SKYNET

DO GOOD

YELLOWPAPER

Version 1.1

skynetworld.org

Introduction

Skynet is a decentralized lending platform, connecting borrowers and lenders in a decentralized fashion. Skynet is built on top of Tezos, and will provide the following collateralized lending options:

1) Crypto-backed loans – initially Skynet will focus on crypto-backed loans, allowing borrowers to stake their crypto as collateral and receive a fiat loan (from one of our lenders)
2) Home-backed loans – we will be working toward providing a solution where users can purchase homes using Skynet, with titles being managed on-chain. The home therefore becomes the collateral
3) Unsecured loans – finally, we aim to provide regulated unsecure loans

Skynet will use a mix of decentralized nodes and centralized API servers to run this platform, located around the globe. Our platform will be built on Tezos and take advantage of the advanced DPoS protocol and smart contract layer. We will be using a mix of different languages to construct the Skynet platform and utilise a number of "best-practises" to ensure our platform is secure and efficient.

The Skynet Decentralized Application (dApp) will have the following layers:

UI LAYER
this is the final layer that end users will be interfacing with – this could be our web app, or our mobile apps

API LAYER
this layer connects with the Blockchain layer and contains logic/code that doesn't need to be run on the Smart Contract layer (i.e. data storage, smart contract relaying etc)

SMART CONTRACT LAYER
this is the trustless layer of code that we will run on top of the Blockchain layer, and will handle the financial transactions on the blockchain

BLOCKHAIN LAYER
this is the core decentralized infrastructure that our platform will be run on, which will be the Tezos protocol

There are a number of services and processes that also take place in-between these layers.

Technology

Our platform will be built on Tezos, a secure and functional blockchain. Smart contracts are written in Michelson, where mathematical proofs can be produced to ensure that our contracts are safe. Our lending contract will allow users to interact with our platform on-chain.

We will develop our smart contracts using a high-level language (Liquduity and fi), as well as a few raw Michelson contracts where it makes sense. We will look to verify each contract to ensure mathematical correctness, security and safety.

Our API servers will be nodejs based, and will store data securely within databases that are backed up and replicated across multiple machines. All data will be stored securely and encrypted.

Our platform will also utilise UI/UX languages including HTML/CSS, JS. We are currently looking at using Angular 6 and Material design for our frontend frameworks for the web app. Skynet will also utilise either ReactNative or Flutter as the language for our mobile apps.

For general communication, we will be utilising the eztz.js library – this provides an easy to use and direct line of communication with the core blockchain protocol.

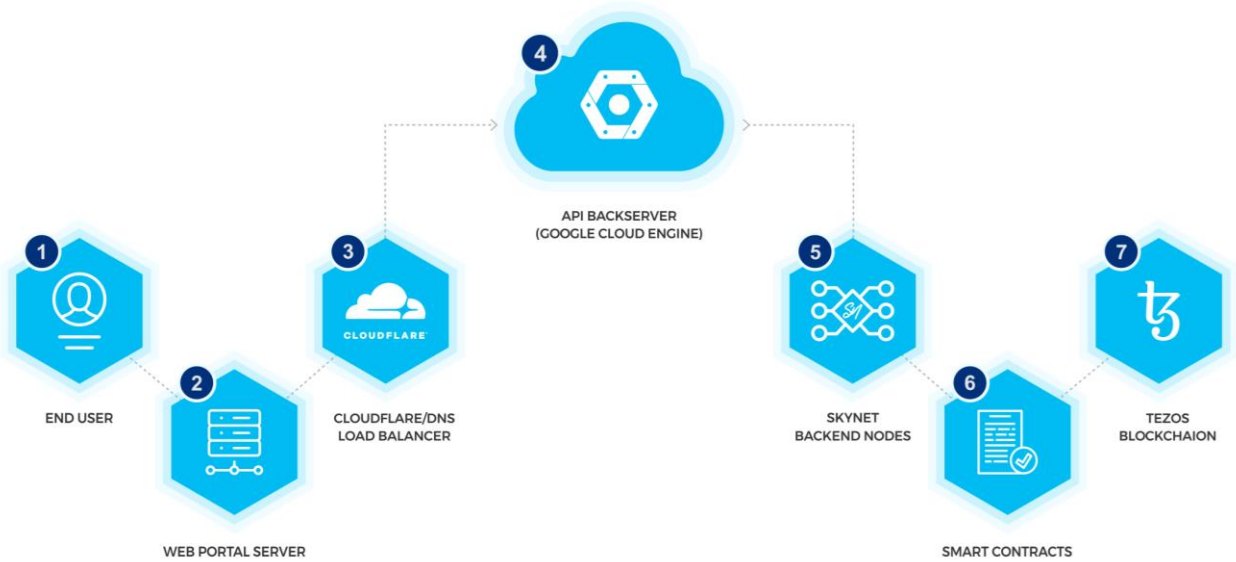
Infrastructure

Skynet will utilise Google Cloud Engine to run Tezos-nodes globally, using load-balancing techniques and DDoS protection to ensure our setup is available and robust. We will also look to partner with well-known public node providers, and offer our servers to the public as well. We will look at hiring experts in the fields to manage and oversee the development and deployment of these nodes.

Our nodes will be built on Debian 9 (Stretch), and will run compiled versions of the Tezos-node protocol released by DLS. These nodes will be secured using a 2048bit security certificate and utilise a load balancing server (with SSL between the back and front nodes).

All off-chain data and logic will be run on independent AWS nodes, balanced using the AWS load-balancers and Cloudflare (for DDOS protection). This gives us a solid base infrastructure to build from, where we can scale with our user base.

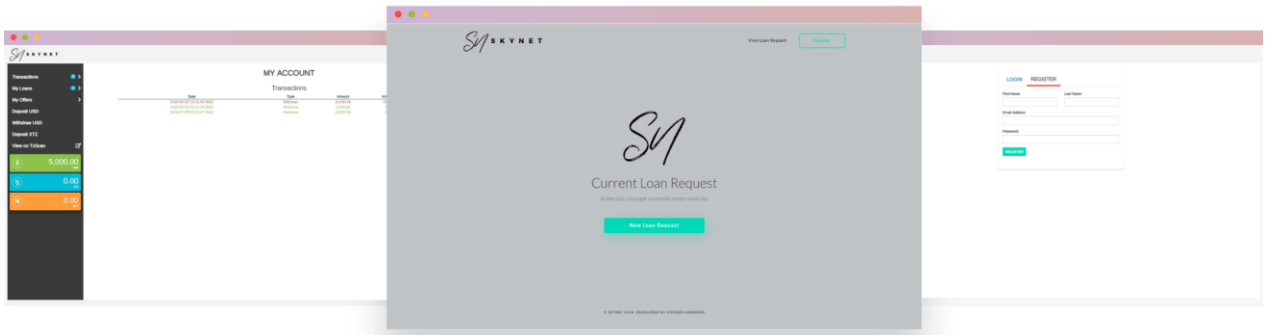
We will look to use industry experts to assist in the formal verification and auditing of our code-base and infrastructure, and have started scouting potential firms to assist with this.

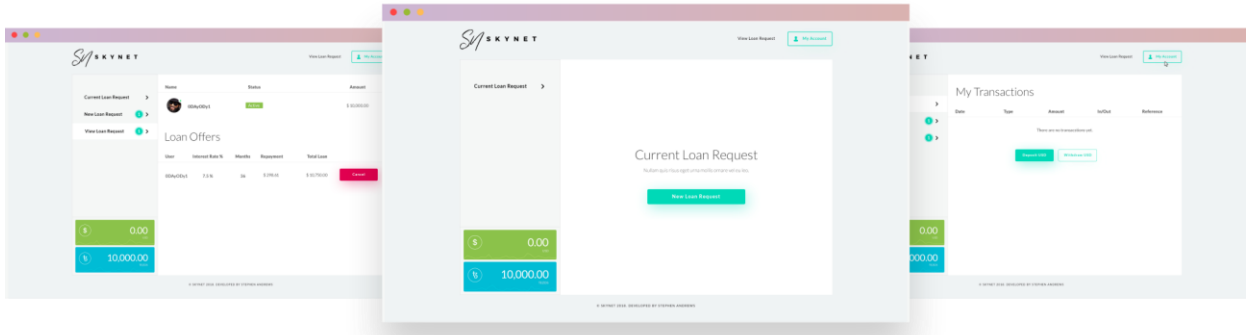


User Interface

We will be employing key experts to assist with the User Experience and User Interface side of Skynet – currently our initial beta release will utilise Material design concepts mixed with custom designs, but we will work toward a more general set of design guidelines for our platform.

We want to ensure that Skynet is simple to use, with appropriate messaging and functionality for our target audience.





Smart Contract

Skynet will utilise three core smart contacts, with the ability to add additional contracts in the future. These contracts will aim to achieve the following:

1) Core Lending Contract – this handles the loan and collateral logic, and allows for signed requests to make repayments on a loan, or liquidate to meet the LVR
2) Token contract – this will be a standard ERC20-like token contract for the NET token on top of Tezos
3) Exchange Contract – this contract will allow for the liquidation of collateral to be instantly settled at a price, and will act as an "order book" to cover liquidations.

Core Lending Contract

This smart contract will handle the day to day running of Skynet on the blockchain, and complete the following tasks:

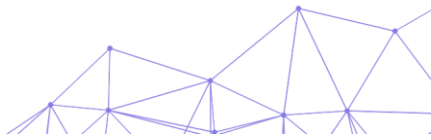
1) Allows for the creation and funding of a crypto-loan – users will interact directly with our smart contract, sending their crypto (collateral) and creating a loan on-chain. The creation process must be signed by the core Skynet key to ensure that the loan cones from the Skynet platform.
2) Cancelling a loan – a loan can be cancelled if the status remains "Pending" and 7 days have passed since the creation of the loan.
3) Accepting a loan – when a loan request becomes funded, Skynet will send a message to the contract which accepts the loan. This makes the loan-contract live.
4) Repayments – periodically, as a borrower makes repayments, Skynet will mark this against the loan in the contract to update the on-chain tracking of the loan.
5) Liquidation – Skynet can request the smart contract to liquidate the loan to ensure the correct LVRs are being maintained – this is all done internally, and is triggered by our oracle.
6) Close Loan – Skynet can also close a loan, liquidating and covering the remaining balance of a loan.

All liquidated tez or sent to the Exchange contract, which is where we can convert the crypto into fiat to repay the lender.

```
const key_hash LOAN_ADMIN "tz1...";
const nat MAINTAIN_RATE 2;
const account EXCHANGE_ACCOUNT "TZ1...";
object Loan{
  account borrower,
  nat balance,
  tez collateral,
  timestamp started
};
storage map(string=>Loan) loans;
storage nat usdRate;
storage tez xtzRate;
storage tez exchangeBalance;
depositCollateral(string loanId){
  if (AMOUNT <= tez 0) fail;
  if (!storage.loans.in(input.loanId)) fail;
  Loan loan = storage.loans.get(input.loanId);
  loan.collateral = add(loan.collateral, AMOUNT);
  storage.loans.push(input.loanId, loan);
}
withdrawCollateral(string loanId, tez amount){
  if (!storage.loans.in(input.loanId)) fail;
  Loan loan = storage.loans.get(input.loanId);
  if (SENDER != loan.borrower) fail;
  if (input.amount > loan.collateral) fail;
  var tez newCollateral = sub(loan.collateral, input.amount);
  var nat newCollateralUsd = mul(newCollateral, storage.usdRate);
  if (newCollateralUsd <= mul(loan.balance, MAINTAIN_RATE)) fail;
  loan.collateral = newCollateral;
  storage.loans.push(input.loanId, loan);
  pay(loan.borrower, input.amount);
}

newLoan(string loanId, nat balance){
  if (storage.loans.in(input.loanId)) fail;
  var nat newCollateralUsd = mul(AMOUNT, storage.usdRate);
  if (newCollateralUsd < mul(input.balance, MAINTAIN_RATE)) fail;
  storage.loans.push(input.loanId, new Loan(SENDER, input.balance, AMOUNT, NOW));
}
updateRates(nat usdRate, tez xtzRate){
```

```
  if (MANAGER != LOAN_ADMIN) fail;
  stroage.usdRate = input.usdRate;
  stroage.xtzRate = input.xtzRate;
}
loanRepayment(string loanId, nat amount){
  if (MANAGER != LOAN_ADMIN) fail;
  if (!storage.loans.in(input.loanId)) fail;
  var Loan loan = storage.loans.get(input.loanId);
  if (input.amount > loan.balance) fail;
  if (input.amount > loan.owed){
    loan.owed = nat 0;
  } else {
    loan.owed = _toNat(sub(loan.owed, input.amount));
  }
  loan.balance =  _toNat(sub(loan.balance, input.amount));
  storage.loans.push(input.loanId, loan);
}
closeLoan(string loanId){
  if (MANAGER != LOAN_ADMIN) fail;
  if (!storage.loans.in(input.loanId)) fail;
  Loan loan = storage.loans.get(input.loanId);
  storage.loans.drop(input.loanId);
  var tez owedXtz = mul(loan.balance, storage.xtzRate);
  if (owedXtz > loan.collateral) owedXtz = loan.collateral;
  storage.exchangeBalance.add(owedXtz);
  var tez repayXtz = sub(loan.collateral, owedXtz);
  pay(loan.borrowerAccount, repayXtz);
  pay(EXCHANGE_ACCOUNT, storage.exchangeBalance);
  storage.exchangeBalance = 0;
}
```

Token Contract

There is currently no standardised token contract for Tezos, however we will be working directly with developers on such a standard. This contract can be viewed below – we will look to add additional functionality to this basic standard in the future.

```
storage map(address=>nat) balances;
storage string name;
storage string sumbol;
storage nat decimals;
public transfer(address to, nat amount, ?bytes contractData){
  assert(storage.balances.in(SENDER)) string "No token balance found";
  assert(input.to != SENDER) string "Sender is receiver";
  let nat senderBalance = storage.balances.get(SENDER);
  assert(senderBalance < input.amount) string "Balance is to low";
  let nat receiverBalance = nat 0;
  if (storage.balances.in(input.to)){
    receiverBalance = storage.balances.get(input.to);
  }
  senderBalance = sub(senderBalance, input.amount);
  senderBalance = abs(senderBalance);
  receiverBalance.add(input.amount);
  storage.balances.push(SENDER, senderBalance);
  storage.balances.push(input.to, receiverBalance);
  isset(input.contractData){
    transfer(mutez 0, input.to, input.contractData);
  }
}
```

Exchange Contract

The exchange contract will act like an order book, allowing Skynet to liquidate collateral as required. This area is still being worked on, but will act much like an exchange buy order book.

Fiat Partner

One aspect we are still working on is the integration with a fiat-partner – this will allow users to deposit funds into their Skynet account for the following purposes:

1) To fund an offer toward a loan request (i.e. to become a lender)
2) To make a repayment toward a loan (borrower)

This integration may take some time depending on the backend systems and procedures required, however we have full faith that we can complete this integration within a timely manner.

We will look at eventually tokenizing fiat deposits and withdraws, allowing tracking such transactions on the blockchain. This provides a more transparent and auditable trail, however is not a type of “Stable coin”.

How it works

Skynet pairs lenders and borrowers through the blockchain – initially we will allow the lending of fiat currencies to borrowers who can stake collateral in the form of crypto-currency. We will accept XTZ (Tezos) and NET (the Skynet token).

Users will be required to maintain a certain Loan to Value ratio (LVR), based on the crypto currency being used as collateral (essentially, the more volatile a currency is, the higher the LVR will be). Typically, users will need to maintain a LVR or approximately 66% - this means that will need to hold about 1.5x the value borrowed as crypto-currency.

Repayments are calculated and spread evenly over the length of the loan – missing a payment incurs a small fee. Missed payments will simply be liquidated from the collateral until the LVR is met.

As crypto-currency values moves up and down, so will the LVR – the SkyNet platform will track this changes, and make excess collateral available for withdraw when the underlying crypto increases. SkyNet will also trigger liquidations of small amounts of the collateral when the value drops.

All of the liquidation and management of the collateral is handled by the core lending smart contract, and will provide a “trust-less” service to our users.

All repayments incur interest, which is decided upon between the borrower and lender(s), and a small service fee of 1% p.a. (subject to change).

Future Plans

In future, we want to achieve the following goals for SkyNet:

- Provide loans, secured with homes – we are currently exploring the potential of having titles stored on the blockchain
- Provide loans, secured with other assets (e.g. car loans)
- Unsecured lending, peer-to-peer
- Insurance services
- Debit card services (for accessing NET tokens directly)