting their privacy by choosing whether they want to enter regulated domains.

The cryptocurrency has become strong in the academic field, becoming the first cryptocurrency to have one of its academic papers accepted at Crypto 2017, the leading cryptography conference. IOHK have built research centres and made research partners in several top academic institutions in the world, with the top institutes being the University of Edinburgh and the Tokyo Institute of Technology.

Transaction Fees in Cardano SL

1.11 Motivation

There are two main reasons why transaction fees are needed for Cardano SL:

1. People who run full Cardano SL nodes spend time, money and effort to run the protocol, for which they should be compensated and rewarded. In contrast to Bitcoin, where new currency is created with each mined block, in Cardano SL, transaction fees are the only source of income for participants in the protocol.
2. The second reason is the prevention of DDoS (Distributed Denial of Service) attacks. In a DDoS attack, an attacker tries to flood the network with dummy transactions, and if he has to pay a sufficiently high fee for each of those dummy transactions, this form of attack will become prohibitively expensive for him.

1.12 How transaction fees work

Whenever somebody wants to transfer an amount of Ada, some minimal fees are computed for that transaction. In order for the transaction to be valid, these minimal fees have to be included, although the sender is free to pay higher fees if he so wishes.

Please also read about transaction distribution below.

1.13 Minimal transaction fees

The minimal fees for a transaction are calculated according to the formula:

a + b × size

where:

- a is a special constant, at the moment it is 0.155381 ADA;
- b is a special constant, at the moment it is 0.000043946 ADA/byte;
- size is the size of the transaction in bytes.

This means that each transaction costs at least 0.155381 ADA, with an additional cost of 0.000043946 ADA per byte of transaction size. For example, a transaction of size 200 bytes (a fairly typical size) costs:

0.155381 ADA + 0.000043946 ADA/byte × 200 byte = 0.1641702 ADA.

The reason for having parameter a is the prevention of DDoS attacks mentioned above: even a very small dummy transaction should cost enough to hurt an attacker who tries to generate many thousands of them.

Parameter b has been introduced to reflect actual costs: storing larger transactions needs more computer memory than storing smaller transactions, so larger transactions should be more expensive than smaller ones.

Although particular values for parameters a and b were calculated, these values will probably be adjusted in future to better reflect actual costs.

NEO

What is NEO?

An introduction to the cryptocurrency and blockchain platform formerly known as AntShare.

NEO is the use of blockchain technology and digital identity to digitize assets, the use of smart contracts for digital assets to be self-managed, to achieve "smart economy" with a distributed network.

Where did it come from!?

There's a fair amount of confusion surrounding the Neo platform. Not surprising when you consider the project's complicated history.

Neo began life as AntShares (ANS) in 2014. AntShares, founded by Da Hongfei and Erik Zhang, has been referred to as 'China's first blockchain platform'. In 2016, supposedly in response to growing interest in AntShares, and a need for blockchain solutions that meet the requirements of both government regulators and private companies, Da and Erik founded OnChain, a venture-backed company that provides blockchain-based financial services. In 2017, AntShares was rebranded as Neo.

Neo and OnChain are based in Shanghai. It's certainly the case that Chinese regulation can have far-reaching effects on cryptocurrency markets and development. Neo is equal parts vulnerable to and well-positioned to inform and cooperate with Chinese oversight.

A Smart Economy

The Neo whitepaper is our key resource in understanding the platform. Unfortunately, aspects of Neo are still in development, and certain details are unclear. At times, the whitepaper reads more as an overview of smart contracts in general than a specific guide to Neo's inner workings.

In concept, Neo is a smart contracts ecosystem, similar to Ethereum. It allows users to automate the storage and exchange of digital assets. In order to compete with more established smart contracts implementations, Neo takes advantage of evolving technology and cooperation with Chinese authorities towards the stated goal of a 'smart economy'.

Digital Identity

In 2005, China's 'Digital Signature Act' allowed digital signatures to be legally binding in theory. The trouble here is that a means of digital identification that meets the requirements of this regulation has been hard to come by. In 2016, partnering with Microsoft China, OnChain founded Legal Chain with the goal of providing this means of identification. Legal Chain intends to apply the immutability and transparency of blockchain systems to meet these requirements, with the aim of integrating face and voice recognition along the way.

This concept of digital identity is a key feature in Neo's proposed smart economy. Maintaining a trusted link between digital and physical entities means that you should be able to follow abuse of the system right back to a legally-binding identity.

1.13.1 Economic Model

NEO has two native tokens, NEO (abbreviated symbol NEO) and NeoGas (abbreviated symbol GAS).

NEO, with a total of 100 million tokens, represents the right to manage the network. Management rights include voting for bookkeeping, NEO network parameter changes, and so on. The minimum unit of NEO is 1 and tokens cannot be subdivided.

GAS is the fuel token for the realization of NEO network resource control, with a maximum total limit of 100 million. The NEO network charges for the operation and storage of tokens and smart contracts, thereby creating economic incentives for bookkeepers and preventing the abuse of resources. The minimum unit of GAS is 0.00000001.

In the genesis block of the NEO network, 100 million NEOs are generated, GAS has not yet been generated. 100 million GAS, corresponding to the 100 million NEO, will be generated through a decay algorithm in about 22 years time to address holding NEO. If NEO is transferred to a new address, the subsequent GAS generated will be credited to the new address.

Consensus

Neo employs a consensus mechanism called Delegated Byzantine Fault Tolerance (dBFT). Participants in the system are able to designate certain nodes as bookkeepers. A bookkeeper node must maintain a minimum balance of NEO and meet certain performance requirements.

Bookkeepers are tasked with verifying the blocks that are written to the blockchain. If two-thirds of the nodes on the network can agree with a bookkeeper's version of the blockchain, consensus is achieved and the proposed version of the blockchain is validated. If consensus fails, an alternate bookkeeper is called and the process is repeated.

Because this consensus only needs to be replicated across a subset of the network, it is said to be more efficient than classic Byzantine Fault Tolerance. The network as a whole consumes fewer resources and can handle higher transaction volumes.

With dBFT and some other key optimizations, Neo claims to be able to handle over 1,000 transactions per second, with a goal of optimizing to over 10,000 transactions per second. Compare that to Ethereum's current rate of 15 transactions per second.

That's a big advantage but it could be argued that these gains come at the cost of centralization. Digital Identification and dBFT may serve to limit control of the system to a select group.

The dBFT combines digital identity technology, meaning the bookkeepers can be a real name of the individual or institution. Thus, it is possible to freeze, revoke, inherit, retrieve, and ownership transfer due to judicial decisons on them. This facilitates the registration of compliant financial assets in the NEO network. The NEO network plans to support such operations when necessary.

NeoContracts

Neo's smart contracts are called NeoContracts. One of the big obstacles to designing smart contracts is that their results need to be reproduced reliably across a network.

If a contract is referenced on a blockchain and it yields different results on different systems, the network can't reliably agree on what the blockchain looks like and blocks will be stalled. But a smart contract can't perform meaningful operations without accessing some variables.

Timestamps—Maybe you want to use smart contracts to automate weekly payments to an employee or settle an account with a distributor every 30 days. Your contract will need to know what time it is. To provide consistent access to time data, Neo registers a timestamp to every new block that is generated. A new block is added every 15 seconds, so contracts can access the current time to within 15 seconds.

Randomness—Also useful is the ability to generate random numbers. But how do you provide a random number while still ensuring that the same random number is identified across the network? To provide smart contracts with access to randomness, a random number is inserted into the Nonce field of every new block. Contracts can reference this Nonce field to access this random number.

Data Storage—Data in NeoContracts can be stored privately, accessible only to the contract with which it is associated. Data may also be stored in a global context, accessible to all of the contracts on the network. External data must be transferred to the Neo blockchain and passed on to these private or public data stores in order to be referenced by contracts.

The Tokens

The platform involves 2 different tokens. NEO and GAS are the cryptographic currencies that drive the Neo network. Both NEO and GAS are capped at 100 million tokens each.

The NEO token is representative of shares in the Neo market, and cannot be divided. NEO holders get voting rights in the NEO ecosystem as well as rights to dividends in the form of GAS. 50 million NEO were distributed through initial crowd funding. The remaining 50 million tokens are fixed with a 1-year lockout period, expiring October 16, 2017.

These lockout tokens are to be managed by the NEO Council (A group of the project's founders) to support development and maintenance of the ecosystem. Specifically, 10 million tokens are earmarked to reward core developers and members of the NEO Council, another 10 million are to be used to stimulate the Neo development ecosystem, 15 million tokens are to be retained as a 'contingency', and the remaining 15 million are to be cross-invested in blockchain ecosystems supporting Neo.

Neo's alternate token, GAS, is generated at a rate of 8 GAS per block with the construction of the blockchain. The rate of production is reduced by 1 token for every 2 million blocks generated.

Sometime around 2039, GAS circulation will reach 100 million and production will cease. Unlike NEO, GAS can be divided.

GAS dividends also accumulate as fees to the network. Users pay in GAS to deploy and run smart contracts. Fees are proportional to the computing resources consumed by the contract. These fees are distributed to 'bookkeepers' as reward for their activity on the network.

Superconducting Transactions:

In a traditional currency exchange, orders are placed and matched in a centralized marketplace. The process is efficient but it requires that the user release control of their funds to the exchange.

By automating the placement and matching of orders across a consensus network, you can ensure that orders are matched and processed fairly and transparently, effectively creating a decentralized exchange. But this results in slow transactions as adjustments must to be validated across the network.

Neo proposes a system whereby exchange transactions are settled on the blockchain but order matching is handled off-chain by a central exchange. Neo calls these transactions 'Superconducting Transactions'. This is intended to provide the efficiency of centralized exchanges with the security of a decentralized exchange.

NeoX

NeoX allows transactions to traverse blockchains. I can't find much in detail about this protocol. Similar protocols involve generating smart contracts that serve to lock funds on one blockchain in return for access to funds on an alternate chain.

NeoFS

NeoFS allows large files to be divided and distributed across the network. Users can specify the level of reliability they expect of a file. Files with low reliability requirements can be stored and retrieved at minimal cost. For a higher fee, data can be stored on more reliable nodes.

NeoQS

Quantum computers threaten the security of certain cryptographic techniques. Neo uses a lattice based cryptographic mechanism that it calls NeoQS (Quantum Safe) which is theoretically resistant to attacks from quantum computers. It's not likely that quantum computing will affect cryptographic systems in the near future, but it does offer some peace of mind.

1.14 Fees

NEO's distributed architecture provides high redundancy of storage capacity, and the use of this capacity is not free. Deploying a smart contract on the NEO network requires a fee, currently fixed at 500GAS, which is collected by the system and recorded as a system gain. Future fees will be adjusted according to the actual operating cost in the system. The smart contract deployed on the blockchain can be used multiple times, until the contract is destroyed by the deployer.

Advantages:

- Superconducting Transactions
- Cross-chain Interoperability

- Less Transection cost
- High Scalability
- Low Coupling
- Provide Neo VM

HYPERLEDGER (FABRIC)

The Linux Foundation founded Hyperledger in 2015 to advance cross-industry blockchain technologies. Rather than declaring a single blockchain standard, it encourages a collaborative approach to developing blockchain technologies via a community process, with intellectual property rights that encourage open development and the adoption of key standards over time.
Hyperledger Fabric is one of the blockchain projects within Hyperledger. Like other blockchain technologies, it has a ledger, uses smart contracts, and is a system by which participants manage their transactions.

Where Hyperledger Fabric breaks from some other blockchain systems is that it is private and permissioned. Rather than an open permission less system that allows unknown identities to participate in the network (requiring protocols like Proof of Work to validate transactions and secure the network), the members of a Hyperledger Fabric network enrol through a trusted Membership Service Provider (MSP).

Hyperledger Fabric also offers several pluggable options. Ledger data can be stored in multiple formats, consensus mechanisms can be swapped in and out, and different MSPs are supported.
Hyperledger Fabric also offers the ability to create channels, allowing a group of participants to create a separate ledger of transactions. This is an especially important option for networks where some participants might be competitors and not want every transaction they make - a special price they're offering to some participants and not others, for example - known to every participant. If two participants form a channel, then those participants – and no others – have copies of the ledger for that channel.

Shared Ledger

Hyperledger Fabric has a ledger subsystem comprising two components: the world state and the transaction log. Each participant has a copy of the ledger to every Hyperledger Fabric network they belong to.
The world state component describes the state of the ledger at a given point in time. It's the database of the ledger. The transaction log component records all transactions which have resulted in the current value of the world state; it's the update history for the world state. The ledger, then, is a combination of the world state database and the transaction log history.
The ledger has a replaceable data store for the world state. By default, this is a LevelDB key-value store database. The transaction log does not need to be pluggable. It simply records the before and after values of the ledger database being used by the blockchain network.

Smart Contracts

Hyperledger Fabric smart contracts are written in Chaincode and are invoked by an application external to the blockchain when that application needs to interact with the ledger. In most cases, Chaincode interacts only with the database component of the ledger, the world state (querying it, for example), and not the transaction log.
Chaincode can be implemented in several programming languages. The currently supported Chaincode language is Go with support for Java and other languages coming in future releases.

Privacy

Depending on the needs of a network, participants in a Business-to-Business (B2B) network might be extremely sensitive about how much information they share. For other networks, privacy will not be a top concern.
Hyperledger Fabric supports networks where privacy (using channels) is a key operational requirement as well as networks that are comparatively open.

Consensus

Transactions must be written to the ledger in the order in which they occur, even though they might be between different sets of participants within the network. For this to happen, the order of transactions must be established and a method for rejecting bad transactions that have been inserted into the ledger in error (or maliciously) must be put into place.
This is a thoroughly researched area of computer science, and there are many ways to achieve it, each with different trade-offs. For example, PBFT (Practical Byzantine Fault Tolerance) can provide a mechanism for file replicas to communicate with each other to keep each copy consistent, even in the event of corruption. Alternatively, in Bitcoin, ordering happens through a process called mining where competing computers race to solve a cryptographic puzzle which defines the order that all processes subsequently build upon.

Hyperledger Fabric has been designed to allow network starters to choose a consensus mechanism that best represents the relationships that exist between participants. As with privacy, there is a spectrum of needs; from networks that are highly structured in their relationships to those that are more peer-to-peer.

Architecture Explained

The Hyperledger Fabric architecture delivers the following advantages:
Chaincode trust flexibility. The architecture separates trust assumptions for chaincodes (blockchain applications) from trust assumptions for ordering. In other words, the ordering service may be provided by one set of nodes (orderers) and tolerate some of them to fail or misbehave, and the endorsers may be different for each chaincode.
Scalability. As the endorser nodes responsible for particular chaincode are orthogonal to the orderers, the system may scale better than if these functions were done by the same nodes. In particular, this results when different chaincodes specify disjoint endorsers, which introduces a partitioning of chaincodes between endorsers and allows parallel chaincode execution (endorsement). Besides, chaincode execution, which can potentially be costly, is removed from the critical path of the ordering service.
Confidentiality. The architecture facilitates deployment of chaincodes that
have confidentiality requirements with respect to the content and state updates of its transactions.
Consensus modularity. The architecture is modular and allows pluggable consensus (i.e., ordering service) implementations.
Part I: Elements of the architecture relevant to Hyperledger Fabric v1
System architecture
Basic workflow of transaction endorsement
Endorsement policies
Part II: Post-v1 elements of the architecture
Ledger checkpointing (pruning)

1. System architecture

The blockchain is a distributed system consisting of many nodes that communicate with each other. The blockchain runs programs called chaincode, holds state and ledger data, and executes transactions. The chaincode is the central element as transactions are operations invoked on the chaincode. Transactions have to be “endorsed” and only endorsed transactions may be committed and have an effect on the state. There may exist one or more special chaincodes for management functions and parameters, collectively called system chaincodes.

1.1. Transactions

Transactions may be of two types:
Deploy transactions create new chaincode and take a program as parameter. When a deploy transaction executes successfully, the chaincode has been installed “on” the blockchain.
Invoke transactions perform an operation in the context of previously deployed chaincode. An invoke transaction refers to a chaincode and to one of its provided functions. When successful, the chaincode executes the specified function - which may involve modifying the corresponding state, and returning an output.
As described later, deploy transactions are special cases of invoke transactions, where a deploy transaction that creates new chaincode, corresponds to an invoke transaction on a system chaincode.

1.2. Blockchain datastructures

1.2.1. State

The latest state of the blockchain (or, simply, state) is modeled as a versioned key-value store (KVS), where keys are names and values are arbitrary blobs. These entries are manipulated by the chaincodes (applications) running on the blockchain through put and get KVS-operations. The state is stored persistently and updates to the state are logged. Notice that versioned KVS is adopted as state model, an implementation may use actual KVSs, but also RDBMSs or any other solution.
More formally, state s is modeled as an element of a mapping K -> (V X N), where:
K is a set of keys
V is a set of values
N is an infinite ordered set of version numbers. Injective function next: N -> N takes an element of N and returns the next version number.
Both V and N contain a special element $\perp$ (empty type), which is in case of N the lowest element. Initially all keys are mapped to ($\perp$, $\perp$). For s(k)=(v,ver) we denote v by s(k).value, and ver by s(k).version.
KVS operations are modeled as follows:
put(k,v) for $k \in K$ and $v \in V$, takes the blockchain state s and changes it to s' such that s'(k)=(v,next(s(k).version)) with s'(k')=s(k') for all k'!=k.
get(k) returns s(k).
State is maintained by peers, but not by orderers and clients.
State partitioning. Keys in the KVS can be recognized from their name to belong to a particular chaincode, in the sense that only transaction of a certain chaincode may modify the keys belonging to this chaincode. In principle, any chaincode can read the keys belonging to other chaincodes. Support for cross-chaincode transactions, that modify the state belonging to two or more chaincodes is a post-v1 feature.

1.2.2 Ledger

Ledger provides a verifiable history of all successful state changes (we talk about validtransactions) and unsuccessful attempts to change state (we talk about invalid transactions), occurring during the operation of the system.
Ledger is constructed by the ordering service as a totally ordered hashchain of blocks of (valid or invalid) transactions. The hashchain imposes the total order of blocks in a ledger and each block contains an array of totally ordered transactions. This imposes total order across all transactions.
Ledger is kept at all peers and, optionally, at a subset of orderers. In the context of an orderer we refer to the Ledger as to OrdererLedger, whereas in the context of a peer we refer to the ledger as to PeerLedger. PeerLedger differs from the OrdererLedger in that peers locally maintain a bitmask that tells apart valid transactions from invalid ones.
Peers may prune PeerLedger as described in Section XX (post-v1 feature). Orderers maintain OrdererLedger for fault-tolerance and availability (of the PeerLedger) and may decide to prune it at anytime, provided that properties of the ordering service are maintained.
The ledger allows peers to replay the history of all transactions and to reconstruct the state.

1.3. Nodes

Nodes are the communication entities of the blockchain. A “node” is only a logical function in the sense that multiple nodes of different types can run on the same physical server. What counts is how nodes are grouped in “trust domains” and associated to logical entities that control them.

There are three types of nodes:
Client or submitting-client: a client that submits an actual transaction-invocation to the endorsers, and broadcasts transaction-proposals to the ordering service.
Peer: a node that commits transactions and maintains the state and a copy of the ledger. Besides, peers can have a special endorser role.
Ordering-service-node or orderer: a node running the communication service that implements a delivery guarantee, such as atomic or total order broadcast.
The types of nodes are explained next in more detail.

1.3.1. Client

The client represents the entity that acts on behalf of an end-user. It must connect to a peer for communicating with the blockchain. The client may connect to any peer of its choice. Clients create and thereby invoke transactions.
As detailed in Section 2, clients communicate with both peers and the ordering service.

1.3.2. Peer

A peer receives ordered state updates in the form of blocks from the ordering service and maintain the state and the ledger.
Peers can additionally take up a special role of an endorsing peer, or an endorser. The special function of an endorsing peer occurs with respect to a particular chaincode and consists in endorsing a transaction before it is committed. Every chaincode may specify an endorsement policy that may refer to a set of endorsing peers. The policy defines the necessary and sufficient conditions for a valid transaction endorsement (typically a set of endorsers’ signatures), as described later in Sections 2 and 3. In the special case of deploy transactions that install new chaincode the (deployment) endorsement policy is specified as an endorsement policy of the system chaincode.

1.3.3. Ordering service nodes (Orderers)

The orderers form the ordering service, i.e., a communication fabric that provides delivery guarantees. The ordering service can be implemented in different ways: ranging from a centralized service (used e.g., in development and testing) to distributed protocols that target different network and node fault models.
Ordering service provides a shared communication channel to clients and peers, offering a broadcast service for messages containing transactions. Clients connect to the channel and may broadcast messages on the channel which are then delivered to all peers. The channel supports atomic delivery of all messages, that is, message communication with total-order delivery and (implementation specific) reliability. In other words, the channel outputs the same messages to all connected peers and outputs them to all peers in the same logical order. This atomic communication guarantee is also called total-order broadcast, atomic broadcast, or consensus in the context of distributed systems. The communicated messages are the candidate transactions for inclusion in the blockchain state.
Partitioning (ordering service channels). Ordering service may support multiple channels similar to the topics of a publish/subscribe (pub/sub) messaging system. Clients can connect to a given channel and can then send messages and obtain the messages that arrive. Channels can be thought of as partitions - clients connecting to one channel are unaware of the existence of other channels, but clients may connect to multiple channels. Even though some ordering service implementations included with Hyperledger Fabric support multiple channels, for simplicity of presentation, in the rest of this document, we assume ordering service consists of a single channel/topic.
Ordering service API. Peers connect to the channel provided by the ordering service, via the interface provided by the ordering service. The ordering service API consists of two basic operations (more generally asynchronous events):

TODO add the part of the API for fetching particular blocks under client/peer specified sequence numbers.
broadcast(blob): a client calls this to broadcast an arbitrary message blob for dissemination over the channel. This is also called request(blob) in the BFT context, when sending a request to a service.
deliver(seqno, prevhash, blob): the ordering service calls this on the peer to deliver the message blob with the specified non-negative integer sequence number (seqno) and hash of the most recently delivered blob (prevhash). In other words, it is an output event from the ordering service. deliver() is also sometimes called notify() in pub-sub systems or commit() in BFT systems.
Ledger and block formation. The ledger (see also Sec. 1.2.2) contains all data output by the ordering service. In a nutshell, it is a sequence of deliver(seqno, prevhash, blob) events, which form a hash chain according to the computation of prevhash described before.
Most of the time, for efficiency reasons, instead of outputting individual transactions (blobs), the ordering service will group (batch) the blobs and output blocks within a single deliver event. In this case, the ordering service must impose and convey a deterministic ordering of the blobs within each block. The number of blobs in a block may be chosen dynamically by an ordering service implementation.
In the following, for ease of presentation, we define ordering service properties (rest of this subsection) and explain the workflow of transaction endorsement (Section 2) assuming one blob per deliver event. These are easily extended to blocks, assuming that a deliver event for a block corresponds to a sequence of individual deliver events for each blob within a block, according to the above mentioned deterministic ordering of blobs within a blocs.

Ordering service properties

The guarantees of the ordering service (or atomic-broadcast channel) stipulate what happens to a broadcasted message and what relations exist among delivered messages. These guarantees are as follows:

Safety (consistency guarantees):
As long as peers are connected for sufficiently long periods of time to the channel (they can disconnect or crash, but will restart and reconnect), they will see an identical series of delivered (seqno, prevhash, blob) messages. This means the outputs (deliver() events) occur in the same order on all peers and according to sequence number and carry identical content (blob and prevhash) for the same sequence number. Note this is only a logical order, and a deliver(seqno, prevhash, blob) on one peer is not required to occur in any real-time relation to deliver(seqno, prevhash, blob) that outputs the same message at another peer. Put differently, given a particular seqno, no two correct peers deliver different prevhash or blob values. Moreover, no value blob is delivered unless some client (peer) actually called broadcast(blob) and, preferably, every broadcasted blob is only delivered once.
Furthermore, the deliver() event contains the cryptographic hash of the data in the previous deliver() event (prevhash). When the ordering service implements atomic broadcast guarantees, prevhash is the cryptographic hash of the parameters from thedeliver() event with sequence number seqno-1. This establishes a hash chain across deliver() events, which is used to help verify the integrity of the ordering service output. In the special case of the first deliver() event, prevhashhas a default value.

Liveness (delivery guarantee):
Liveness guarantees of the ordering service are specified by a ordering service implementation. The exact guarantees may depend on the network and node fault model.
In principle, if the submitting client does not fail, the ordering service should guarantee that every correct peer that connects to the ordering service eventually delivers every submitted transaction.
To summarize, the ordering service ensures the following properties:
Agreement. For any two events at correct peers deliver(seqno, prevhash0, blob0) anddeliver(seqno, prevhash1, blob1) with the same seqno, prevhash0==prevhash1 and blob0==blob1;

Hashchain integrity. For any two events at correct peersdeliver(seqno-1, prevhash0, blob0) and deliver(seqno, prevhash, blob),prevhash = HASH(seqno-1||prevhash0||blob0).
No skipping. If an ordering service outputs deliver(seqno, prevhash, blob) at a correct peer p, such that seqno>0, then p already delivered an eventdeliver(seqno-1, prevhash0, blob0).
No creation. Any event deliver(seqno, prevhash, blob) at a correct peer must be preceded by a broadcast(blob) event at some (possibly distinct) peer;
No duplication (optional, yet desirable). For any two events broadcast(blob) and broadcast(blob'), when two events deliver(seqno0, prevhash0, blob) and deliver(seqno1, prevhash1, blob') occur at correct peers and blob == blob', then seqno0==seqno1 andprevhash0==prevhash1.
Liveness. If a correct client invokes an event broadcast(blob) then every correct peer “eventually” issues an event deliver(*, *, blob), where * denotes an arbitrary value.

1.15 Document Conventions

- Font size for heading is 14 and for text is 12.
- Style used for headings is Times and for text is Arial.
- The headings are bold and subheadings are mentioned using the related version.
- Title page and name of product in headers are bold and Italicized.

Chapter 2. Overall Description

2.1 Product Features

CRYPact is a web based platform which allow users to obtain transfer letter and other required documents against their properties, produce e-stamp papers and to transfer property ownerships. All the records will be saved and maintain on Blockchain.

2.2 Functional Description

CRYPact is a web based platform which allow users to obtain transfer letter and other required documents against their properties, produce e-stamp papers and to transfer property ownerships. All the records will be saved and maintain on Blockchain.

2.3 User Classes and Characteristics

- Primary:
  - Buyer
  - Seller

- Secondary:
  - LDA

- Tertiary:
  - None

2.4 Design and Implementation Constraints

The design implemented according to the needs and rules of LDA for transfer of property ownership change. There are some constraints and boundaries of our system which are discussed below:

- The system will be in English language only and users must know the language in order to use it.
- Some of the features and functions don't work on Internet Explorer.
- A stable Internet connection is required.

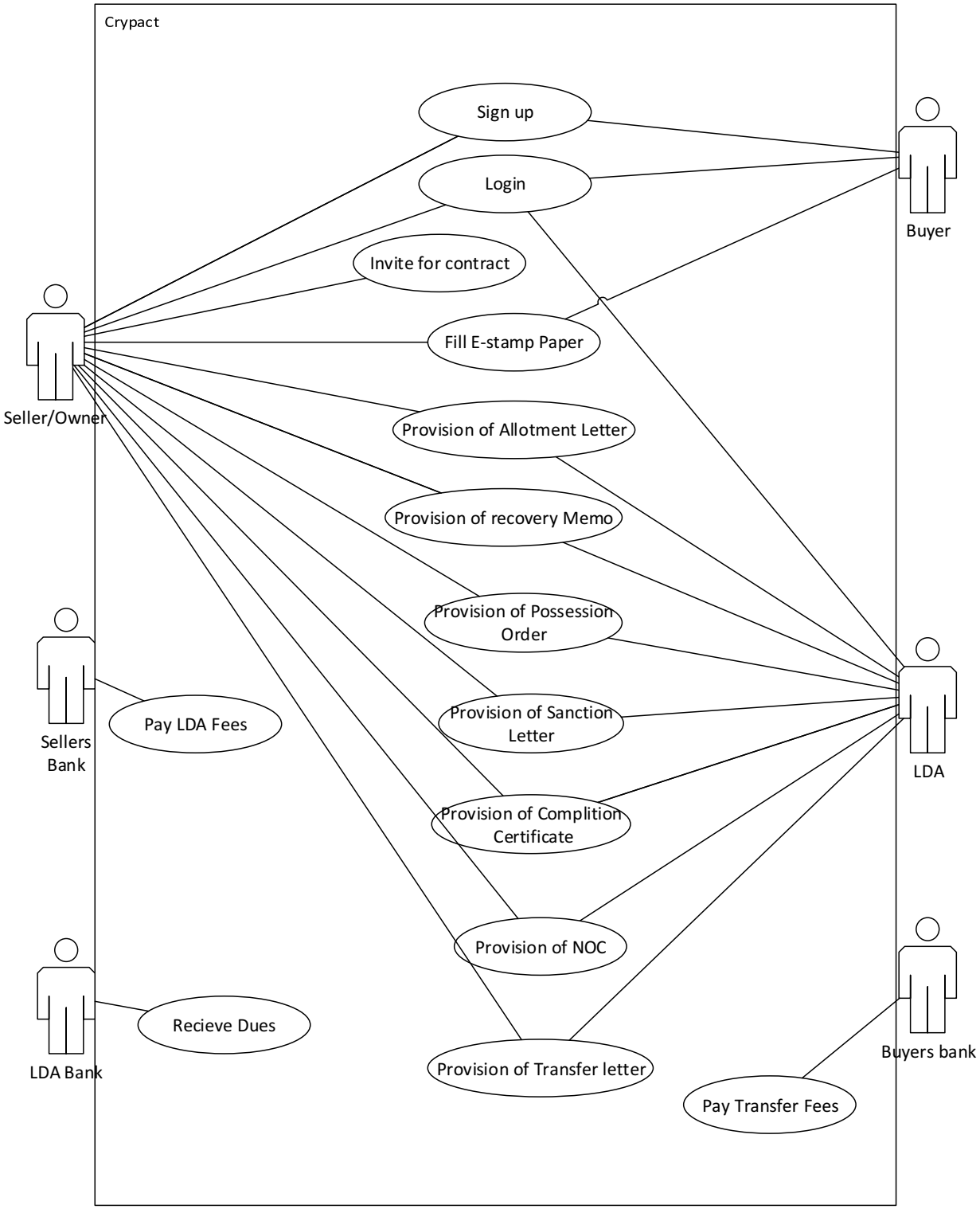
2.5 Assumptions and Dependencies

Following factors are assumed:

- The user must know English.
- Payment gateways are supposed to be integrated and is out of scope of this system.
- Digital Identity is supposed to be done and is out of our project scope.
- The system is proposed to LDA.
- The system is covering "Allottee plots" only which is one category under LDA plot schemes.
- The system is providing ownership transfer to one and one party only. No mechanism is provided which can handle ownership of more than one person for one single property.
- Ownership changed on behalf of Inheritance is supposed to be done and no mechanism is provided by our system to cop with such problem and is out of scope.
- A user should have basic knowledge of process for sale and purchase of property and change of property ownership.

Chapter 3. System Requirements

3.1 Functional Requirements



3.1.1 Sign up

Identifier		Sign up
Purpose		To Use Crypact System
Priority		High
Pre-conditions		Webpage Loaded Successfully
Post-conditions		User Signed up successfully
Typical Course of Action		
S#	**Actor Action**	**System Response**
1	Enter Signup Details	
2		Display Success message on screen
3		Send Verification Email
…	Verify Email.	
Alternate Course of Action		
S#	**Actor Action**	**System Response**
1-A	Enter Wrong Information	
2		Display Error Message
3		
…		

3.1.2 Login

Identifier		Login
Purpose		To buy or sell some property
Priority		High
Pre-conditions		Webpage Loaded Successfully
Post-conditions		Users Home page loaded Successfully
Typical Course of Action		
S#	**Actor Action**	**System Response**
1	Enter Login Information	
2		Display Home Page of user
Alternate Course of Action		
S#	**Actor Action**	**System Response**
1-A	Enter Wrong Information	
2		Display Error Message

3.1.3. Invite for Contract

Identifier	Invite for Contract	
Purpose	To invite a buyer and sell property.	
Priority	High	
Pre-conditions	Seller/owner is logged in.	
Post-conditions	Invitation is sent.	
Typical Course of Action		
S#	**Actor Action**	**System Response**
1	Seller clicks on the invite button.	
2		Display Invitation screen.
3	Searches the buys through his public key.	
…		Shows the matching public key holders.
	Seller selects a user and sends the request.	
Alternate Course of Action		
S#	**Actor Action**	**System Response**
1-A	Seller clicks on the invite button.	
2		Display Invitation screen.
3	Searches the buys through his public key.	
…		No match found.

3.1.4. Provision of Transfer Letter

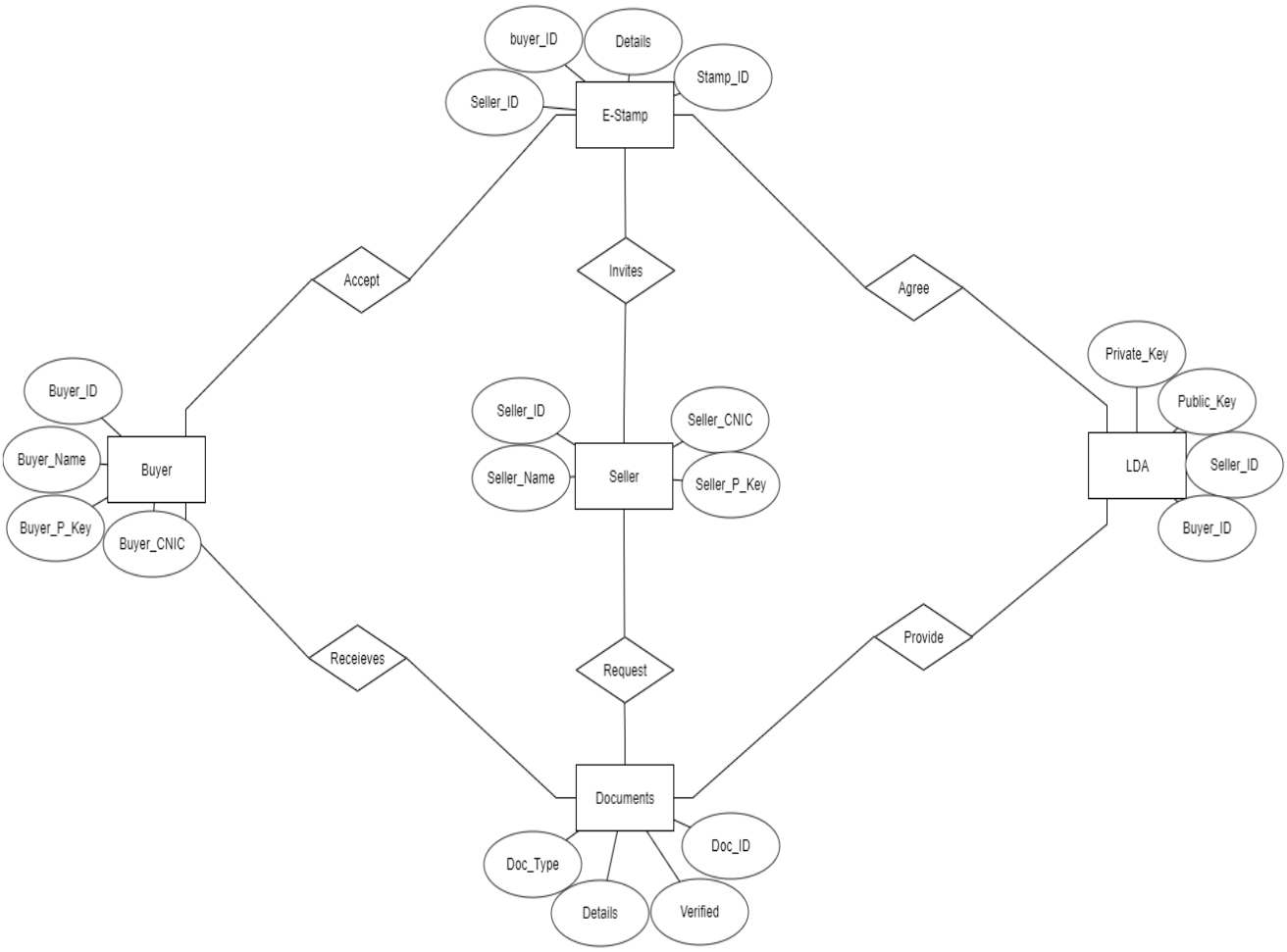
Identifier	Provision of Allotment Letter	
Purpose	To give ownership of land to the owner who had requested for this land.	
Priority	High	
Pre-conditions	Owner has paid the allotment fees.	
Post-conditions	Allotment letter is provided to the owner.	
Typical Course of Action		
S#	**Actor Action**	**System Response**
1	Seller/Owner requests Allotment letter.	
2		System forwards the request to LDA.
	LDA provides the allotment letter for a plot to owner through Crypact.	
Alternate Course of Action		
S#	**Actor Action**	**System Response**
1-A	Seller/Owner requests for a plot.	
2		System forwards the request to LDA.
	Allotment letter not provided due to	

3.1.5. Fill E-Stamp paper

Identifier	Fill request for transfer ownership.	
Purpose	Fill all the headings of smart contract	
Priority	High	
Pre-conditions	Both buyer and seller are logged in.	
Post-conditions	All headings of Smart Contract filled	
Typical Course of Action		
S#	**Actor Action**	**System Response**
1	Buyer files the categories related to him.	
2	Seller files the categories related to him.	
Alternate Course of Action		
S#	**Actor Action**	**System Response**
1-A	Buyer files the categories related to him.	
2	Seller files the categories related to him.	

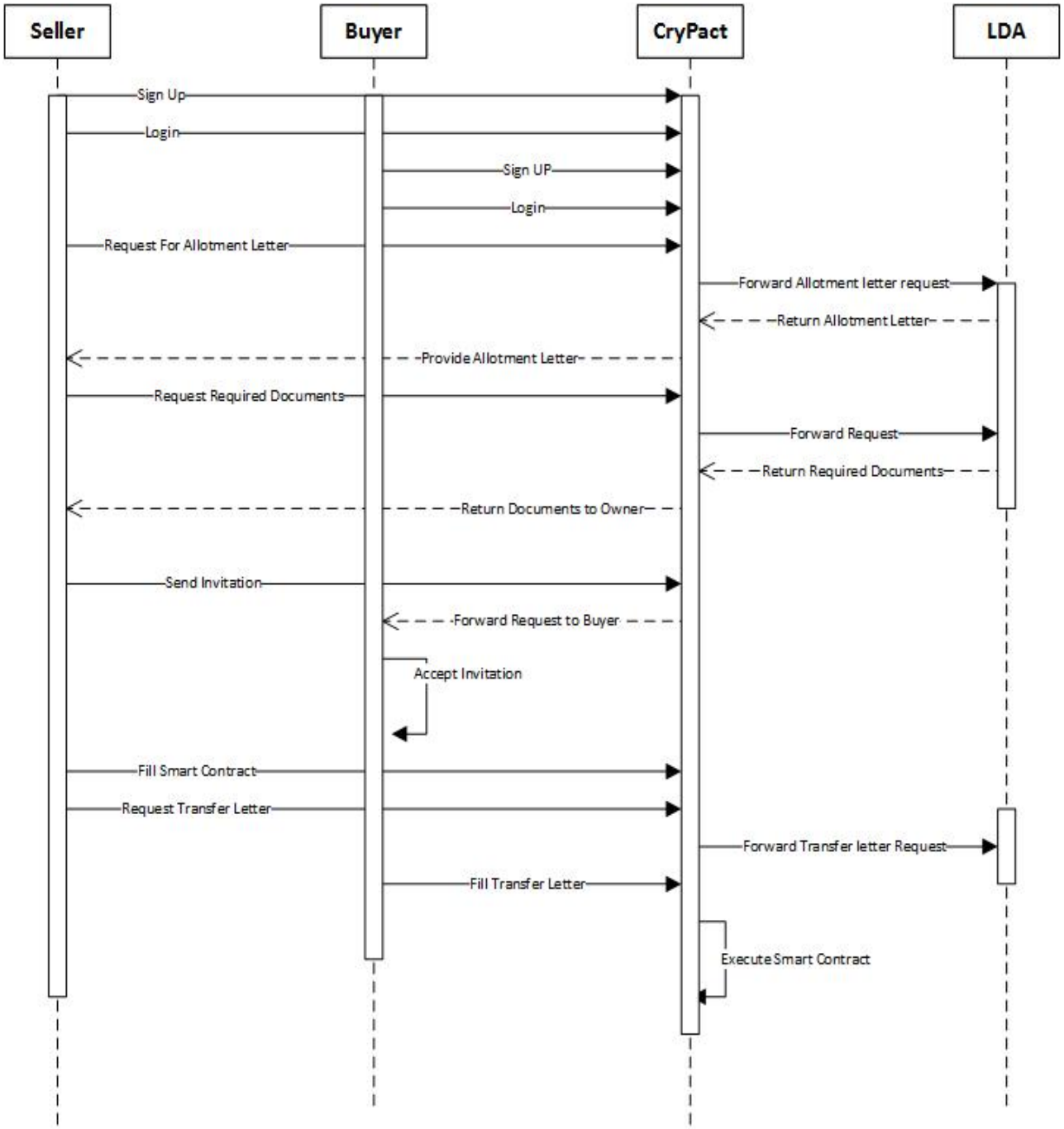
3.2. Requirements Analysis and Modeling

Entity Relationship Diagram:



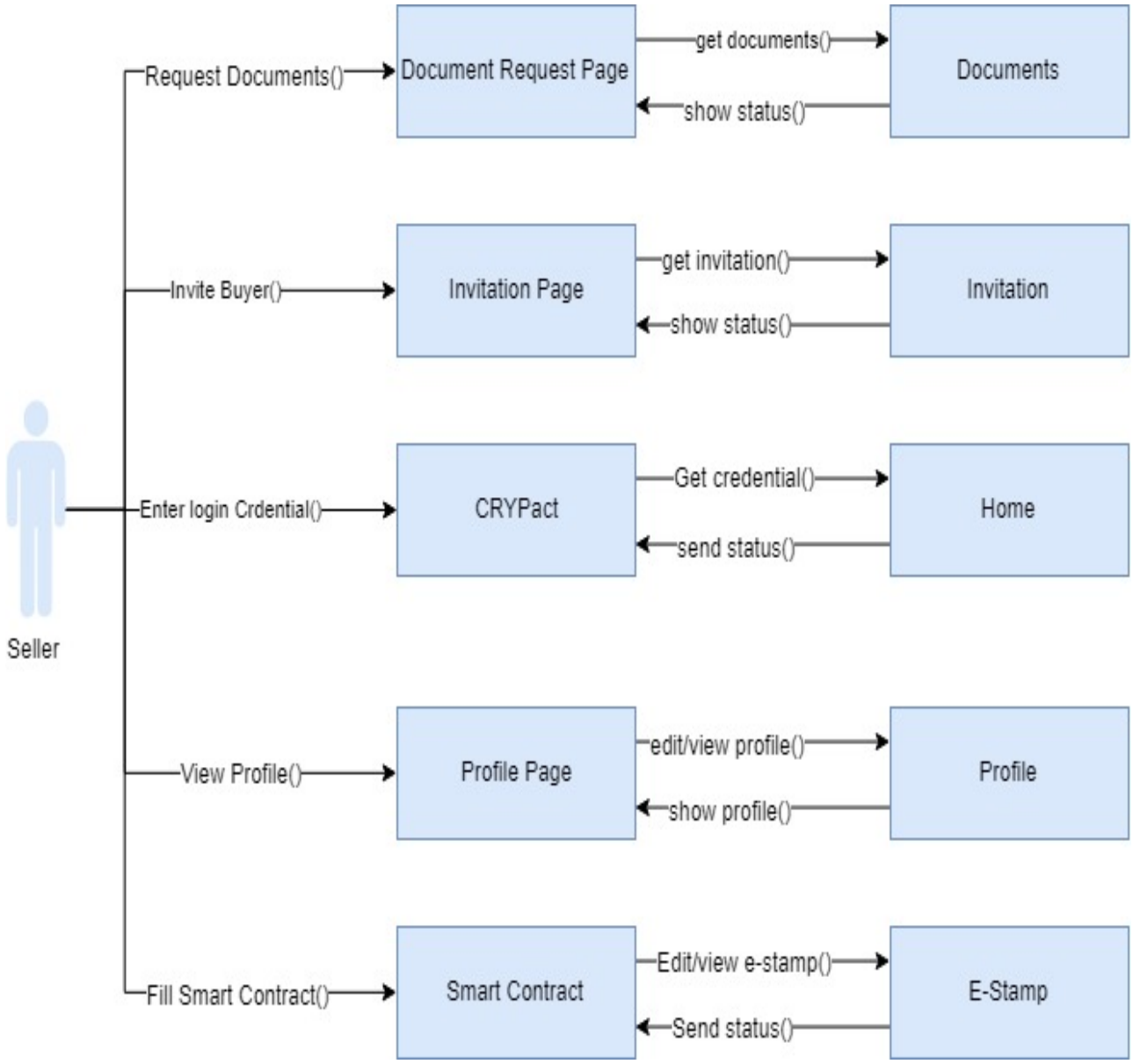
3.2.1. Component Interactions and Collaborations

3.1.3 Design level sequence diagram

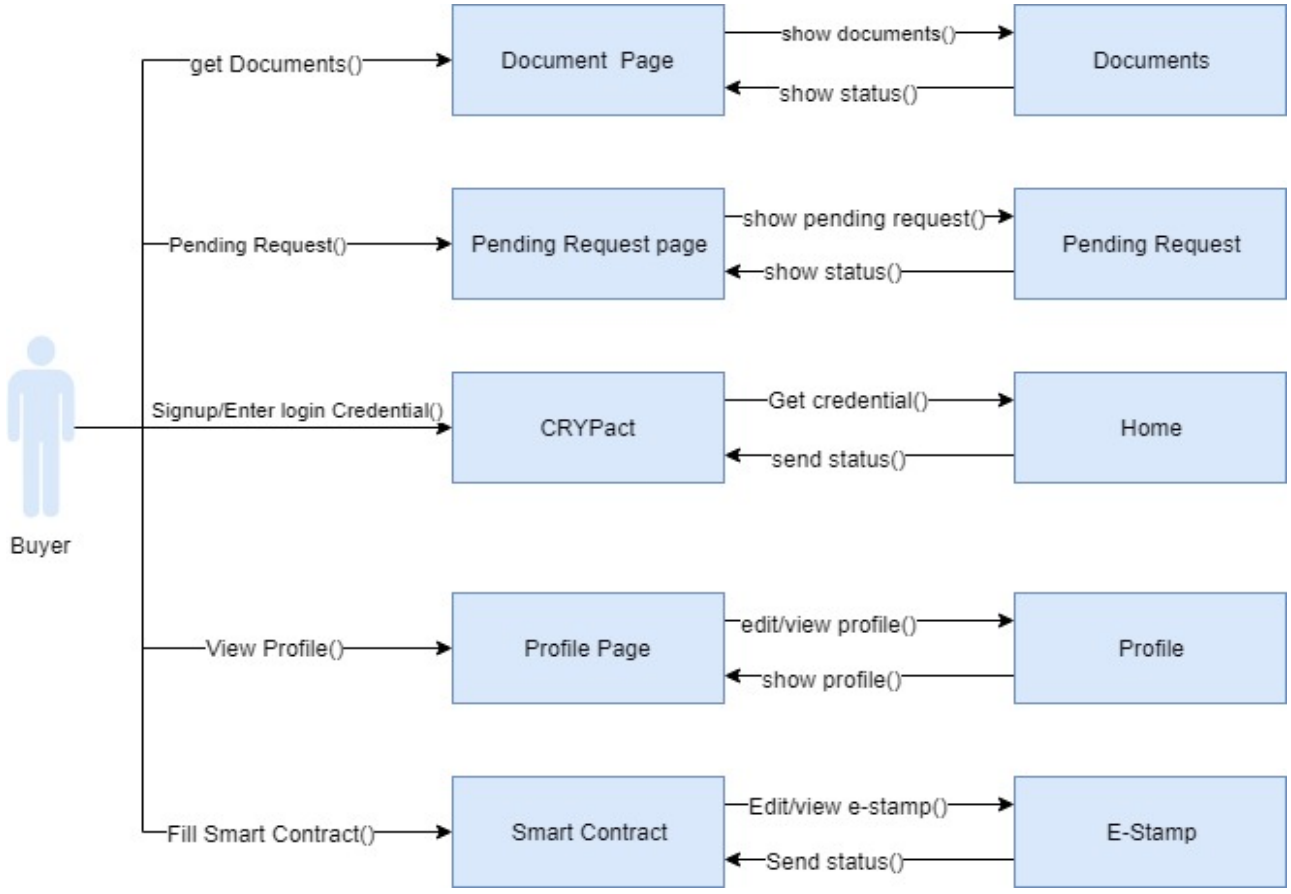


3.1.4 Collaboration Diagram:

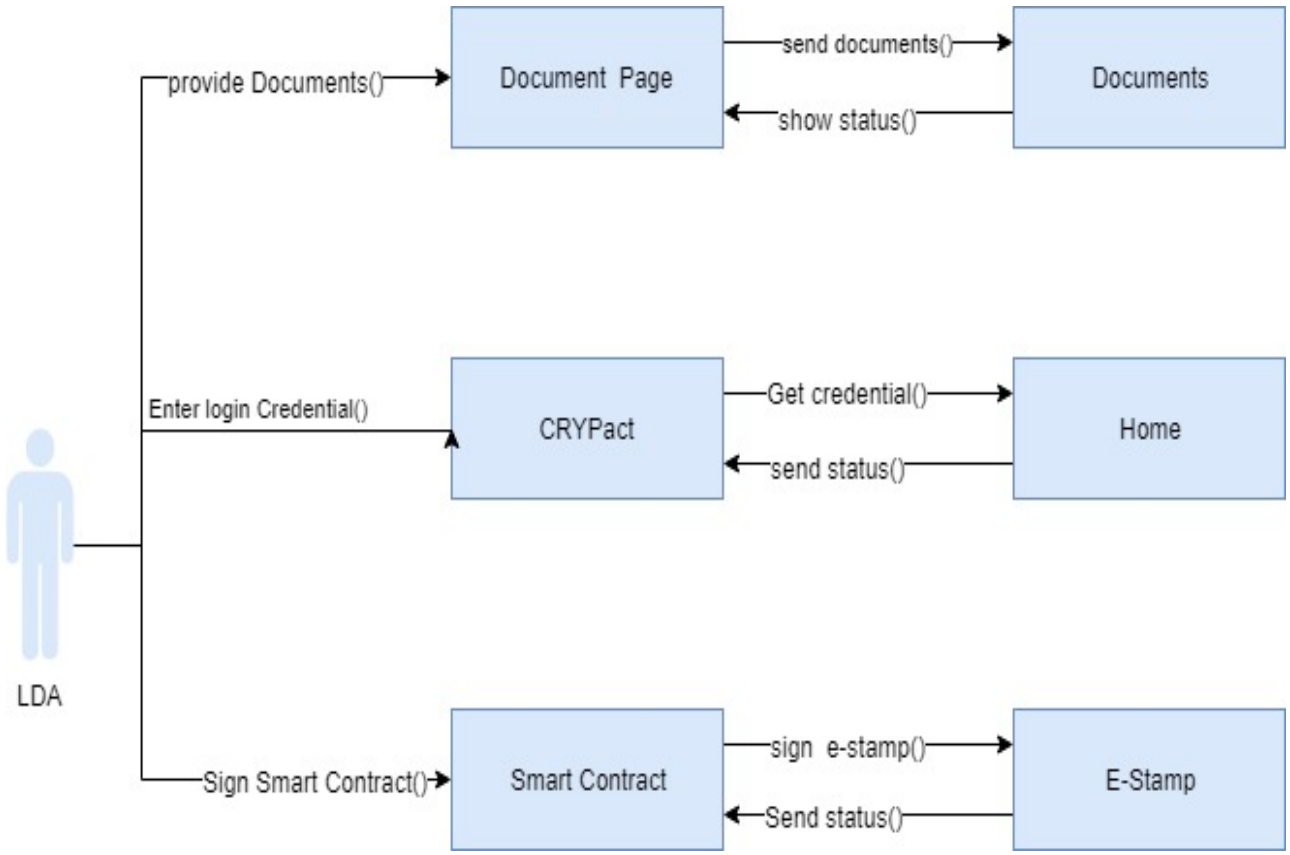
Seller:



Buyer:

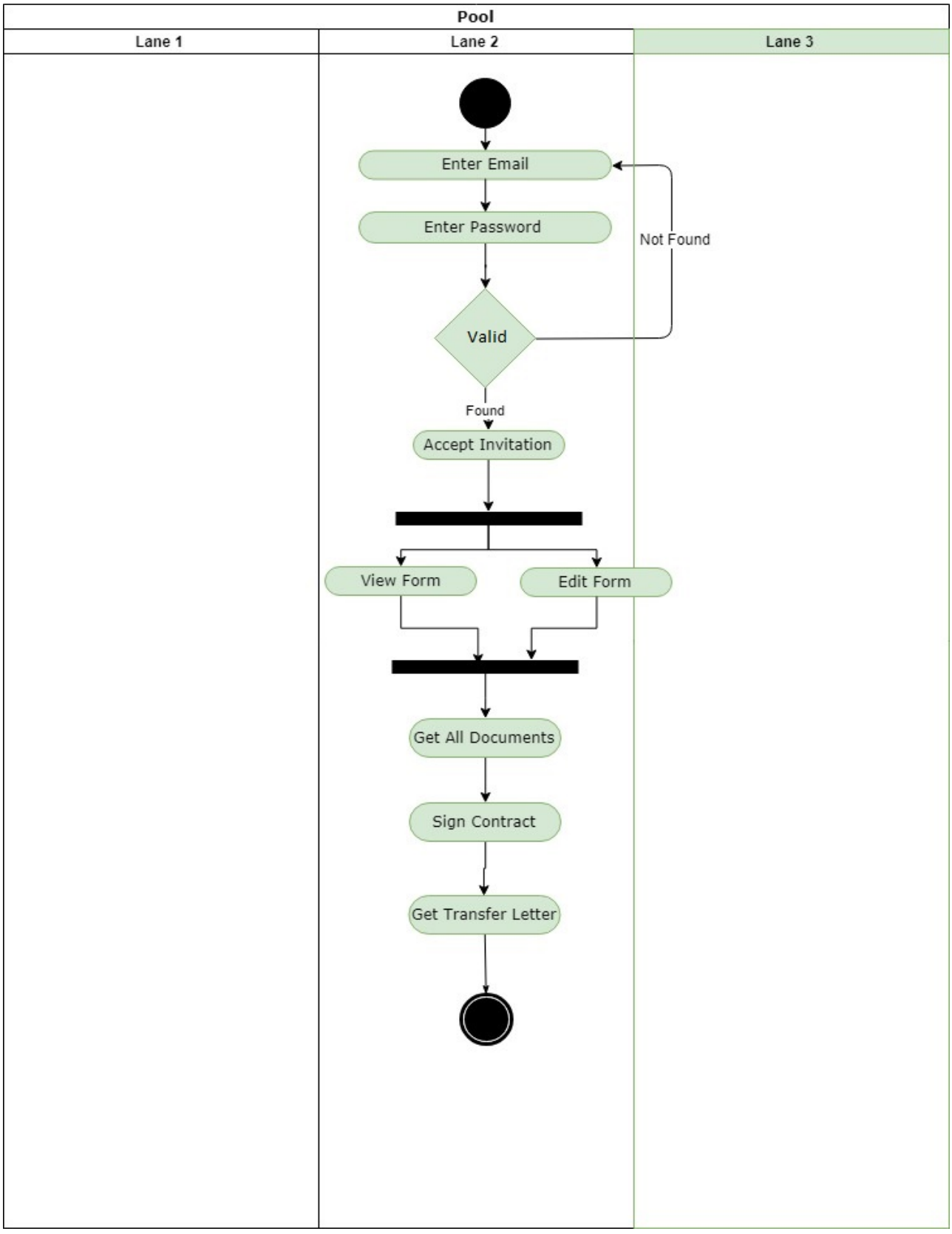

Buyer
Signup/Enter login Credential()
CRYPact
Get credential()
send status()
Home
get Documents()
Document Page
show documents()
show status()
Documents
Pending Request()
Pending Request page
show pending request()
show status()
Pending Request
View Profile()
Profile Page
edit/view profile()
show profile()
Profile
Fill Smart Contract()
Smart Contract
Edit/view e-stamp()
Send status()
E-Stamp

LDA:

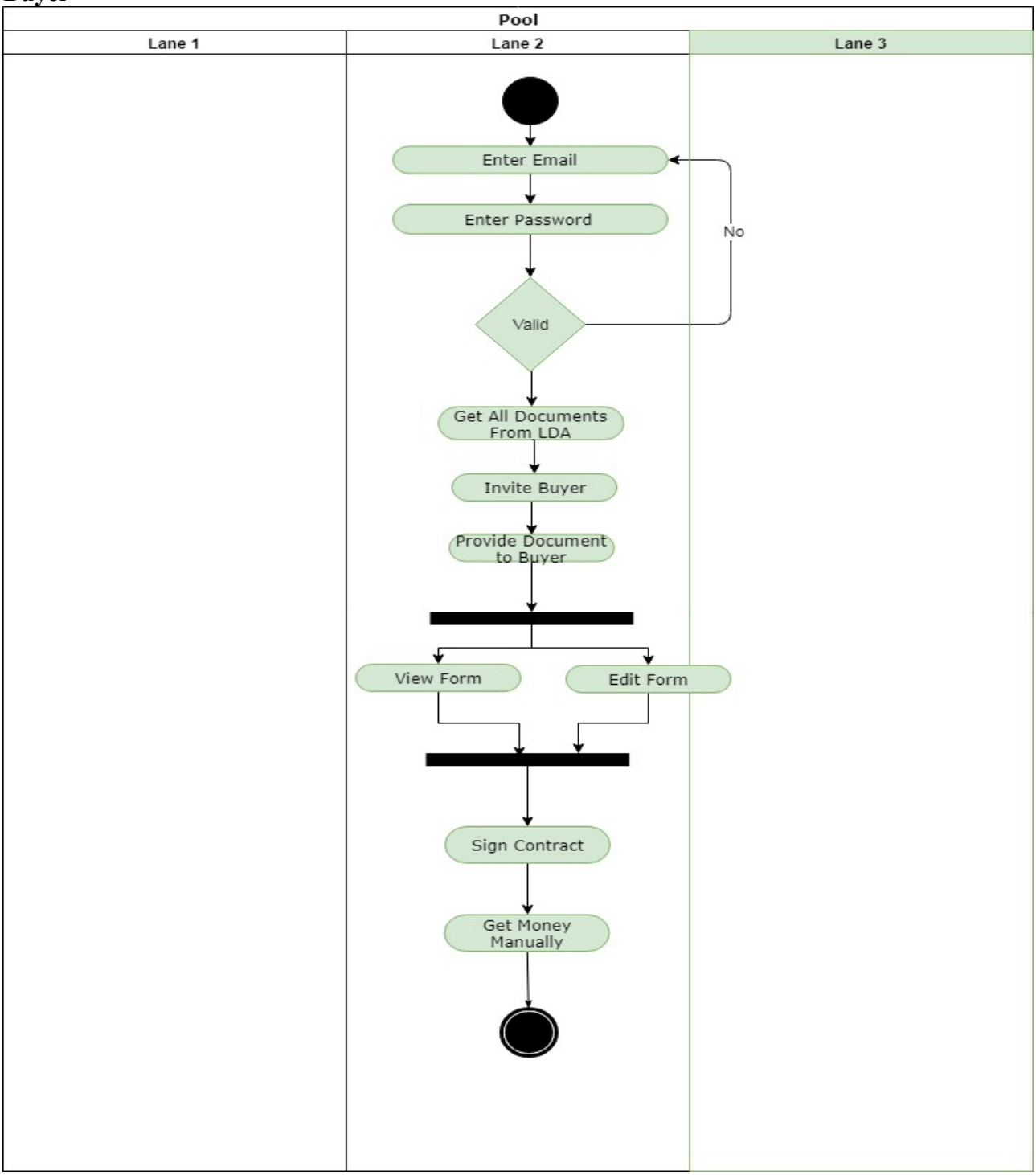

provide Documents()
Document Page
send documents()
Documents
show status()
Enter login Credential()
CRYPact
Get credential()
Home
send status()
LDA
Sign Smart Contract()
Smart Contract
sign e-stamp()
E-Stamp
Send status()

Activity diagrams

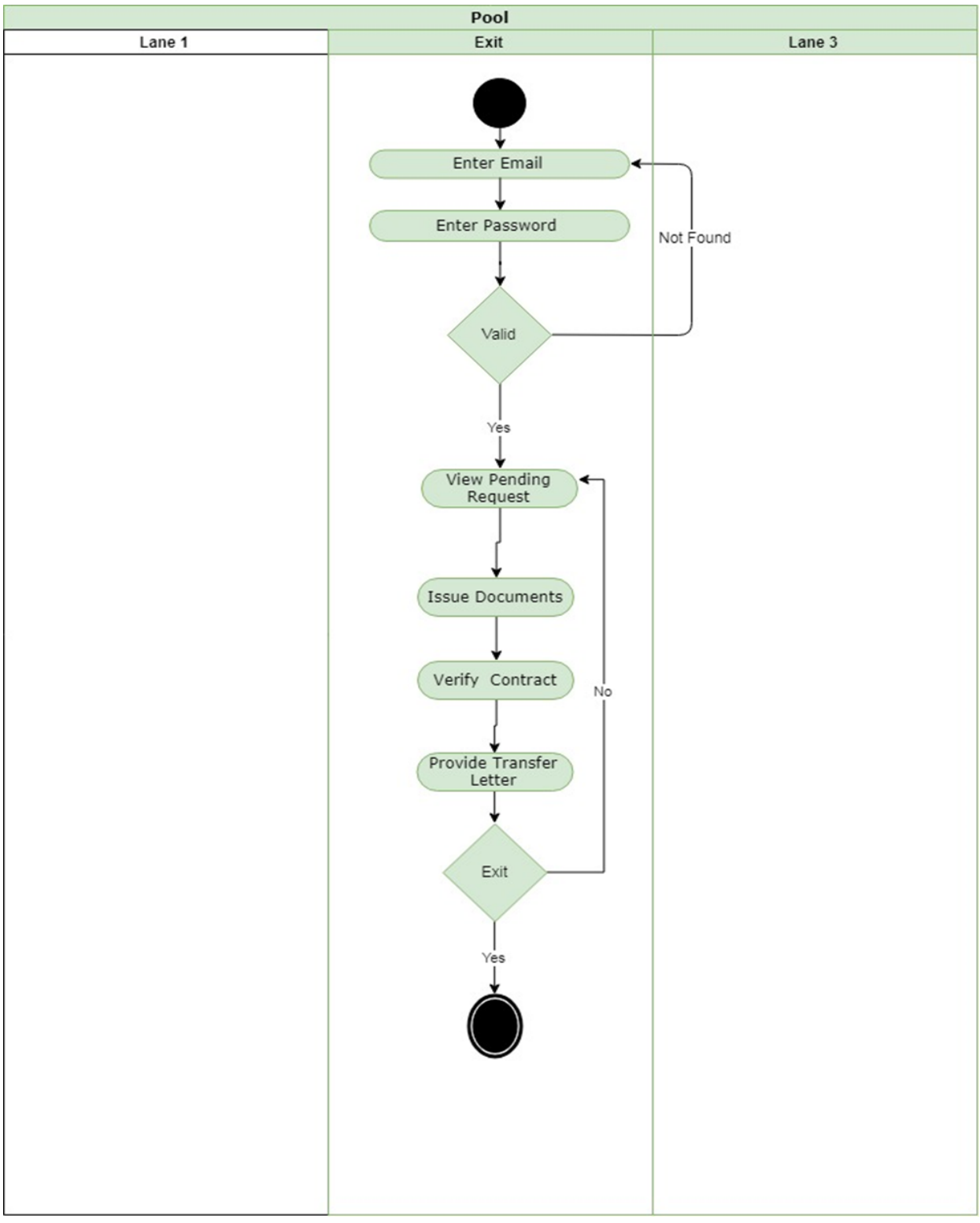
Seller



Buyer


Pool
Lane 1
Lane 2
Lane 3
Enter Email
Enter Password
No
Valid
Get All Documents From LDA
Invite Buyer
Provide Document to Buyer
View Form
Edit Form
Sign Contract
Get Money Manually

LDA:


Pool
Lane 1
Exit
Lane 3
Enter Email
Enter Password
Not Found
Valid
Yes
View Pending Request
Issue Documents
Verify Contract
No
Provide Transfer Letter
Exit
Yes

3.3. Nonfunctional Requirements

3.3.3. Performance Requirements

The users must fill up the e-stamp papers, bank information and transfer letter with proper format and correct information according to the opposed respective rules. These are the requirements that are must in order to have the system working properly. Users must know the process of transferring the ownership under the laws and rules of LDA.

3.3.4. Safety Requirements

The system must be incorporated with robust and verified database server which provides reliable performance and integrity of data. Identities must be verified through “NADRA” database which is duty of LDA. System must be kept in check as there are possibilities of certain attacks on smart contracts. If any unknown and unexpected inconvenience occurs or even when the operation of a user is not successful, no fees shall be deducted and all process must be reverted and users will start over again.

3.3.5. Security Requirements

Sensitive information is entered into the blockchain such as Personal information about users and their profiles & the data about various properties. All private keys must be generated through CRYPact system randomly through cryptography and cryptosystem. All information must go through encryption and decryption so that all data remains safe and secure. Mismanagement of information might cause participant dissatisfaction that will eventually lead to loss, only because of mistakes on giving information.
To protect the confidential data from hackers or unknown users the technology is implemented with strict user assess criteria. System is facilitated with digital identity and complete 2FA in order to keep it save from unauthorized other users. Operational rights must be designed that a particular user can only have approach to specific tasks.

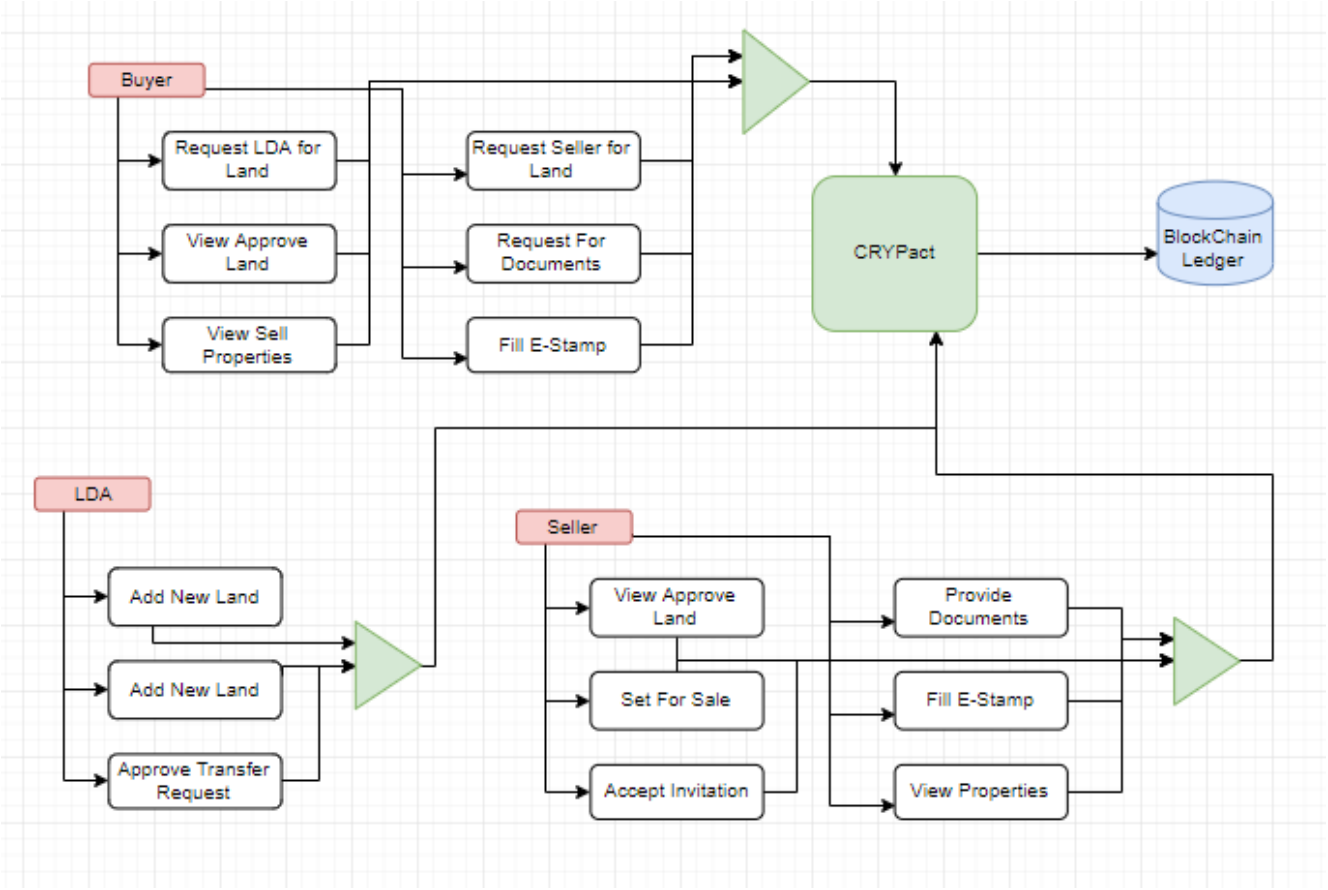
3.3.6. Additional Software Quality Attributes

- The reliability of system is high.
- Usability is high due to interactive interface design.
- Database is more secure and reliable due to Blockchain technology.
- System is transparent.

3.4. Other Requirements

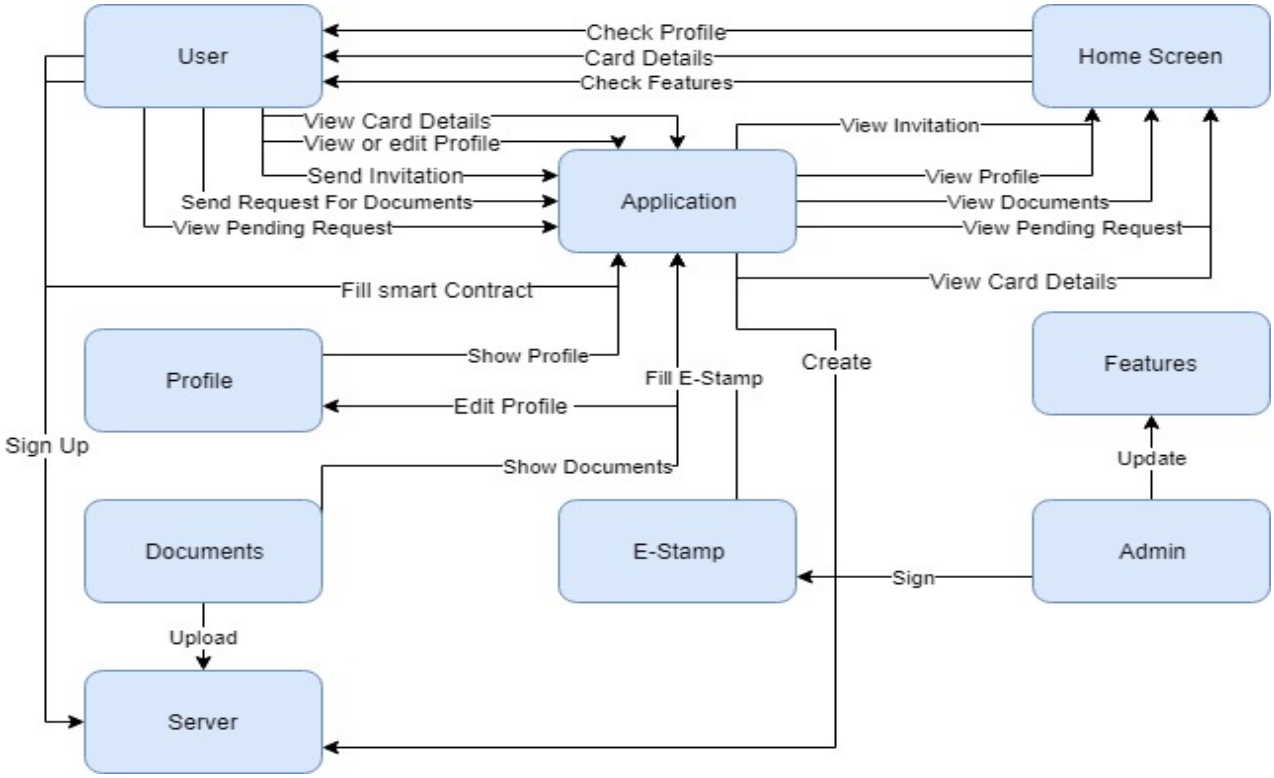
Might also be using server for local database to store login credentials, different API’s for data communication, encryption and decryption of system and some other stuff according to the needs of project based on research.

4. Technical Architecture

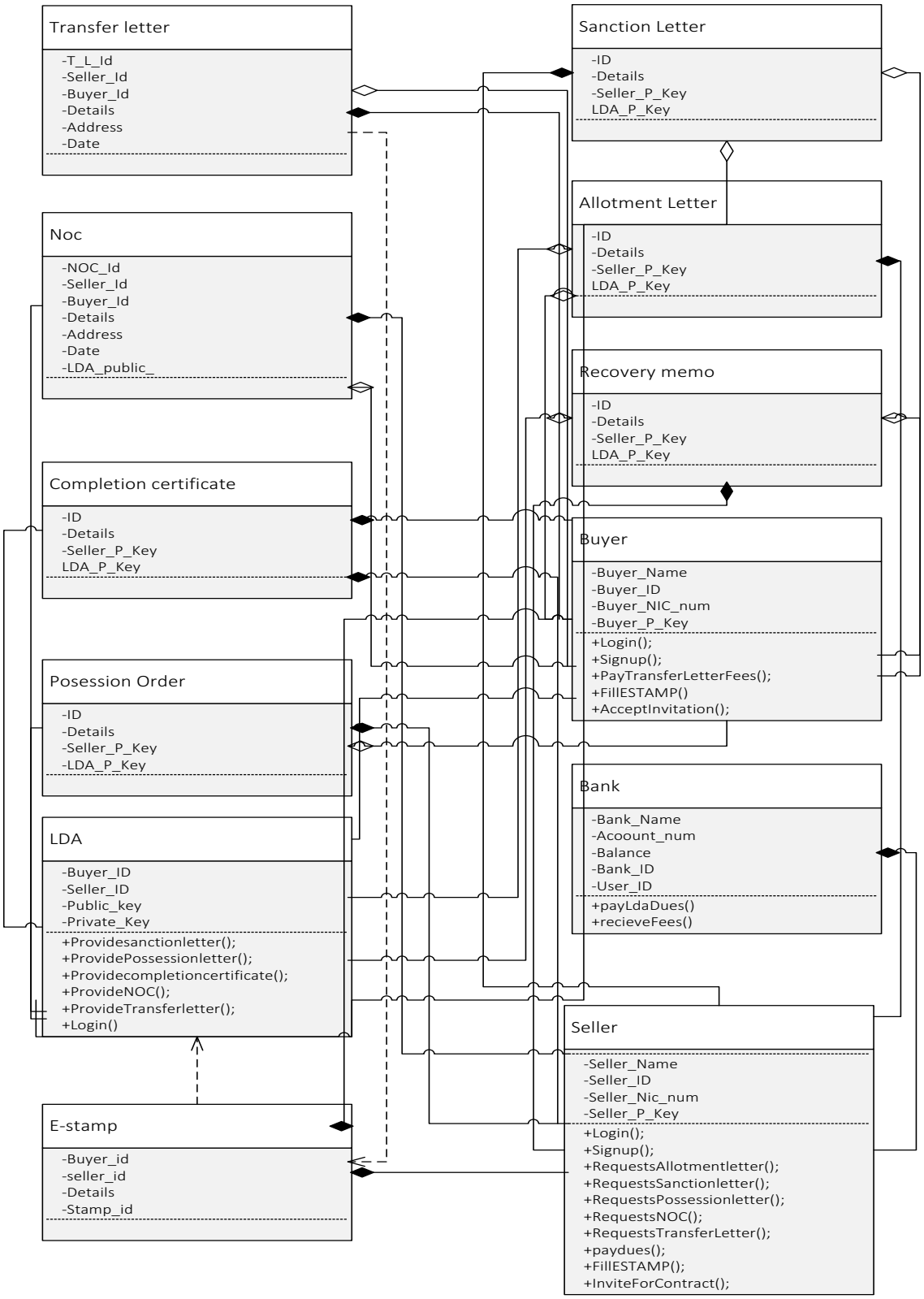


4.1. Application and Data Architecture

4.1.1. Component Diagram:

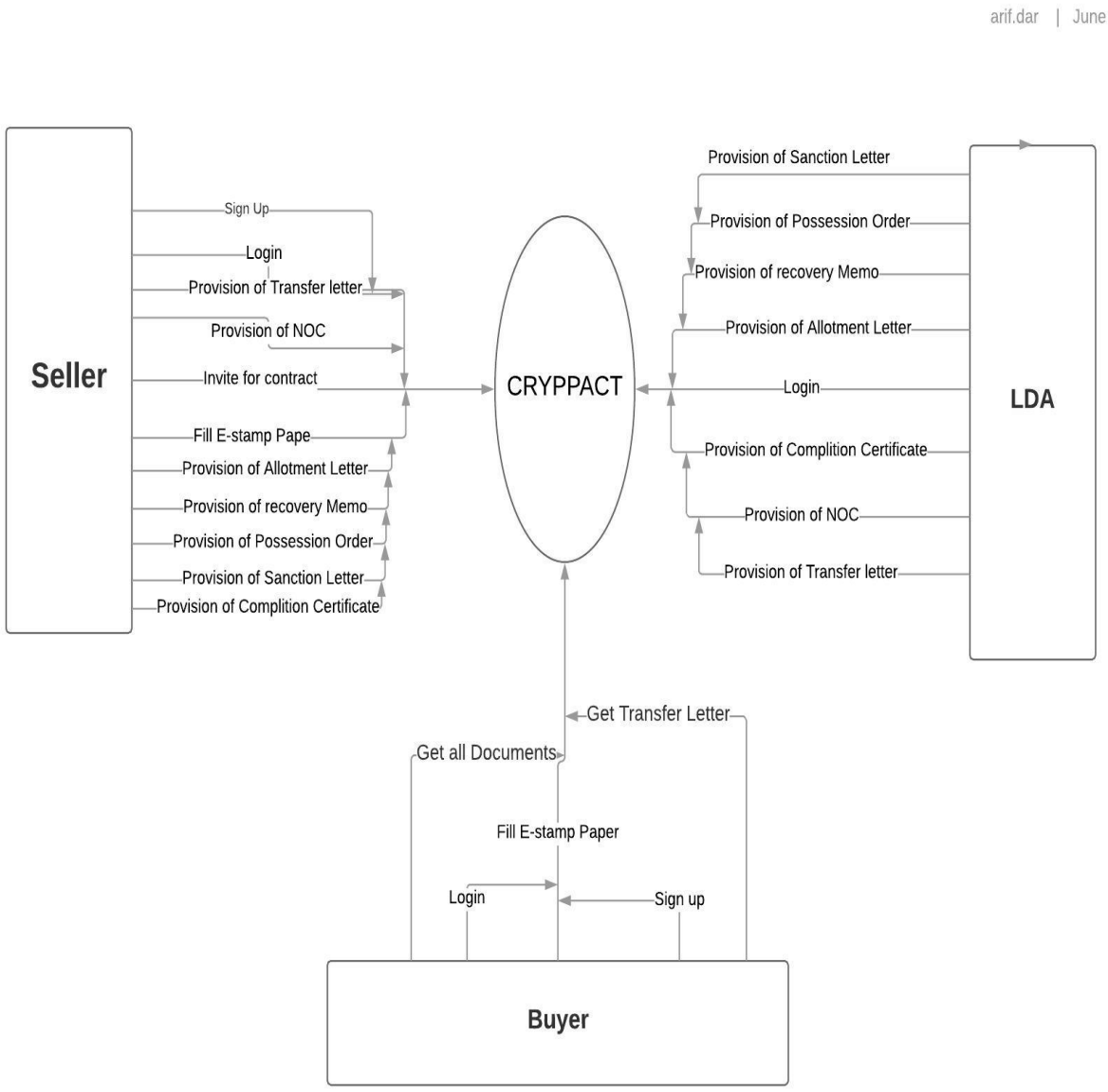


4.1.2. Class Diagram:

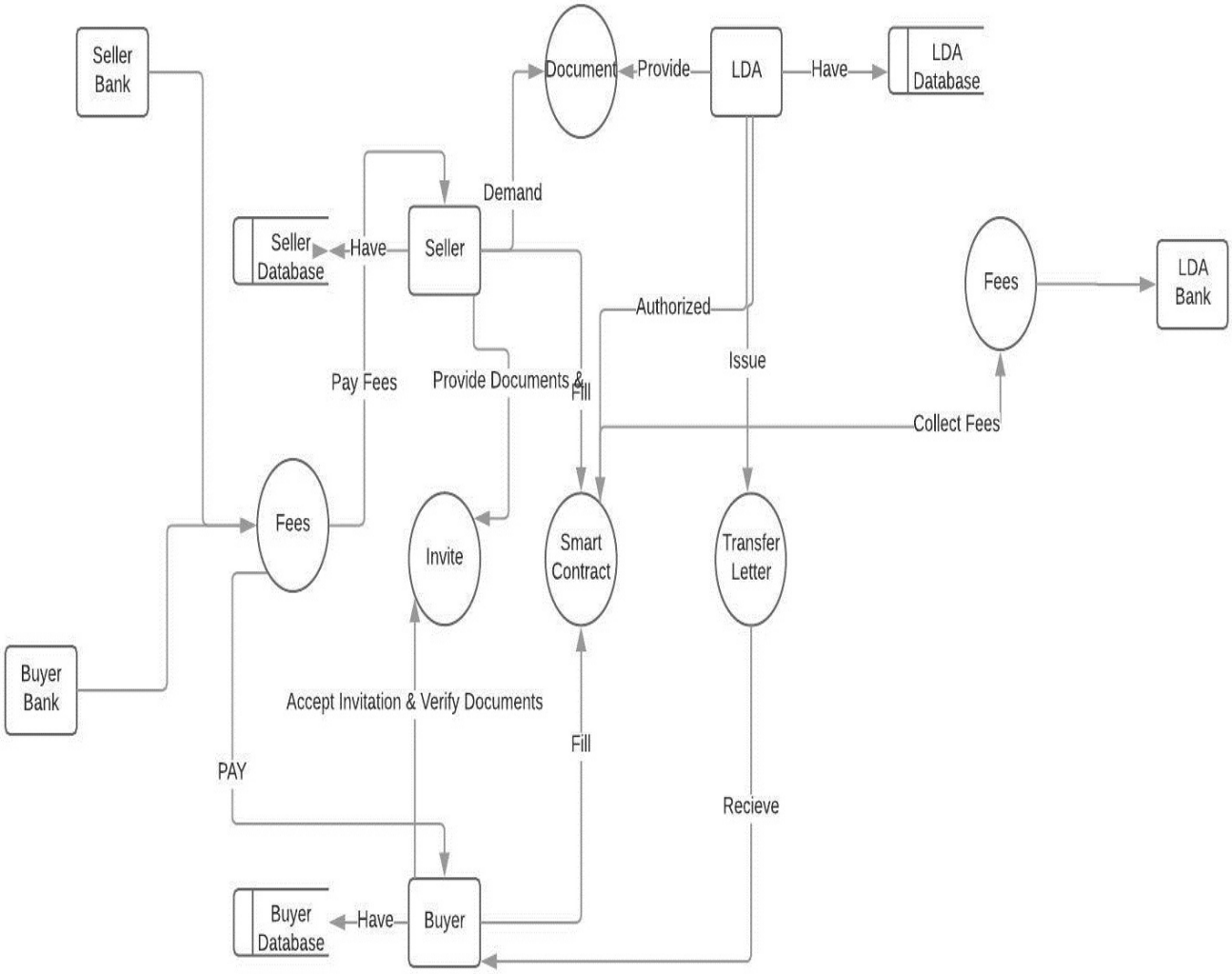


4.1.3. Data Flow Diagram:

Context Diagram (Level 0)



Context Diagram (Level 0)



4.2. Technology Architecture

- CRYPact website will be on a hosting server.
- The Front-End is written in HTML, CSS, Bootstrap and mainly in React JS.
- Backend will be written in MEAN stack mainly in Node JS to create and run Blockchain which will act as distributed ledger.
- There may also be central database to save login credentials and some other stuff.
- TCP/IP as the IPS for Client-Server Communication.

4.3. Architecture Evaluation

The front end of the system is implemented using a very popular Javascript framework known as React js. Developers using other frameworks have the challenge of having to rework on most codes even when crafting components that changed frequently. What they wanted was a framework that could allow them to break down complex components and reuse the codes to complete their projects faster.

ReactJS provided the solution that developers were looking for. It uses JSX (a unique syntax that allows HTML quotes as well as HTML tag syntax application for rendering specific subcomponents) This is very helpful in promoting construction of machine-readable codes and at the same time compounding components into a single-time verifiable file.

The backend of the project is implemented in Blockchain which helps keeping everything decentralized and distributed. There are a few key benefits to decentralizing things instead of keeping it in a centralized server/database:

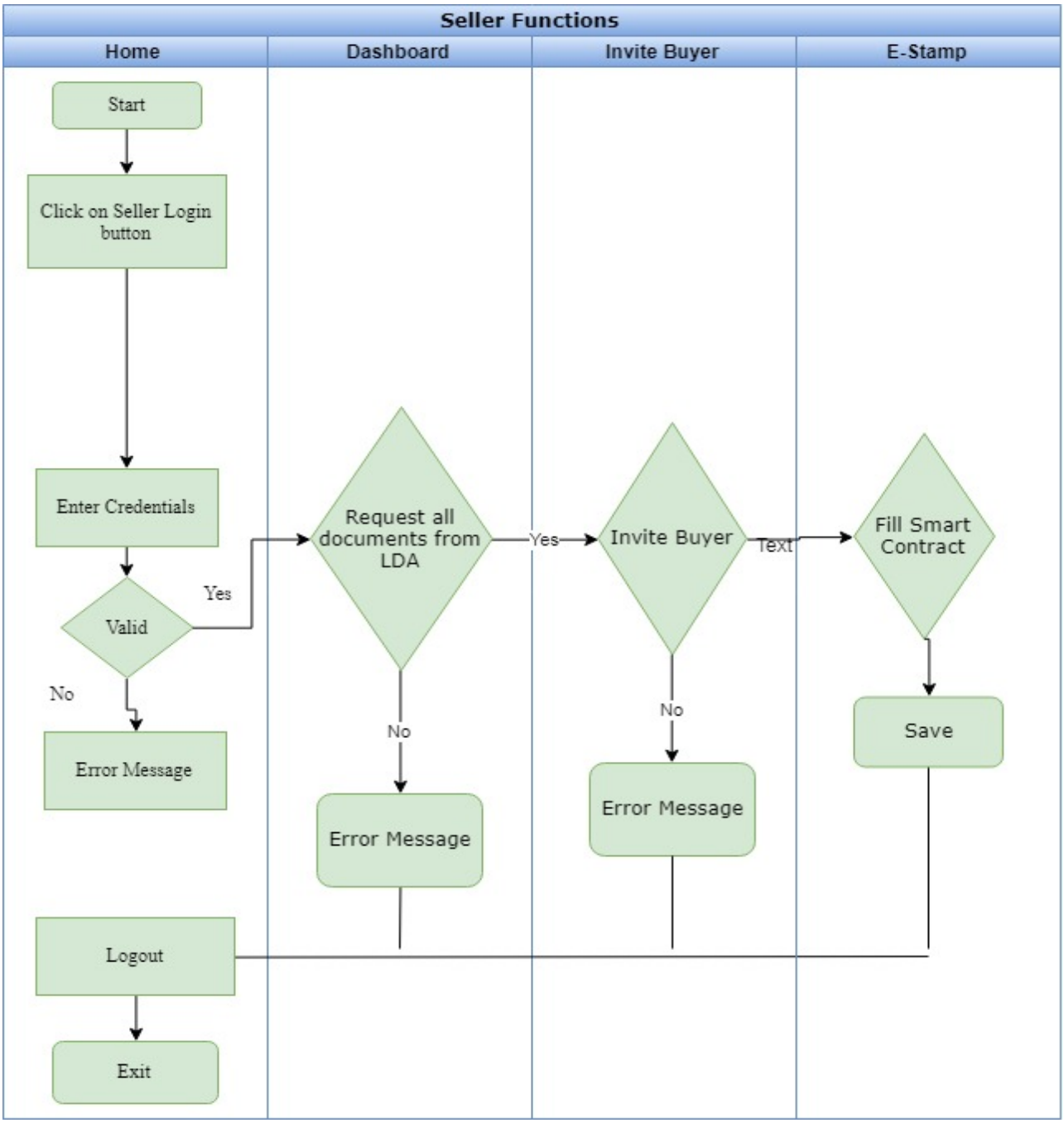
- Immutability
- Security
- Redundancy
- Overhead/cost reduction
- Accountability/transparency

5. Detailed Design and Implementation

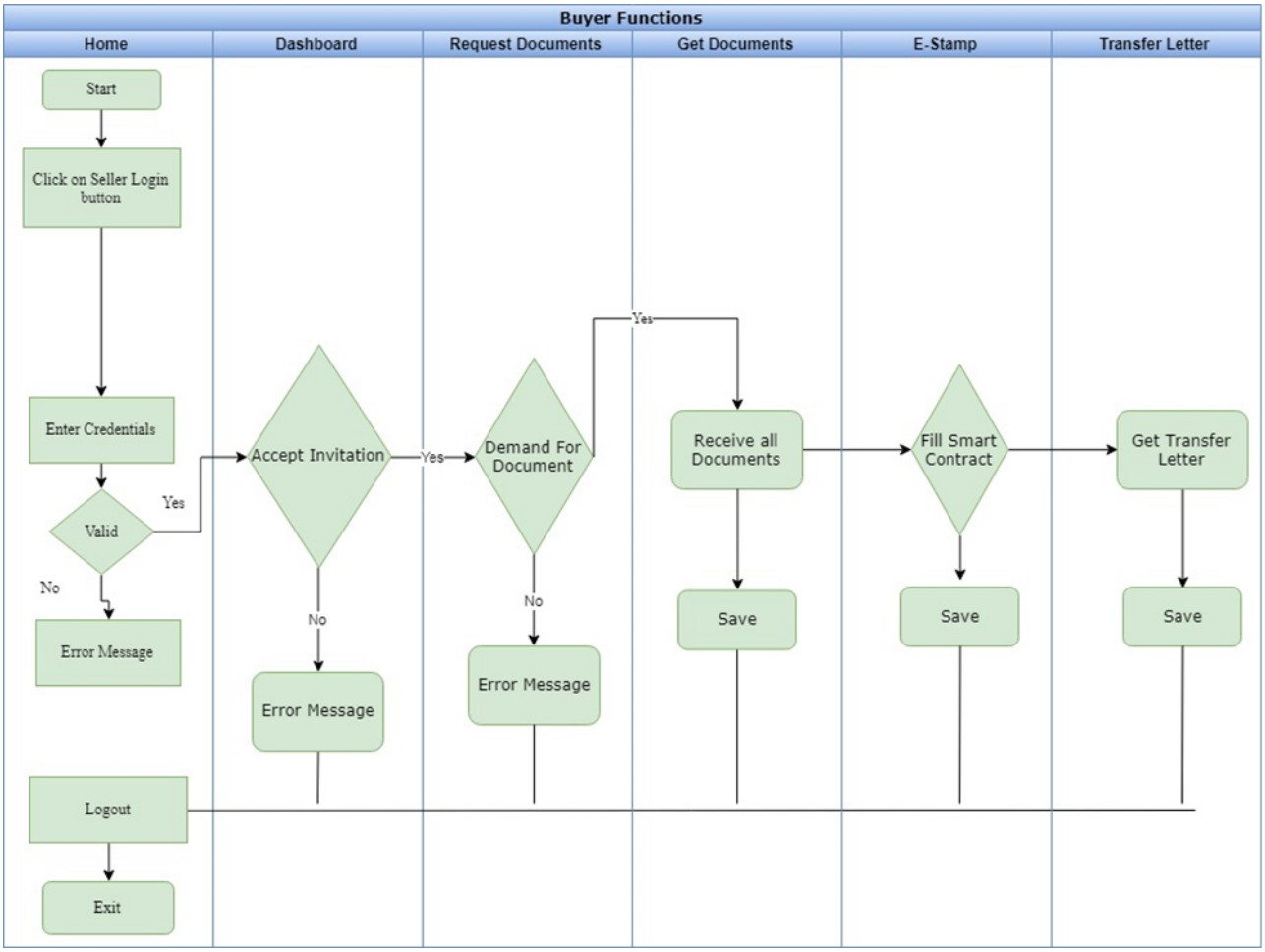
5.1. Screenshots/Prototype

5.1.3. Workflow

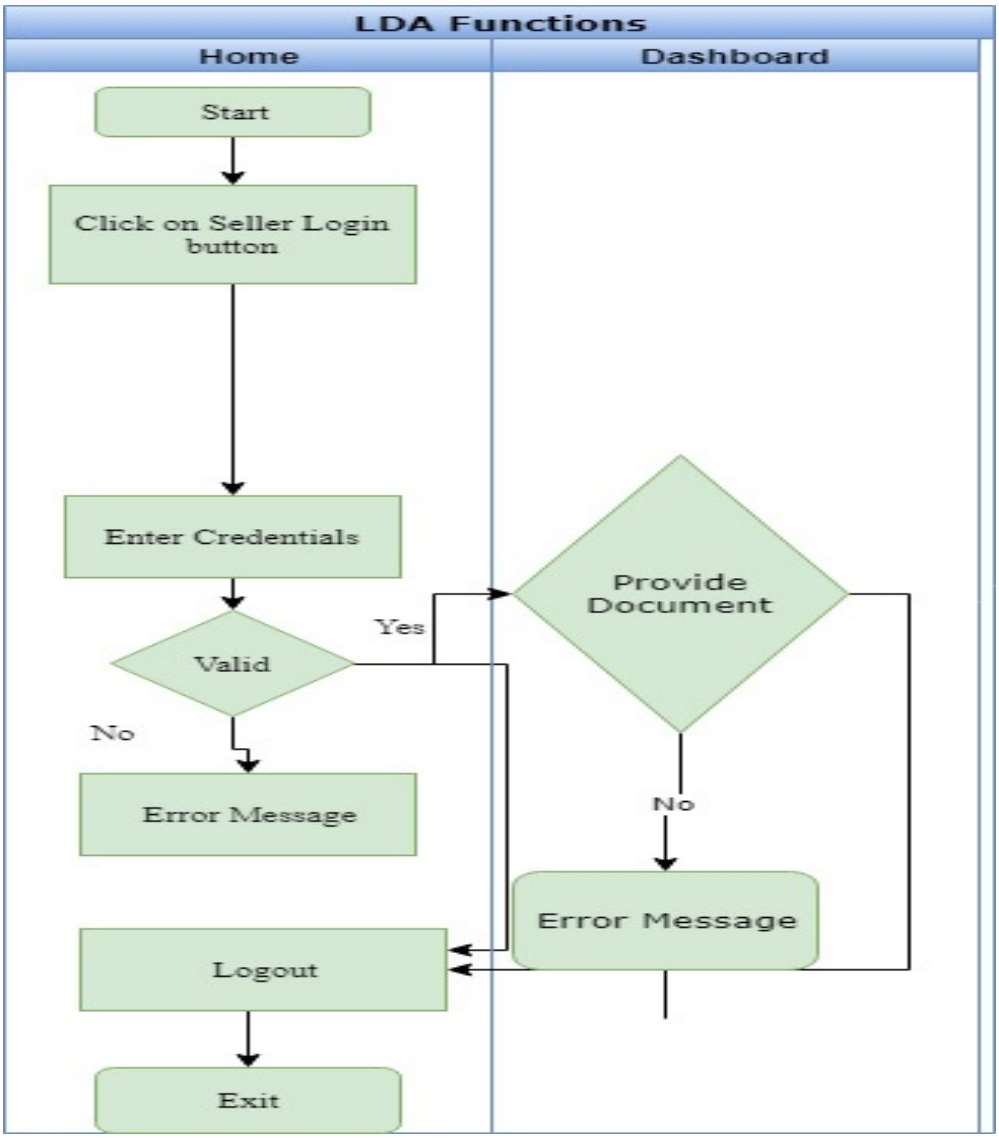
Seller



Buyer

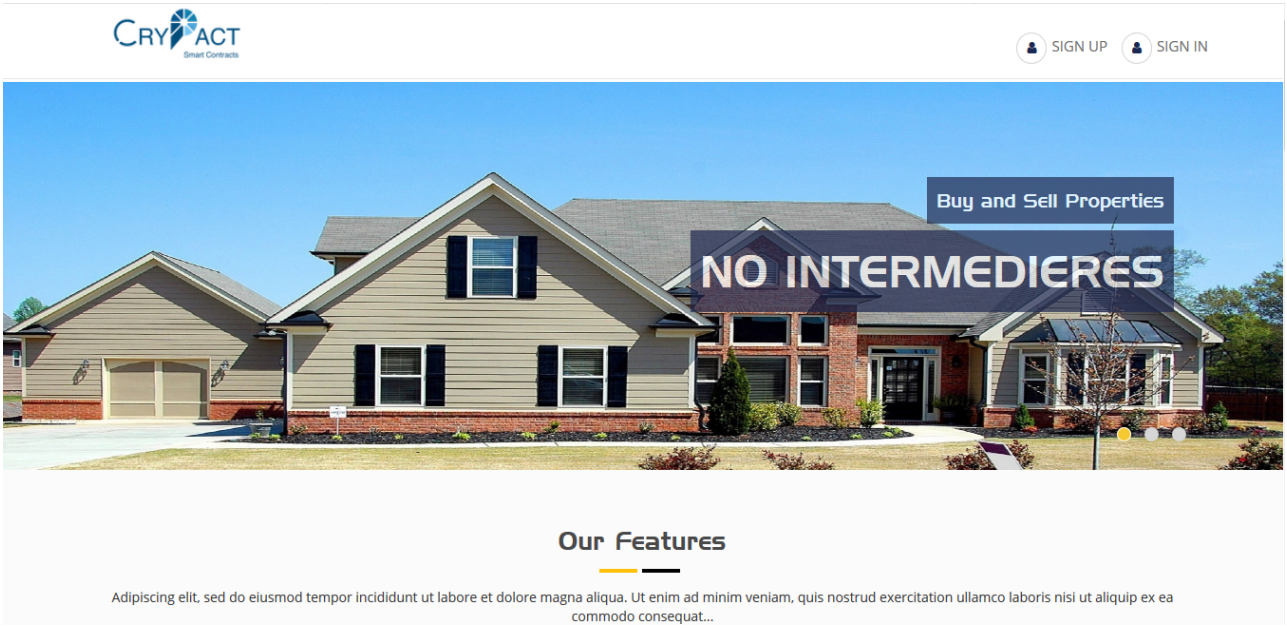

Buyer Functions
Home
Dashboard
Request Documents
Get Documents
E-Stamp
Transfer Letter
Start
Click on Seller Login button
Enter Credentials
Valid
Yes
No
Error Message
Logout
Exit
Accept Invitation
Yes
No
Error Message
Demand For Document
Yes
No
Error Message
Receive all Documents
Save
Fill Smart Contract
Save
Get Transfer Letter
Save

LDA

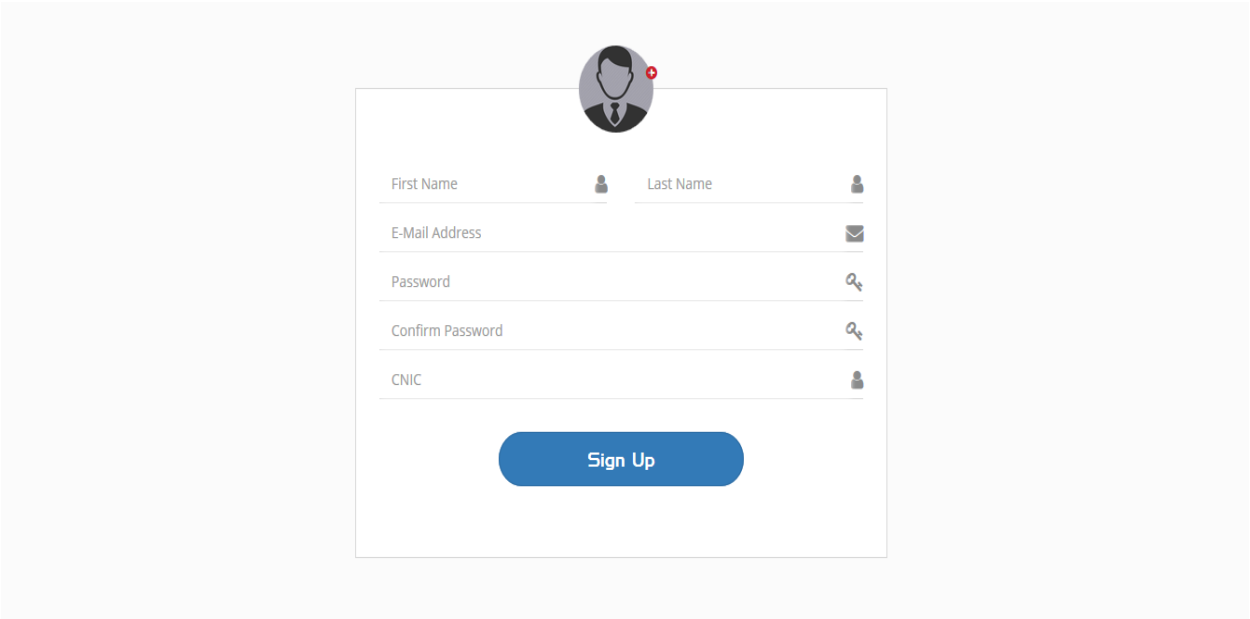

LDA Functions
Home
Dashboard
Start
Click on Seller Login button
Enter Credentials
Valid
Yes
No
Error Message
Provide Document
No
Error Message
Logout
Exit

5.1.4. Screens

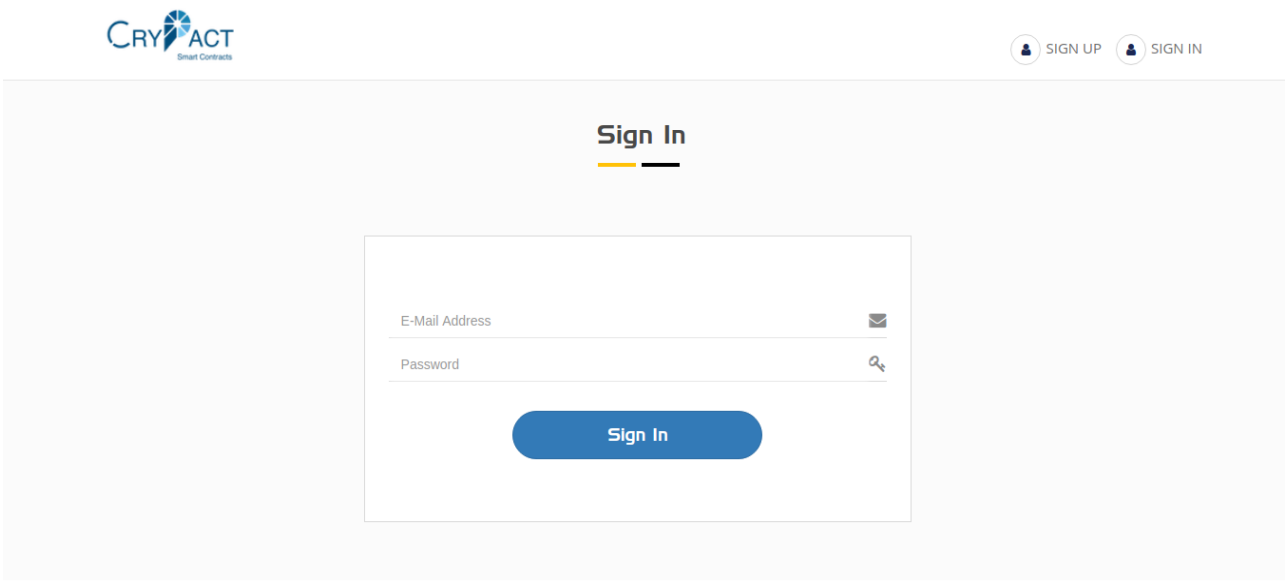
a. Home



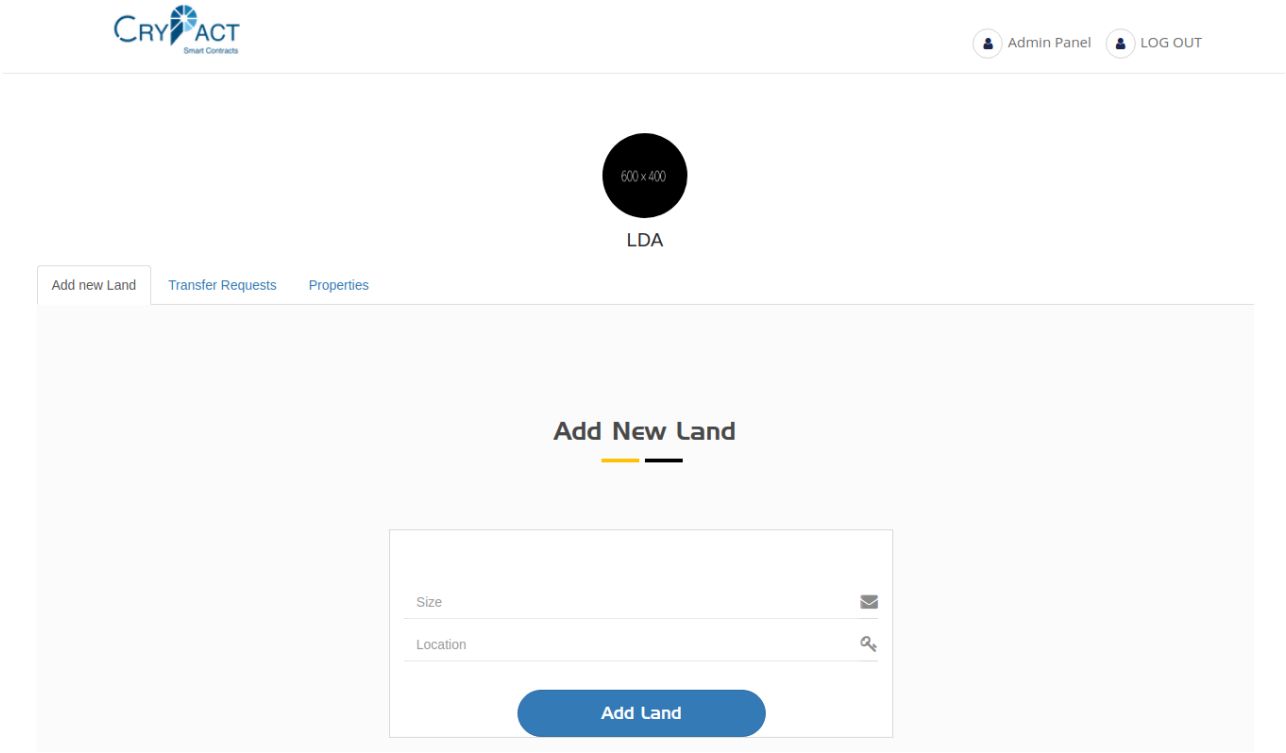
b. SignUp



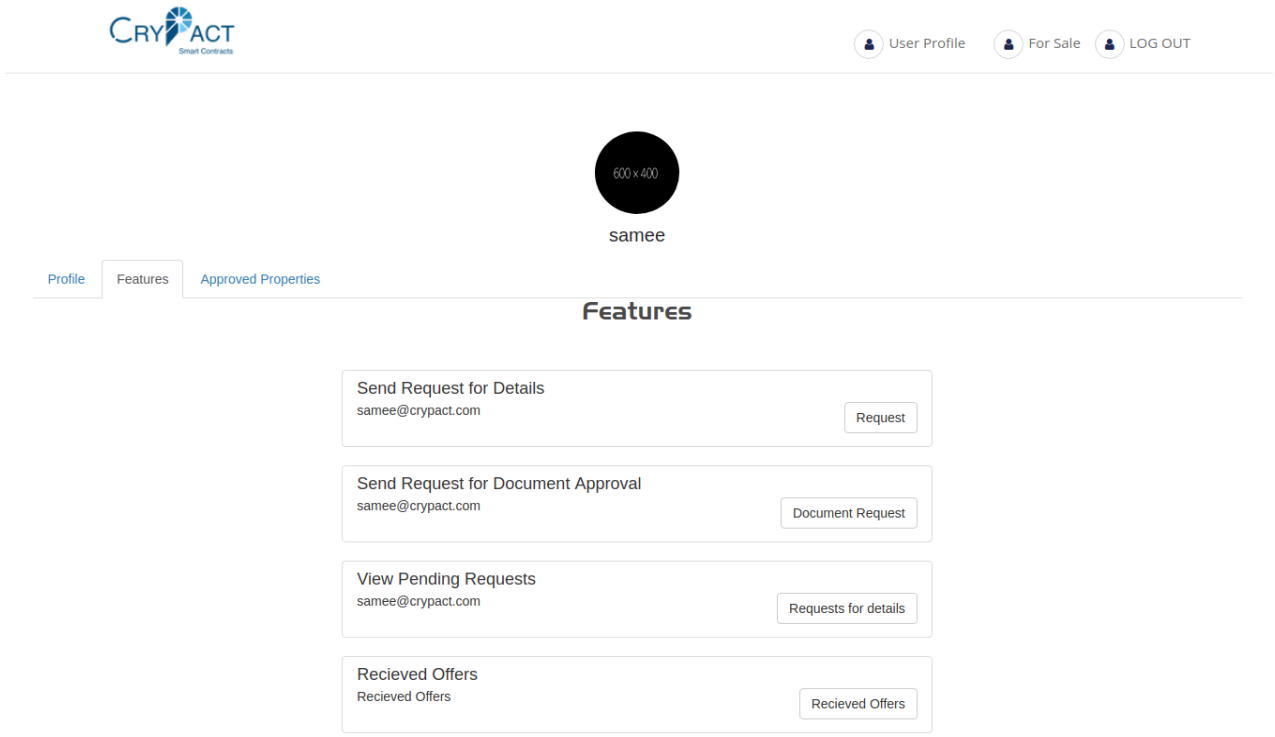
c. Sign in



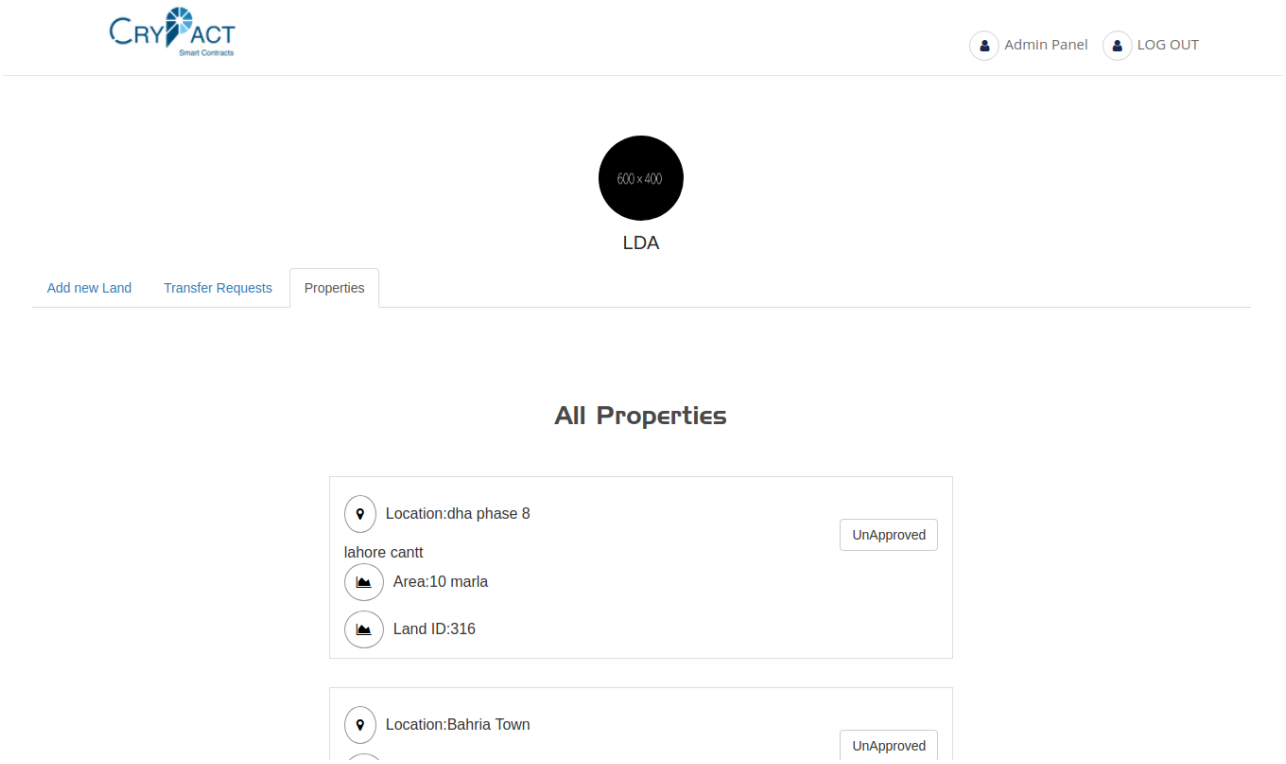
d. Add New Land



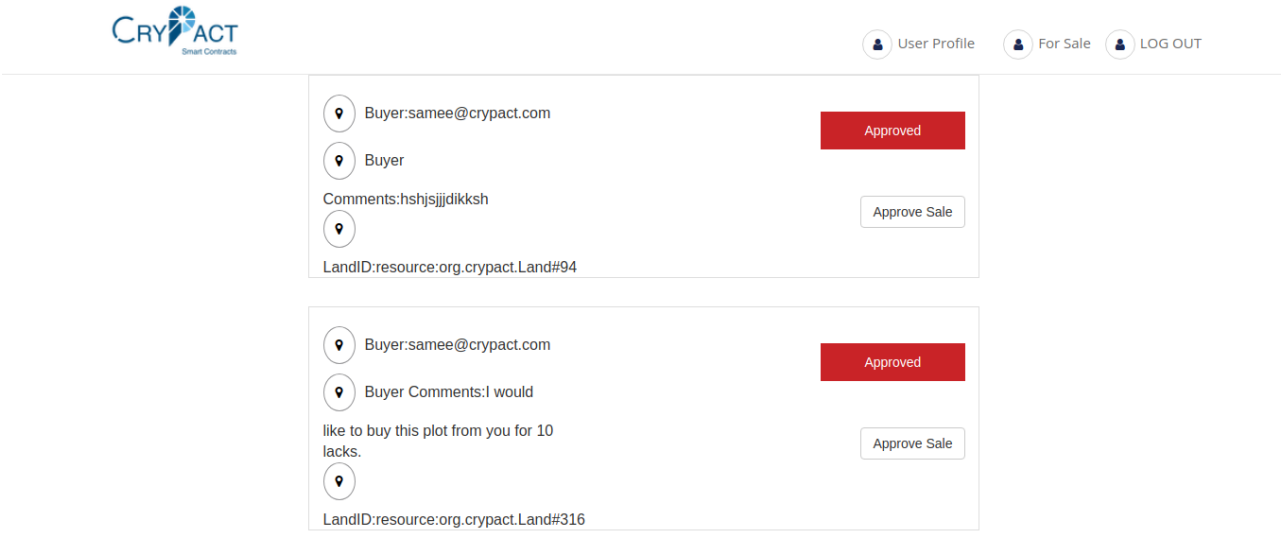
e. User Features



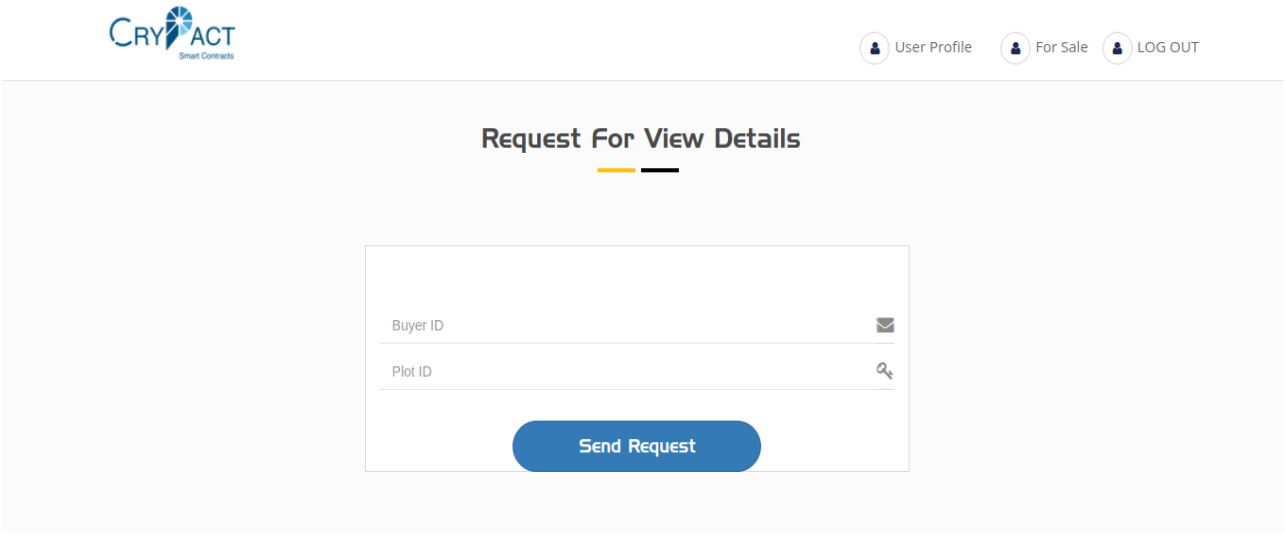
f. LDA Panel



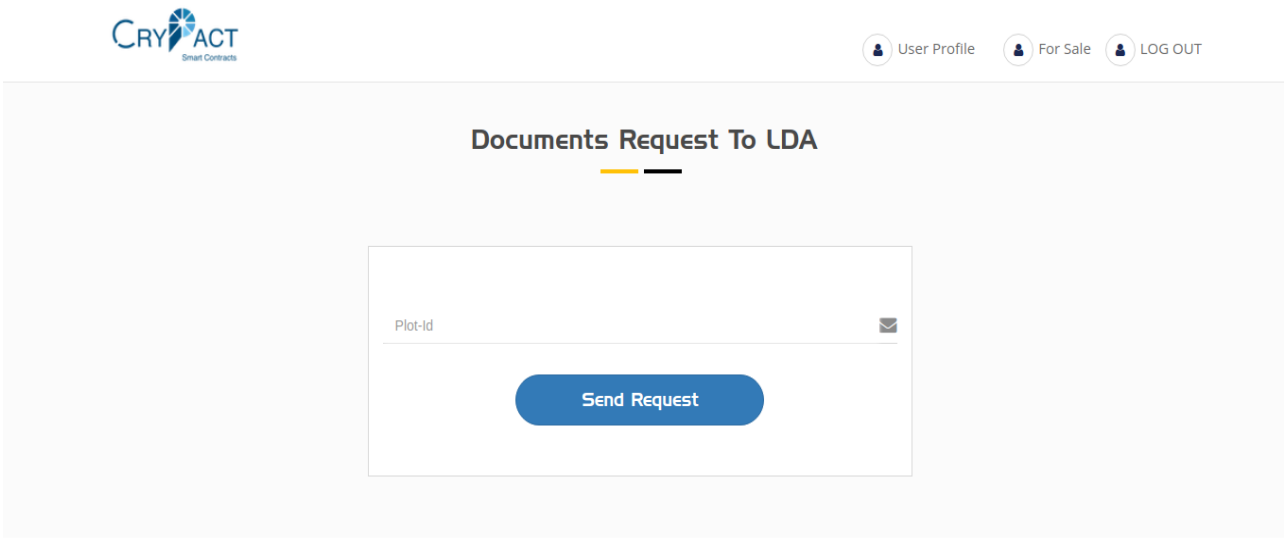
g. Received Offers



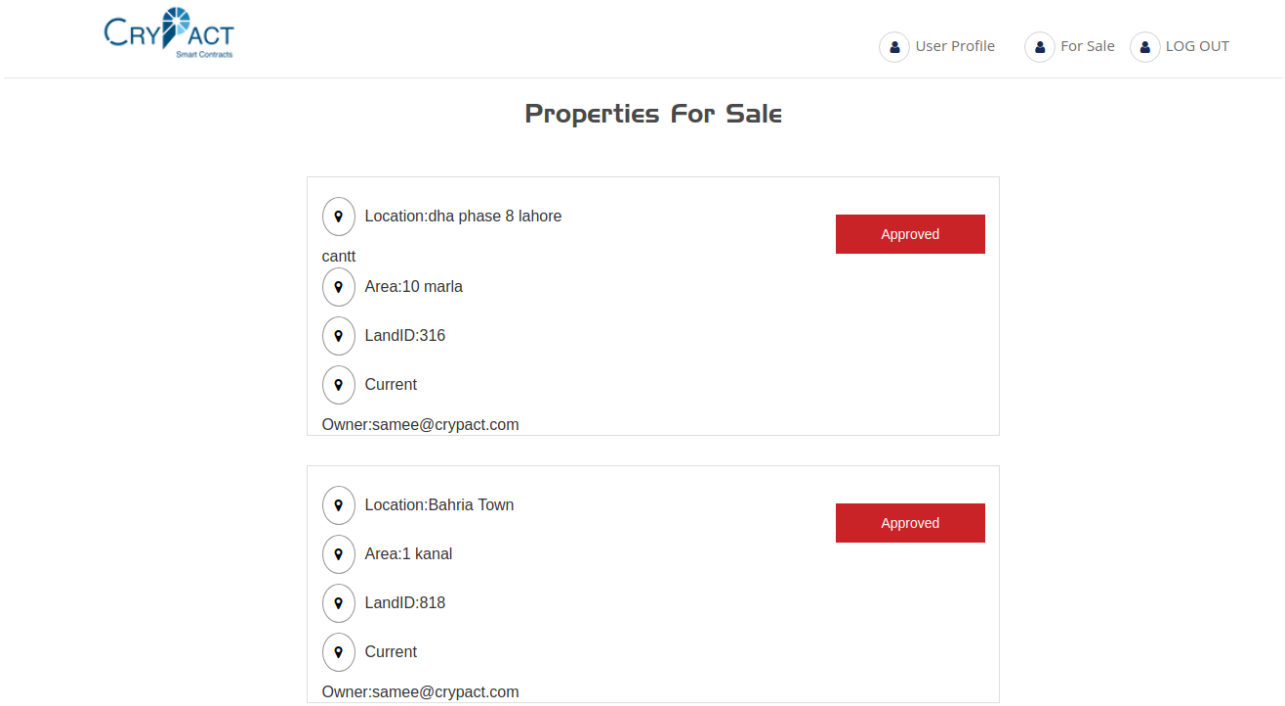
h. Request for Details



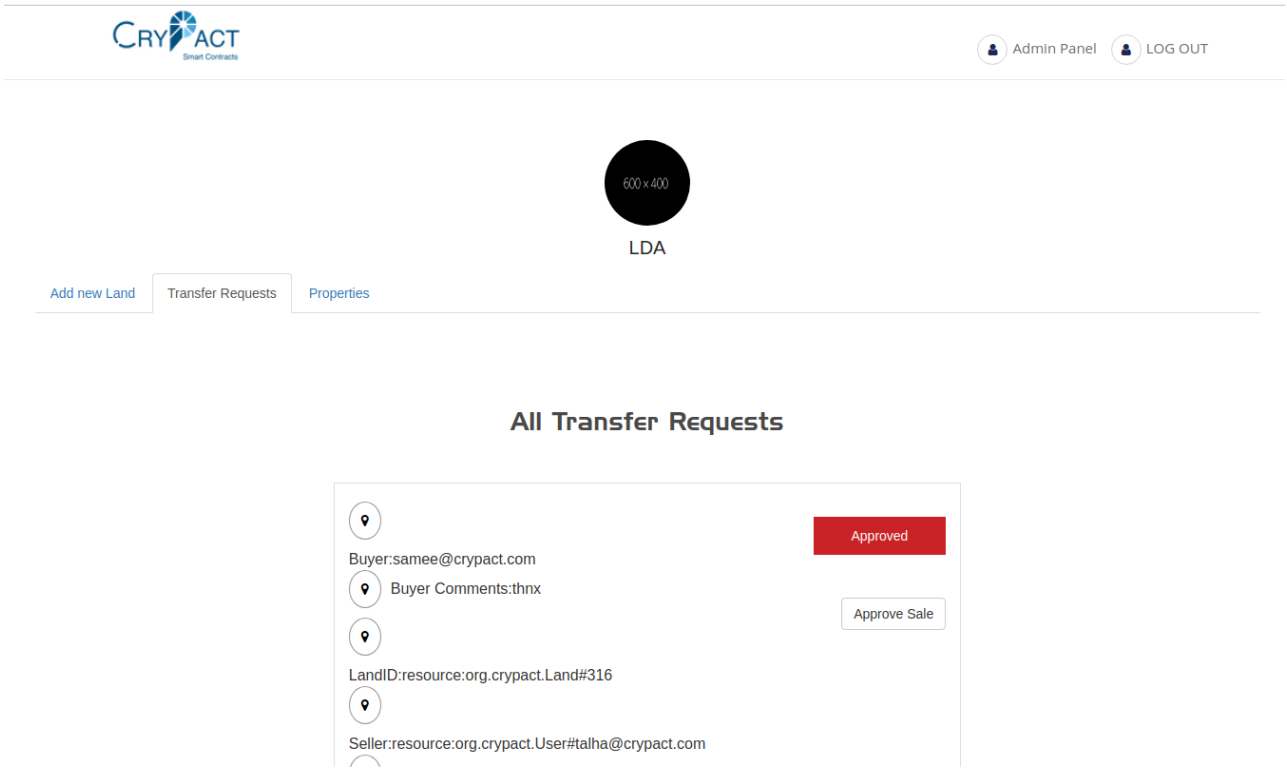
i. Buy from LDA (Request to LDA)



j. Properties for Sale (Public List)



k. Transfer Requests to LDA



5.2. Additional Information

All the system – Crypact – Smart Contracts is just to demonstrate the automation of property ownership change through Smart Contracts and data being record on Blockchain. We are not covering each and every aspect of the field because it's out of the range so many of the things like payment gateways, digital identity, templates for letters may be excluded.

6. Test Specification and Results

6.1. Test Case Specification

Sign Up

Identifier	TC1
Short description	User will enter credential and sign up
Pre-condition(s)	Web page should be open.
Input data	Muhammad Talha, TALHA@gmail.com, Password, Password. 35202-536283707
Detailed steps	User will enter all credential first name ,last name ,password , cnic for sign up
Expected result(s)	User will successfully sign up
Actual result(s)	User will successfully sign up
Post-condition(s)	Form Cleared.
Test Case Result	Pass

Sign In

Identifier	TC2
Short description	User will enter credential and sign In
Pre-condition(s)	Web page should be open.
Input data	Tbasit.explorer@gmail.com, Password
Detailed steps	User will enter all credential first , password for sign In
Expected result(s)	User will successfully sign In
Actual result(s)	User will successfully sign In
Post-condition(s)	Form Cleared.
Test Case Result	Pass

Send invitation for contracts

Identifier	TC3
Related requirements(s)	Invite For Contract
Short description	The users will be able to send invitation for the smart contract.
Pre-condition(s)	User must Be logged in.
Input data	Buyer id, Plot id.
Detailed steps	1:Entered Buyer id. 2:Entered plot id. 3:Click on Send request.
Expected result(s)	User is expected to be shown a success message.
Post-condition(s)	User is expected to be shown a success or faliure message.
Actual result(s)	The user was shown success message.
Test Case Result	Pass

Send Request for Documents.

Identifier	TC4
Related requirements(s)	Request for Documents
Short description	The users will be able to send a request for documents approval to LDA.
Pre-condition(s)	User must Be logged in.
Input data	Plot id.
Detailed steps	1:Entered Plot id.
Expected result(s)	User is expected to be shown a success message.
Post-condition(s)	User is expected to be shown a success or faliure message.
Actual result(s)	The user was shown success message.
Test Case Result	Pass

View pending requests.

Identifier	TC5
Related requirements(s)	Pending Requests
Short description	The users will be able to view all the pending requests which he has received for sale and purchase of property.
Pre-condition(s)	User must Be logged in.
Input data	No input data.
Detailed steps	1:Click on Pending requests button on user profile.
Expected result(s)	User is expected to be shown all pending requests.
Post-condition(s)	User is taken to pending requests page.
Actual result(s)	The user was taken to pending requests page.
Test Case Result	Pass

Approved Properties.

Identifier	TC6
Related requirements(s)	Show Approved Properties
Short description	The users will be able to see all his approved properties by LDA.
Pre-condition(s)	User must Be logged in.
Input data	No input data.
Detailed steps	1:Click on Approved properties button on user profile screen.
Expected result(s)	User is expected to be shown all Approved properties.
Post-condition(s)	User is taken to pending requests page.
Actual result(s)	The user was taken to Approved properties page.
Test Case Result	Pass

Accept Smart contract request.

Identifier	TC7
Related requirements(s)	Accept Invitation
Short description	The users will be able to see the Estamp page.
Pre-condition(s)	User must Be logged in.
Input data	No input data.
Detailed steps	1:Click on Accept button on pending requests page.
Expected result(s)	User is expected to be shown the estamp page.
Post-condition(s)	User is taken to estamp page.
Actual result(s)	The user was taken to estamp paper page.
Test Case Result	Pass

Execute smart contract.

Identifier	TC8
Short description	The users will be able to execute smart contracts.
Pre-condition(s)	User must Be logged in.
Input data	I Muhammad Talha declare that I have sold this property to Arif dar at a price of 1000000.
Detailed steps	1:Click on WE AGREE button.
Expected result(s)	The smart contract is expected to be executed when both users this button.
Post-condition(s)	The smart contract is expected to be executed when both users this button.
Actual result(s)	The smart contract was executed and the ownership of property was changed.
Test Case Result	Pass

Add New Land

Identifier	TC9
Short description	LDA will add a new land
Pre-condition(s)	User must Be logged in.
Input data	Size, location (10kanal,Bahria Town)
Detailed steps	LDA must logged In and should go to new land form and then fill that form with validated inputs and click save button and it should save and a success message will be shown.
Expected result(s)	New Land Added Successfully
Actual result(s)	New Land Added Successfully
Post-condition(s)	Form Cleared.
Test Case Result	Pass

Buy From LDA

Identifier	TC10
Short description	User request for a land to LDA
Pre-condition(s)	User must Be logged in.
Input data	Size, location (10kanal,Bahria Town)
Detailed steps	User will sent a request to LDA for a specific plot and LDA will allot that plot to that specific user.
Expected result(s)	Request will be sent to LDA
Actual result(s)	Request will be sent to LDA
Post-condition(s)	Request will be shown to LDA
Test Case Result	Pass

LDA Approved Land

Identifier	TC11
Short description	LDA approved a land for a specific user
Pre-condition(s)	LDA must Be logged in.
Input data	Size, location (10kanal,Bahria Town)
Detailed steps	LDA logged in and sees pending transfer request and approved a land for a specific request
Expected result(s)	Land will be approved
Actual result(s)	Land will be approved
Post-condition(s)	Form Cleared
Test Case Result	Pass

Set Property For Sale

Identifier	TC12
Short description	User will set a property for sale
Pre-condition(s)	User must Be logged in.
Input data	Size, location (10kanal,Bahria Town)
Detailed steps	User will logged in and set property for sale and other user can buy that property.
Expected result(s)	Property will show in for sale properties.
Actual result(s)	Property will show in for sale properties.
Post-condition(s)	Form Cleared
Test Case Result	Pass

View All Properties

Identifier	T C13
Short description	User can see his properties
Pre-condition(s)	User must Be logged in.
Input data	Click on view properties
Detailed steps	User will logged in and see his properties by clicking on view properties which are shown in his profiles
Expected result(s)	All land will be shown
Actual result(s)	All land will be shown
Post-condition(s)	User can use any feature of dashboard
Test Case Result	Pass

Received Offer

Identifier	TC14
Short description	User will receive offers from other users
Pre-condition(s)	User must Be logged in.
Input data	Size, location (10kanal,Bahria Town)
Detailed steps	User will receive offer from other user and make a deal by accepting that offer.
Expected result(s)	User will accept offer.
Actual result(s)	User will accept offer.
Post-condition(s)	Accept offer
Test Case Result	Pass

Accept Offer

Identifier	TC15
Short description	User will accept offers offered by other user to buy plot
Pre-condition(s)	User must Be logged in.
Input data	Click on accept offer
Detailed steps	User will accept offer by click on accept button and then both of the user will sign a smart contract and the plot ownership will be changed after that by LDA
Expected result(s)	Offer will be accept
Actual result(s)	Offer will be accept
Post-condition(s)	Form Cleared
Test Case Result	Pass

Table 6.2: Results

Module Name	Test cases run	Number of defects found	Number of defects corrected so far	Number of defects still need to be corrected
1- Sign up	TC1	0	0	0
2- Login	TC2	0	0	0
3- Request for land to LDA	TC6, TC9, TC10	0	0	0
4- Buy land from LDA	TC6, TC9, TC10	1	1	0
5- Pending Requests from Users	TC11, 12, 14	0	0	0
6- Approved Properties from LDA	TC10	0	0	0
7- Accept Request from User	TC14	0	0	0
8- e-Stamp Execution	TC6, 7, 8	0	0	0
9- Set Property for Sale Publicly	TC12, 13	0	0	0
10- Show All Properties for Sale	TC13	1	1	0
11- Transfer Request for Land	TC15	1	1	0
Complete System	15	3	3	0

7. Project Completion Status/Conclusion

Table 7.1: Project Completion Status

Module Name	**Status** (Complete, Partially Implemented, Not Implemented)
1- Sign up	Complete
2- Login	Complete
3- Request for land to LDA	Complete
4- Buy land from LDA	Complete
5- Pending Requests from Users	Complete
6- Approved Properties from LDA	Complete
7- Accept Request from User	Complete
8- e-Stamp Execution	Complete
9- Set Property for Sale Publicly	Complete
10- Show All Properties for Sale	Complete
11- Transfer Request for Land	Complete
Complete System	Complete

Table 7.2: Objective(s)/Target(s) Status

Target/Objective	**Status** (Completed, Partially Completed, Not Completed)	**Reason(s)**
Objective 1	Completed	N/A
Objective 2	Completed	N/A
Objective 3	Completed	N/A
Number of Targets Completed	3 (All)	
Number of Targets Partially Completed	0	
Number of Targets Not Completed	0	

References

https://www.investopedia.com/terms/s/smart-contracts.asp#ixzz5DaJ57ITW
https://en.wikipedia.org/wiki/Blockchain
https://github.com/ethereum/wiki/wiki/White-Paper
http://docs.neo.org/en-us/
https://www.cardano.org/en/academic-papers/
http://hyperledger-fabric.readthedocs.io/en/release-1.1/arch-deep-dive.html
https://bitcoin.org/bitcoin.pdf
https://www.npmjs.com/package/gpgme
https://qtum.org/en/
https://hackernoon.com/costs-of-a-real-world-ethereum-contract-2033511b3214
https://www.investopedia.com/terms/s/smart-contracts.asp
https://ethereumreport.org/ethereum/questions-answers-test-your-knowledge/
https://en.insider.pro/tutorials/2017-09-04/what-blockchain-laymans-terms/

Appendix A Glossary

Blockchain:

Blockchain is a public digital ledger of past transactions in order. For the rest of the article, we will consider this to be a ledger of bitcoin transactions. The blockchain is called so because it is a chain of blocks. A blockchain is a hash-linked data structure.

Node:

Nodes are distributed computers in the network that all have a copy of the entire blockchain. As new users enter the blockchain network, copies of the blockchain and the access to it is distributed. The data is replicable, synchronized and shared across all the nodes in the across multiple networks. The data is not controlled by a singular node or network.

Smart Contracts:

A smart contract is a digital agreement stored on the blockchain that is unalterable, once signed. It defines certain logic operations that have to be fulfilled in order to perform tasks such as deposit money or data.

Appendix B IV & V Report

(Independent verification & validation)

IV & V Resource

Name Signature

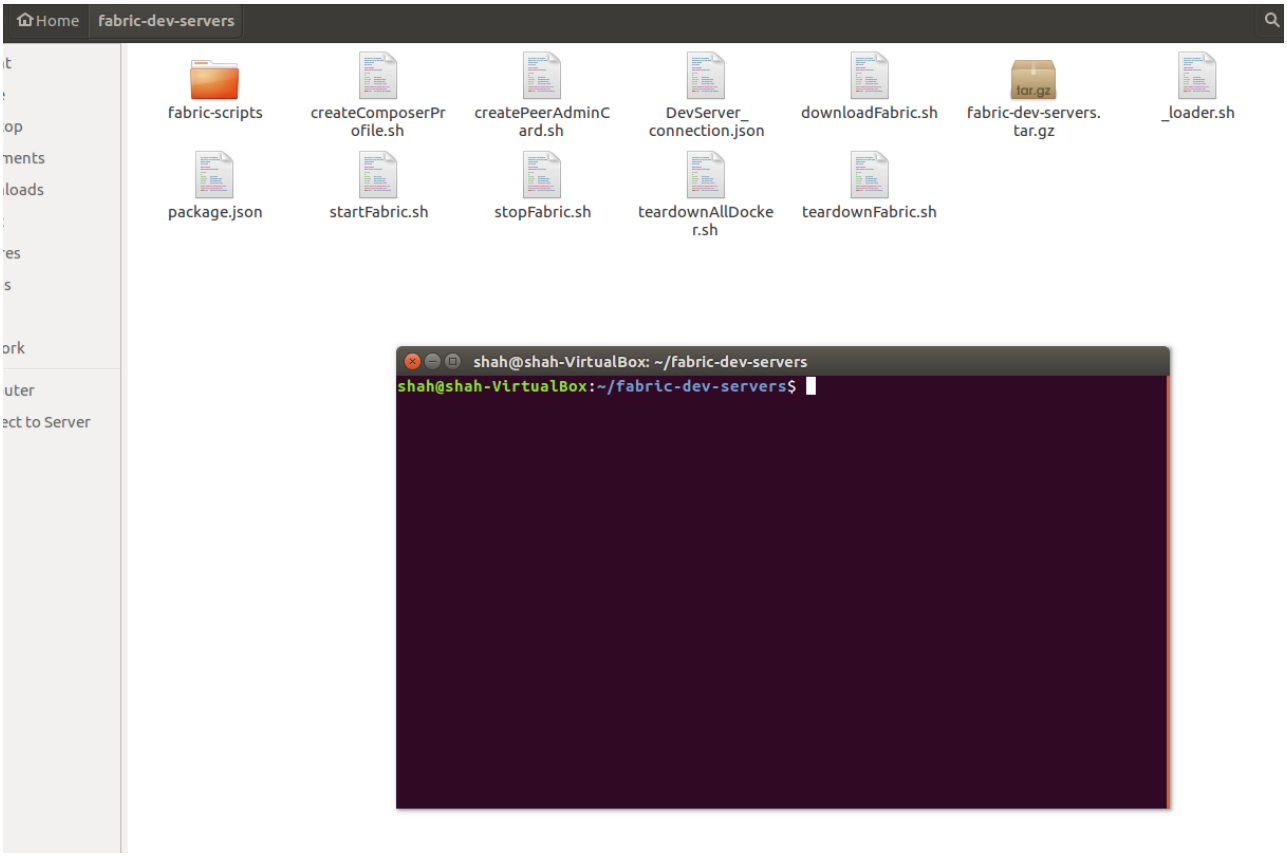
S#	Defect Description	Origin Stage	Status	Fix Time	
				Hours	Minutes
1					
2					
3					
…					

Table B.1: List of non-trivial defects

Appendix C Deployment/Installation Guide

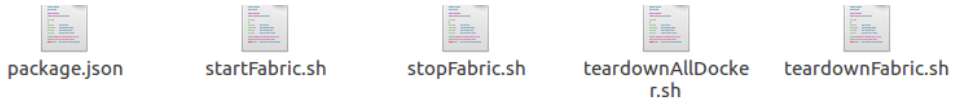
Back End (Blockchain & Network Channel):

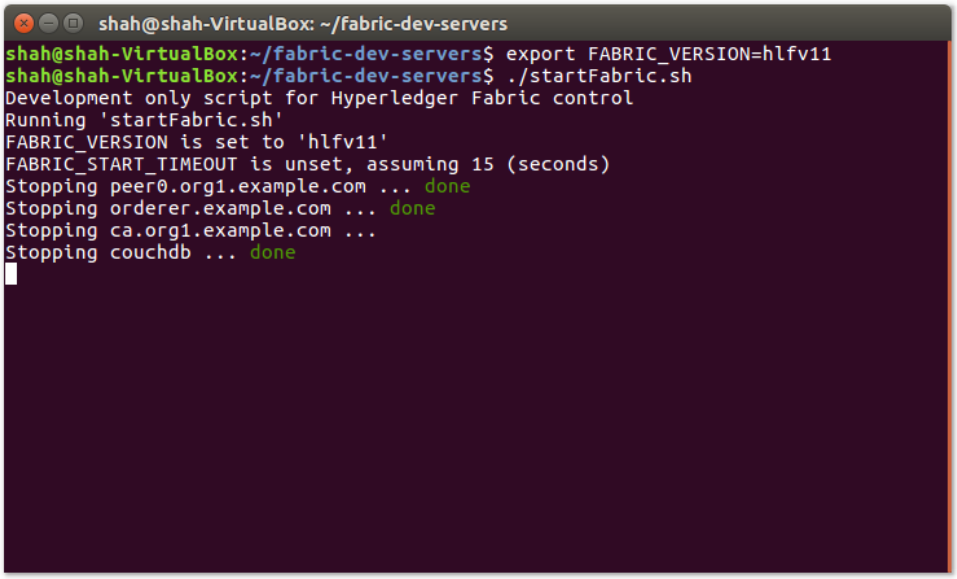
- Head over to "fabric-dev-servers" folders, and open it in terminal as shown below.



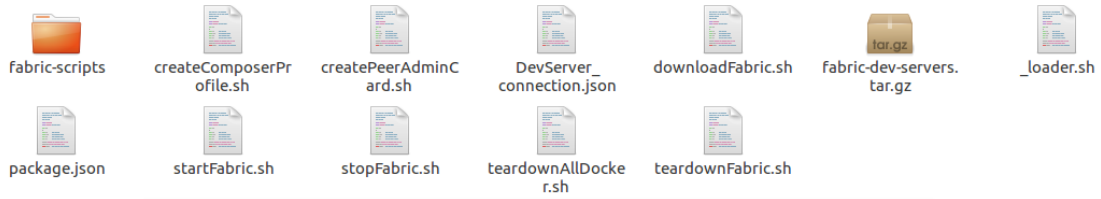
- Enter command: ***export FABRIC_VERSION=hlfv11***

- Then enter command: ***./startFabric.sh***

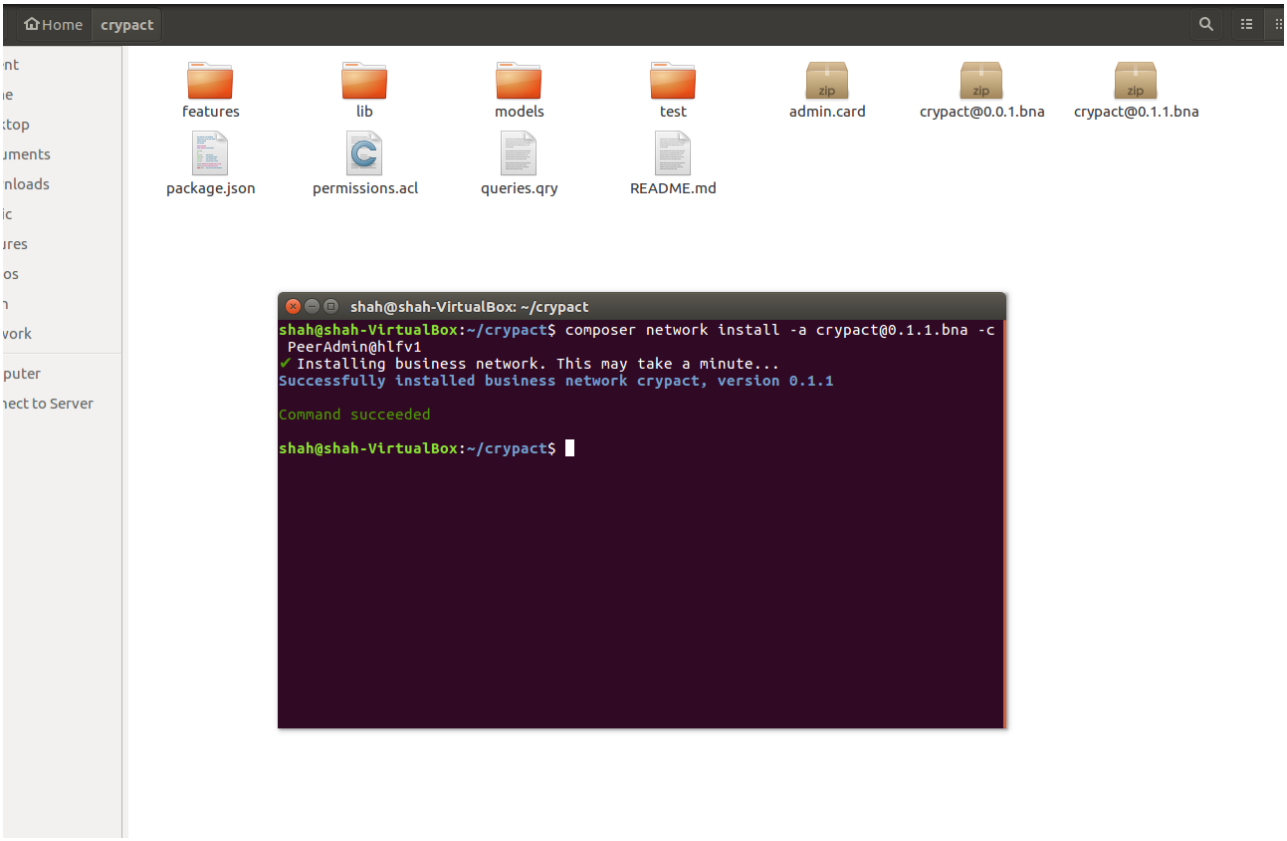




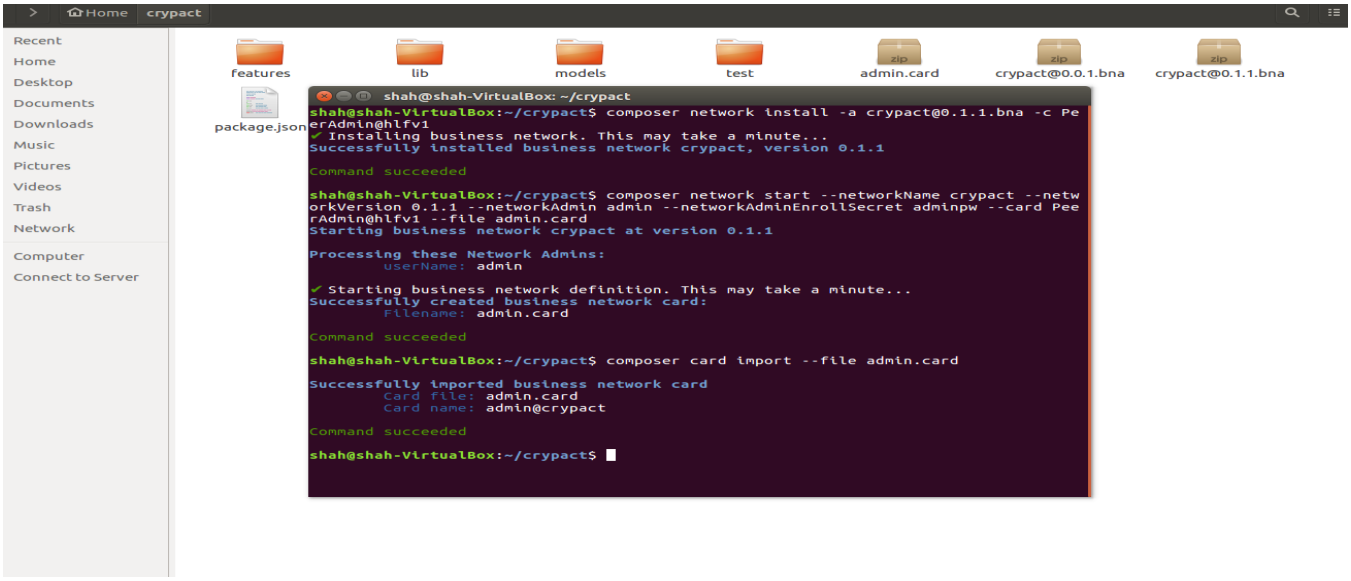
- Then enter command: ***composer card delete -c admin@crypact***
  (if it says card doesn't exist then no issue, just move on with next steps)



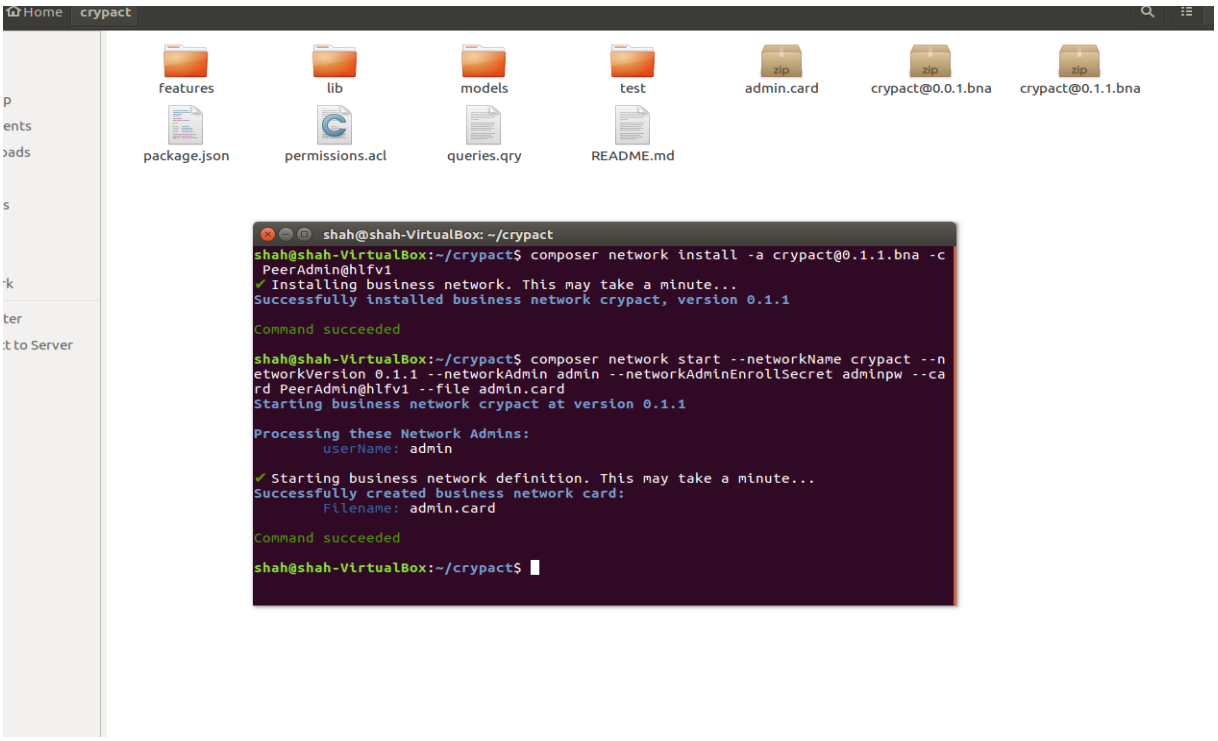
- Then go to “crypact” folder directory and open it in terminal just like we did with “fabric-dev-servers” and then enter command: ***composer network install -a crypact@0.1.1.bna -c PeerAdmin@hlfv1***



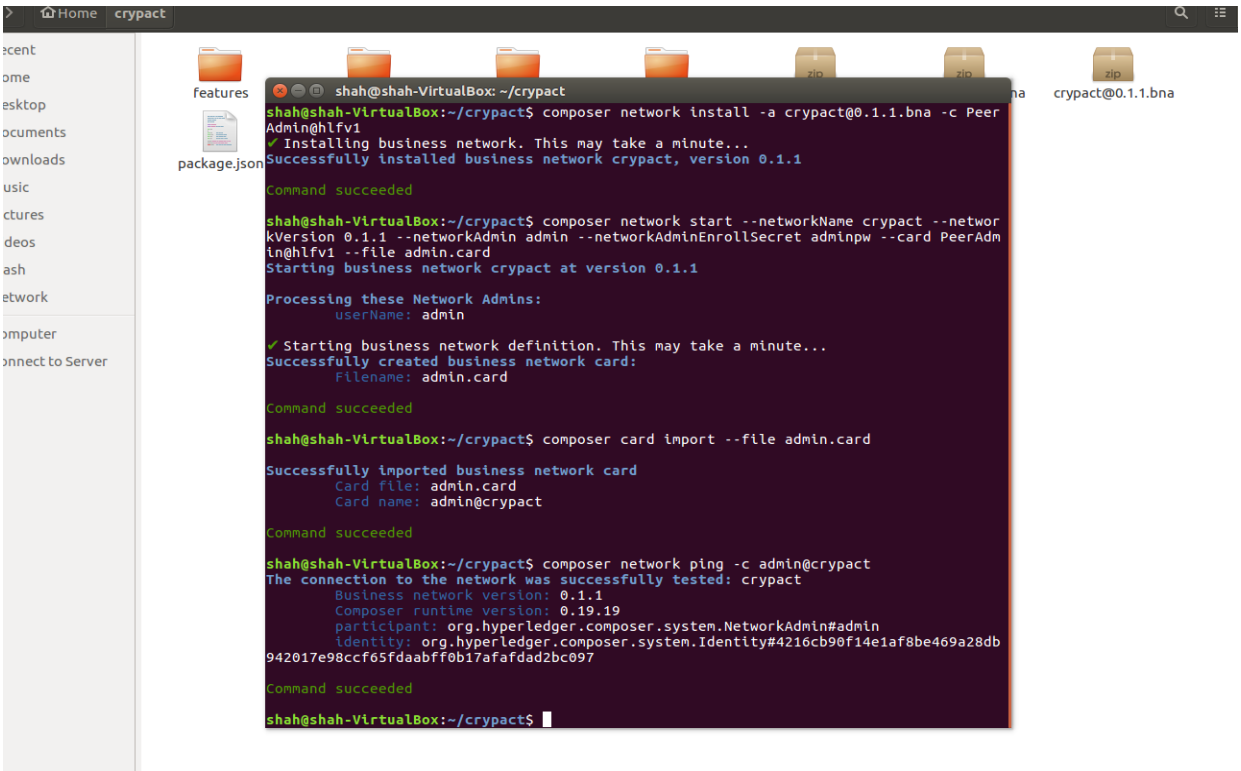
- Then enter command: ***composer network start --networkName crypact --networkVersion 0.1.1 --networkAdmin admin --networkAdminEnrollSecret adminpw --card PeerAdmin@hlfv1 --file admin.card***



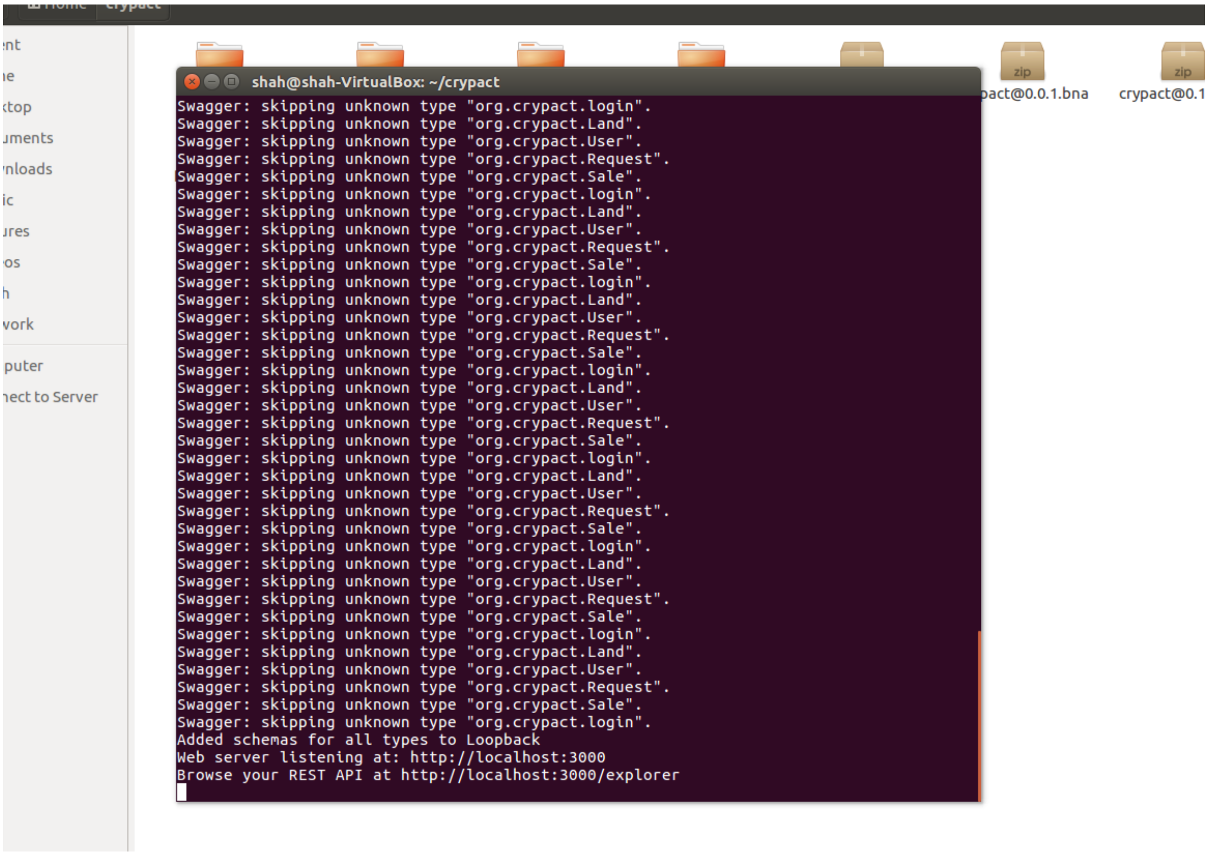
- The enter command: ***composer card import --file admin.card***



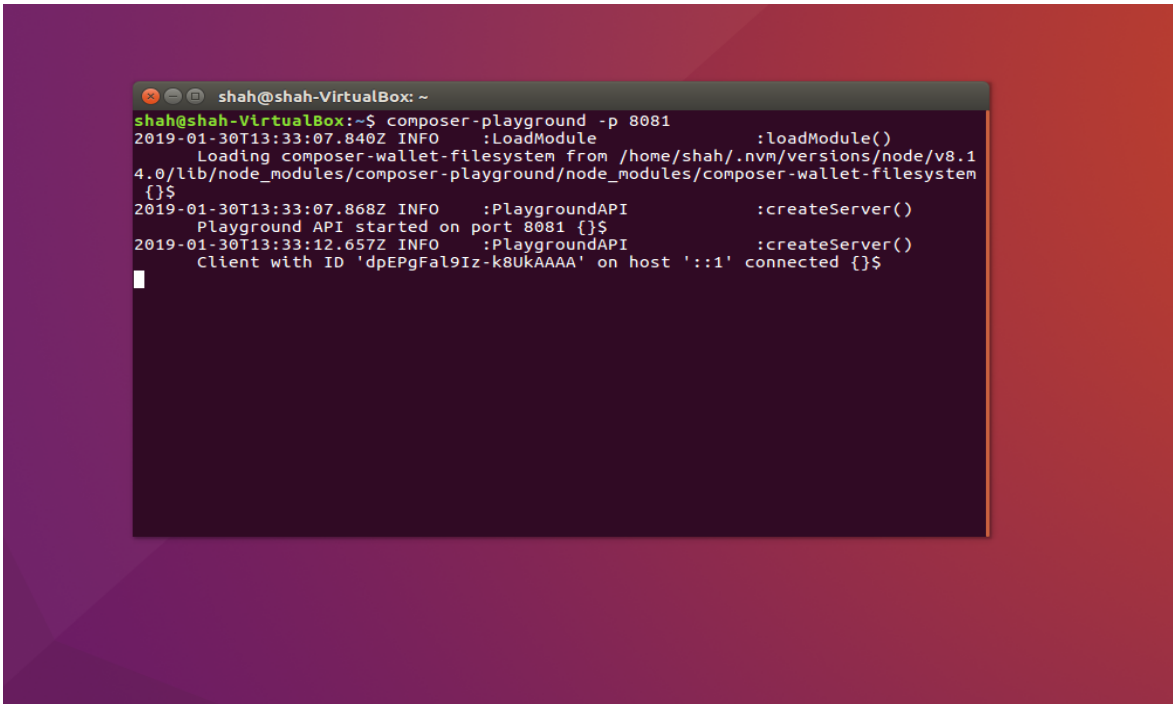
- The enter command: ***composer network ping -c admin@crypact*** (just to ensure everything went smoothly and you should see something like below)



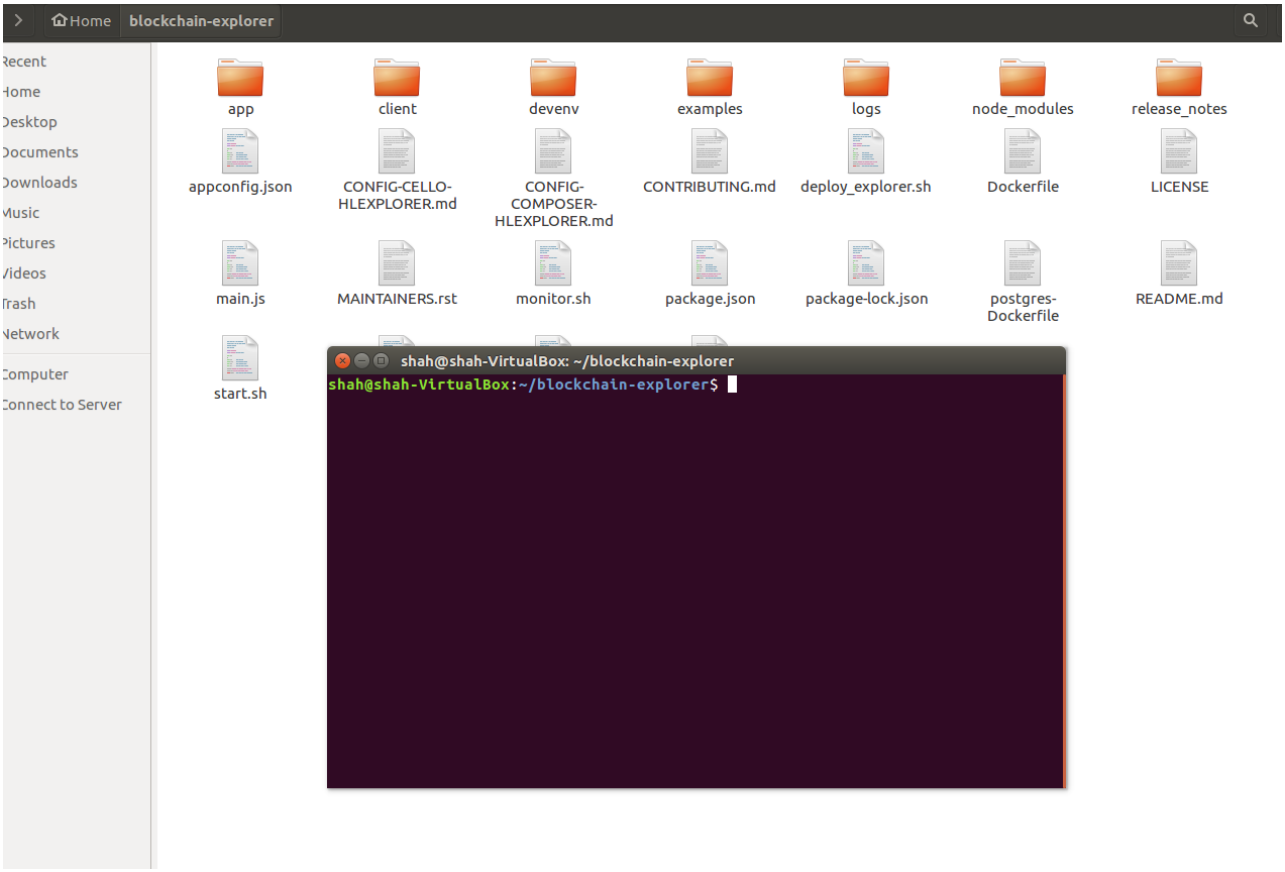
- Now to bring up the API endpoints enter the command: ***composer-rest-server -c admin@crypact*** (you should see something like in screenshot below after success)



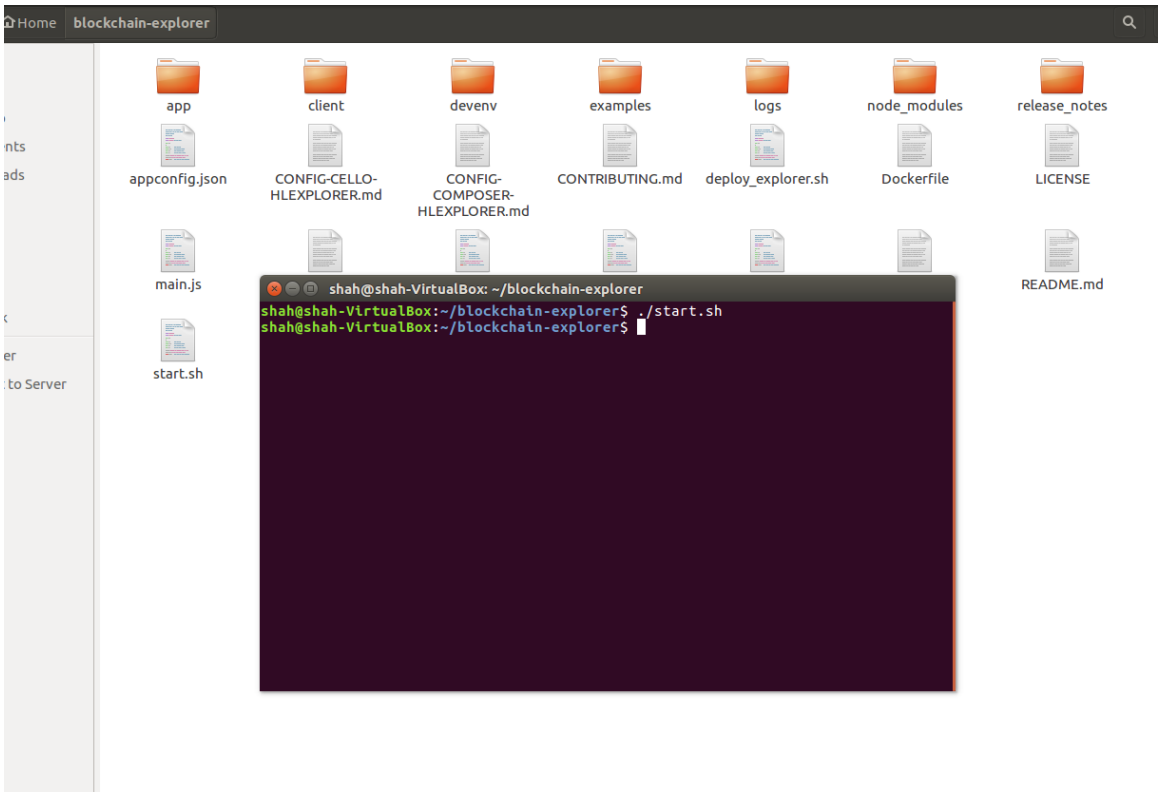
- Open terminals new window and enter the command: ***composer-playground -p 8081*** (To start the playground interface of Hyperledger)



- Now, go to the “blockchain-explorer” folder and open it in terminal.

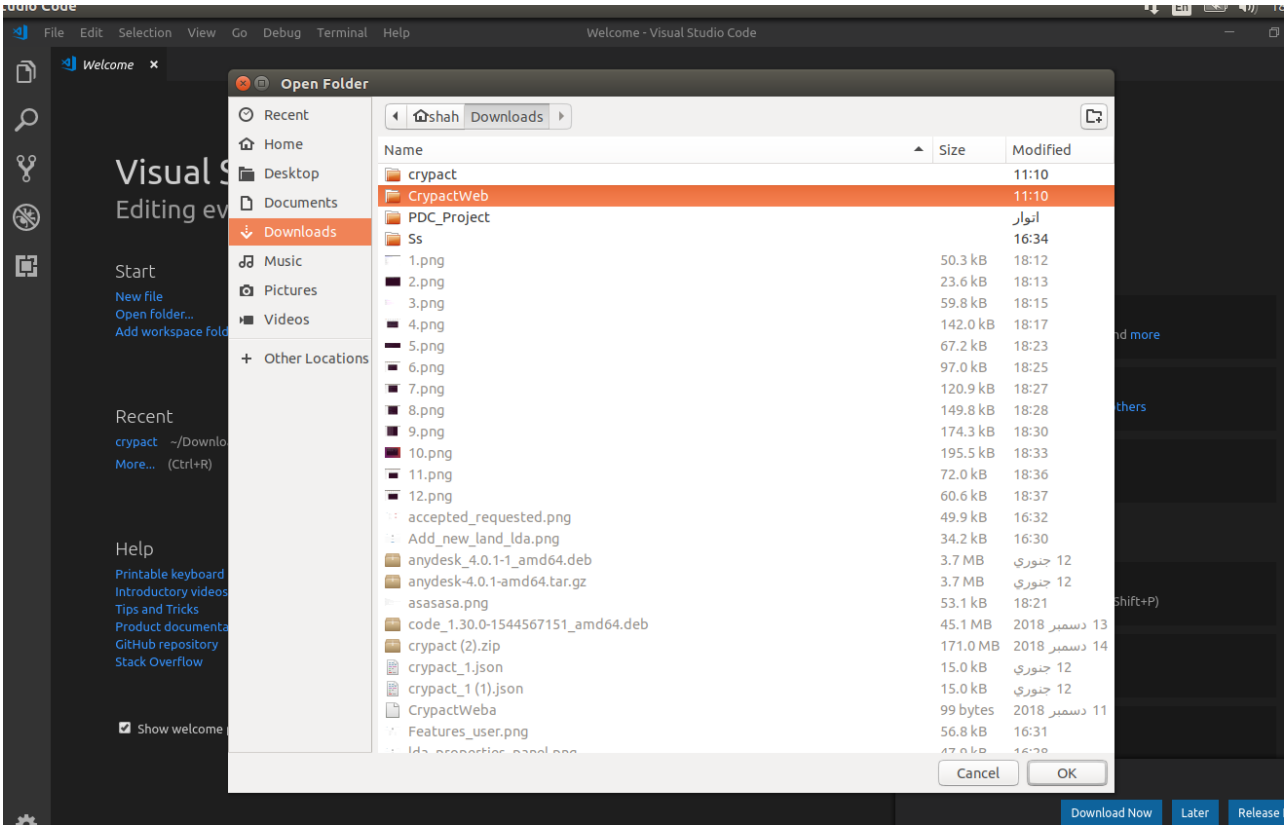


- Run the following command to start the network: ***./start.sh***

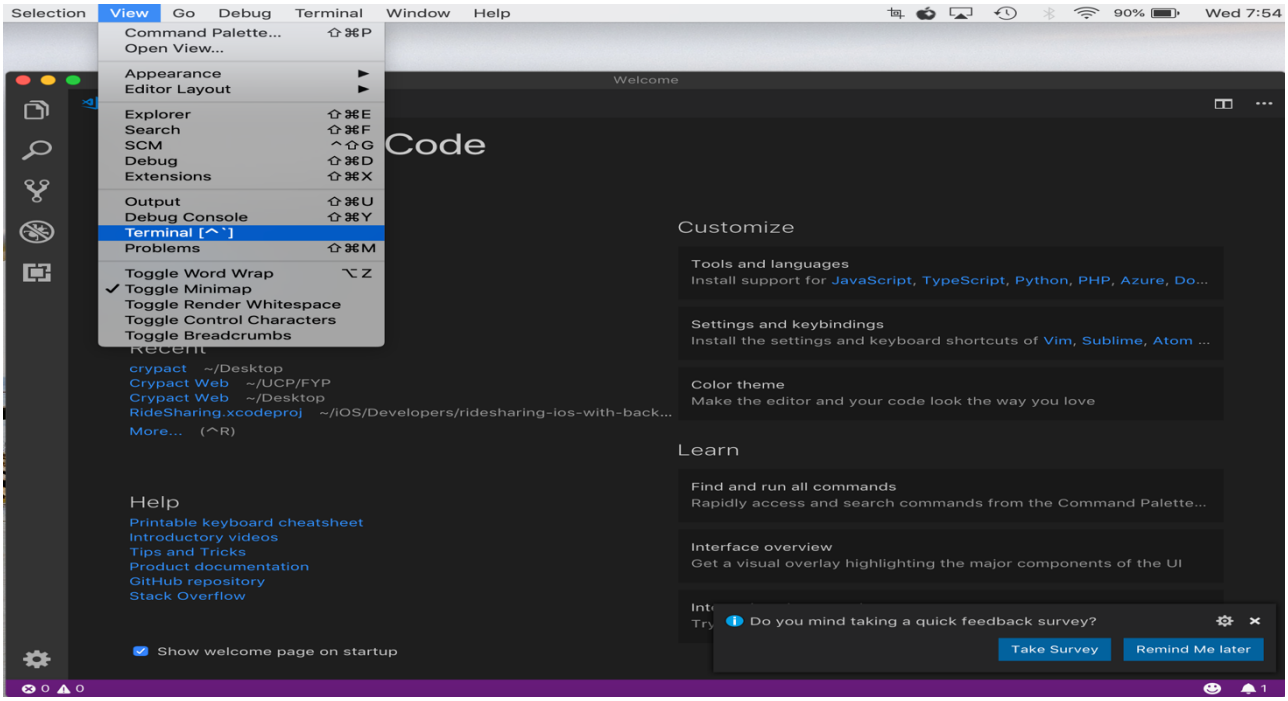


Front End Web:

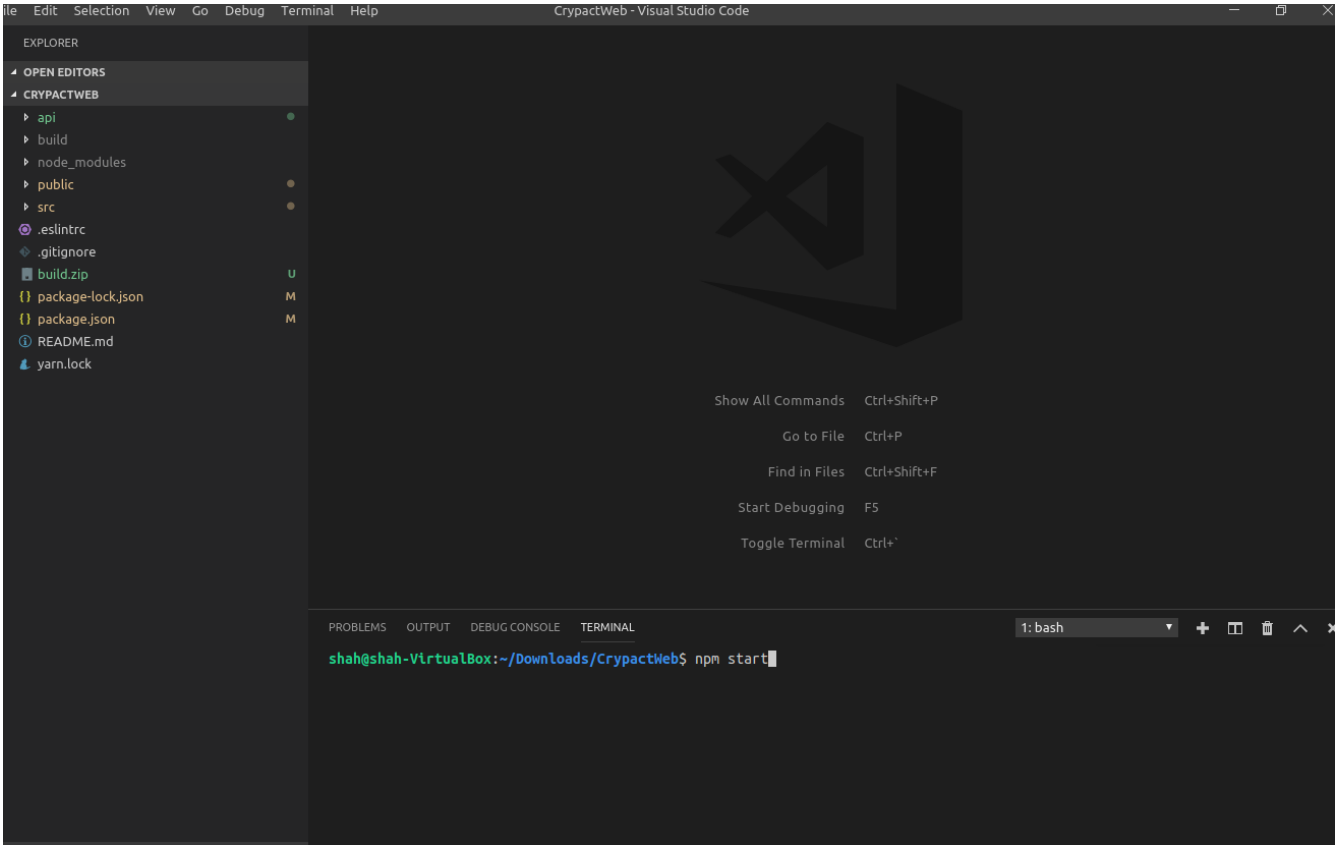
- Download any web editor like Atom, VS Code etc.
- Click on “File” in VS Code menu and then click “Open Folder” and select “CrypactWeb” folder in editor. (Visual Studio Code is preferred. You can do it in any editor).



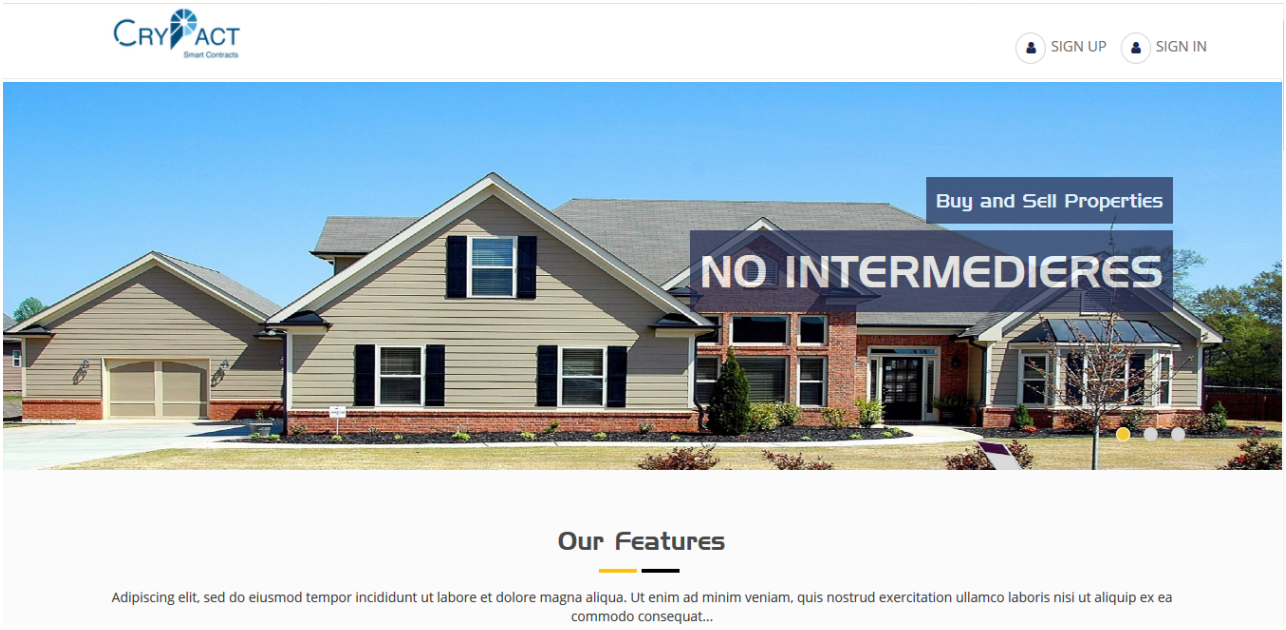
- From the main menu in VS Code, hover to “View” and click “Terminal”.



- In terminal space below which editor is showing, write the command: ***npm start***



- The default browser will automatically redirect you to the web interface OR if not then open any browser and enter ***localhost:3001*** to run the website. You will see something like this which is home page for our website.

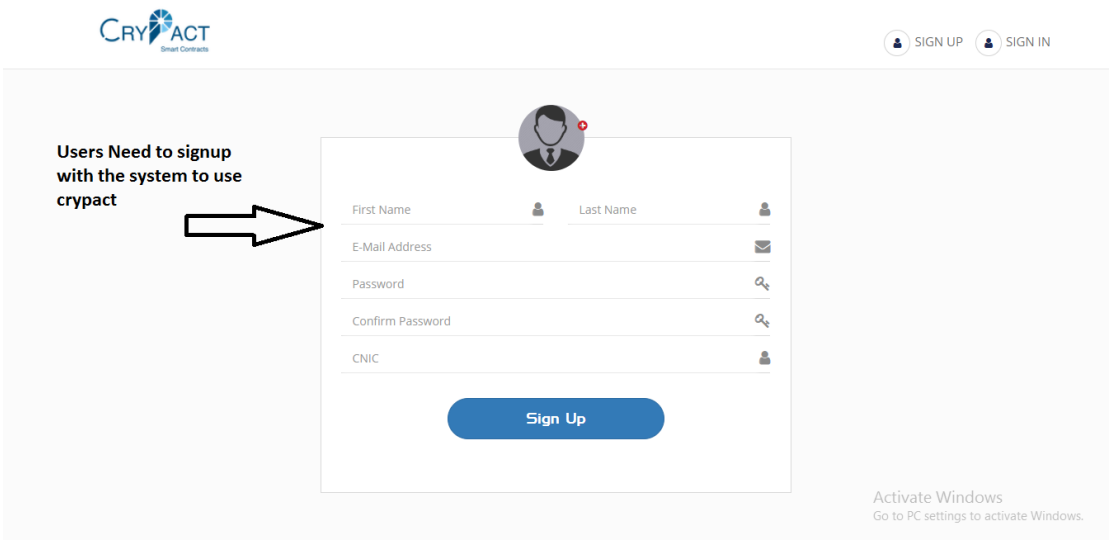


NOTE: The deployment guide is purely for Linux based OS only (like Ubuntu) and it is installed on localhost.

Appendix D User Manual

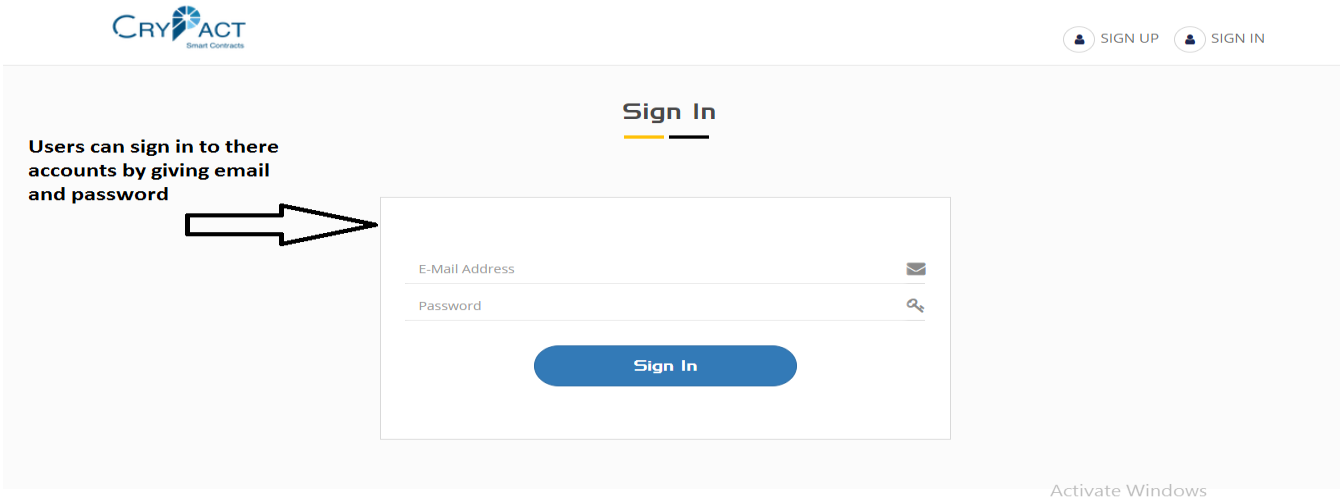
Sign Up:

- Enter the credentials as mentioned.
- Click on the Sign up button
- If you see a success message, then you have signed in with Crypact web system.



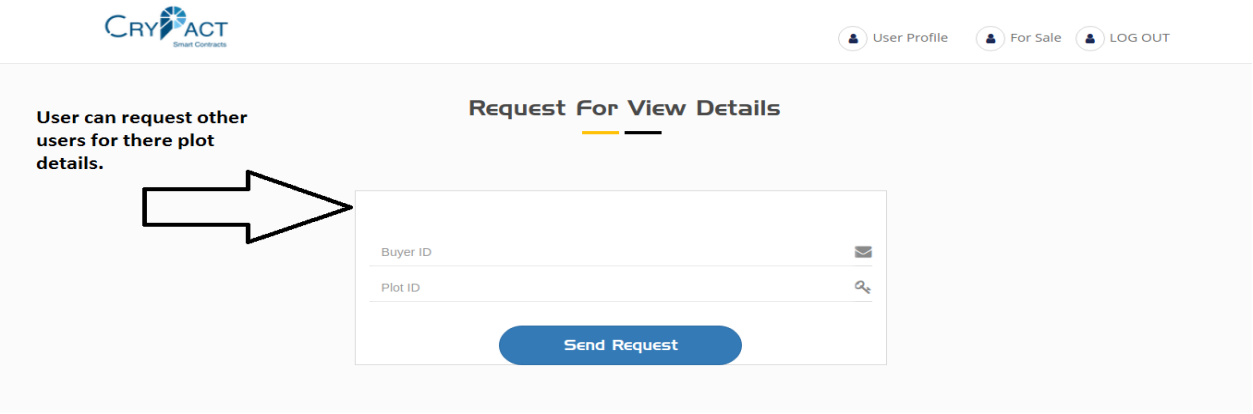
Sign In:

- Enter the email and password
- Press the sign in button.



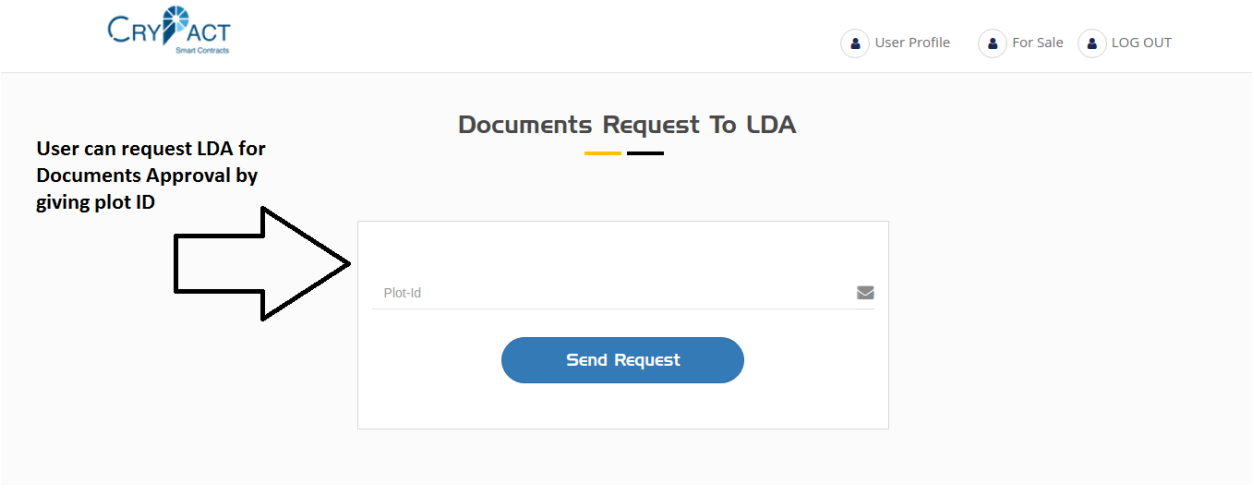
Request For Documents:

- Enter the buyer id.
- Enter the plot id.
- Press the button to send request for details.



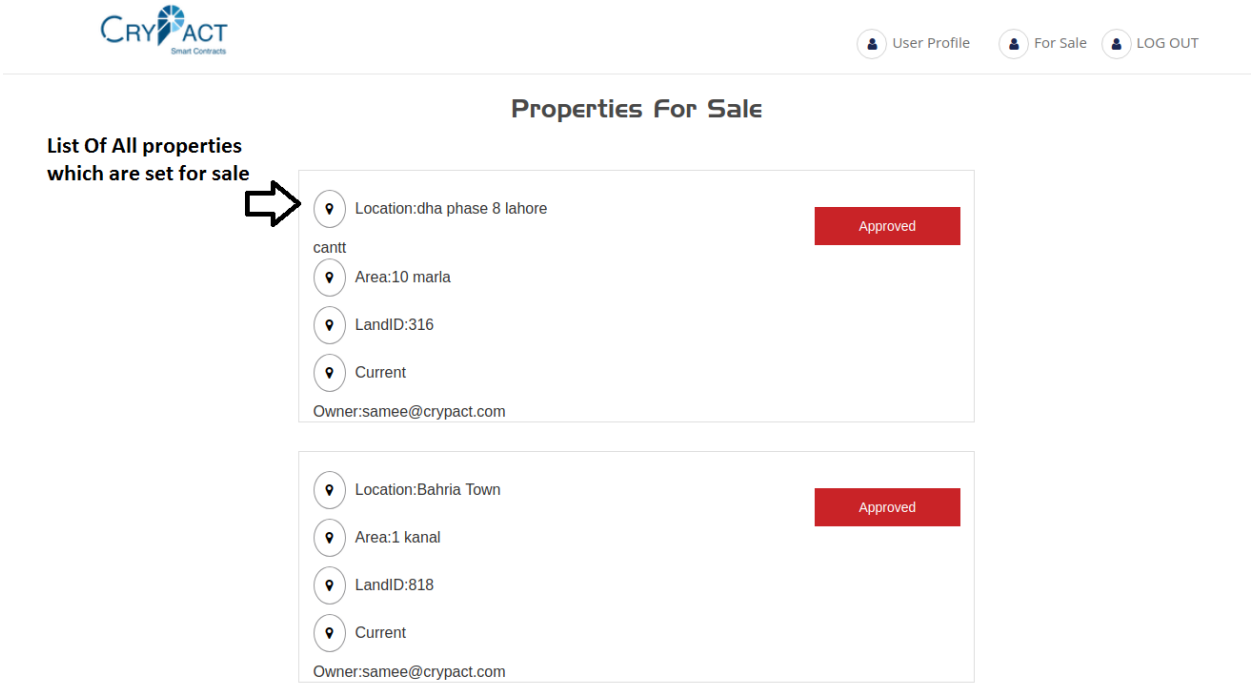
Request For Documents

- Enter the Plot id.
- Press the button to send request to LDA.



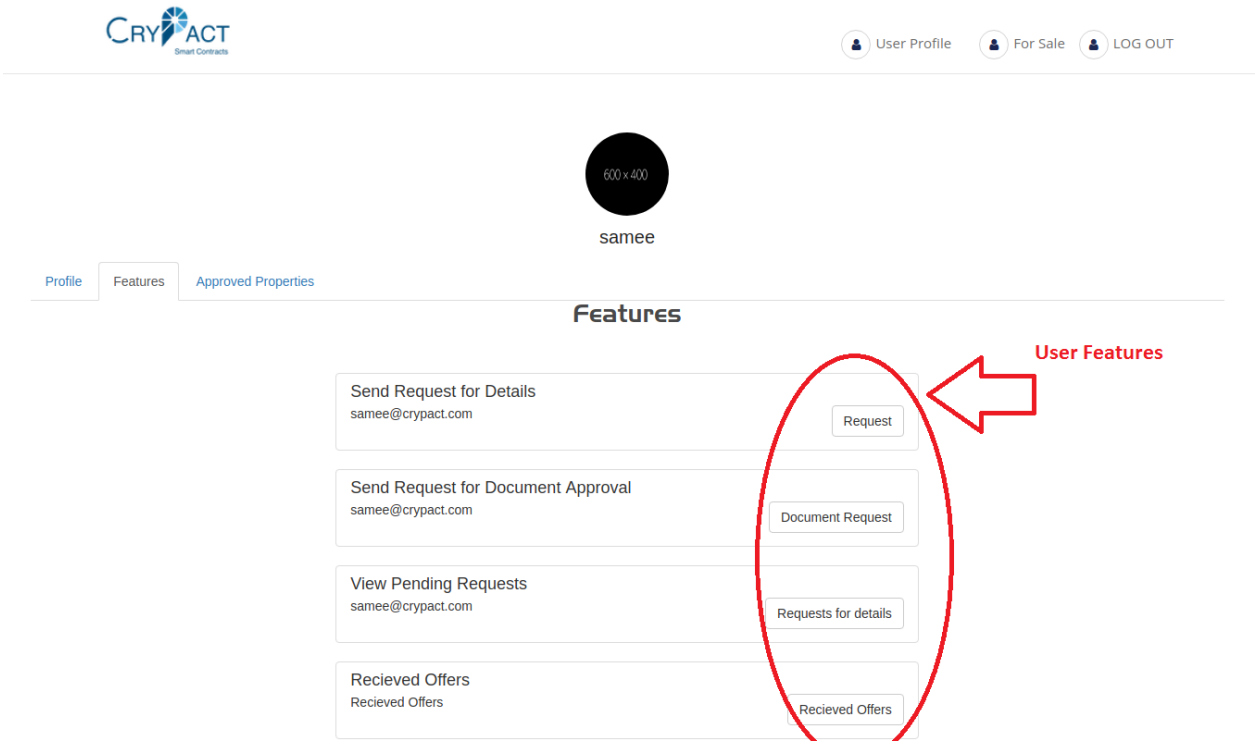
For Sale Properties

- User can view all the properties which are for sale



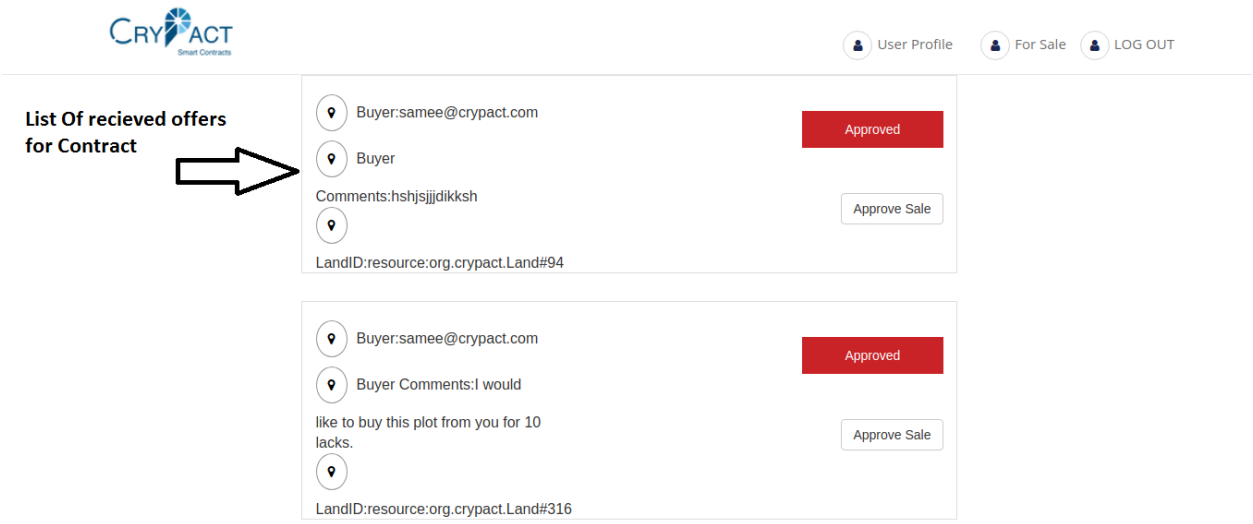
User Features:

- User can perform all of these functions in the profile as shown in the picture below.



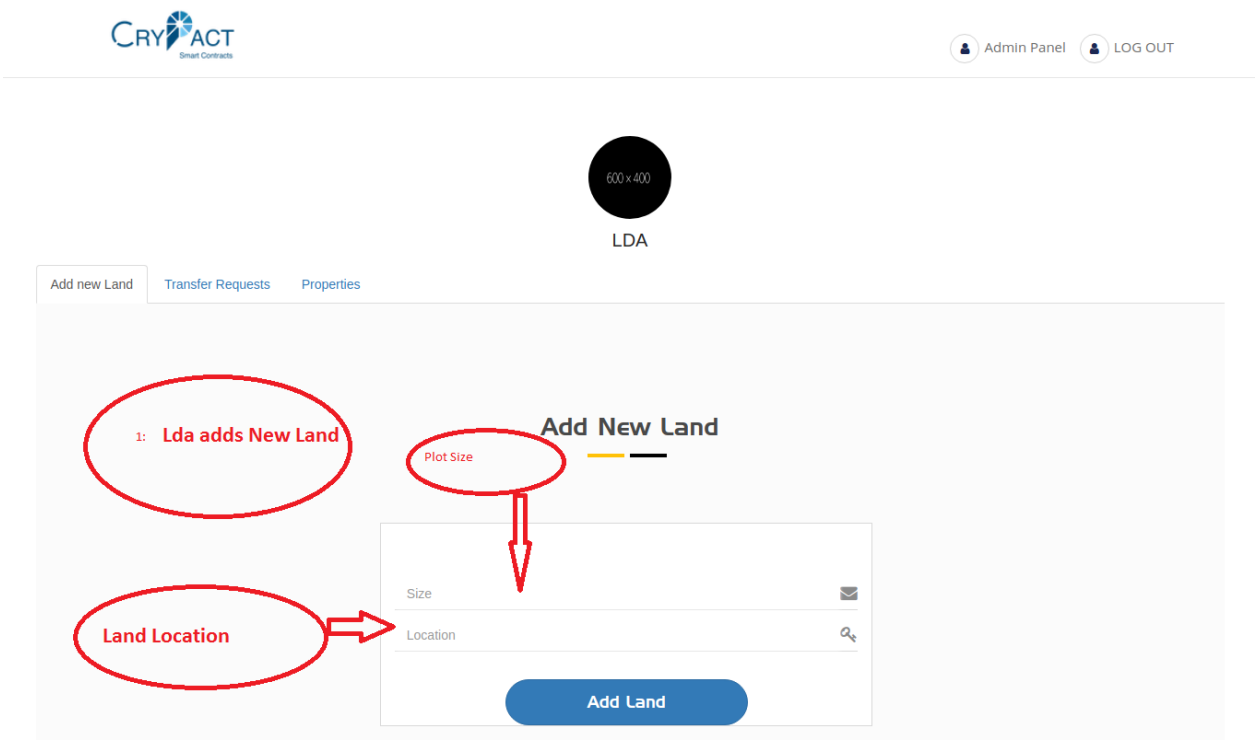
Received Offers:

- User can see the offers he has received for smart contract.



Add new Land:

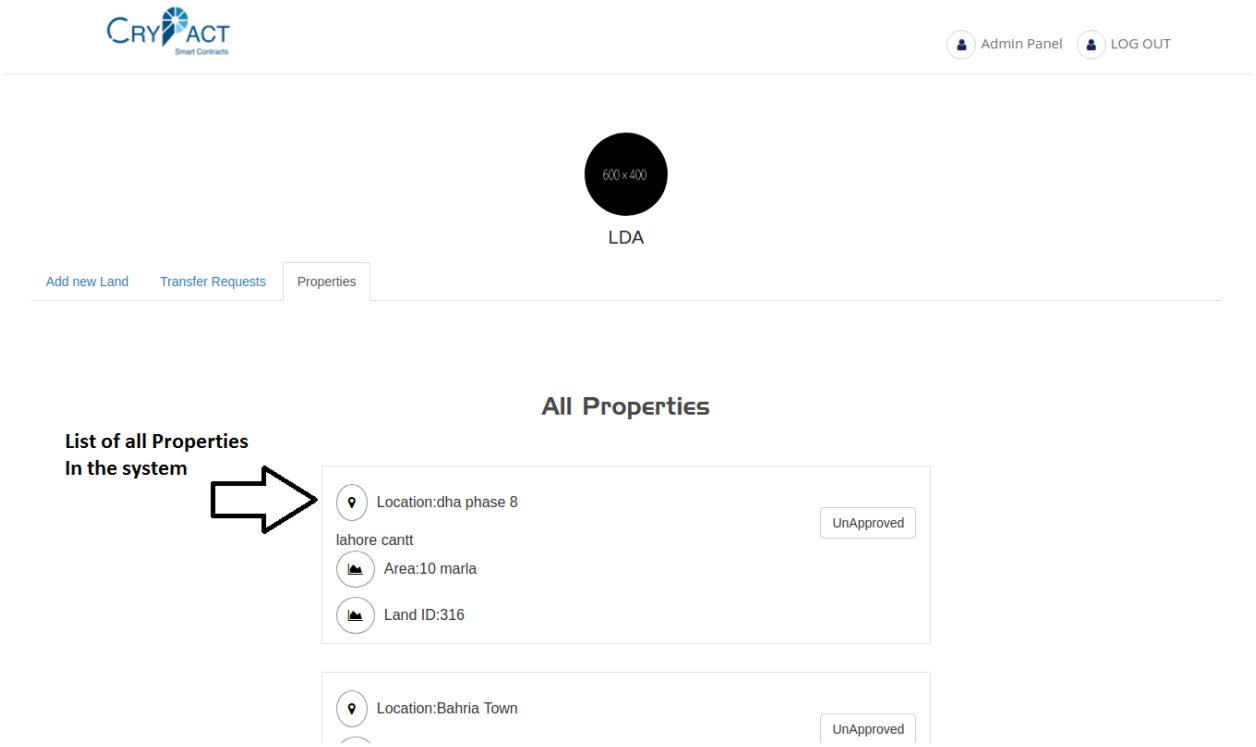
- LDA can add new land.



All

Properties:

- Panel to show all properties in LDA system



Transfer Requests:

- LDA can view all the requests to transfer the ownership of Land.