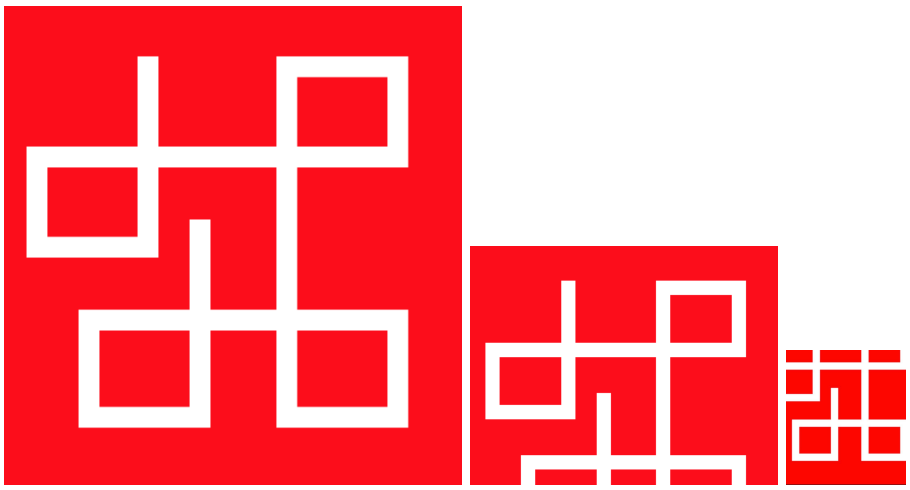
CryptoCount Project Proposal

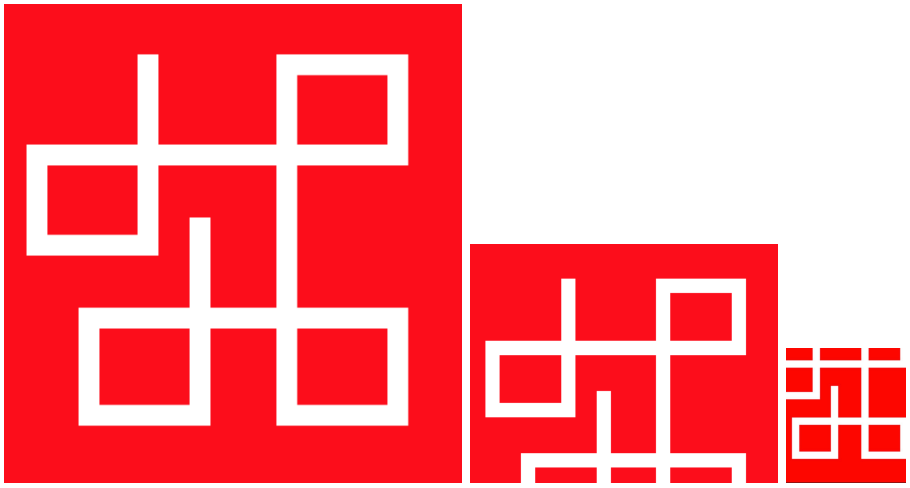


Author: Henrik Moe

Table Of Contents:

Note: The previous name of this project was “PoS Taxation” or “PoS Tax” for short. As of Version 2, the project has been renamed “CryptoCount”. There will be references to ‘PoS Tax’ and ‘CryptoCount’ in this document. The two names should be read as synonyms.

CryptoCount Introduction

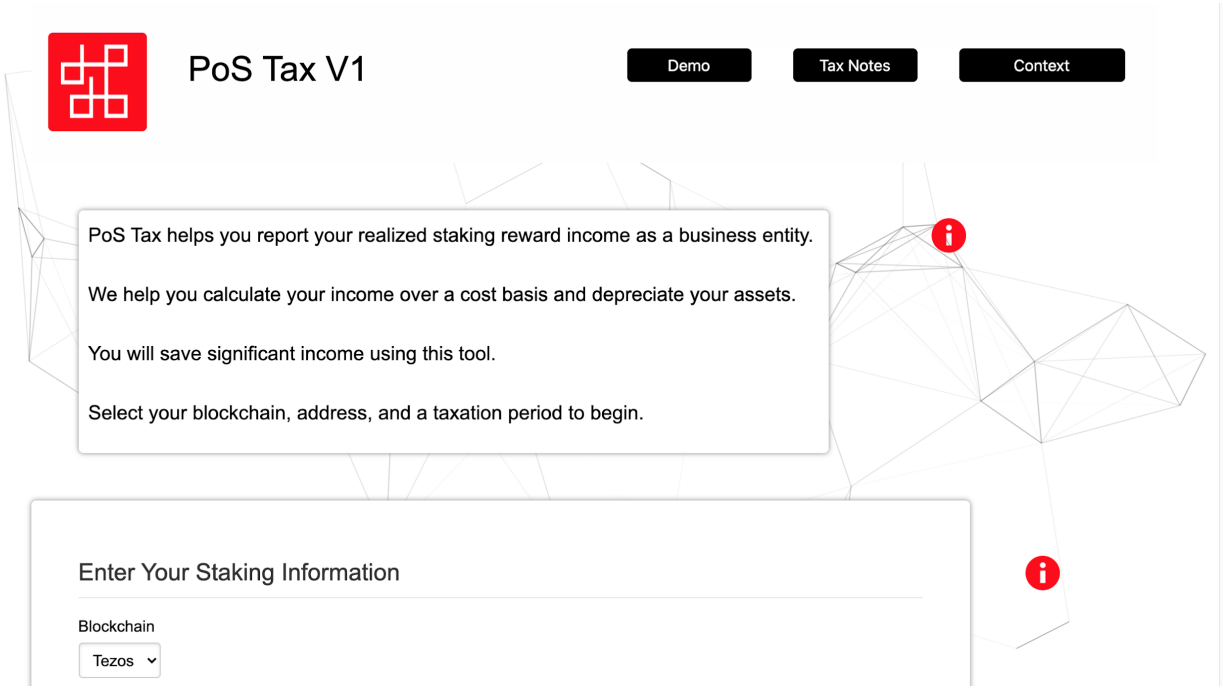


Background

Who we are what our purpose is

We are officers acting for the interests of the Tezos stakeholders and the Tezos community.

We have developed Proof of Stake Tax V1 below: https://postaxation.com



V2 moving environment configuration.

Properly taxing cryptocurrency reward income is a hot topic right now.

There are delegators that should use our accounting to get their right income.

Delegators exist on a spectrum

Hard core - guys that have dense delegation schemes

Middle - ppl that find their delegates and send em money

Barely involved - ppl who just delegate to their exchange’s baker

V2 environment hanging docks for idea/movement synchronization

In order to get this issue to the forefront of the industry, we will use all available dock points from the xtz economy to the fiat economy. Our nature as an open source software will help us integrate into industry positions without any costs to other people's positions. These places are where users spend most of their time. It is also the place to target new users that are just entering cryptocurrency.

API client docks are places where our APIs can connect to commercial clients and boost the effectiveness of them.

We have three content delivery methods to connect to users:

First, There are big exchanges that hold people's wallets. These platforms are where XTZ is realized in fiat. Integrating our use cases into the UX of exchanges through partnership and technical support of the exchange team increases our number of users and awareness. (more on implementation in the to market plan)(client integration module development in the development plan).

Second, we have the site, we have a home base where delegators or people interested in delegating can come and use our tools directly.

Third, the plug in app, we have a mobile home base that will always be with the users as an extension when they are surfing the web. The plug in app makes us more readily available and will allow users to get even more comfortable with our product.

Core Tech Stack

Our core tech stack is made up of the following components:

Interface: HTML, SCSS, Javascript, Bootstrap, CSS, CDN

Server: Node JS, Express JS, Socket IO, RESTful API, MONGODB

Supporting Tech Stack

Our supporting tech stack is made up of the following components:

Version Control System: Git

Communication: Email, WhatsApp

CryptoCount Core Technical Plan

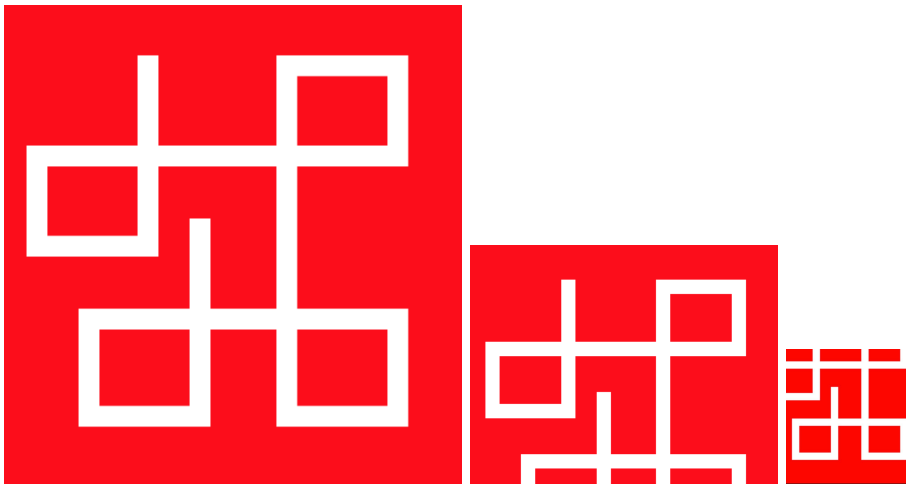


Table Of Contents:

b. Prospective Staker Website
c. Accounting Analysis Client
d. Prospective Staker Client
e. Signature Tag Client
f. Plug In App

1. **Use Case Requirements And Backgrounds**

    a. **Accounting Analysis Background**

    The User: I'm a delegator. I have a bunch of XTZ sitting in my coinbase and I want it to make me more XTZ. So, I sent it all to this baking address. These guys pay me all the time. I've watched my balance in Coinbase increase. I sold 500 of my rewarded XTZ from my coinbase account today. You're a website that will discount my income accounting? Great!

    b. **Prospective Staker Background**

    The user: I'm a prospective staker. I have little sense of what XTZ is but I hear that some of these cryptocurrencies can make me money by just putting it into the blockchain or something? Anyway kickbacks on my money sound great. How much can I get staking with Tezos? Your website has a tool that can show me an expected return if I do this? Great!

    c. **Client Module Background**

    The user: I'm a common cryptocurrency holder that keeps my assets on an exchange. I am interested in staking / or I am staking and I want to get my correct accounting without ever leaving the exchange. Both of your products are already integrated into the UX on the exchange? Great!

    d. **Client Module Integration Background**

    The user: I'm a developer for a big exchange. I do the front end web page work that the top authorizes. I need to know how to plug in PoS Tax to our platform. You have a manual and the whole process is super easy? Great!

    e. **Plug In App Background**

    The user: I'm a delegator / prospective delegator / an person interested in blockchain asset management. I hear you have a browser plug in app. The app allows me to use your two products at any time? Great!

2. **Functional and DataBase Requirements**

    a. **Performance**

    2.a.1 Throughput for requests no matter how large the account should not exceed 5 seconds.

2.a.2 DOM modeling must be made efficient

2.a.3 CDN routing must be made efficient

2.a.4 plug-in app module must be made efficient

b. Tzkt Special Request Path

2.b.1 Our system performance is capped at a request limit to tzkt

2.b.2 The system will connect to tzkt through a special endpoint arranged between our two organizations. This will allow for faster throughput and better system performance. We will likely pay Tzkt for this API.

c. Multiple Fiat Selection

2.c.1 The user will be able to enter a fiat of their preference into the system to return their results.

2.c.2 There will be 20 fiats in V2.

2.c.3 The fiats come from the coingecko data source

d. User Accounts

2.d.1 The user will be able to return to the system with their results preloaded

2.d.2 The system will check if it wrote the results into their browser

2.d.3 The system will propose a button to go to page 2 with their results

2.d.4 The system will keep track of the quantity realized history the user has inputted to always have an accurate depiction of their reward accounting entrant set.

e. Security

2.e.1 The system will have defense against DDoS attacks

2.e.2 The system will have to keep passwords safe and email addresses safe. Through encryption.

3. Use Case Normal Courses

a. Accounting Analysis Normal Course

The user wants to enter their delegation account and get their accounting done.

User Inputs: address, basis date, quantity Realized, fiat

The server gets all user data from data points.

The server performs analysis calculations on the data.

The server sends results back to the front end for rendering and display.

b. Prospective Staker Normal Course

The user wants to enter hypothetical investment amounts and see their expected rewards.

User enters an amount they would stake in XTZ

User enters their Fiat

System calculates the expected yield they would get annually.

Return statements display to user in front end display

Popup box explains our calculation method

c. Client Module Normal Course

User is on an exchange or some third party website/app that they use to interact with the Tezos blockchain.

User navigates to a dialogue box or triggers a dialog box that opens and holds our value products

The user uses our value products without going to our site.

The dialog box is signed with our name and a link to our site.

d. Client Module Integration Normal Course

Add postax.js cdn script tag to your webpage.

Add a canvas tag with the id=’postax’

PoS Tax will route all information and render it within the canvas of the tag

You can also configure your own postax.js script by downloading the file from the website.

We request you add a canvas tag with the id=’postaxSignature’ in this event. This signature is like a watermark that will take users back to our site.

e. Plug In App Normal Course

User downloads the app from the plug in store on from the browser, ex chrome apps

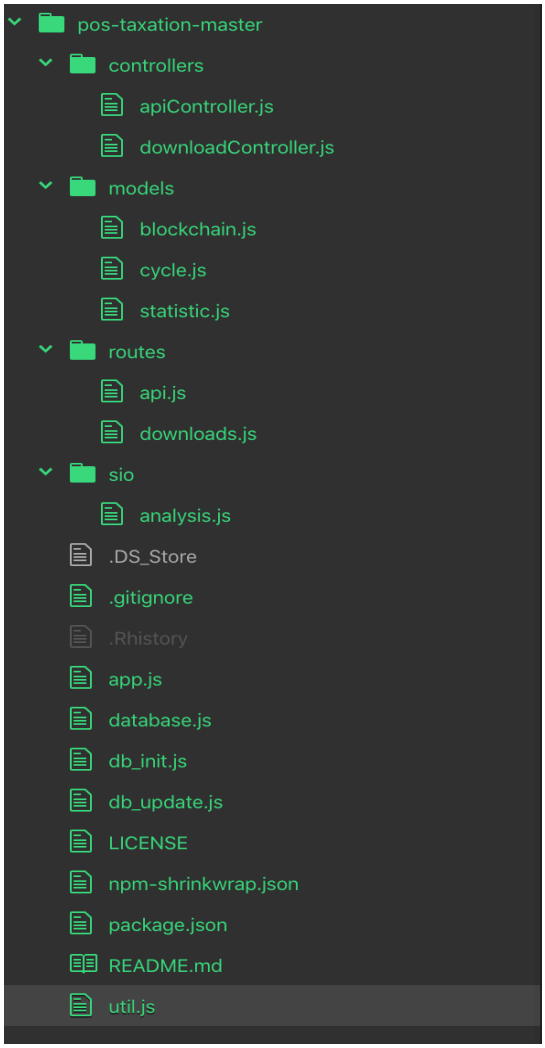
The user then opens the app and the walkthrough display is like the client module.

The app can be opened and operated on any time the user is web browsing

4. Server System Events With Code

The V1.0 Directory structure will be evolved as a working structure for V2.0. Below, the V1.0 server directory is displayed.

There are two API connection methodologies in version 1. There is a RESTful API and a Socket.io API. The following is the directory structure, /routes/api/ is the RESTful API and /sio/analysis.js is the Socket.io API:



The following event cases are based on the user experience and the dependencies that arise to maintain it.

Each of the event sets are organized as branches of development in our version control system git. That means that each of these events will be developed parallel with each other. More information about our development process is available in the development plan section of this document.

a. Accounting Analysis Server Events

1.

Event	Send User Data
Description	Emission from front end client of inputs to server
Location	fe/js/socket.js
Primary Actor	User
Trigger	User enters fields and selects go
Dependencies	The user has entered information and that information has been assigned local storage
Normal Course	1. User enters information in fields in front end 2. The fields populate memory spaces 3. The memory spaces get emitted over the socket connection when triggered by the front end main
Output	Message over the socket
Alternative Courses	1. A pending socket connection with no on message event sent through because the user has not entered anything
Exceptions	1. The user navigates to the context page or the demo page. 2. The user did not enter all of the fields and must enter all of the fields
Example	Socket.io: a. socket.emit({address, fiat, qRealized, startdate, enddate})

2.

Event	Server Data Catching
Description	Data caught from client connection
Location	be/sio/analysis.js
Primary Actor	Node/Express JS App
Trigger	Socket listening and receives payload from a client
Dependencies	API connection, either REST or socket
Normal Course	1. Data is emitted over the socket connection from the client 2. The socket is configured to catch the message 3. The handling event of the catch caches the data into the server instance. 4. The data is checked for irregularities
Output	start date, end date, address, qRealized, fiat written into instance memory
Alternative Courses	Demo button hit, redirect and auto populate displays
Exceptions	Demo button hit, redirect and auto populate displays 1. On data is checked for irregularities: A start date and end date crossover error in

	the incoming data is detected. a. An error message is thrown back to the front end 2. A start date and end date out of existence range error in the incoming data is detected. b. An error message is thrown back to the front end
Example	socket.on('message')=>{address = address, startdate = startdate, enddate = enddate, qRealized = qRealized, fiat = fiat }

3.

Event	Date To Cycles Query To DB
Description	Get proper cycle set from database
Location	be/app.js be/sio/analysis.js
Primary Actor	Node/Express JS App
Trigger	New day 00:00UTC
Dependencies	start date and end date From user
Normal Course	1. Get cycle list object from db 2. Make cycle set for analysis object 3. Integrate into analysis

	payload
Output	cycle query set into url
Alternative Courses	none
Exceptions	none

Example	be/sio/analysis.js:304->309
	
	const cycleDocs = await CycleModel.find();
	let cycles = {};
	for (let i = 0; i < cycleDocs.length; i++) {
	const d = Math.floor(new Date(Date.parse(cycleDocs[i].dateString)).getTime() / (1000 * 60 * 60 * 24));
	cycles[d] = cycleDocs[i].cycleNumber;
	}

4.

Event	Fiat set query
Description	get entered fiat set from database
Location	be/app.js be/sio/analysis.js be/models/blockchain.js
Primary Actor	Node JS app.js
Trigger	Analysis is being conducted
Dependencies	startdate, enddate, fiat
Normal Course	1. Import the database model 2. Add parameter for specific fiat set
Output	price set for the selected fiat

Alternative Courses	none
Exceptions	none

Example:	be/sio/analysis.js:212->218
	
	//note this example uses the v1.0 database model that exclusively holds USD prices
	
	const priceDocs = await BlockchainModel.find();
	let prices = {};
	// convert document to dictionary for better find performance (date -> int: price -> number; date -> int: marketCap -> number)
	for (let i = 0; i < priceDocs.length; i++) {
	const d = Math.floor(priceDocs[i].date.getTime() / (1000 * 60 * 60 * 24));
	prices[d] = priceDocs[i].price;
	}

5.

Event	Supply set query
Description	Get the supply set from the database
Location	be/app.js be/sio/analysis.js be/models/statistic.js
Primary Actor	Node JS
Trigger	Analysis is being conducted
Dependencies	the database storing the daily supply data, start date, enddate
Normal Course	1. Import supply statistic

	model from database 2. Integrate into analysis
Output	supply set into analysis
Alternative Courses	none
Exceptions	none

Example	be/sio/analysis.js:237->244
	
	const supplyDocs = await StatisticModel.find();
	let totalSupplys = {};
	let marketCaps = {};
	for (let i = 0; i < supplyDocs.length; i++) {
	const d = Math.floor(new Date(Date.parse(supplyDocs[i].dateString)).getTime() / (1000 * 60 * 60 * 24));
	totalSupplys[d] = supplyDocs[i].totalSupply;
	marketCaps[d] = (d in prices) ? totalSupplys[d] * prices[d] : 0;
	}

6.

Event	Market Value Query
Description	Get the fiat market value set from the database
Location	be/app.js be/sio/analysis.js be/models/statistic.js
Primary Actor	Node JS
Trigger	Analysis is being conducted
Dependencies	fiat, the database storing the market value set

Normal Course	1. Import the database statistic model 2. Perform maket value calculator for each day with price and supply
Output	market value set
Alternative Courses	none
Exceptions	none

Example	be/sio/analysis.js:237->244
	
	const supplyDocs = await StatisticModel.find();
	let totalSupplys = {};
	let marketCaps = {};
	for (let i = 0; i < supplyDocs.length; i++) {
	const d = Math.floor(new Date(Date.parse(supplyDocs[i].dateString)).getTime() / (1000 * 60 * 60 * 24));
	totalSupplys[d] = supplyDocs[i].totalSupply;
	marketCaps[d] = (d in prices) ? totalSupplys[d] * prices[d] : 0;
	}

7.

Event	Reward Set and Baker Info Query TZKT
Description	Obtain the reward set, the baker address, and the baker alias from data source
Location	be/app.js be/sio/analysis.js

Primary Actor	Node JS
Trigger	Analysis is being conducted
Dependencies	address,
Normal Course	1. Query tzkt with proper URLs 2. Build reward object
Output	Reward object with reward set in xtz and baker alias and baker address
Alternative Courses	none
Exceptions	none

Example	be/sio/analysis.js:6->24
	
	//the url in the example below is not proper the proper url for this query is: https://api.tzkt.io/v1/rewards/delegators/{address}/{cycle}
	
	async function getRewards(address)
	{
	let rewards = [];
	let flowins = {};
	let flowouts = {};
	let lastId = 0;
	while (true) {
	try {
	let url = `https://api.tzkt.io/v1/accounts/${address}/operations?type=endorsement,baking,nonce_revelation,double_baking,double_endorsing,transaction,origination,delegation,reveal,revelation_penalty&lastId=${lastId}&limit=1000&sort=0`;
	// let url = `https://api.tzkt.io/v1/accounts/${address}/operations?lastId=${lastId}&limit=1000&sort=0`;
	const response = await axios.get(url);

	lastId = response.data[response.data.length - 1].id; // update lastId
	for (let i = 0; i < response.data.length; i++) {
	const element = response.data[i];
	if ('endorsement' === element.type) {
	rewards.push({
	type: 'endorsement',
	timestamp: new Date(Date.parse(element.timestamp)),
	amount: element.rewards
	});
	}

8.

Event	Basis Obtained
Description	The system obtains the user's basis in order to perform accounting later
Location	be/app.js be/sio/analysis.js
Primary Actor	Node JS
Trigger	Analysis being conducted
Dependencies	users address, start date, fiat[startdate]
Normal Course	3. Price set obtainted from db 4. Price set value on start date is isolated and used for calculations
Output	Basis value for calculations
Alternative Courses	none
Exceptions	none

Example	be/sio/analysis.js:222->223
	
	let cashMethods = {};
	for (let i = 0; i <= days; i++) {
	const d = startDateD + i;
	if (d in prices) {
	if (d in rewards) {
	cashMethods[d] = prices[d] * rewards[d];
	}
	else {
	cashMethods[d] = prices[d] * 0; // days which don't have reward
	}
	}
	}

9.

Event	CV Basis Calculation
Description	calc the reward set over the cash basis
Location	be/app.js be/sio/analysis.js
Primary Actor	Node JS
Trigger	Analysis is being conducted
Dependencies	reward set, fiat price on startdate
Normal Course	1. Obtain the reward set 2. Obtain the fiat price on the start date 3. Calculate
Output	rewardset over basis
Alternative Courses	none
Exceptions	none
Example	be/sio/analysis.js:222->223

	let cashMethods = {};
	for (let i = 0; i <= days; i++) {
	const d = startDateD + i;
	if (d in prices) {
	if (d in rewards) {
	cashMethods[d] = prices[d] * rewards[d];
	}
	else {
	cashMethods[d] = prices[d] * 0; // days which don't have reward
	}
	}
	}

10.

Event	Dilution Analysis On Basis Set
Description	perform the accounting analysis
Location	be/app.js be/sio/analysis.js
Primary Actor	Node JS
Trigger	Analysis is being conducted
Dependencies	fiat price set, supply data set, fiat market value set, reward set over basis, basis
Normal Course	1. Analysis function triggered 2. Transform the cash basis set into the depletion method 3. Transform the cash basis set into the market value dilution set
Output	depletion and mv dilution accounting compounding cv basis set.

Alternative Courses	none
Exceptions	none

Example	be/sio/analysis.js:246->275
	
	//this example is the depletion calculation
	
	let depBookValues = {};
	for (let i = 0; i <= days; i++) {
	const d = startDateD + i;
	if (i === 0) {
	depletions[d] = 0;
	depBookValues[d] = prices[d] * balances[d]; // assume the user gets initial balance at the start date it inputs
	}
	else {
	let flowValue = 0; // it can be positibe; it can be negative too
	// let depBookValueChange = 0;
	
	if (d in flowins) {
	flowValue += flowins[d];
	}
	if (d in flowouts) {
	flowValue -= flowouts[d];
	}
	if (flowValue > 0) {
	const dayInitBookValue = depBookValues[d - 1] + Math.abs(flowValue) * prices[d - 1]; // assume the flow-in happened in the first second of the day
	depletions[d] = -dayInitBookValue * (1 - totalSupplys[d - 1] / totalSupplys[d]); // this is a negative value
	depBookValues[d]

	= dayInitBookValue + cashMethods[d] + depletions[d]; // assume the reward comes at the end of the day
	} else {
	const dayInitBookValue = (1 - Math.abs(flowValue) / balances[d - 1]) * depBookValues[d - 1]; // assume the flow-out happened in the first second of the day
	depletions[d] = -dayInitBookValue * (1 - totalSupplys[d - 1] / totalSupplys[d]); // this is a negative value
	depBookValues[d] = dayInitBookValue + cashMethods[d] + depletions[d]; // assume the reward comes at the end of the day
	}
	// depletions[d] = -depBookValues[d - 1] * (1 - totalSupplys[d - 1] / totalSupplys[d]); // this is a negative value
	// depBookValues[d] = depBookValues[d - 1] + cashMethods[d] + depletions[d] - depBookValueLoss;
	}
	}

11.

Event	Fifo analysis
Description	perform fifo on top of the analysis for the quantity realized
Location	be/app.js be/sio/analysis.js
Primary Actor	Node JS
Trigger	Analysis is being conducted

Dependencies	depletion, mv dilution sets, basis set, reward quantity set, qRealized
Normal Course	1. The analysis sets have been transformed 2. Take the first in values of the transformed sets and return them to the user up to the amount of the quantity realized 3. Push the returned value to the analysis object
Output	the first in first out accounted sets
Alternative Courses	none
Exceptions	none

Example	be/sio/analysis.js:312->326
	
	let analysisResults = [];
	for (let i = 0; i < days; i++) {
	const d = Math.floor(startDate.getTime() / (1000 * 60 * 60 * 24)) + i;
	const d_obj = new Date(d * 1000 * 60 * 60 * 24);
	let element = {
	date: d_obj.toISOString().substring(0, 10),
	cycle: (d in cycles ? cycles[d] : null),
	balance: (d in balances ? balances[d] / 1000000 : null),
	reward: (d in rewards ? rewards[d] / 1000000 : 0),
	cashMethod: (d in cashMethods ? cashMethods[d] / 1000000 : null),
	depMethod: (d in depMethods ? depMethods[d] / 1000000 : null),

	MVDMethod: (d in MVDMethods ? MVDMethods[d] / 1000000 : null)
	}
	
	analysisResults.push(element);
	}
	
	//new code
	
	fifo(qRealized){
	For values in sets range 1-qRealized(i++){
	Fifoset3 += rewsetOverBasis[
	Fifoset1 +=Depletionset[
	Fifoset2 += mvidilutionset[
	}

12.

Event	Lifo analysis
Description	perform lifo on top of the analysis for the quantity realized
Location	be/app.js be/sio/analysis.js
Primary Actor	Node JS
Trigger	Analysis being conducted
Dependencies	depletion, mv dilution sets, basis set, reward quantity set
Normal Course	1. The analysis sets have been transformed 2. Take the last in values of the sets and present them as the realized value (for loop up to realized value)

	3. Push the returned value to the object payload
Output	the last in first out set
Alternative Courses	none
Exceptions	none

Example	be/sio/analysis.js:312->326
	
	let analysisResults = [];
	for (let i = 0; i < days; i++) {
	const d = Math.floor(startDate.getTime() / (1000 * 60 * 60 * 24)) + i;
	const d_obj = new Date(d * 1000 * 60 * 60 * 24);
	let element = {
	date: d_obj.toISOString().substring(0, 10),
	cycle: (d in cycles ? cycles[d] : null),
	balance: (d in balances ? balances[d] / 1000000 : null),
	reward: (d in rewards ? rewards[d] / 1000000 : 0),
	cashMethod: (d in cashMethods ? cashMethods[d] / 1000000 : null),
	depMethod: (d in depMethods ? depMethods[d] / 1000000 : null),
	MVDMethod: (d in MVDMethods ? MVDMethods[d] / 1000000 : null)
	}
	
	analysisResults.push(element);
	}
	
	//new code
	//new lifo logic needed
	Lifo(qRealized){

	For values in sets range 1-
	qRealized(i++){
	Lifoset3 += rewsetOverBasis[
	Lifoset1 +=Depletionset[
	Lifoset2 += mvidilutionset[
	}

13.

Event	Weight Avg Analysis
Description	perform weighted average on top of the analysis for the quantity realized
Location	be/app.js be/sio/analysis.js
Primary Actor	Node JS
Trigger	Analysis is being conducted
Dependencies	depletion, mv dilution sets, basis set, reward quantity set
Normal Course	1. The analysis sets have been transformed 2. Take the average value of the sets 3. Push the average of QRealized to the object
Output	the weighted average set
Alternative Courses	none
Exceptions	none

Example	be/sio/analysis.js:312->326
	
	let analysisResults = [];
	for (let i = 0; i < days; i++) {
	const d = Math.floor(startDate.getTime() / (1000 * 60 * 60 * 24)) + i;

	const d_obj = new Date(d * 1000 * 60 * 60 * 24);
	let element = {
	date: d_obj.toISOString().substring(0, 10),
	cycle: (d in cycles ? cycles[d] : null),
	balance: (d in balances ? balances[d] / 1000000 : null),
	reward: (d in rewards ? rewards[d] / 1000000 : 0),
	cashMethod: (d in cashMethods ? cashMethods[d] / 1000000 : null),
	depMethod: (d in depMethods ? depMethods[d] / 1000000 : null),
	MVDMethod: (d in MVDMethods ? MVDMethods[d] / 1000000 : null)
	}
	
	analysisResults.push(element);
	}
	
	//new code
	//new avg logic needed
	Avg(qRealized){
	For values in sets range 1-qRealized(i++){
	Avgset3 += rewsetOverBasis[
	Avgset1 +=Depletionset[
	Avgset2 += mvidilutionset[
	}

14.

Event	Optimized Analysis

Description	perform optimized on top of the analysis for the quantity realized
Location	be/app.js be/sio/analysis.js
Primary Actor	Node JS
Trigger	Analysis is being conducted
Dependencies	depletion, mv dilution sets, basis set, reward quantity set
Normal Course	1. Analysis has been conducted and the reward sets have been transformed 2. Find the lowest values of the available entries 3. Return these values up to the qRealized
Output	the minimum value set
Alternative Courses	none
Exceptions	none

Example	be/sio/analysis.js:312->326
	
	let analysisResults = [];
	for (let i = 0; i < days; i++) {
	const d = Math.floor(startDate.getTime() / (1000 * 60 * 60 * 24)) + i;
	const d_obj = new Date(d * 1000 * 60 * 60 * 24);
	let element = {
	date: d_obj.toISOString().substring(0, 10),
	cycle: (d in cycles ? cycles[d] : null),
	balance: (d in balances ? balances[d] / 1000000 : null),
	reward: (d in rewards ? rewards[d] / 1000000 : 0),

	cashMethod: (d in cashMethods ? cashMethods[d] / 1000000 : null),
	depMethod: (d in depMethods ? depMethods[d] / 1000000 : null),
	MVDMethod: (d in MVDMethods ? MVDMethods[d] / 1000000 : null)
	}
	
	analysisResults.push(element);
	}
	
	//new code
	//new optimized logic needed
	Opt(qRealized){
	For values in sets range 1-qRealized(i++){
	Optset3 += rewsetOverBasis[
	Optset1 +=Depletionset[
	Optset2 += mvidilutionset[
	}

15.

Event	Emit To Client
Description	wrap together and send to socket
Location	be/app.js be/sio/analysis.js
Primary Actor	Node JS
Trigger	Analysis Done
Dependencies	fifo set, lifo set, weightAvgSet, minValueSet

Normal Course	1. The analysis completes 2. The resulting object is emitted over the web socket
Output	Socket packet of data
Alternative Courses	None
Exceptions	Error messages: 1. Analysis Response Error: 'please check input params' 2. Analysis Response Error: 'this address does not have a balance history' 3. Analysis Response Error: 'The start date is earlier than the first balance day
Example	be/sio/analysis.js: 327 socket.emit('analysisRes', analysisResults);

16.

Event	Fe Catching
Description	Catch the emitted values process in the front end
Location	fe/js/socket.io
Primary Actor	Front end web client
Trigger	Server completes analysis and responds to client over the web socket
Dependencies	Socket catches data packet
Normal Course	1. Analysis response is caught

	2. The payload of the response is parsed through 3. The parsed payload is assigned to local memory
Output	Data cached/written to memory
Alternative Courses	Analysis response error: The error messages from the analysis server program are shown on the web page alert menu.
Exceptions	none

Example	fe/js/socket.js:88->97
	
	socket.on('analysisRes', (data) => {
	console.log('got analysis response');
	//iterate through socket object json payload and push elements into the vectors
	$.each(data, function(i, result){
	
	date.push(result.date.toString().slice(0,10))
	
	cashValue.push(result.cashMethod) //whatever the stuff is called in the json
	
	rewDepl.push(result.depMethod)
	
	rewDil.push(result.MVDMethod)
	
	rewardQuantity.push(result.reward)
	})

17.

Event	Fe Render
Description	render the output of the process in the front end
Location	fe/page2.html
Primary Actor	Web Browser Compiler
Trigger	The data has arrived and is good to display on the analysis page
Dependencies	Data values in memory cache/ local storage
Normal Course	1. The socket connection program hs checked that the data is in storage 2. Page 2 is triggered 3. Elements of page 2 display the values in local storage
Output	Web Page of Analysis results
Alternative Courses	none
Exceptions	none

Example	fe/js/socket.js:151->159
	
	var dilutionRewardAccumulated = []
	
	dilutionRewardAccumulated.push(parseFloat(splitDilutionVector[0]))
	for(i = 1; i < splitDilutionVector.length; i++){
	
	dilutionRewardAccumulated.push(dilutionRewardAccumulated[i-1] + parseFloat(splitDilutionVector[i]))
	}

	localStorage.setItem('dilutionRewardAccumulated', dilutionRewardAccumulated) window.location.href = "./page2.html"

18.

Event	User Sends Back The Accounting Method They Used
Description	The user sends back which accounting method they used for their reporting. This for the eligible accounting set calculation for remembered use.
Location	fe/js/sendback.js
Primary Actor	User
Trigger	The User selects a method they used and clicks send
Dependencies	The user has already gotten their results and has decided which accounting method they use (Fifo, lifo, etc)
Normal Course	1. The user selects which mechanism they used and press send 2. The socket emits the accounting method to the server 3. The caught response is fed into the db
Output	Analysis used socket packet
Alternative Courses	The user does not comply and sends nothing
Exceptions	none

Example	fe/js/sendback.js
	//needs to be built
	socket.emit('analysisUsed'){
	optimized}

5.a Summary: Sections 1, 16 - 18 are client processes. 2-15 are server processes. Events 2-8 are data obtaining events. Events 9-10 are the main transformation events. Events 11-14 are accounting events for the quantity realized.

b. Prospective Staker Server Events

1.

Event	Socket Emission
Description	Emission from front end of inputs to backend. User enters their information into the prospective staking product
Location	fe/js/socket.io
Primary Actor	User
Trigger	User enters their information
Dependencies	The user has selected the prospective staker product
Normal Course	1. The user enters their prospective information 2. The socket emits the information
Output	New server execution path/instance. Staking amount, fiat over the socket
Alternative Courses	None
Exceptions	None
Example	fe/js/socket.io //hypothetical

	socket.emit('prospectiveStaker'{quantity: quantity, currency: currency}

2.

Event	Price Query
Description	get prices from database
Location	be/app.js be/sio/prospective.js
Primary Actor	Node JS
Trigger	The socket connection has called on prospective analysis
Dependencies	Fiat, last day
Normal Course	1. Prospective analysis is triggered 2. Query the database for the price in the currency
Output	fiatBasisPrice
Alternative Courses	none
Exceptions	none

Example	be/sio/prospective.js:
	
	//hypothetical example missing the currency selection
	
	Add all currencies to the blockchain model in the db_update
	
	const priceDocs = await BlockchainModel.find();
	let prices = {};
	// convert document to dictionary for better find performance (date -> int: price -> number; date -> int:

	marketCap -> number)
	for (let i = 0; i < priceDocs.length; i++) {
	const d = Math.floor(priceDocs[i].date.getTime() / (1000 * 60 * 60 * 24));
	prices[d] = priceDocs[i].price;
	}

3.

Event	Staked Amount Calculation
Description	translate prospective staking amount from fiat
Location	be/app.js be/sio/prospective.js
Primary Actor	Node Js
Trigger	Prospective analysis underway
Dependencies	fiatBasisprice, qFiat/staking amount
Normal Course	1. selectedCurrencyBasis * qFiatStaked
Output	stakedqXTZ
Alternative Courses	none
Exceptions	Error message: Invalid input error if (qStaked <= 0)

Example	be/sio/prospective.js
	
	f(){
	qXTZ = fiatBasisPrice * qFiat
	}

4.

Event	Stake Yield Calculation
Description	return calculation on the staked amount
Location	be/
Primary Actor	Node JS
Trigger	Prospective staking analysis is underway
Dependencies	fiatBasisPrice, StakedqXTZ
Normal Course	1. The expected amount of xtz reward is calculated 2. The expected amount is subtracted by the average baker fees 3. The new amount of rewardXTz is translated back into the currency the user chose
Output	cash basis prospective income
Alternative Courses	None
Exceptions	none

Example	be/sio/prosepective.js
	
	f(qXTZ, fiatBasisPrice){
	xtzRew = qXTZ * yieldRate - average(bakerfees)
	fiatRew = xtzRew * (fiatBasisPrice)
	Return xtzRew, fiatRew
	}

5.

Event	Be Emission
Description	Emit calculated data over the socket or prepare for REst
Location	be/app.js be//sio/prospective.js

Primary Actor	Node JS
Trigger	Prospective analysis is complete
Dependencies	Server instance xtzRew, fiatRew
Normal Course	1. The output of the prospective analysis is emitted over the socket
Output	Socket data packet with the two objects
Alternative Courses	none
Exceptions	none
Example	be/sio/prospective.js socket.emit('prosRes'{currencyRew: currencyRew, xtzRewQ: xtzRewQ})

6.

Event	Fe Catch Data
Description	Catch the data from the api connection to the server
Location	fe/js/socket.js
Primary Actor	Client
Trigger	Socket Response is Recieved
Dependencies	Local storage xtzRew, fiatRew
Normal Course	1. Socket is listening for response 2. Socket gets response 3. Response parsed into client memory
Output	rendered display for annual rewards
Alternative Courses	none

Exceptions	none
Example	fe/js/socket.js //hypothetical socket.on('prosRes'{$.each(localStorage.setItem('currencyRew, xtzRewQ))}

7.

Event	Fe Render
Description	render in front end
Location	fe/index.html
Primary Actor	Web Page async processor
Trigger	Local storage is populated with prospective results
Dependencies	Local storage xtzRew, fiatRew
Normal Course	1. Local storage is being constantly checked for values 2. Values are rendered in DOM
Output	rendered display for annual rewards
Alternative Courses	none
Exceptions	Error: 1. The prospective information you entered was invalid

Example	fe/index.html
	
	async f(){
	If (localStorage.getItem('currencyRew' == values){
	DOMElement = currencyRew}

	If (localStorage.getItem('rewQXt z == values){
	DOMElement = rewQXTZ}
	
	Update page
	}

c. Functional And DataBase Server Events

1.

Event	DB Fiat Price Update
Description	update all fiat prices each day for XTZ from coingecko
Location	be/db_init.js be/db_update.js be/database.js
Primary Actor	Node JS
Trigger	New Day 00:00 UTC
Dependencies	MongoDB connected and db_init calls update file
Normal Course	1. Run axios restful get request 2. Parse in the get request into the database
Output	prices for xtz in 20 currencies each day
Alternative Courses	none
Exceptions	none
Example	be/db_update.js:25->45 //the axios rest get to coingecko will be updated to incorporate 20 other fiats

```
const fetchData = async () => {
                return await
axios.get(`https://api.coingecko
.com/api/v3/coins/tezos/history
?date=${d.getUTCDate()}-${d.g
etUTCMonth() +
1}-${d.getUTCFullYear()}&locali
zation=false`);
            };

limiter.schedule(fetchData)
                .then(function
(response) {
                    if
(response.data.market_data) {
                        const price =
response.data.market_data.cur
rent_price.usd;
                        const
marketCap =
response.data.market_data.ma
rket_cap.usd;

BlockchainModel

.findOneAndUpdate(
                                {
                                    date:
d
                                },
                                {
                                    price:
price,

marketCap: marketCap
                                },
                                {
                                    new:
true,

upsert: true
                                })
```

2.

Event	DB Market Value Update
Description	market value collection from coingecko
Location	be/db_init.js be/db_update.js be/database.js
Primary Actor	Node JS
Trigger	New day 00:00 UTC
Dependencies	MongoDB connected and db_init calls update file
Normal Course	1. Axios get request to coingecko 2. Parse in the response from the marketCap 3. Write that response to the db
Output	market value for xtz in 20 currencies each day
Alternative Courses	none
Exceptions	none

Example	be/db_update.js:25->45
	
	//the axios rest get to coingecko will be updated to incorporate 20 other fiats
	
	const fetchData = async () => {
	return await axios.get(`https://api.coingecko.com/api/v3/coins/tezos/history?date=${d.getUTCDate()}-${d.getUTCMonth() + 1}-${d.getUTCFullYear()}&localization=false`);

	};
	
	limiter.schedule(fetchData)
	.then(function
	(response) {
	if
	(response.data.market_data) {
	const price =
	response.data.market_data.cur
	rent_price.usd;
	const
	marketCap =
	response.data.market_data.ma
	rket_cap.usd;
	
	BlockchainModel
	
	.findOneAndUpdate(
	{
	date:
	d
	},
	{
	price:
	price,
	
	marketCap: marketCap
	},
	{
	new:
	true,
	
	upsert: true
	})

3.

Event	DB Supply Data Update

Description	collecting supply data for each day from tzkt
Location	be/dbupdate.js
Primary Actor	Node JS
Trigger	New Day 00:00 UTC
Dependencies	Mongo DB connected
Normal Course	1. Run axios request to tzkt for supply stats 2. Update the supply quantity to the database
Output	supply quantity for xtz each day
Alternative Courses	none
Exceptions	none

Example	be/db_update.js:57-107
	
	try {
	let url = `https://api.tzkt.io/v1/statistics?offset=${offset}&limit=10000`;
	const response = await axios.get(url);
	// console.log(url);
	offset = response.data[response.data.length - 1].level + 1; // update for next iteration
	for (let i = 0; i < response.data.length; i++) {
	const element = response.data[i];
	statistics.push(element);
	}
	if (response.data.length < 10000) {
	break;
	}
	} catch (error) {
	console.error(error);

```
      }
   }
   let lastDateNumber = 0;
   let lastTotalSupply = 0;
   let lastDateString = '';
   for (let i = 0; i < statistics.length;
i++) {
      const dateNumber =
Math.floor(new
Date(Date.parse(statistics[i].timest
amp)).getTime() / (1000 * 60 * 60
* 24));
      if (dateNumber >
lastDateNumber) {
         if (lastDateNumber !== 0) {

StatisticModel.findOneAndUpdate
(
               {
                  dateString:
lastDateString
               },
               {
                  dateString:
lastDateString,
                  totalSupply:
lastTotalSupply
               },
               {
                  new: true,
                  upsert: true
               })
               .then((doc) => {

console.log(`totalSupply on date
${doc.dateString} updated in
database`);
               }).catch((err) => {
                  console.error(err);
               })
         }
```

4.

Event	Push Previous Address Analysis To DB
Description	After the system has gone through the events of accounting analysis. The system will store the address history of analysis in the database
Location	be/db_update_address_history.js
Primary Actor	Node JS
Trigger	The user has responded to the accounting analysis with the accounting method they used for the realizedQ
Dependencies	The address's last set of transformed accounting entrants. The 4 sets of qRealized values under fifo,lifo,weightavg,optimized. Ths user's selection of which set they used.
Normal Course	1. Take address string and qRealized set. 2. Delete the corresponding accounting entrants in the transformed cv dep mvdil sets from the qRealized value 3. Update the address's accounting entrant set with
Output	The remaining accounting entrants that can be realized
Alternative Courses	none
Exceptions	Address has no previous analysis 1. Do nothing
Example	be/db_update_address_history.js f(){

	.findOneAndUpdate(
	{
	addressString: lastAddressString
	},
	{
	eligibleEntrants: newEligibleEntrants
	},
	{
	new: true,
	upsert: true
	})
	
	}

5.

Event	Use Your Analysis Evolution?
Description	The user's address query history has evolved over time because of the quantity realized variable and the previously used accounting mechanism(s). The evolving accounting entry pool is what is being sought here, for consistency.
Location	fe/js/socket.js be/sio/analysis.js
Primary Actor	User/Node JS
Trigger	System prompts upon 'let's go' click
Dependencies	User has previously used PoS Tax and have a history of realized gain calculations
Normal Course	1. The user gets prompted the question 'use your original basis and past realized quantities with their accounting mechanisms?' 2. The user clicks yes

	3. The socket.emit() function attaches a ('historyReq') 4. On the BE the DB is queried for the address history 5. Rerun the reward set query for rewards beyond the enddate of the previous analysis 6. The eligible transformed reward accounting entrants on the memory file get assigned for new analysis. 7. Analysis goes back to user
Output	Transformed entrant pool being pushed to the analysis process
Alternative Courses	1. User clicks no 2. The system performs analysis on his address without the remembered sets
Exceptions	1. Address has no previous analysis

Example	be/sio/analysis.js
	//does not exist yet
	
	f(){
	//get the new rewards
	newSet = axios.get('tzkt.rewardsSet&lastDate-newenddate')
	
	analysis('newSet')
	
	oldSet = db.get('eligbleSetFromPrevAnalysis')
	
	.join(newSet, oldSet)
	
	fifo/lifo,weightavg,opt()

	Return to user }

6.

Event	Date To Cycles DB Update
Description	The query to the data source has to be in cycles so we must convert the date range that the user entered into cycles.
Location	be/app.js be/db_update.js be/models/cycles
Primary Actor	Node/Express JS App
Trigger	New day 00:00UTC
Dependencies	start date and end date From user
Normal Course	1. Get the cycle statistic from tzkt 2. Match the date with the timestamp return payload 3. Update database
Output	cycle query set to db
Alternative Courses	Last date to string is updated to database
Exceptions	No exceptions

Example	be/db_update.js:115->185
	
	async function updateCycles()
	{
	let offset = 0;
	let cycles = []
	while (true) {
	try {
	let url = `https://api.tzkt.io/v1/statistics/cyclic?offset=${offset}&limit=10000`;

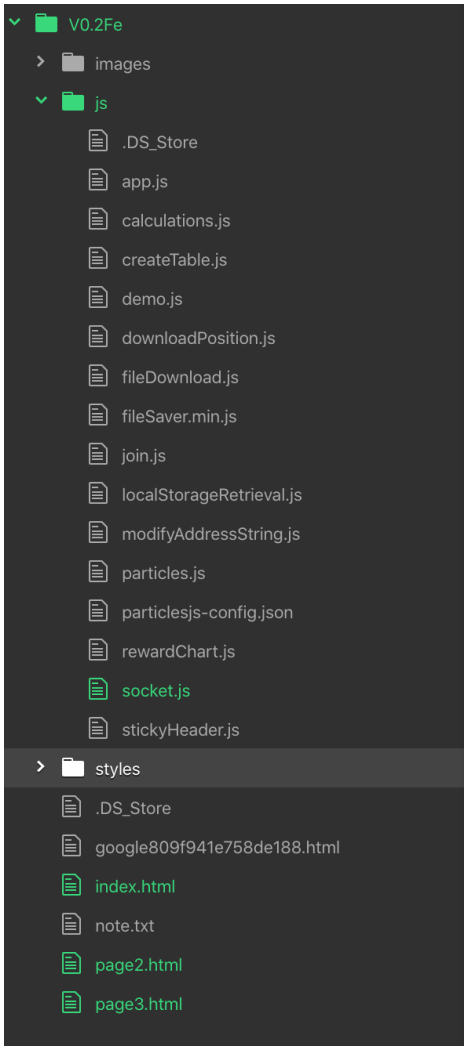
```
            const response = await
axios.get(url);
            // console.log(url);
            offset =
response.data[response.data.l
ength - 1].level + 1;  // update
for next iteration
            for (let i = 0; i <
response.data.length; i++) {
                const element =
response.data[i];

cycles.push(element);
            }
            if
(response.data.length <
10000) {
                break;
            }
        } catch (error) {
            console.error(error);
        }
    }
    let cycleObj = {};
    for (let i = 0; i <
cycles.length; i++) {
        const date = new
Date(Date.parse(cycles[i].time
stamp));
        const dateString =
date.toISOString().substring(0,
10);
        cycleObj[dateString] =
cycles[i].cycle + 1;
    }
    let days = Math.floor((new
Date().getTime() - new
Date(Date.parse("2018-06-30"
)).getTime()) / (1000 * 60 * 60 *
24));
    for (let i = 0; i <= days; i++) {
        let date = new
Date(Date.parse("2018-06-30"
) + i * 24 * 60 * 60 * 1000);
        let dateString =
date.toISOString().substring(0,
10);
        if (dateString in cycleObj)
{
            // handle with current
```

	cycle (latest cycle)
	if (dateString === cycles[cycles.length-1].timestamp.substring(0, 10)) { // if this is the last cycle in returned data
	while (i <= days) {
	date = new Date(date.getTime() + 1000 * 60 * 60 * 24);
	
	cycleObj[date.toISOString().substring(0, 10)] = cycles[cycles.length-1].cycle + 1;
	i += 1;
	}
	// break out of the loop
	}
	}
	else {
	const lastDate = new Date(date.getTime() - 1000 * 60 * 60 * 24);
	const lastDateString = lastDate.toISOString().substring(0, 10);
	cycleObj[dateString] = lastDateString in cycleObj? cycleObj[lastDateString]: 0;
	}
	}
	for (let i = 0; i <= days; i++) {
	const date = new Date(Date.parse("2018-06-30") + i * 24 * 60 * 60 * 1000);
	const dateString = date.toISOString().substring(0, 10);
	
	CycleModel.findOneAndUpdate(
	{
	dateString: dateString
	},
	{
	dateString: dateString,

	cycleNumber: cycleObj[dateString] }, { new: true, upsert: true }) .then((doc) => { console.log(`cycleNumber on date ${doc.dateString} updated in database`); }).catch((err) => { console.error(err); }); } } module.exports = { updatePrices, updateTotalSupplys, updateCycles };

5. Client System Events With Code

This is the V1.0 front end directory structure.



a. Accounting Analysis Interface Events

1.

Event	Enter Address
Description	User enters address into system
Location	fe/js/moduleSorting.js
Primary Actor	User
Trigger	User enters address and

	selects go
Dependencies	The system has loaded its pathways into the users interface
Normal Course	1. The user enters their email 2. The user clicks continue 3. The module sorter stores the element information 4. The module sorter switches interface modules
Output	1. New interface module to user 2. Address is written into local storage
Alternative Courses	1. The user enters email 2. The user selects create account
Exceptions	none
Example	fe/jsmoduleSorting.js: localStorage.setItem('address') .toggle Switch to next interface module element

2.

Event	Enter Basis Day
Description	User enters basis date into system
Location	fe/js/moduleSorting.js
Primary Actor	User
Trigger	User enters basis date and selects go
Dependencies	The system has loaded its

	pathways into the users interface
Normal Course	1. The user enters their basis date 2. The user clicks continue 3. The module sorter stores the element information 4. The module sorter switches interface modules
Output	1. New interface module to user 2. Basis date is written into local storage
Alternative Courses	1. The user selects back button
Exceptions	1. If the basis date is before XTZ existence 2. If the basis date is in the future Both of these bounce back an error to the user
Example	fe/jsmoduleSorting.js: localStorage.setItem('basisDate') .toggle Switch to next interface module element

3.

Event	Enter Quantity Realized and Fiat
Description	User enters q realized and the fiat currency they would like to observe their results in into the system
Location	fe/js/moduleSorting.js

Primary Actor	User
Trigger	User enters quantity realized and their fiat and selects go
Dependencies	The system has loaded its pathways into the users interface
Normal Course	1. The user enters their quantity realized 2. The user enters their fiat currency 3. The user confirms they are not a DDoS bot 4. The user selects continue 5. The module sorter stores the element information 6. The socket program now holds all information that it needs to get analysis from the server 7. The data is emitted from the client to the server 8. The module sorter switches interface modules
Output	1. Emission of user input data to server. Connection to Section 4.a.1 2. Upon response, New interface module to user
Alternative Courses	1. The user selects back 2. The server responds with an error
Exceptions	1. Captcha not met
Example	fe/jsmoduleSorting.js: localStorage.setItem(qRealized, fiat)

	socket.emit('inputParams'){
	Fe emission of inputs to server}
	
	if (error == yes){
	alert(error)}
	
	else{
	.toggle Switch to next interface module element
	}

4.

Event	User Selects the analysis they want to view
Description	User selects the analysis they want to view from the server response
Location	fe/js/moduleSorting.js
Primary Actor	Client System
Trigger	The front end emission is met with a non error response
Dependencies	The server has taken all user inputs from 5.a.3 and has performed analysis from 4.a and has returned the valid data to the client
Normal Course	1. The analysis is rendered ready for user selection 2. The user selects which accounting method they would like to use. 3. The user selects which depletion calculation they would like to use or basis
Output	1. 4.a.17 2. Basis Analysis Income 3. Depletion Analysis Income 3. Market Value Dilution

	Analysis Income
Alternative Courses	none
Exceptions	none
Example	fe/jsmoduleSorting.js: renderFunctions(){} .toggle Switch to next interface module element

5.

Event	Submit Analysis Used to update account history in the system
Description	The user selects which combination of accounting they utilized. This is done in order to update the user's account in the system.
Location	fe/js/moduleSorting.js
Primary Actor	User
Trigger	User enters email and selects go
Dependencies	The system has loaded its pathways into the users interface
Normal Course	5. The user enters their email 6. The user clicks continue 7. The module sorter stores the element information 8. The module sorter switches interface modules
Output	1. The server section 4.a.18 begins
Alternative Courses	1. The user does not

	interact with this section and nothing is updated 2. The user has not made an account and needs to do so in order to do this
Exceptions	none
Example	

B. User Account Interaction Events

1.

Event	Create Account
Description	User enters an email and a password to create an account in the system
Location	fe/js/moduleSorting.js
Primary Actor	User
Trigger	User selects create account from the analysis page or from the index page
Dependencies	None or the user has already obtained analysis on an address they entered
Normal Course	1. The user enters their email 2. The user enters their password 3. The user reenters their password 4. The user verifies they're not a DDoS bot 5. The user selects contine 6. The elements the user entered are bundled into a socket emission 7. The server creates

	their account in the database
Output	1. The user has a new account in the system 2. The module sorter changes to the account interface module
Alternative Courses	1. The user entered an email that already has account 2. The server responds with this and alerts the user to either use the password for this account or a new email or reset the password
Exceptions	none
Example	

2.

Event	Sign In To Account
Description	User enters their email and their password
Location	fe/js/moduleSorting.js
Primary Actor	User
Trigger	User enters email and password
Dependencies	The system has loaded its pathways into the users interface 1. The user has a preexisting account in the system
Normal Course	1. The user enters their email 2. The user gets prompted a password box 3. The user enters their

	password and selects contine 4. They get signed in to their account
Output	1. The user goes to the account interface module
Alternative Courses	1. The user can't remember their password and initiates password reset section x.x.x
Exceptions	1. The user keeps reentering passwords until they get it right
Example	

3.

Event	User Account's Interface Module
Description	The user account's interface module is the home base for a user. Here they see all of their addresses, can enter a new address for analysis or go back to old analysis
Location	fe/js/moduleSorting.js
Primary Actor	User
Trigger	User has signed in 5.b.2 or created account 5.b.1
Dependencies	The system has loaded its pathways into the users interface 1. 5.b.1 2. 5.b.2
Normal Course	1. The user can enter a new address for analysis a. They then get redirected to

	5.a.2 2. The user can select previous address with analysis
Output	1. Interface Module sorter switches to 5.a.2 2. Interface module sorter switches to 5.b.4
Alternative Courses	1. The user can sign out of their account and go back to section 5.a.1
Exceptions	none
Example	

4.

Event	Previous Address New / View Analysis
Description	The user seeks to conduct more quantity realized analysis on a previous address.
Location	fe/js/moduleSorting.js
Primary Actor	User
Trigger	The user has selected a previously analyzed address
Dependencies	1. The address analysis history is fetched from the db
Normal Course	1. The user enters their new quantity realized and the fiat they want to see it in.
Output	1. The user is directed to analysis interface module - section 5.a.4
Alternative Courses	1. The user selects no new analysis show a previous analysis

	a. A table opens and shows their previous realization events b. The user can select one of these events and review it in the analysis interface module 2. The user selects a new basis day to recalculate the incomes for the address
Exceptions	none
Example	

C. Prospective Staker Interface Events

1.

Event	Enter Prospective Staker Product
Description	The user selects prospective staker product
Location	fe/js/moduleSorting.js
Primary Actor	User
Trigger	The user selects prospective staker product
Dependencies	1. The system has loaded its interface paths
Normal Course	1. The user selects the prospective staker interface path
Output	1. The prospective staker form input is loaded
Alternative Courses	1. The user does anything

	but select prospective staker product
Exceptions	none
Example	

2.

Event	Prospective Form Fill
Description	The user enters information into the form
Location	fe/js/moduleSorting.js
Primary Actor	User
Trigger	The user is prompted the prospective form
Dependencies	The user has navigated to the form
Normal Course	1. The user enters the quantity they want to stake 2. The user specifies what currency they would be staking from 3. The client system sends the user data into section 4.b.1
Output	The client system navigates to the result module
Alternative Courses	1. The user selects the back button
Exceptions	1. The user enters a value <= 0
Example	

3.

Event	Prospective Results
Description	The user gets their prospective annual return results
Location	fe/js/moduleSorting.js

Primary Actor	User
Trigger	The user has entered valid information into the from and has selected continue
Dependencies	The client system has gotten the results from server section 4.b.7
Normal Course	1. Results are caught from the server 2. The results are rendered
Output	The expected annual yield from the basis quantity is rendered to the user
Alternative Courses	1. The user selects the back button
Exceptions	none
Example	

D. Information Interface Events

1.

Event	Demo Selection
Description	The user selects the demo button and the client system navigates to an analysis demo
Location	fe/js/moduleSorting.js
Primary Actor	User
Trigger	Demo button selection
Dependencies	System map is set and demo is preloaded into the client
Normal Course	1. The user selects demo 2. The client system directs to the analysis

	module 3. The analysis module displays the demo data
Output	A full demo of the analysis
Alternative Courses	1. The user does not select demo
Exceptions	none
Example	

2.

Event	Context
Description	The user selects the context button that takes the user to the context page
Location	fe/js/moduleSorting.js
Primary Actor	User
Trigger	User selects context
Dependencies	The context interface module is loaded into the client system
Normal Course	1. The user selects context 2. The context interface module is navigated to
Output	The context interface module
Alternative Courses	None
Exceptions	none
Example	

3.

Event	Link Selection
Description	The user selects any of the context related links on the index module interface

Location	fe/js/moduleSorting.js
Primary Actor	User
Trigger	User selects one of the link elements
Dependencies	The client interface module system
Normal Course	1. User selects a link 2. The user is taken to the outside site
Output	User on a new website
Alternative Courses	none
Exceptions	none
Example	

E. Client Integration Events

1.

Event	Module Instance Request
Description	emit user entered inputs over the socket connection
Location	Fe /clientModule/js/socket.js
Primary Actor	Client
Trigger	The client webage has loaded with our content on it
Dependencies	cdn postax, canvas tag, client input
Normal Course	1. The client has on boarded our content into their system 2. The cdn tag gets our programs

	3.
Output	Begin server instance of client module
Alternative Courses	none
Exceptions	none
Example	fe/clientmodule/js/socket.js socket.emit('analysisReq',{address: address, startdate: startedate, enddate: enddate, fiat: fiat, qRealized: qRealized}

2.

Event	Product Sent Back To Client
Description	processing of server payload from the server
Location	fe/clientmodule/js/socket.js
Primary Actor	Client
Trigger	Socket data is Recieved
Dependencies	socket.Onmessgae() to be running
Normal Course	1. Product analysis has been completed in the server 2. The resulting object has been emitted over the socket 3. The socket has caught the package
Output	DOM memory/elements populated with product content
Alternative Courses	none
Exceptions	none

Example	fe/clientModule/js/socket.js f(){socket.on('analysisRes', (data)=>{$.each(data, function(i,result){some iterative processing}) pageElement = value }

3.

Event	Render Client Module
Description	css package renders js
Location	fe/clientModule/styles/client.css
Primary Actor	Client
Trigger	The client's memory and dom elements have been updated
Dependencies	Product outputted data elements written into local storage
Normal Course	1. Client module css executes
Output	Client module window display render
Alternative Courses	A client has elected to hardwire our products into their display 1. The PoS Tax signature is rendered
Exceptions	none
Example	fe/clientModule/styles/client.css .box{ background = x

	}

4.

Event	CDN Instance Activated
Description	CDN is activated by the developers of a client solution
Location	be/app.js be/sio/analysis.js be/sio/prospective.js
Primary Actor	Node JS
Trigger	A client connection has been requested
Dependencies	Cdn established
Normal Course	1. App.js has been instantiated
Output	Socket connection to client module
Alternative Courses	none
Exceptions	If the connection is an attack 1. Deny service
Example	be/app.js const app = express()

5.

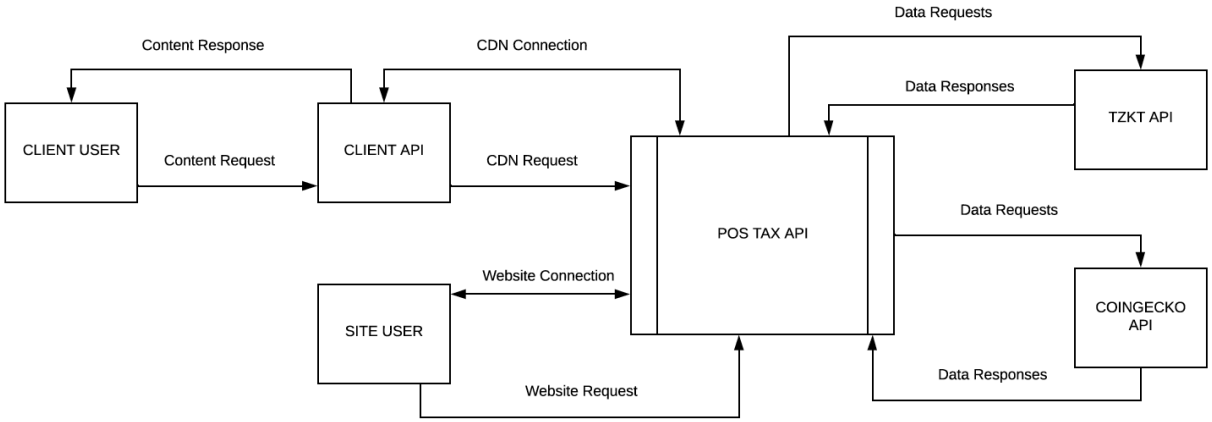
Event	Signature Tag
Description	Signature tag posting and use
Location	clientFe/clientmodule/js/signatureTag

Primary Actor	Client
Trigger	The client has elected to integrate our products into their DOM
Dependencies	Client module integration into client's DOM display, therefore -> socket payload arrival, client event handling
Normal Course	1. Page loads with our signature on it
Output	client handling display, PoS Tax signature to client webpage
Alternative Courses	none
Exceptions	none

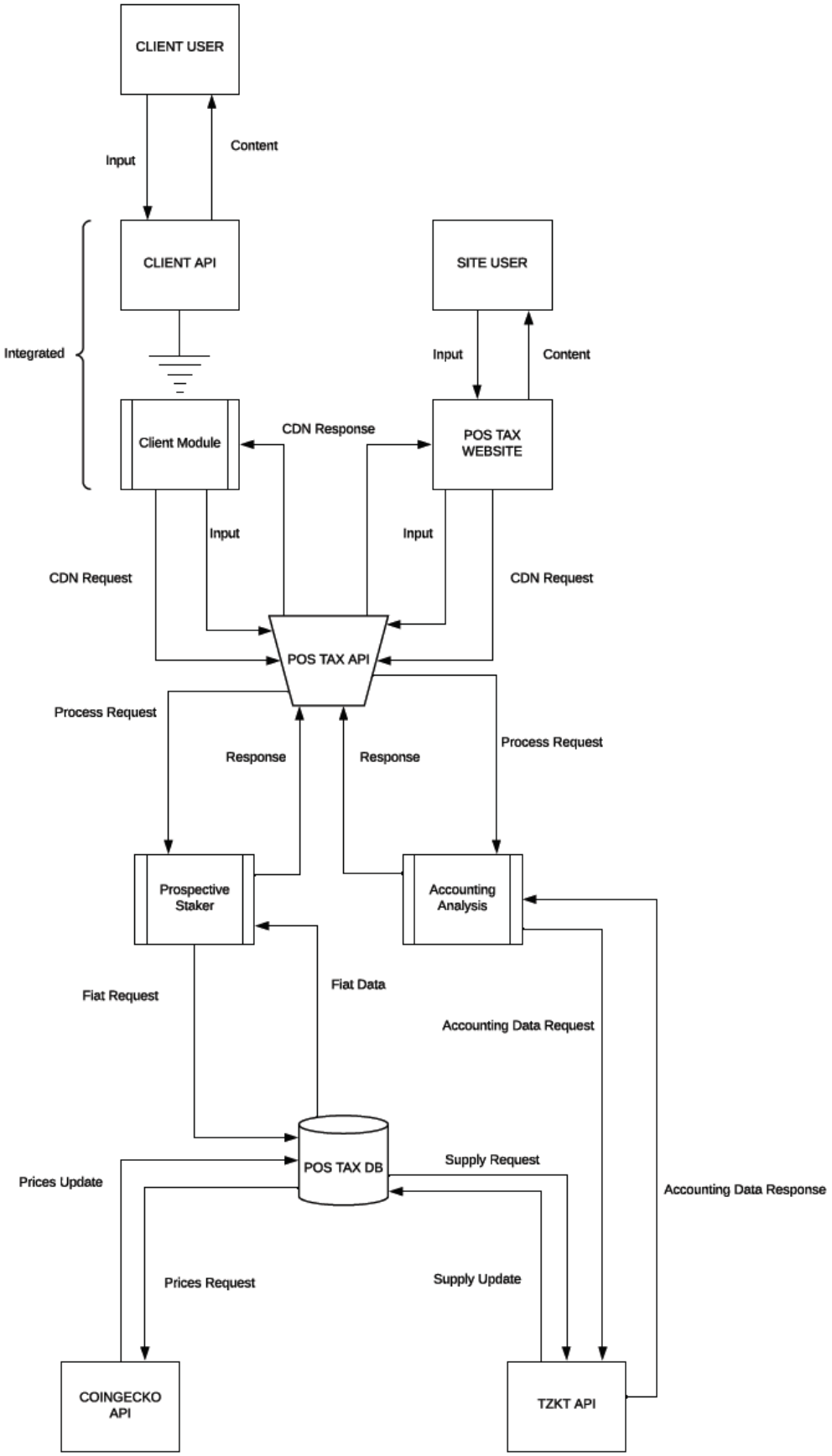
Example	clientfe/index.html
	
	<canvas id = 'postaxSig'></canvas>
	//js
	on.click('postaxation.com')

6. Diagrams

a. Context Diagram

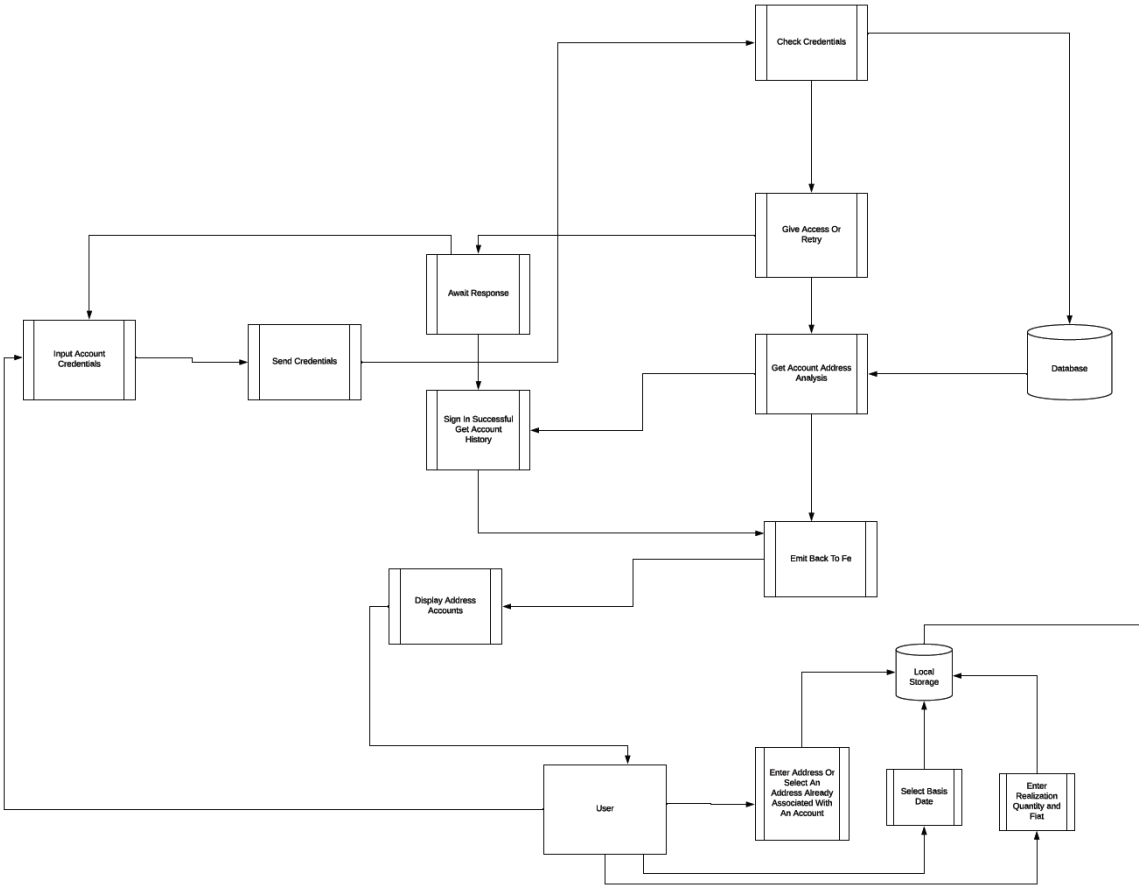


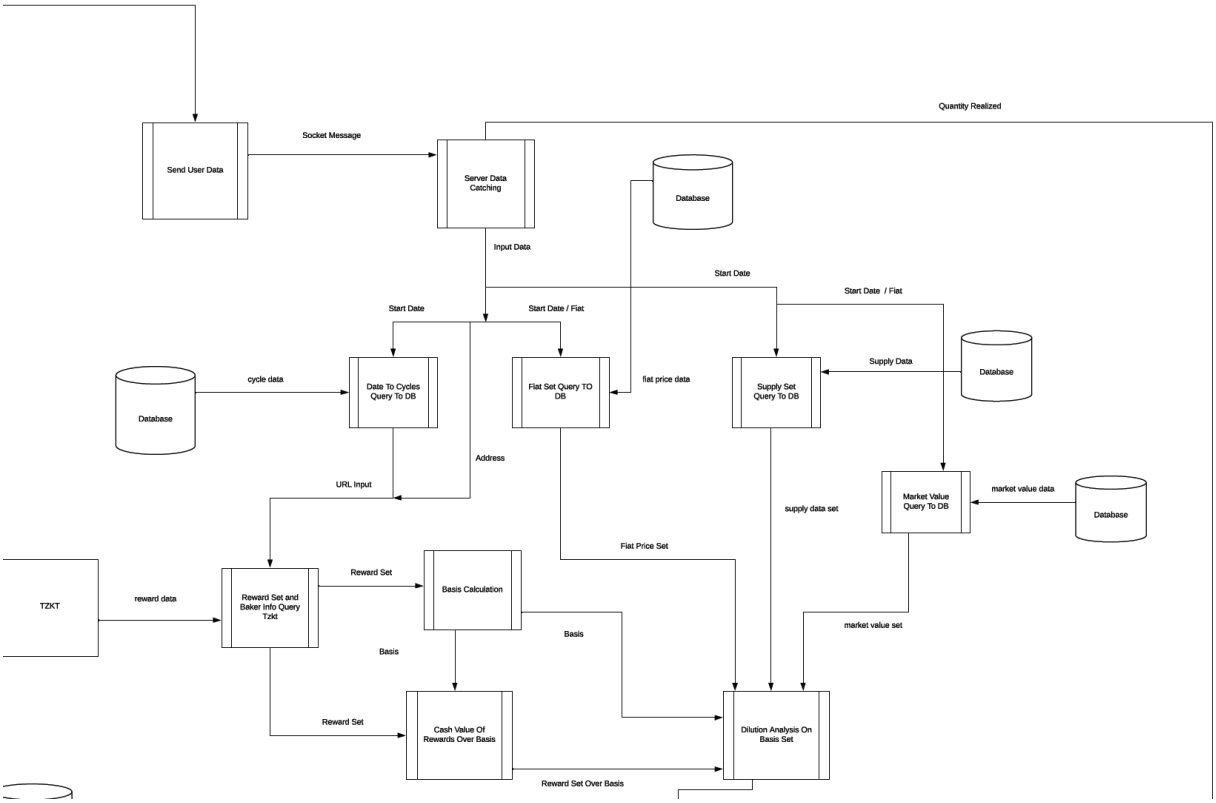
b. Level 0 Data Flow Diagram

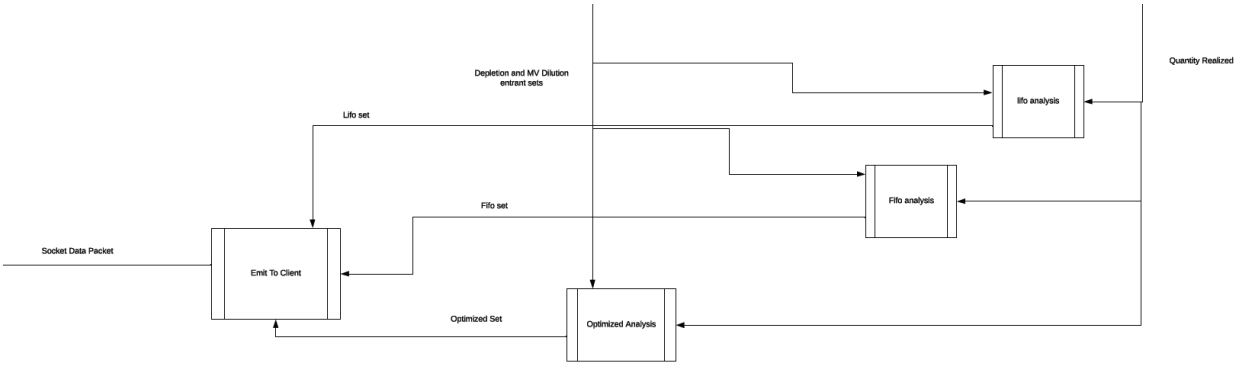

CLIENT USER
Input
Content
CLIENT API
SITE USER
Integrated
Input
Content
Client Module
CDN Response
POS TAX WEBSITE
Input
Input
CDN Request
CDN Request
POS TAX API
Process Request
Response
Response
Process Request
Prospective Staker
Accounting Analysis
Fiat Request
Fiat Data
Accounting Data Request
POS TAX DB
Supply Request
Prices Update
Accounting Data Response
Prices Request
Supply Update
COINGECKO API
TZKT API

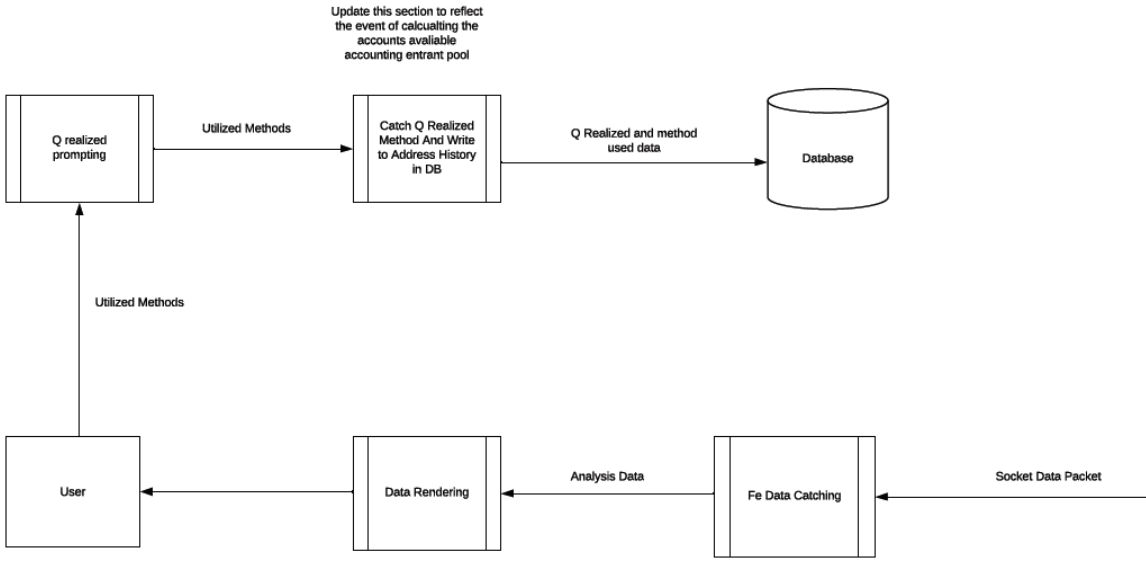
c. Process Flow Diagram

1. Accounting Process Flow



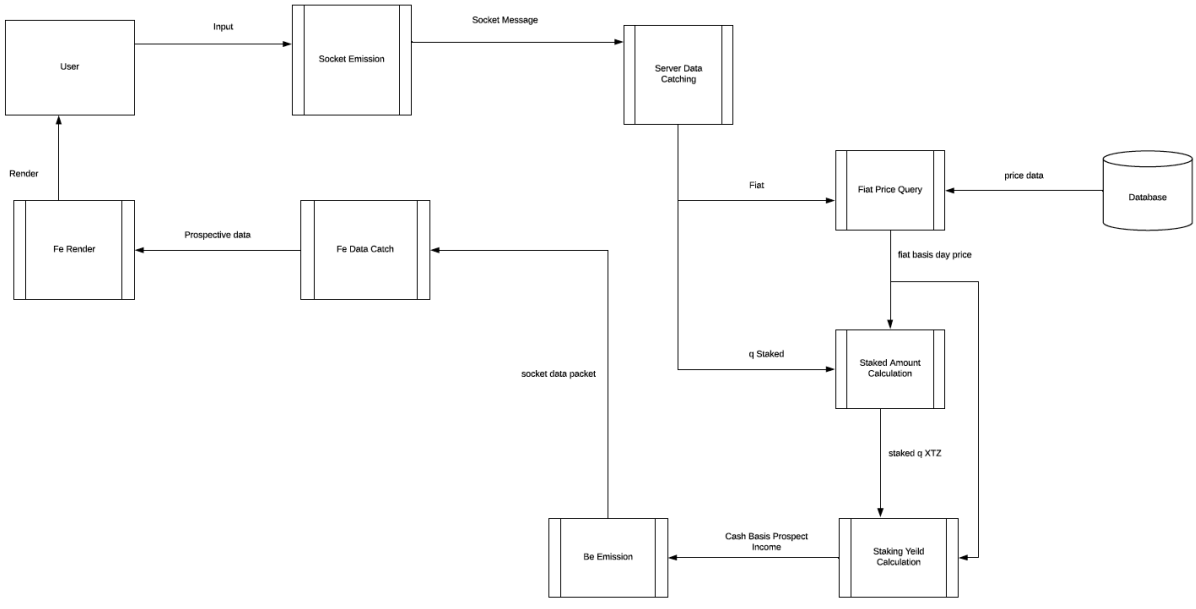

Quantity Realized
Send User Data
Socket Message
Server Data Catching
Database
Input Data
Start Date
Start Date / Fiat
Start Date
Start Date / Fiat
Supply Data
Database
cycle data
Database
Date To Cycles Query To DB
Fiat Set Query TO DB
fiat price data
Supply Set Query To DB
Address
URL Input
supply data set
Market Value Query To DB
market value data
Database
Fiat Price Set
TZKT
reward data
Reward Set and Baker Info Query Tzkt
Reward Set
Basis Calculation
Basis
market value set
Basis
Reward Set
Cash Value Of Rewards Over Basis
Dilution Analysis On Basis Set
Reward Set Over Basis


Depletion and MV Dilution entrant sets
Quantity Realized
lifo analysis
Lifo set
Fifo analysis
Fifo set
Socket Data Packet
Emit To Client
Optimized Set
Optimized Analysis

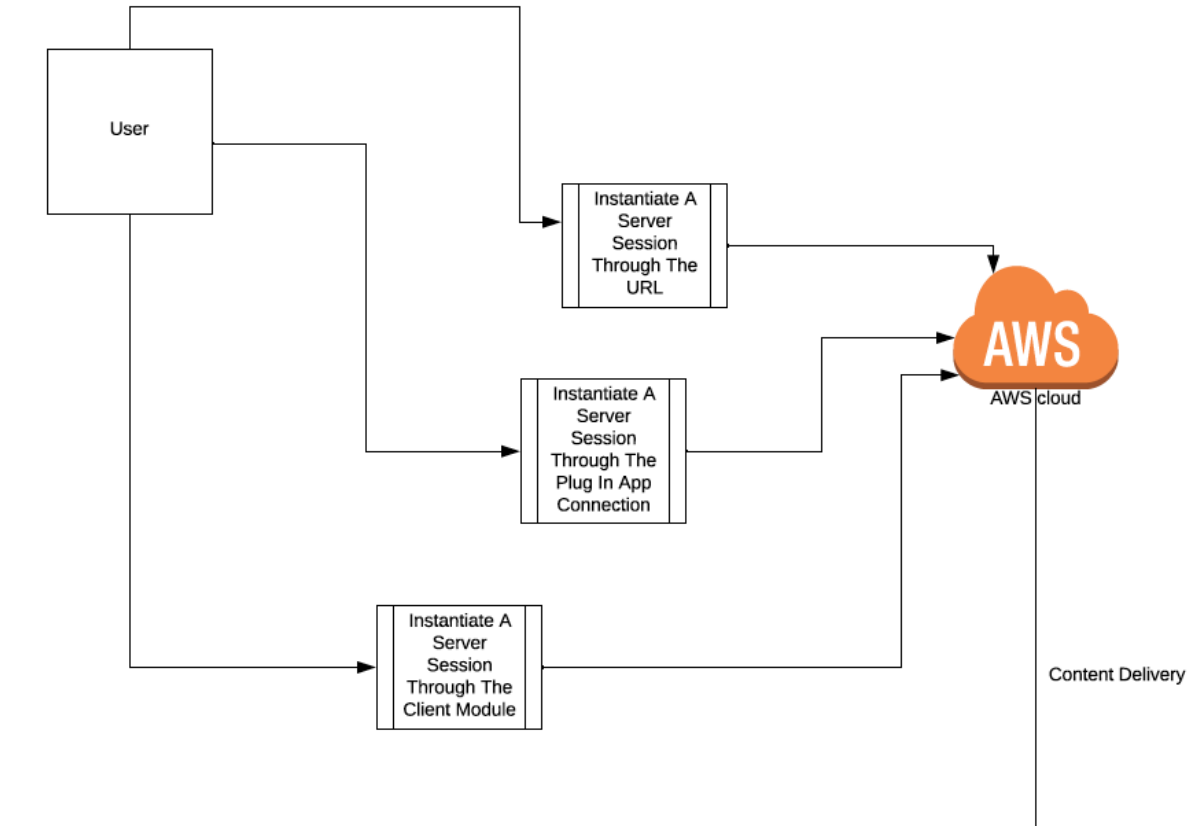


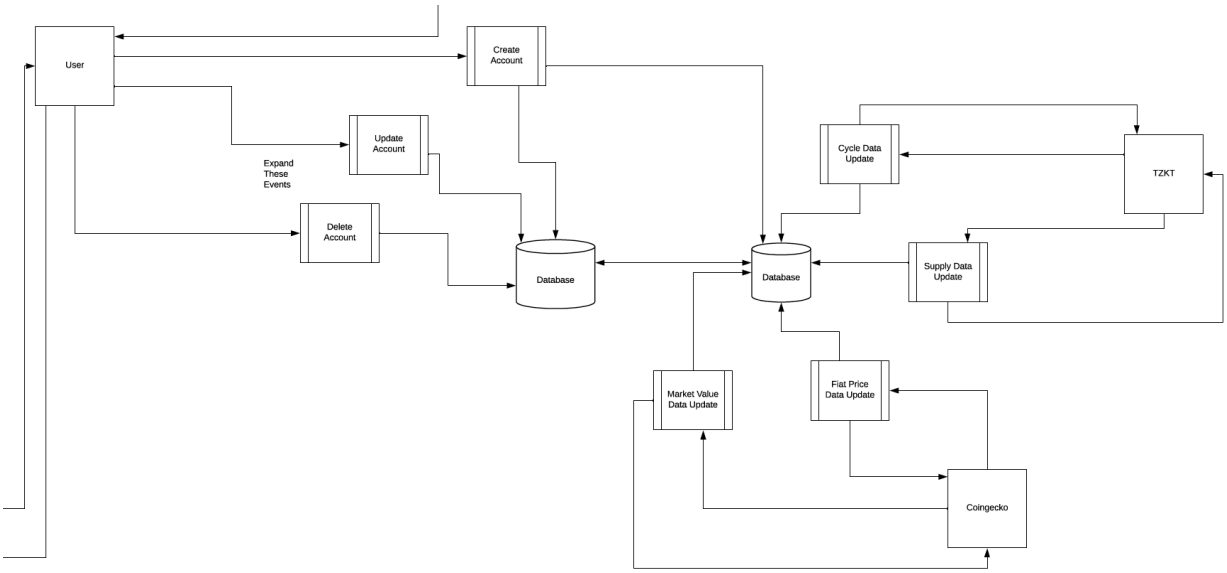
Fin.

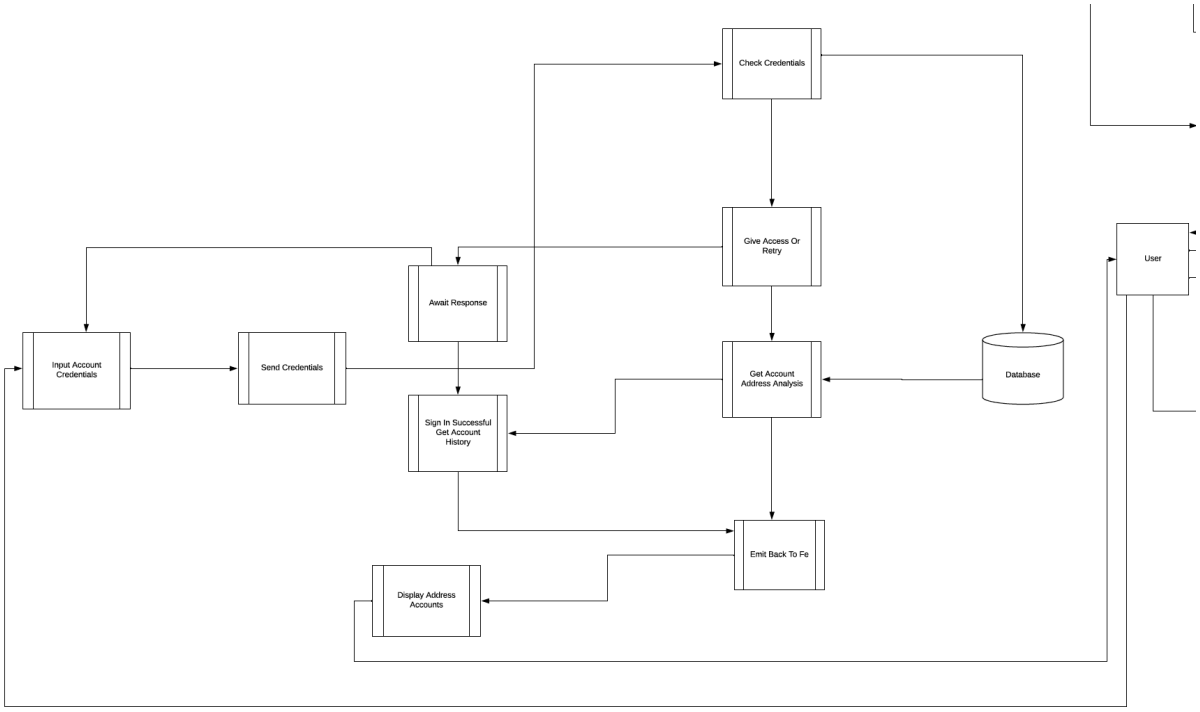
2. Prospective Staker Event Flow



3. Functional And Database Flow







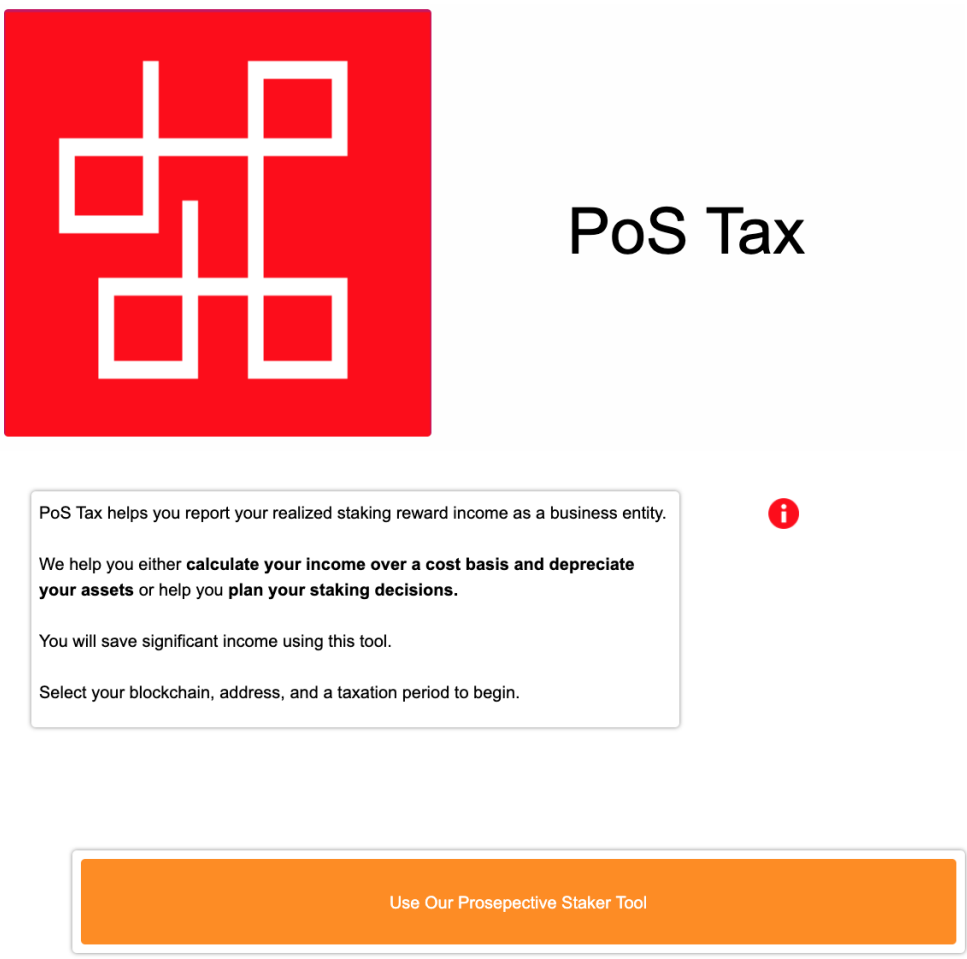
7. MockUps

Note that in all user experiences, the total number of user clicks and transformations of content interface is always less than 3. This low level of engagement effort makes PoS Tax products easily usable.

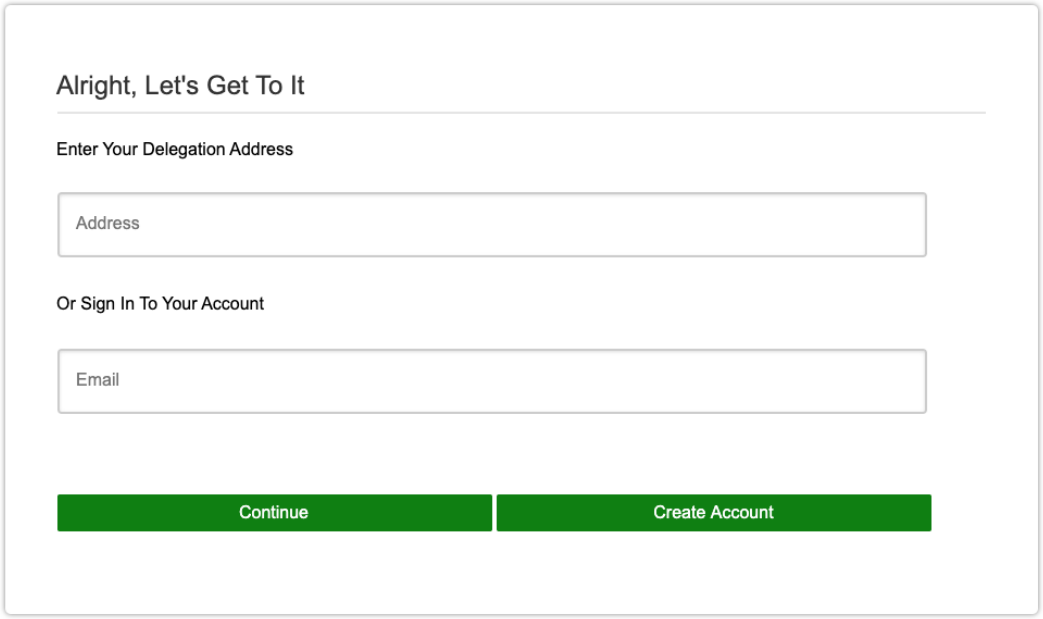
a. Accounting Analysis Website

The user navigates to the site and reads the site description.

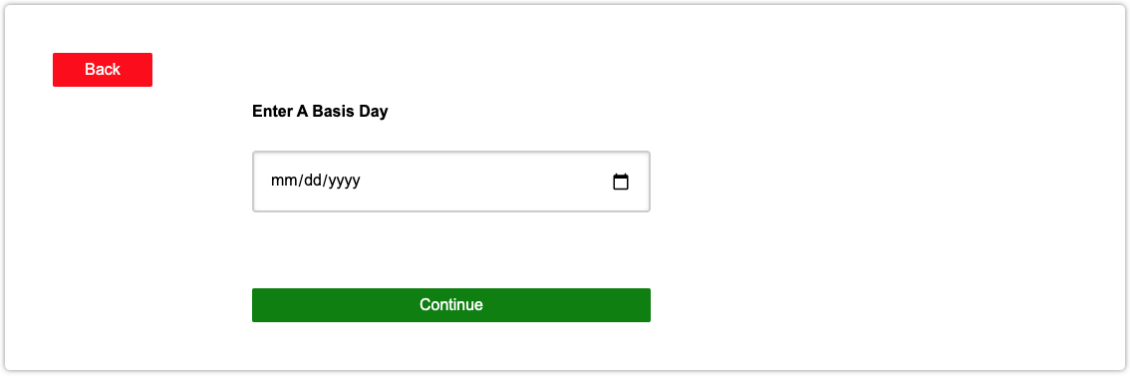
The user has the option here to begin the prospective staker user experience.



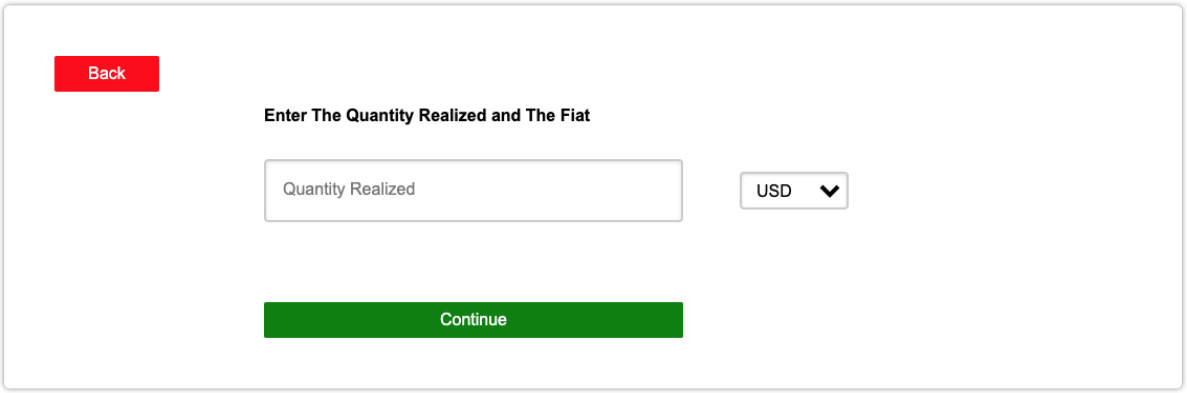
The user continues to scroll down to the delegator experience and enters their address in the form and selects continue.



From the address input the user is directed to enter a basis day.

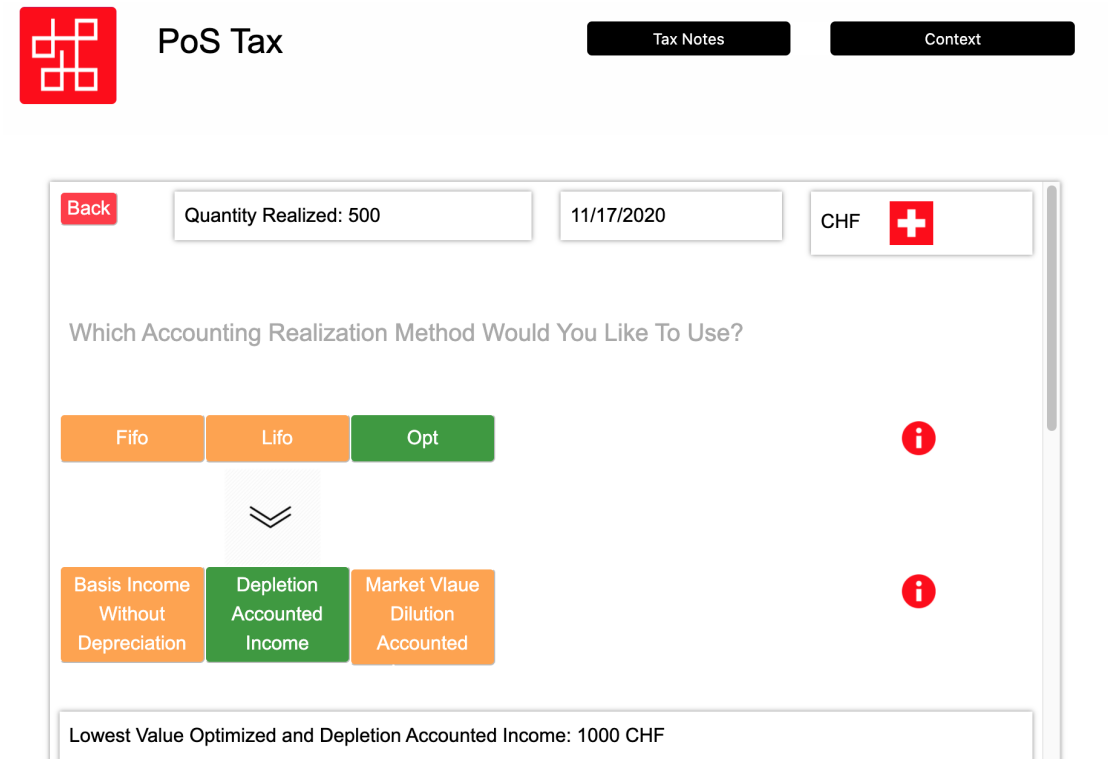


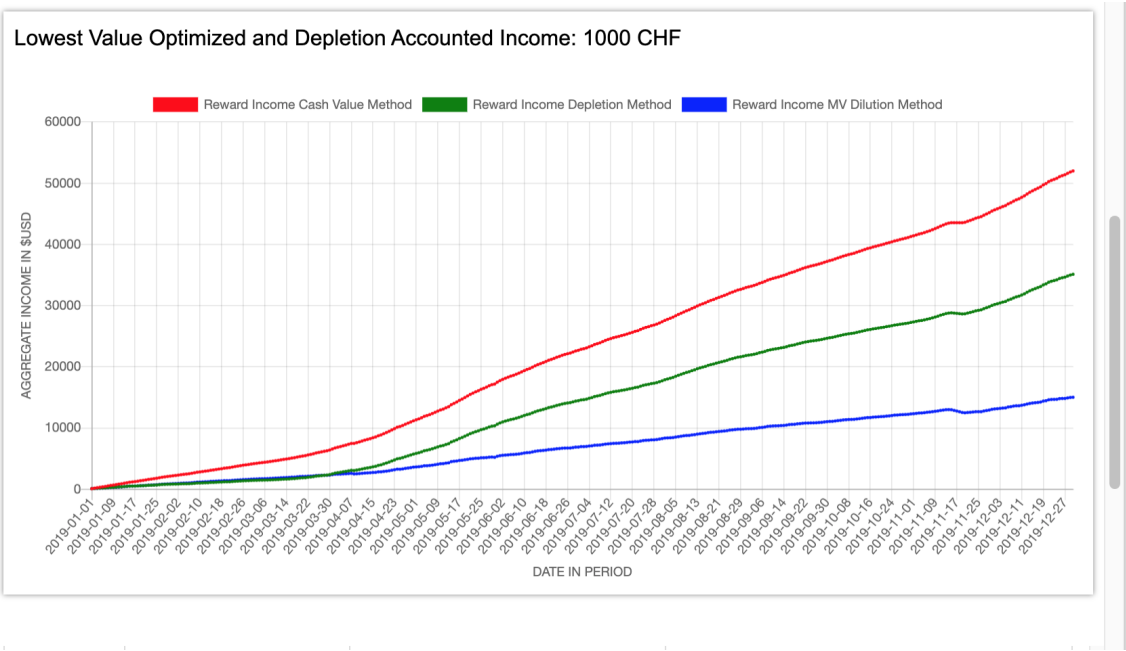
After the user enters a basis day for their address they are instructed to enter their quantity realized and fiat they realized it in.



The user's data is then processed on the server and the site switches to the analysis interface module. In this module, all of the user's accounting metrics are displayed to her.

The user can also send back the method combination they selected to the system so it can store it for later use.





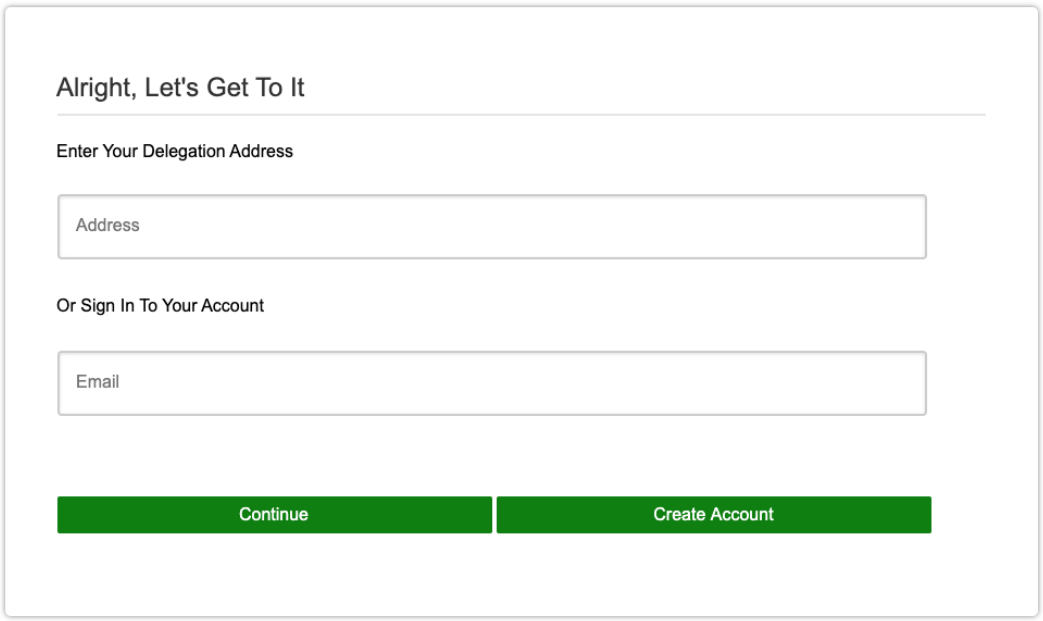
Date	Reward Quantity (XTZ)	Reward Value Cash Basis ($USD)	Reward Value Depletion Accounting ($USD)
2019-01-07	203	97.65	51.05
2019-01-08	155.67	73.19	27.72
2019-01-09	114	55.1	10.22
2019-01-10	163	80.85	35.73
2019-01-11	172.67	77.26	34.09
2019-01-12	134	59.46	13.41

Download Complete Statement

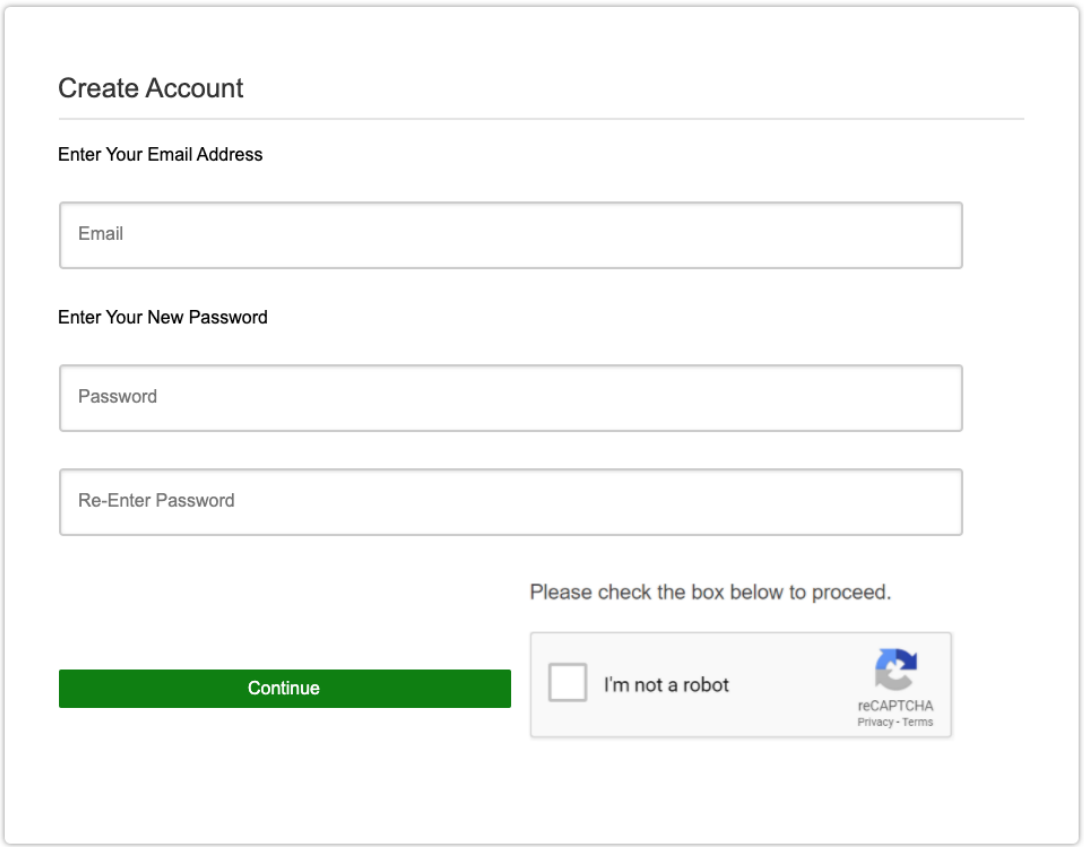
Update your account with your realizaiton method

Send Back This Combination

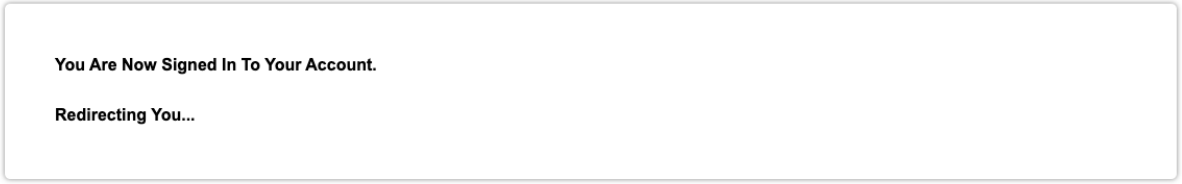
But now, let's assume the user wants to create an account in the system. The user selects the "Create Account" button below from the index/landing page.



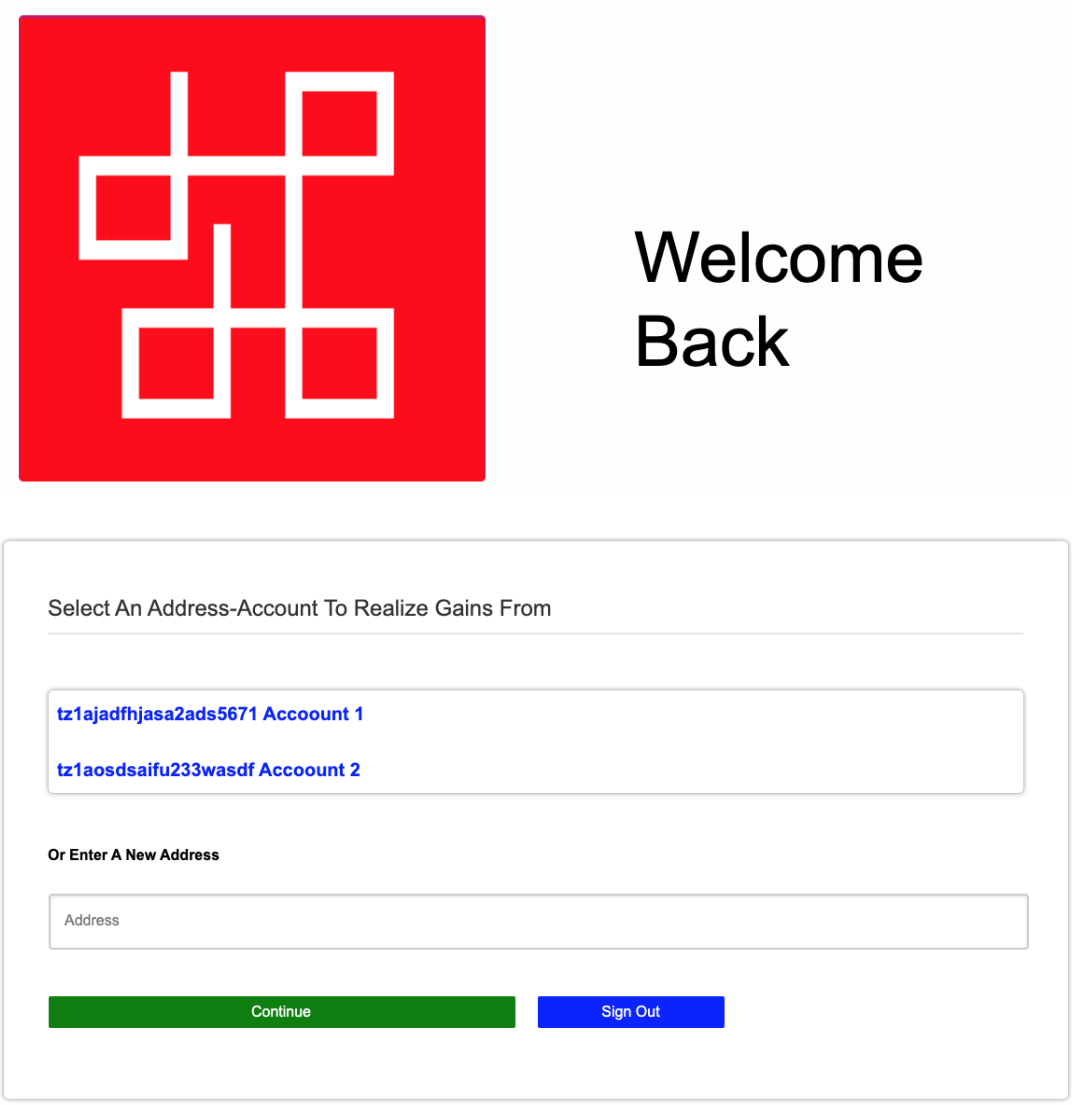
The user then enters their email address and their new password, twice to make sure they don't make a mistake. They'll also prove that they are not a DDoS bot trying to wreck our server.



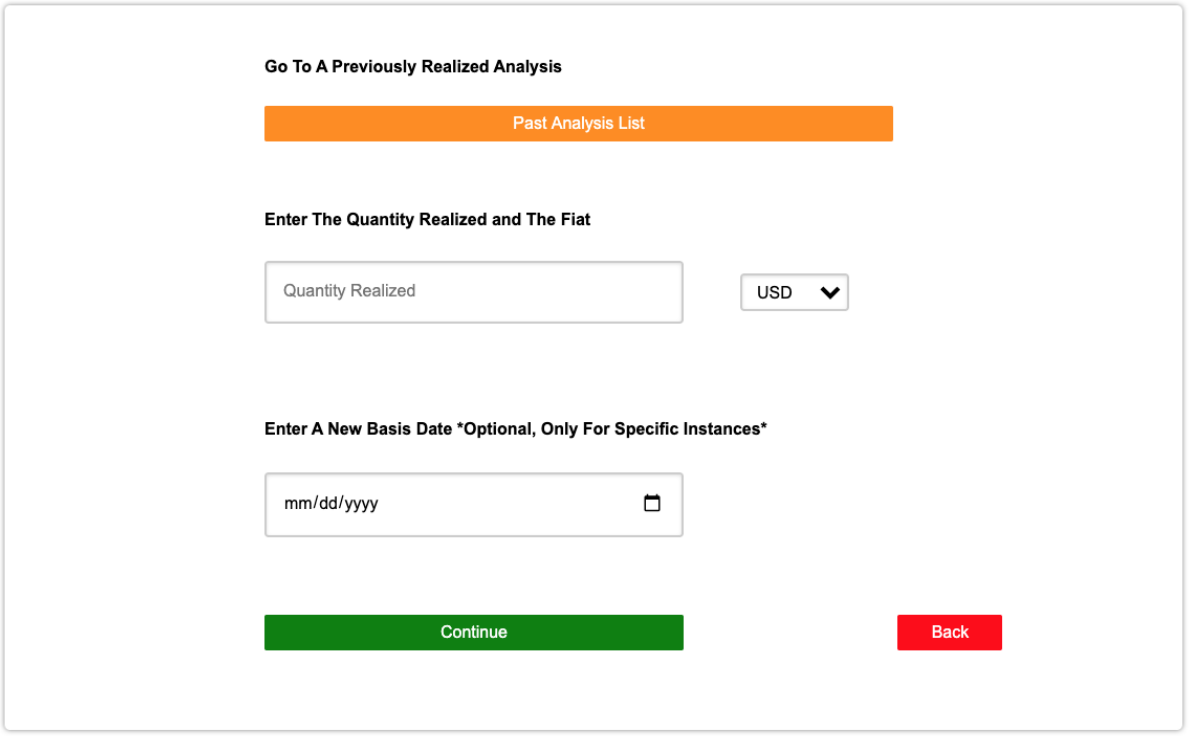
The user is then redirected.



They are redirected to the account home interface module. If the user enters a new address in the module below, they will be directed to the enter basis day and enter quantity realized and fiat modules that are mocked up above.



The user selects on an account they have in the system. Once they have done that and selected continue, the user is prompted the module below.



If the user selects on the past analysis list, then a table opens showing their past realization analysis events will be displayed in the module.

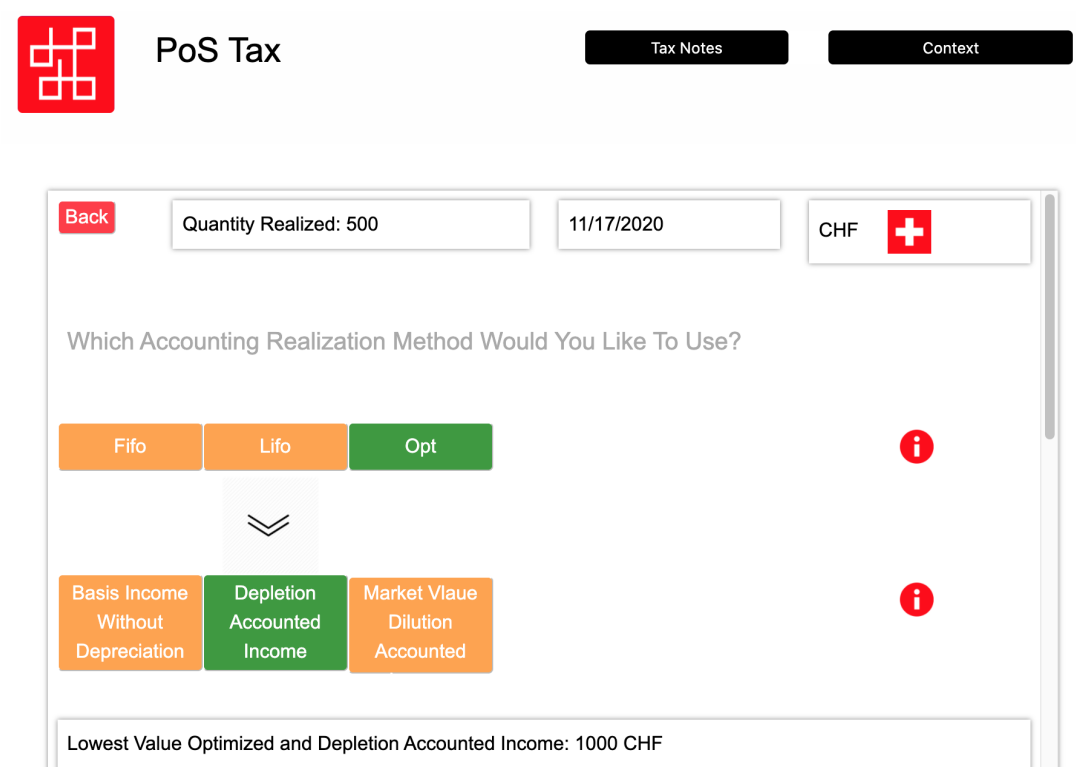
Date	Calculated Income Of Quantity Realized (FIAT)	Method Combination
10/10/2020	1000 CHF	FIFO / Depletion Method
10/12/2020	400 USD	LIFO / Depletion Method
10/30/2020	3100 CHF	OPT / Market Value Dilution
11/10/2020	55500 YEN	OPT / Depletion
11/17/2020	69 USD	OPT / Market Value Dilution

Accounted Total Realized Income: 4100 CHF

Accounted Total Realized Income: 55500 YEN

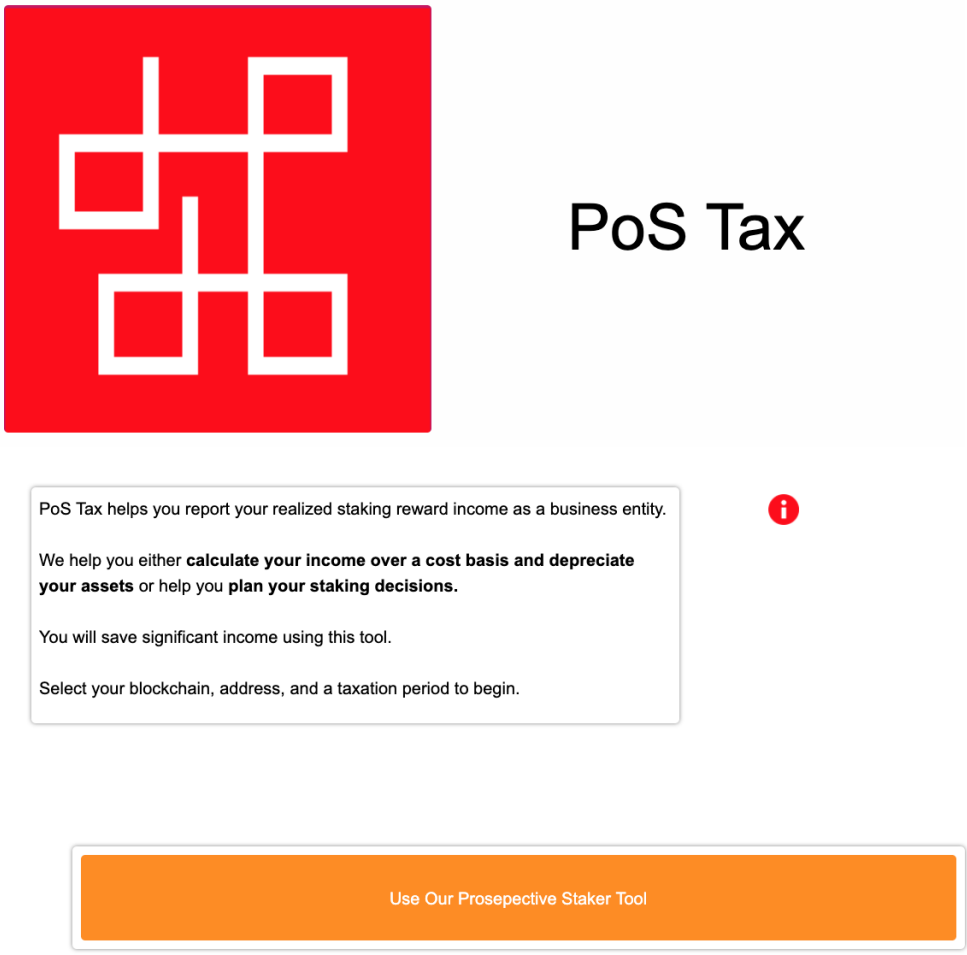
Accounted Total Realized Income: 69 USD

The user can then select the analysis event and view the results. They can then view the results in a different way and push that new way they want to have the gain realized back to the server.

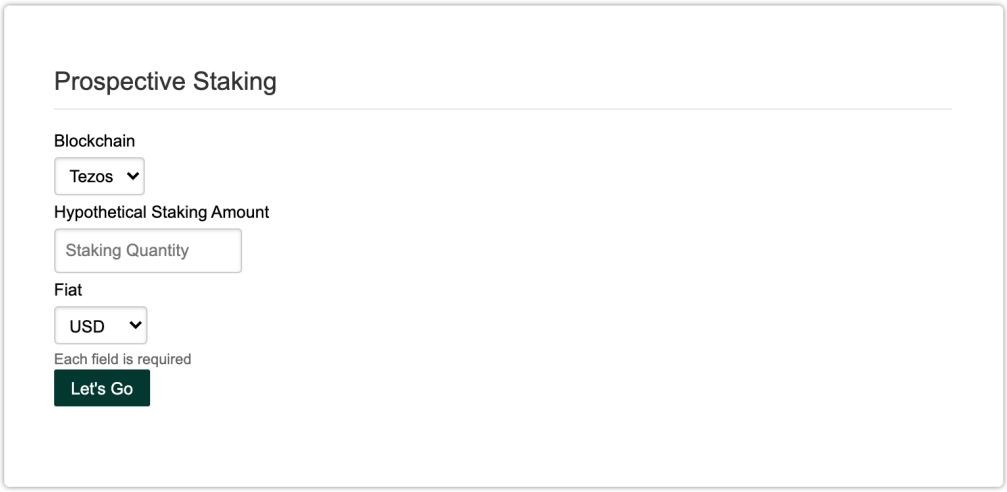


b. Prospective Staker Website

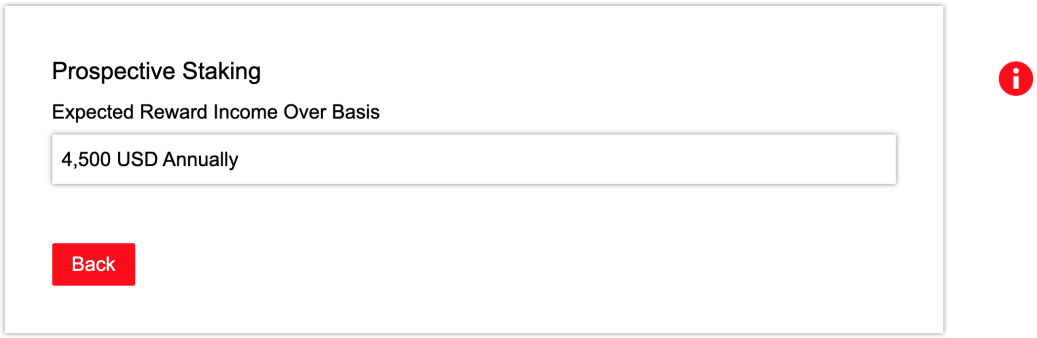
First the user selects the prospective staking experience from the experience prompt.



Then the user enters their information into the prospective staking form.



Then the user gets their projected earnings over their basis.

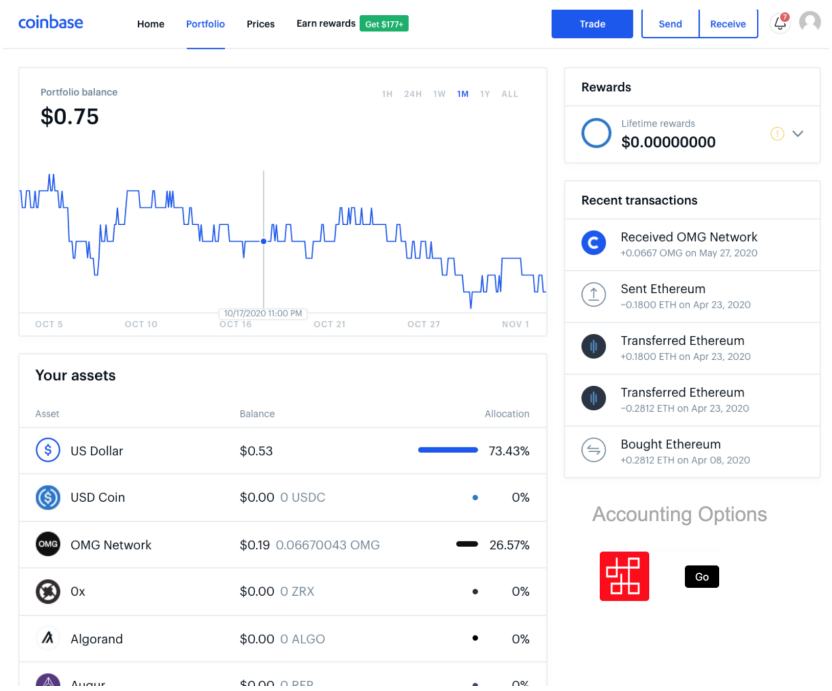


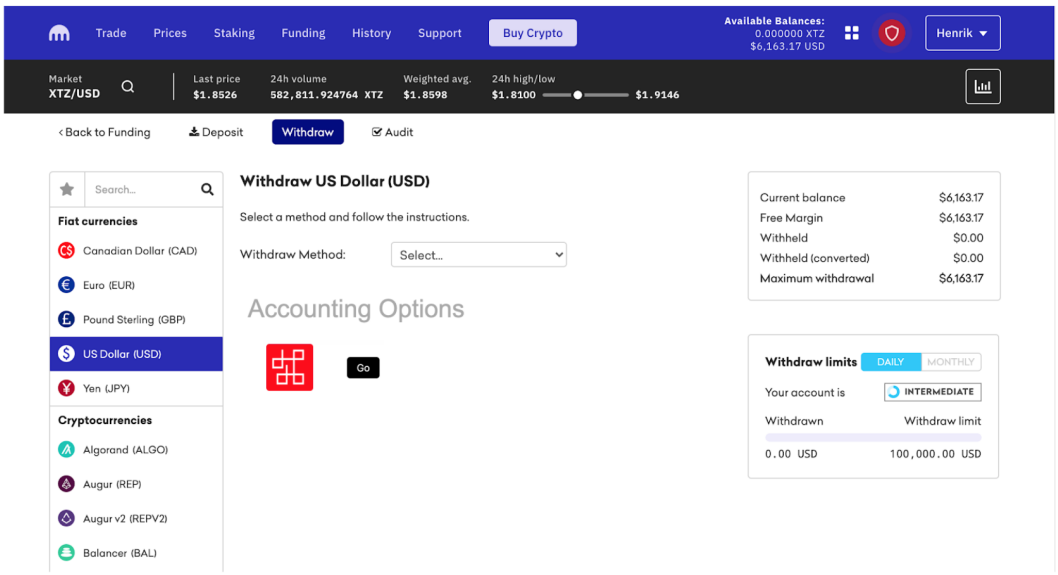
c. Accounting Analysis Client

For context, recall that these content windows are within windows on another client's DOM.

This user experience mockup walkthrough begins if the client decides to use our interface package. Without the interface package, simply our logo signature will be displayed on the webpage that the developers have hardwired our servers to.

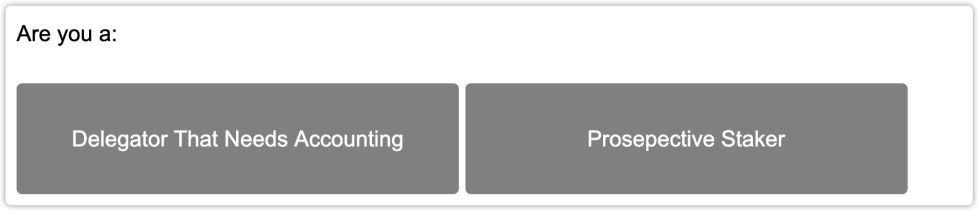
Here is the user experience beginning on a client's webpage. Note that we're not affiliated with Kraken or Coinbase, I use them as examples.



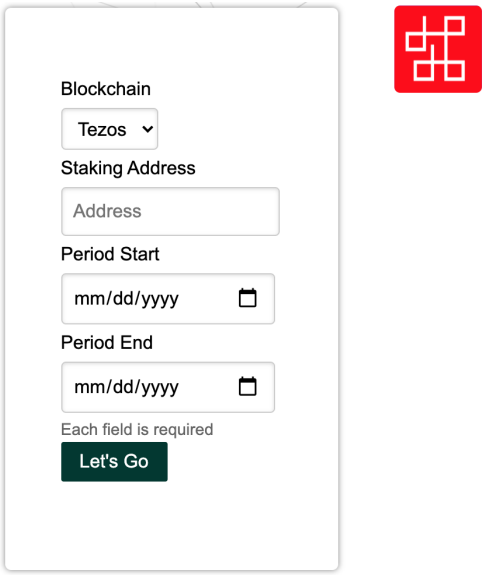


Below is the opening of the user experience client package.

The logo will animate to the right revealing the question content.



After the user selects delegator services, the content box transitions to the input form. The form might get preloaded with client information by our client package by our collaborations with devs who make the client client. The user will be able to scroll through this window to input, fiat, quantity realized, and accounting method. The example below is slightly deprecated.



The system processes the information and returns the final data to the interface.

Your Analysis

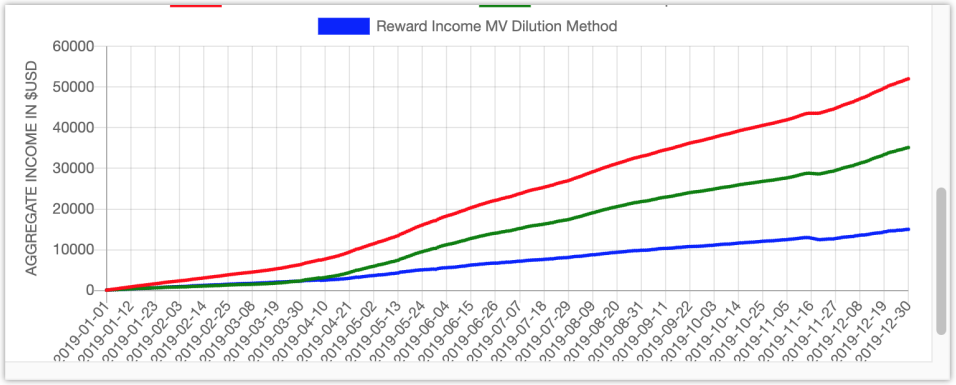
Accounting Method : Income Result

Cash Value Basis, Reward Income: $51990.23 USD

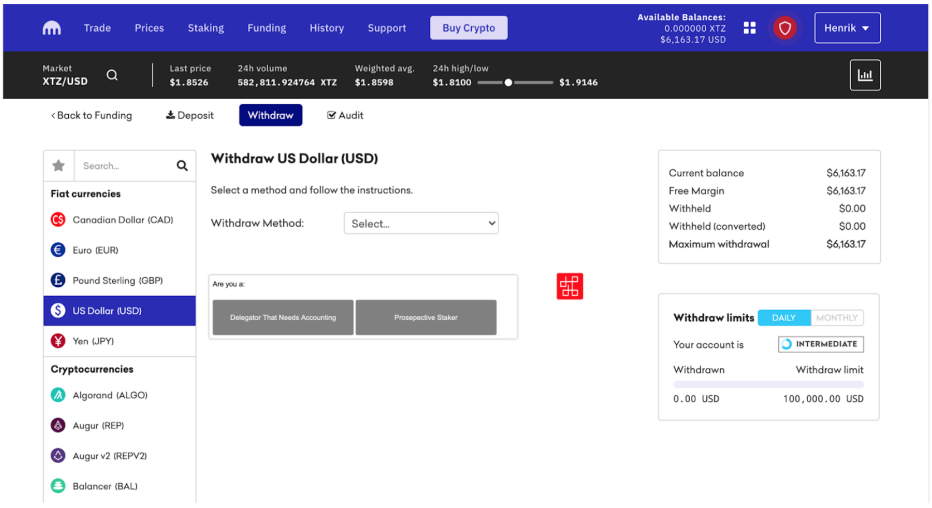
Accounting For Depletion, Reward Income: $35110.46 USD

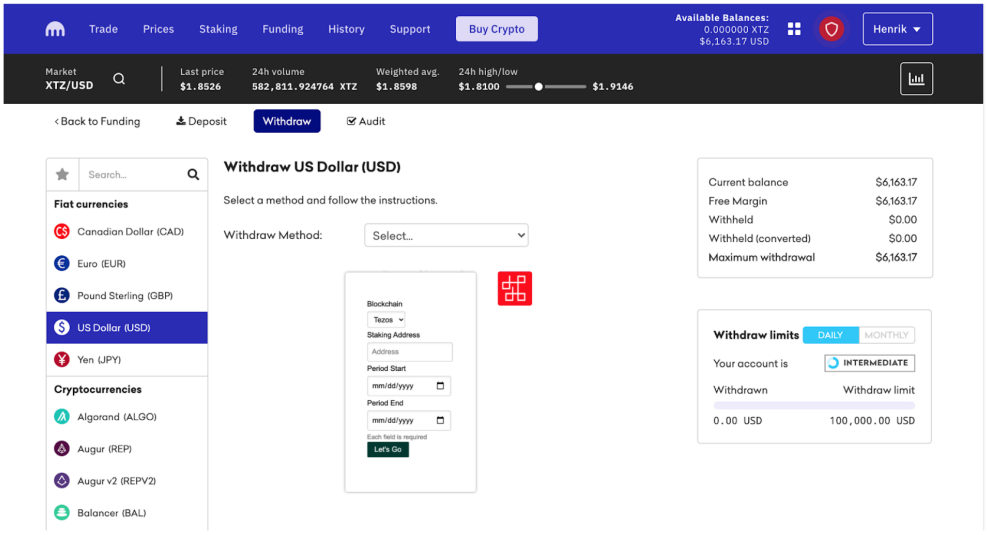
Accounting For Market Value Dilution, Reward Income: $15012.2 USD

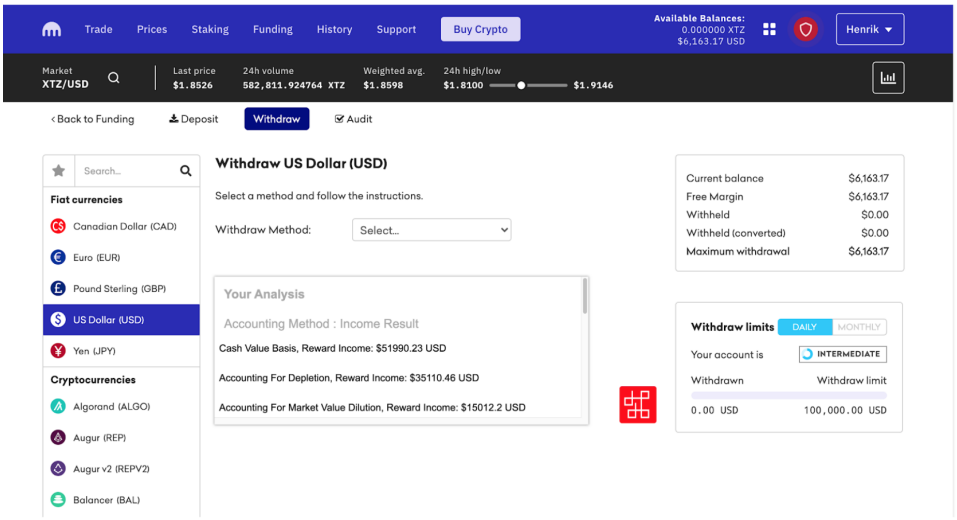
On window scroll down:

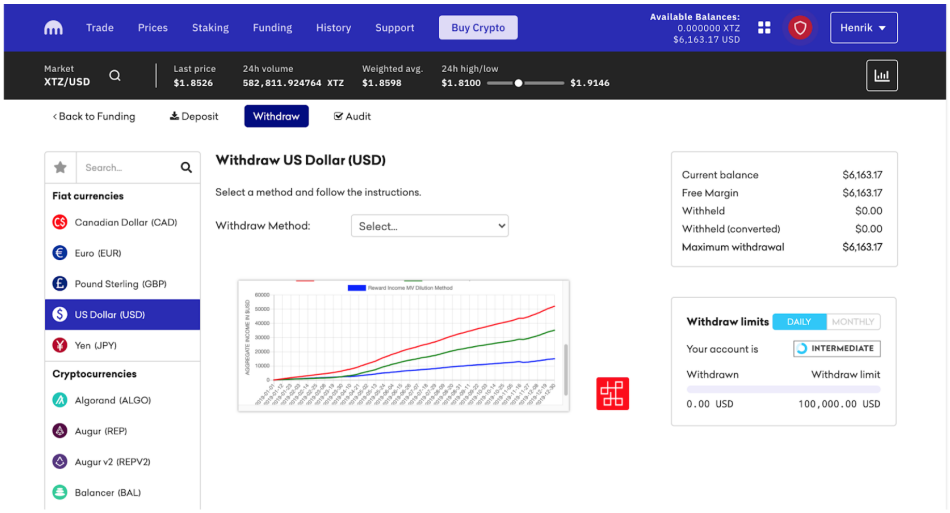


In context:




Trade
Prices
Staking
Funding
History
Support
Buy Crypto
Available Balances:
0.000000 XTZ
$6,163.17 USD
Henrik
Market
XTZ/USD
Last price
$1.8526
24h volume
582,811.924764 XTZ
Weighted avg.
$1.8598
24h high/low
$1.8100
$1.9146
Back to Funding
Deposit
Withdraw
Audit
Search...
Fiat currencies
Canadian Dollar (CAD)
Euro (EUR)
Pound Sterling (GBP)
US Dollar (USD)
Yen (JPY)
Cryptocurrencies
Algorand (ALGO)
Augur (REP)
Augur v2 (REPV2)
Balancer (BAL)
Withdraw US Dollar (USD)
Select a method and follow the instructions.
Withdraw Method:
Select...
Blockchain
Tezos
Staking Address
Address
Period Start
mm/dd/yyyy
Period End
mm/dd/yyyy
Each field is required
Let's Go
Current balance $6,163.17
Free Margin $6,163.17
Withheld $0.00
Withheld (converted) $0.00
Maximum withdrawal $6,163.17
Withdraw limits
DAILY
MONTHLY
Your account is
INTERMEDIATE
Withdrawn
Withdraw limit
0.00 USD
100,000.00 USD


Trade
Prices
Staking
Funding
History
Support
Buy Crypto
Available Balances:
0.000000 XTZ
$6,163.17 USD
Henrik
Market
XTZ/USD
Last price
$1.8526
24h volume
582,811.924764 XTZ
Weighted avg.
$1.8598
24h high/low
$1.8100
$1.9146
Back to Funding
Deposit
Withdraw
Audit
Search...
Fiat currencies
Canadian Dollar (CAD)
Euro (EUR)
Pound Sterling (GBP)
US Dollar (USD)
Yen (JPY)
Cryptocurrencies
Algorand (ALGO)
Augur (REP)
Augur v2 (REPV2)
Balancer (BAL)
Withdraw US Dollar (USD)
Select a method and follow the instructions.
Withdraw Method:
Select...
Your Analysis
Accounting Method : Income Result
Cash Value Basis, Reward Income: $51990.23 USD
Accounting For Depletion, Reward Income: $35110.46 USD
Accounting For Market Value Dilution, Reward Income: $15012.2 USD
Current balance $6,163.17
Free Margin $6,163.17
Withheld $0.00
Withheld (converted) $0.00
Maximum withdrawal $6,163.17
Withdraw limits
DAILY
MONTHLY
Your account is
INTERMEDIATE
Withdrawn
Withdraw limit
0.00 USD
100,000.00 USD

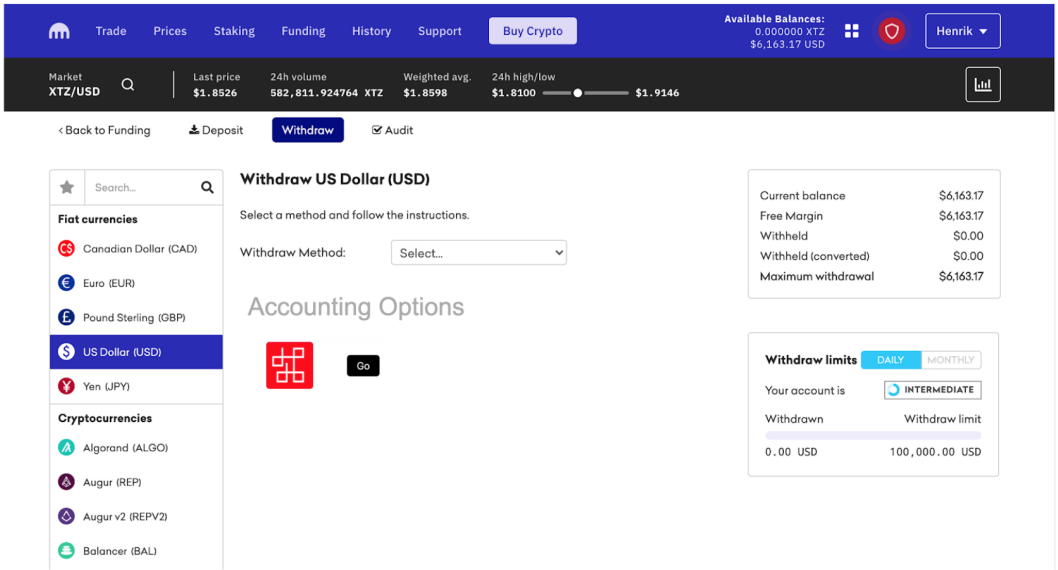


d. Prospective Staker Client

Like the accounting client, recall that these content windows are within windows on another client DOM website.

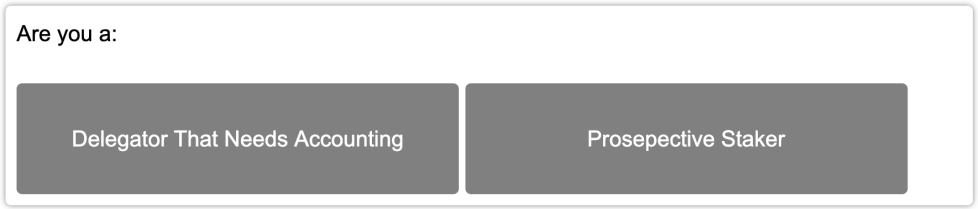
This user experience mockup walkthrough begins if the client decides to use our interface package. Without the interface package, simply our logo signature will be displayed on the webpage that the developers have hardwired our servers to.

Here is the user experience beginning on a client's webpage. Note that we're not affiliated with Kraken, I use them as an example.

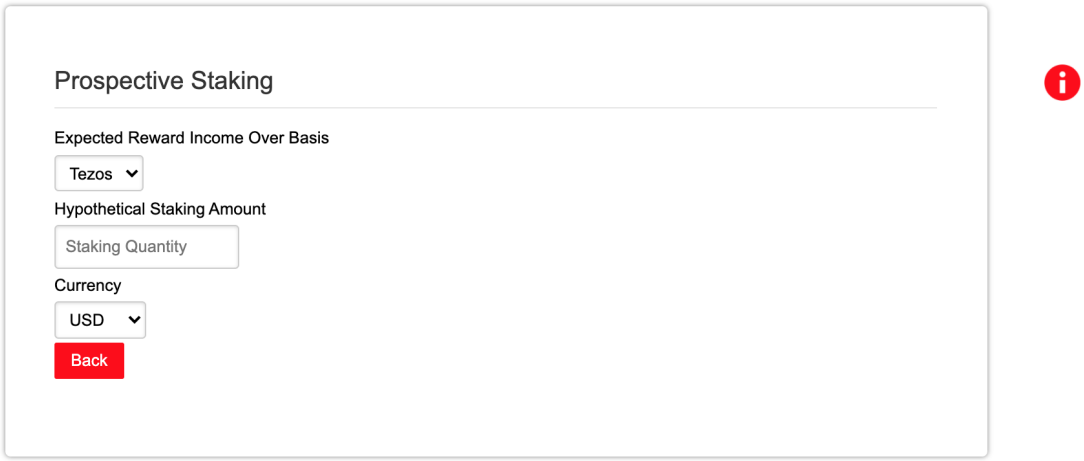


Below is the opening of the user experience client package.

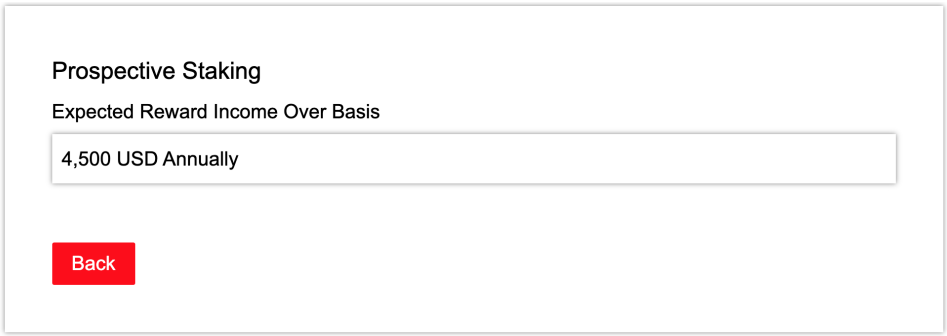
The logo will animate to the right revealing the question content.



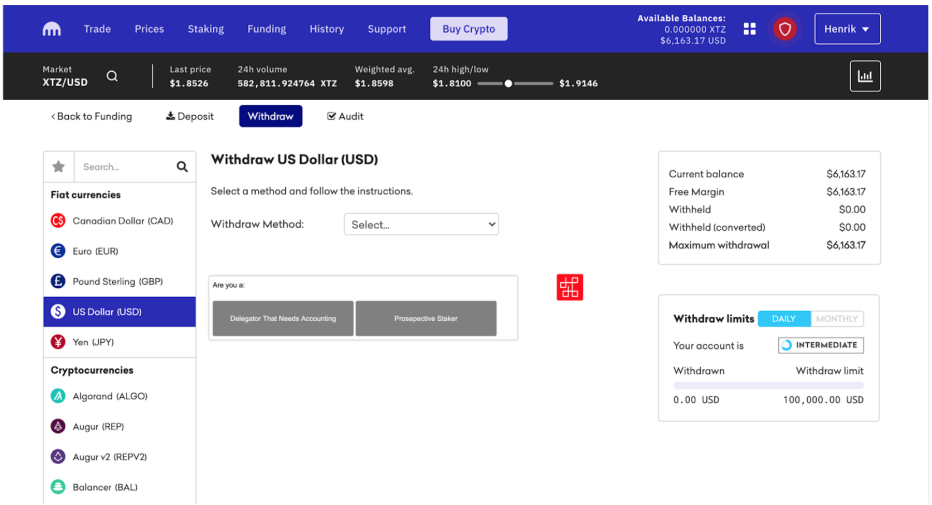
Then the client user will be able to enter their hypothetical metrics

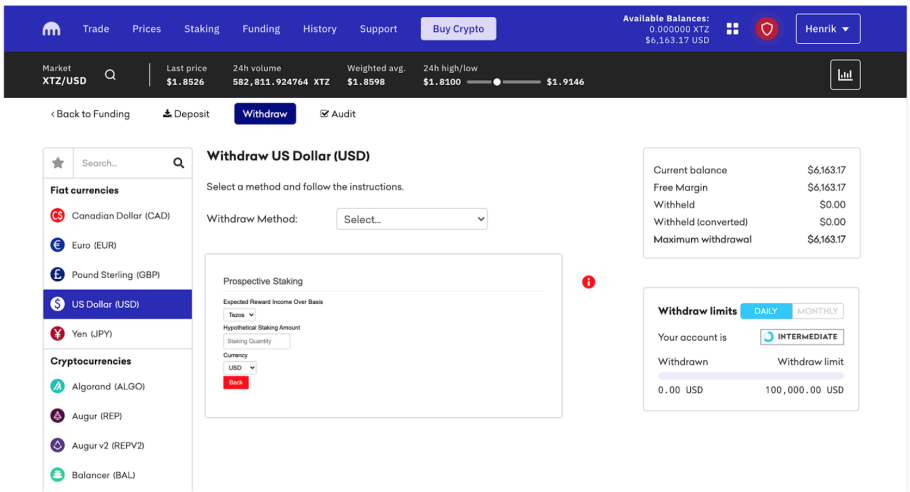


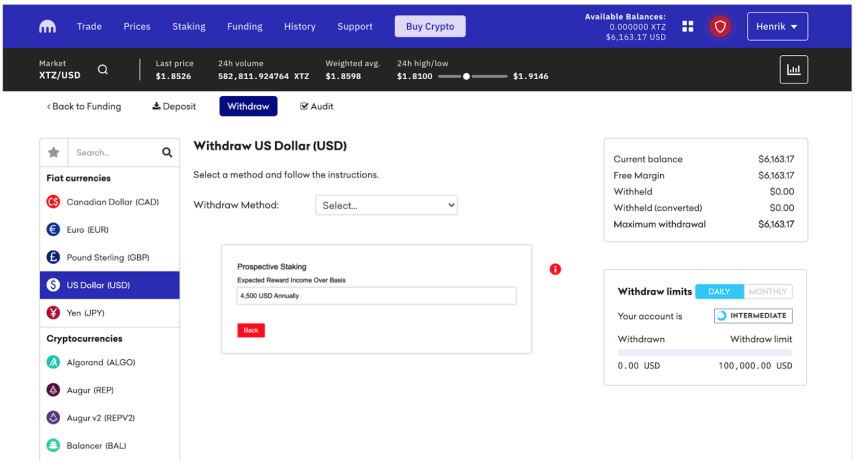
Then the user will observe the total


Prospective Staking
Expected Reward Income Over Basis
4,500 USD Annually
Back

In context:

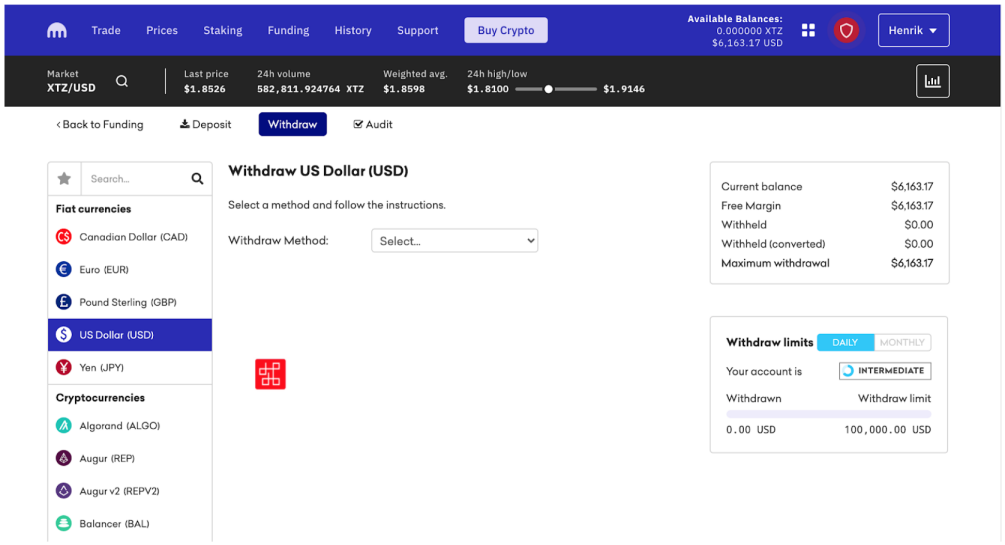

Trade
Prices
Staking
Funding
History
Support
Buy Crypto
Available Balances:
0.000000 XTZ
$6,163.17 USD
Henrik
Market
XTZ/USD
Last price
$1.8526
24h volume
582,811.924764 XTZ
Weighted avg.
$1.8598
24h high/low
$1.8100
$1.9146
Back to Funding
Deposit
Withdraw
Audit
Search...
Fiat currencies
Canadian Dollar (CAD)
Euro (EUR)
Pound Sterling (GBP)
US Dollar (USD)
Yen (JPY)
Cryptocurrencies
Algorand (ALGO)
Augur (REP)
Augur v2 (REPV2)
Balancer (BAL)
Withdraw US Dollar (USD)
Select a method and follow the instructions.
Withdraw Method:
Select...
Are you a:
Delegator That Needs Accounting
Prospective Staker
Current balance
$6,163.17
Free Margin
$6,163.17
Withheld
$0.00
Withheld (converted)
$0.00
Maximum withdrawal
$6,163.17
Withdraw limits
DAILY
MONTHLY
Your account is
INTERMEDIATE
Withdrawn
Withdraw limit
0.00 USD
100,000.00 USD


Trade
Prices
Staking
Funding
History
Support
Buy Crypto
Available Balances:
0.000000 XTZ
$6,163.17 USD
Henrik
Market
XTZ/USD
Last price
$1.8526
24h volume
582,811.924764 XTZ
Weighted avg.
$1.8598
24h high/low
$1.8100
$1.9146
Back to Funding
Deposit
Withdraw
Audit
Search...
Fiat currencies
Canadian Dollar (CAD)
Euro (EUR)
Pound Sterling (GBP)
US Dollar (USD)
Yen (JPY)
Cryptocurrencies
Algorand (ALGO)
Augur (REP)
Augur v2 (REPV2)
Balancer (BAL)
Withdraw US Dollar (USD)
Select a method and follow the instructions.
Withdraw Method:
Select...
Prospective Staking
Expected Reward Income Over Basis
Tezos
Hypothetical Staking Amount
Staking Quantity
Currency
USD
Back
Current balance
$6,163.17
Free Margin
$6,163.17
Withheld
$0.00
Withheld (converted)
$0.00
Maximum withdrawal
$6,163.17
Withdraw limits
DAILY
MONTHLY
Your account is
INTERMEDIATE
Withdrawn
Withdraw limit
0.00 USD
100,000.00 USD



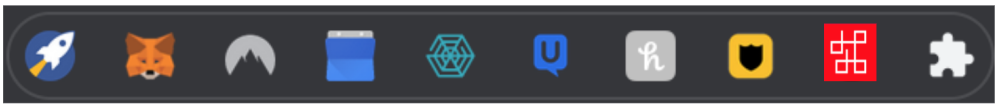
e. Signature Tag Client

The signature tag will be anywhere on the client webpage. The user experience will use our servers but the client will have their own interfacing display. Large platforms will likely be able to recreate our interface in their own display schema, flow, and appearance. This signature tag client module interface gives clients greater flexibility. We will also support client's developers in integrating our client module to their webpages.

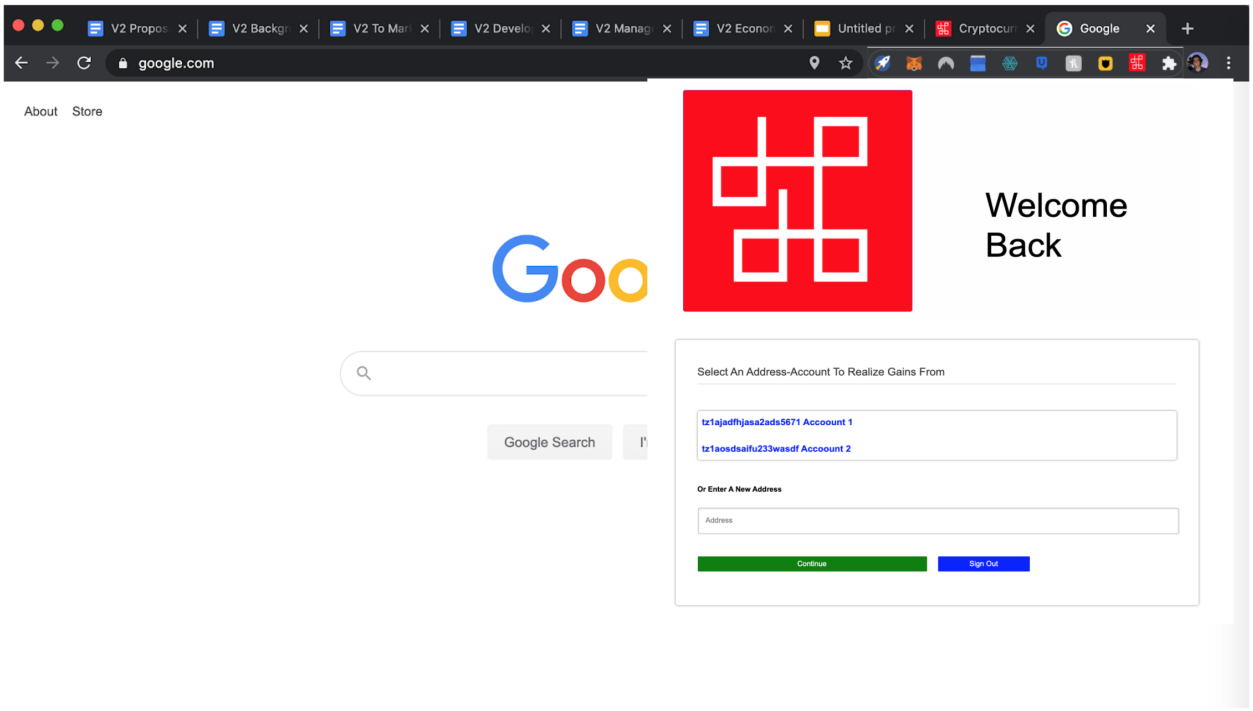


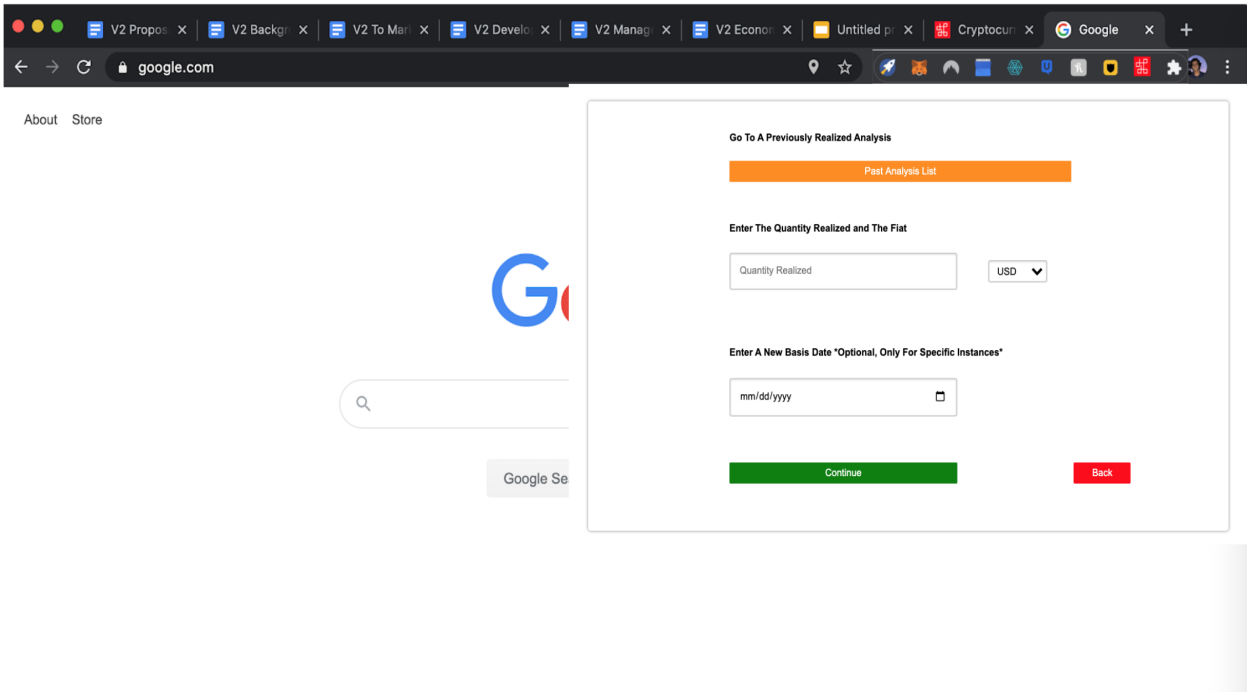
f. Plug In App

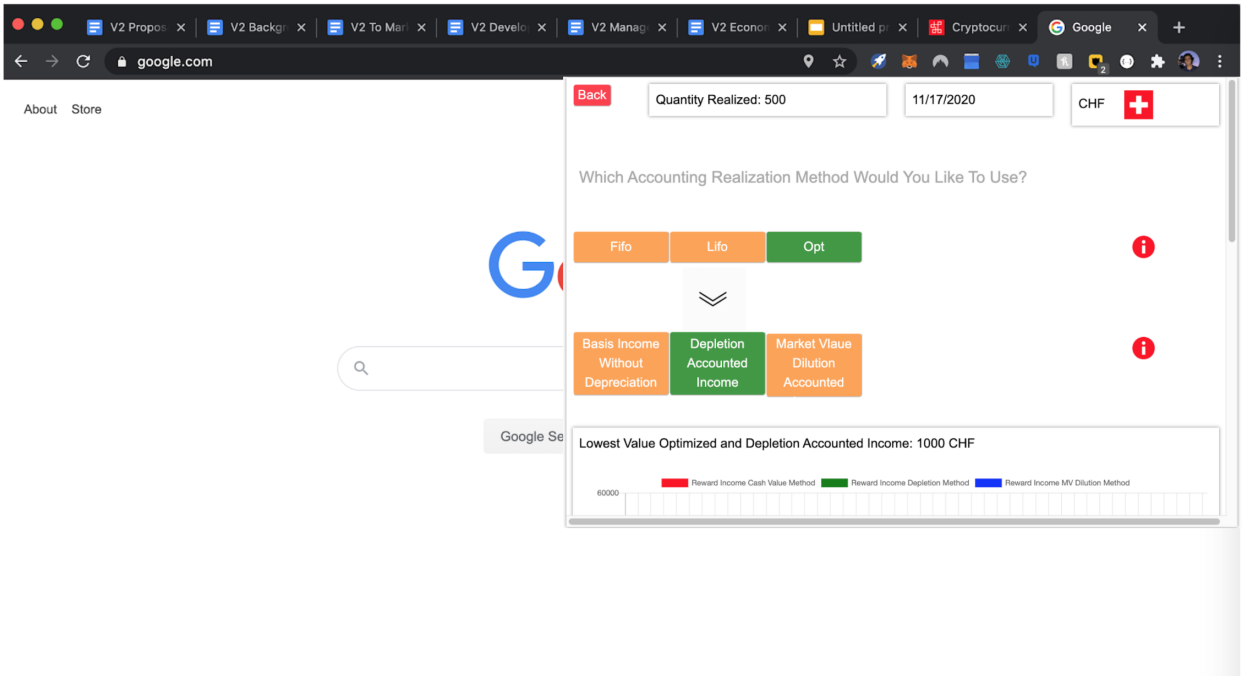
The displays on the Plug In Browser App will be similar to the client. With a drop down from the plug in bar on browsers.

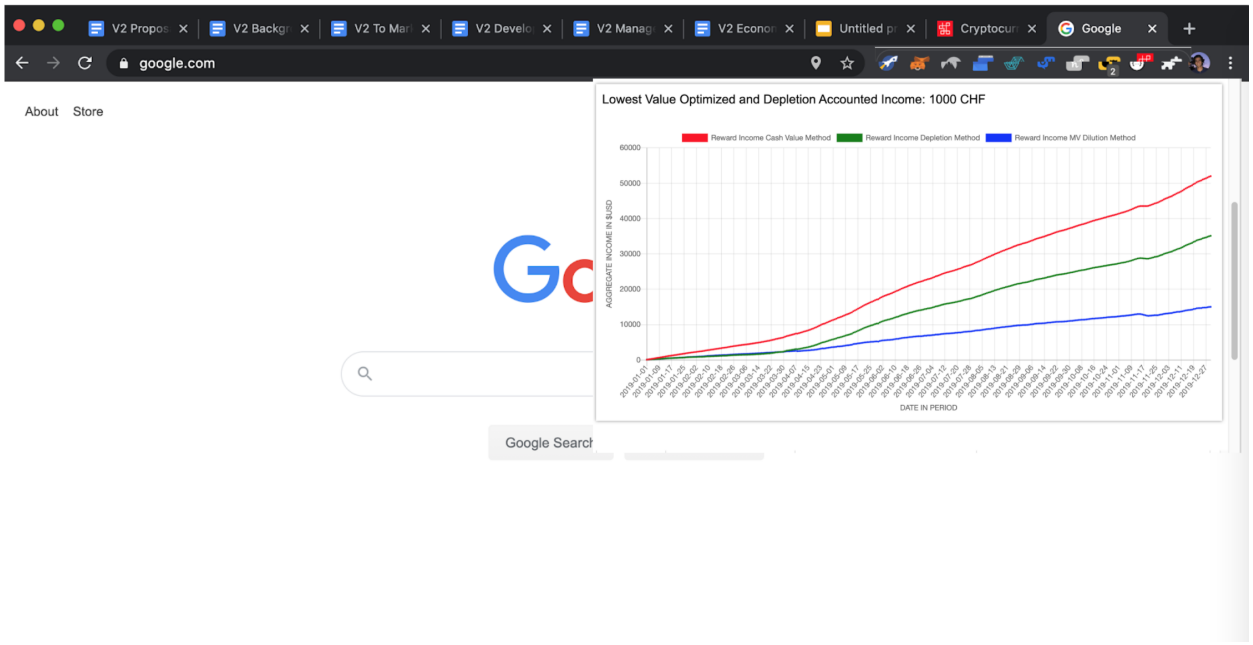


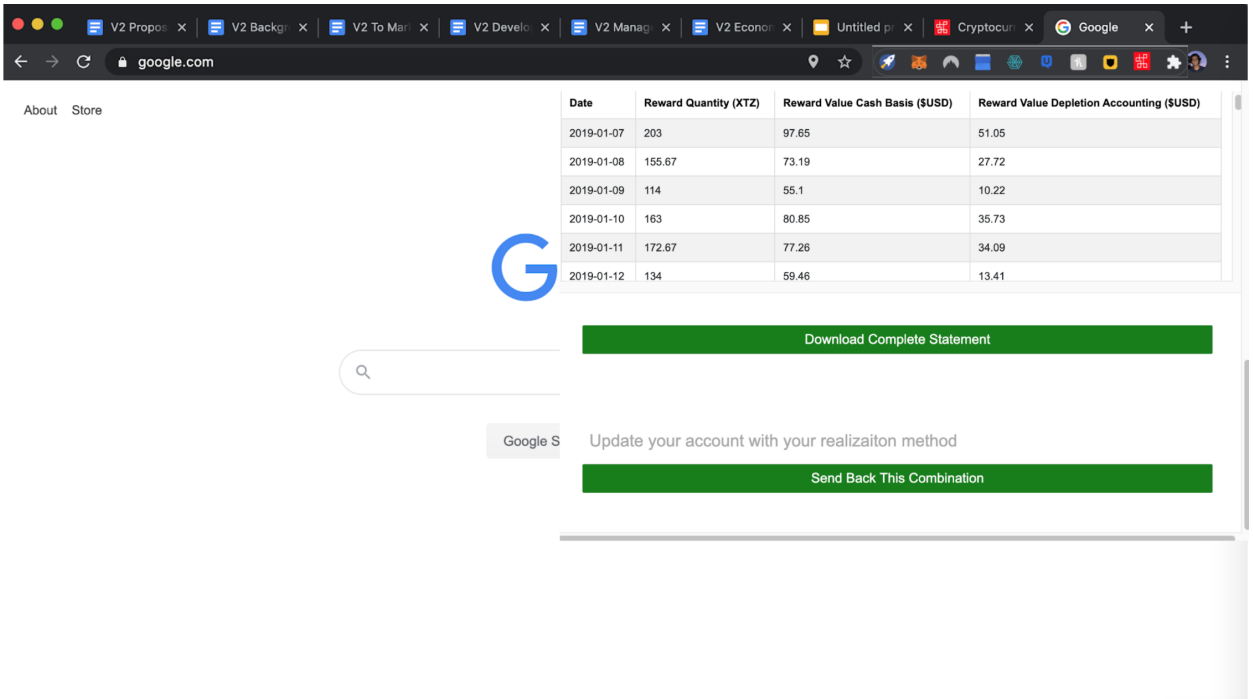
Once the user selects the PoS Tax app the dropdown with the experience selector will appear. The mockups below only show the accounting user experience and not the prospective staking experience.




google.com
About
Store
Google Se
Go To A Previously Realized Analysis
Past Analysis List
Enter The Quantity Realized and The Fiat
Quantity Realized
USD
Enter A New Basis Date "Optional, Only For Specific Instances"
mm/dd/yyyy
Continue
Back


google.com
About
Store
Google Se
Back
Quantity Realized: 500
11/17/2020
CHF
Which Accounting Realization Method Would You Like To Use?
Fifo
Lifo
Opt
Basis Income Without Depreciation
Depletion Accounted Income
Market Vlaue Dilution Accounted
Lowest Value Optimized and Depletion Accounted Income: 1000 CHF
Reward Income Cash Value Method
Reward Income Depletion Method
Reward Income MV Dilution Method
60000


google.com
About
Store
Lowest Value Optimized and Depletion Accounted Income: 1000 CHF
Reward Income Cash Value Method
Reward Income Depletion Method
Reward Income MV Dilution Method
AGGREGATE INCOME IN $USD
DATE IN PERIOD
Google Search


google.com
About
Store
Date
Reward Quantity (XTZ)
Reward Value Cash Basis ($USD)
Reward Value Depletion Accounting ($USD)
2019-01-07 203 97.65 51.05
2019-01-08 155.67 73.19 27.72
2019-01-09 114 55.1 10.22
2019-01-10 163 80.85 35.73
2019-01-11 172.67 77.26 34.09
2019-01-12 134 59.46 13.41
Download Complete Statement
Update your account with your realizaiton method
Send Back This Combination

CryptoCount Core Development Plan

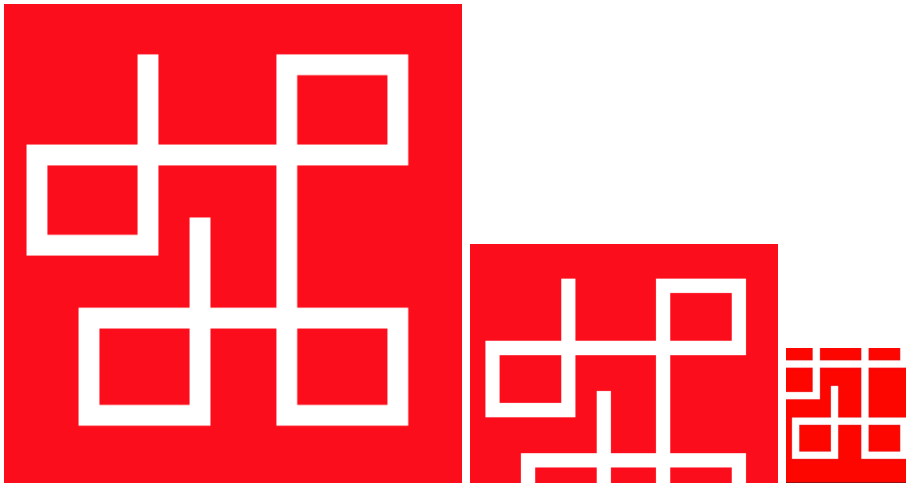


Table Of Contents:

1. Development Plan

a. Git Development Structure

Each of the system event processes listed in the core technical plan above represents a branch of our git structure.

Our git development structure is similar to many projects of similar build team size (less than 3).

Our git branches are organized as levels as follows.

Level 0: Main Branch - this branch holds all stable code for the project and is the external publicly viewable code branch.

Level 1: Integration Branch - this branch takes topic branches and tests them for stability and integration.

Level 2: Topic Branches - topic branches coincide with the event processes listed in the core technical document. These branches all pertain to a set of subfunctions of the whole system. Once development is completed on one of the topics it is pushed up to the integration branch.

Level 3: Topic Sub Issue Branches - These branches are for development of small sub problems from the main topic branch development. A developer can create several of these a day to tackle micro details of the development. These sub issue branches are tested at this level and are integrated into the topic branches.

b. Scheduling Methodology

Development of a system is not a linear process. In this section I will outline our mechanism for evaluating the length of time that a component will take to be developed. Then, I will use a selected group of components to evaluate the development timeline for all components in the system.

There is an average development time of each component that can be identified. We will take every component and multiply it by the average development time of each component.

c. Functional and DataBase Development Plan

Developing Technical Section 4.e.2 DB Market Value Update

What we got to do for this part?

For this part we have to update the db_update.js file in the back end.
We have to update the queries to coingecko to get 19 more fiat currencies.
We have to assign space in the mongo db to put these 19 other currencies.
We have to push the fiats into the database.

How long this gonna take?

First we have to get all our queries down to the coingecko data source. This will take some time, probably 10 hours.

Then we have to integrate all of those queries into the program - approximately another 5 hours.

Then we have to write the code to update the database with the newly found axios queries, another 4 hours.

All together ~ 3 days.

Developing Technical Section 4.e.4 Push Previous Address Analysis To DB

What we got to do for this part?

We got to wait for the front end client to ping us back with what they did or did not use.

We have to build the event handling on the front end from the interface down to the socket emission to the back end where the socket catches the front emission and handles it by transforming the accounting set with the remaining eligible pool of entries for future realization.

How long this gonna take?

There is the event handling in the interface which we will take 3 hours to get all up to speed. Developing the backend data handling will take a couple days to ensure that everything is done correctly.

All together ~ 2.5 days.

d. Accounting Analysis Development Plan

Developing Technical Section 4.a.8

What we got to do for this part?

We have to query the TZKT database to obtain the reward set of the user, the baker address, and the baker alias. We then have to output this for integration into the next phase of analysis.

How long this gonna take?

We have to structure our request to TZKT with the user data. We must also establish a special API to TZKT so that our system can obtain and process data in a timely manner for large accounts.

The development for this problem will take approximately 2 days. The partnership for this problem will take an unpredictable amount of time, but hopefully, it is relatively brief (less than 3 weeks).

We'll call this a 2.5 day development time.

Developing Technical Section 4.a.9

What we got to do for this part?

This part we calculate the basis and the reward set relative to the basis. We are taking the start date or basis date from the client obtaining the price of the cryptocurrency that day, then we are scaling the cryptocurrency rewards by that basis price.

How long this gonna take?

This process integrates all 20 supported fiats. To integrate the fiat coingecko query and the reward set query outputs into this process will take approximately 4 hours. Then writing the code to perform the calculation will take 2 hours. Then

integrating the process back to the next point of the system for additional analysis will likely take another 4 hours.

All together we'll call this 2 days.

Developing Technical Section 4.a.16

What we got to do for this part?

This section is catching the returned data objects from the server. We have to establish either an asynchronous REST connection or a web socket to obtain the data. Then the client system has to write the returned data into client storage.

How long this gonna take?

The connection catching part will take about a day. The handling of the data and getting it prepared for its next step in the interface will take about a day as well.

Total time for this development will be about 2.5 days.

e. Prospective Staker Development Plan

Developing Technical Section 4.b.2

What we got to do for this part?

The system must internally obtain the price set from its database. The price set is determined by the fiat the user is requesting.

How long this gonna take?

The system must query its internal database. The query code will take 3 hours to write. Then there is the integration into the next step of the prospective staker path. This will take another 3 hours to develop.

This section will take 1 day to develop.

Developing Technical Section 4.b.4

What we got to do for this part?

We have to calculate the expected yield of staking Tezos for a prospective user. We need the fiat basis price and the quantity of XTZ the user is expecting to stake.

We then calculate the expected yield through simple arithmetic.

How long this gonna take?

We have to integrate this process. The staked quantity and fiat basis price should already be stored as variables at this point. The code for the calculation is quite simple. That will take probably less than 1 hour.

However, making sure the whole sub system is working smoothly will likely take 1 day.

This process will take 1 day.

Developing Technical Section 4.b.7

What we got to do for this part?

This part is interfacing to the user through the front end. The asynchronous event loop of web pages will be reloading the page looking for active changes to the DOM.

We must prepare the DOM for more variables of data to be added in.

We must assign proper local storage and event handling for data population increases.

How long this gonna take?

Organizing rendering processes and handling will change constantly as more core functionality branches are developed.

This process is dynamic and will likely take 2 months because it is dependent on server side processes. The time for this process will accumulate to 7 days.

f. Client Integration Module Development Plan

Developing Technical Section 4.c.1

What we got to do for this part?

This part is beginning the server process for an instance of our system. This part is triggered by a user requesting data from our server.

The data can come through our website, the client cdn, or through the plug in app.

The job for this part is making sure our content delivery channels are all functioning properly.

How long this gonna take?

To get a strong CDN package, additional research is needed. This will likely take 7 days.

Development from the research will then take 1-2 days.

All together this section will take 9 days.

Developing Technical Section 4.c.3

What we got to do for this part?

The section is a css package rendering solution for client systems. Luckily, most of our interfacing and rendering solutions cross over with each other. This makes development less unique to each content delivery method.

The css package will be developed with scss.

How long this gonna take?

Although the interface packages have high crossover, each one will be unique to itself by a sub set of characteristics.

The development for the identifying characteristics in a scss package will likely be 7 days. The longer time is to make sure that the colors and details of user interfaces are perfect.

Over the span of 2-3 months this process will accumulate to about 7 days.

g. Plug In App Development Plan

Developing Technical Section 4.b.7

What we got to do for this part?

This is the interfacing for the pug in app. As mentioned above, there is a lot of crossover between interface channels, lowering development time.

However, for the plug in app, integration of a DOM model into each specific browser extension format will take additional research because Chrome, Safari, FireFox, and the like all have different integration code.

How long this gonna take?

To research each browser's extension integration and creation code/methods will take about 7 days. Integrating all the syntax obtained from the research will take an additional 3 days.

Altogether this will take 10 days.

h. Development Plan Calculation and Summary

Now in order to get an idea of a time scale, I will take the average of all the components already discussed.

Above, we analyzed 11 development sections. The average development time for a section is 2 days.

Total Dev time = Avg * (number of process events) = 2.5 * (51) = 127.5 days.

Stability testing and code integration are included in the total development time.

I believe that his development timeline is an accurate representation of the time it will take to develop this product package.

CryptoCount V2 Core Management Plan

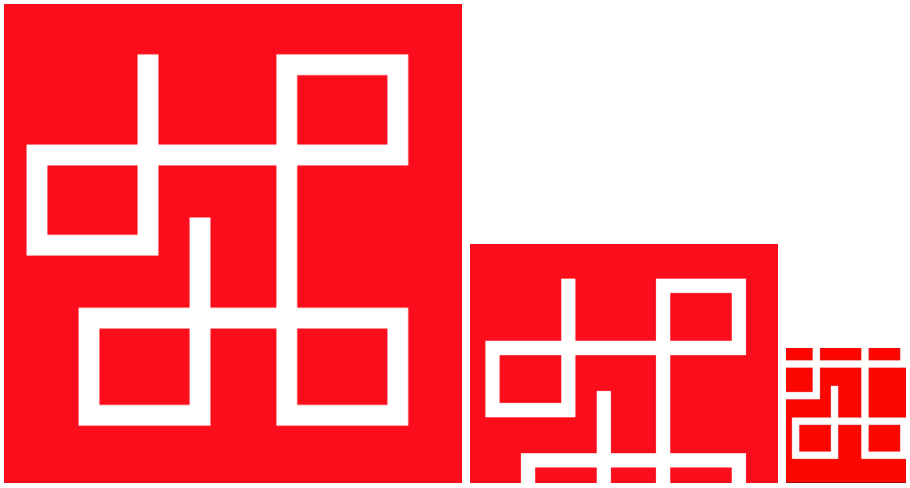


Table Of Contents:

1. Deployment and Management

a. Site Deployment Process

1. Server Caddy

Caddy is a server management stack that allows developers to deploy automatic RESTful APIs. Caddy configuration allows for the integration of our front end and back end together on the server.

2. AWS EC2

The hardware of this project is housed by Amazon. AWS's EC2 server instance product is where we host our server. AWS has a shell that allows you to take control of the EC2. This is where we deploy our caddy config with the front end and the back end.

b. Google Site Management

1. SEO

Search engine optimization will always be a crucial component of any software or website product. Our website will be constantly refined with the best combination of efficient and concise content, meta tags to drive google's crawling engine, and user retainment for significant periods of time on the website.

2. Crawling Web Pages

As management, I will send our web pages into Google's crawling queue for processing into their search engine. Any time any of the web pages are updated, I will resubmit the pages to Google for crawling.

3. Ads

As CryptoCount's marketing campaign and flooding strategy takes effect, Google Ads will be a perfect place for us to promote our website to google users. Using targeted google ads based on market / industry keywords will help us drive more traffic and add more users. At the time of writing this document, the URL postaxation.com is on the sixth page of google results for the search query "postaxation".

c. Plug In App Deployment Process

The plug in app will be minimal additional front end work for a whole new set of functionalities for the users. The plug in app requires that we walk through each browser's system of deploying and maintaining the plug in apps. These apps will be deployed to the chrome store and safari extensions.

d. Plug In App Management

The extension/plug in app will be another product to promote on the website. The app will receive reviews where we can see if users think our products are worthwhile and function well. We're developing personal accounting tools, so user satisfaction has to be 100%.

e. Client Integration Management

A big part of PoS Tax V2 strategy is integrating into other client's systems. In order to achieve this, the PoS Tax officer team will be talking to developers of client systems and the officers of the client systems to create a pathway for integration that makes sense for all parties involved.

Critically, the integration into client's softwares will be free because this project is open source, so any involvement from PoS Tax to clients will solely benefit the client's users, therefore also the client, and of course the Tezos ecosystem which gets more exposure.

f. Future Version

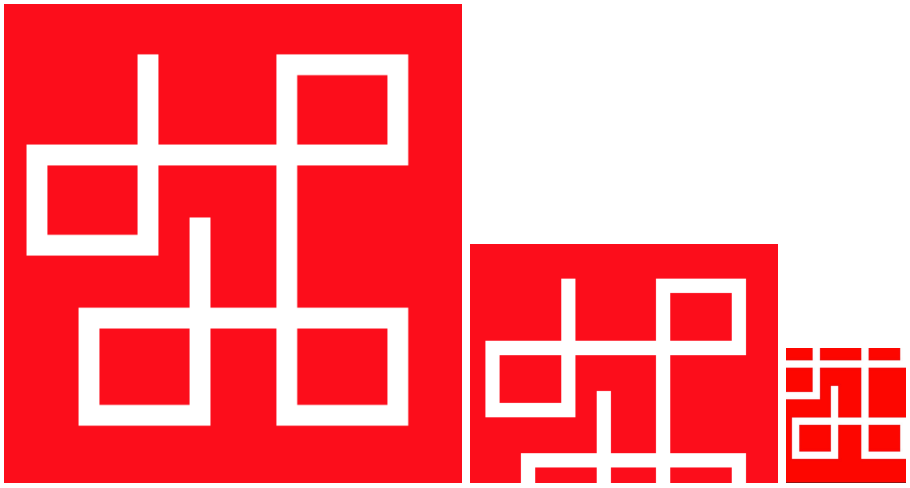
The future version of PoS Tax can cast projections and injections into the blockchains to return users a type of yield asset.

Process Execution Officers

Executive Officer - Henrik Moe

Technology Officer - Wei Wang

CryptoCount Core To Market, Integration, and Flood Plan



To Market Plan

In our market and use plan, we are specifically targeting delegators who need discount accounting for their blockchain rewards.

Media

We will target widespread awareness and masses of delegators and prospective delegators through media releases. We will conduct a keynote with our products.

Direct Targeting

We will target delegators by going into group chats, reddits, slacks, discords, etc. throughout the Tezos ecosystem. Google ads also fall in here.

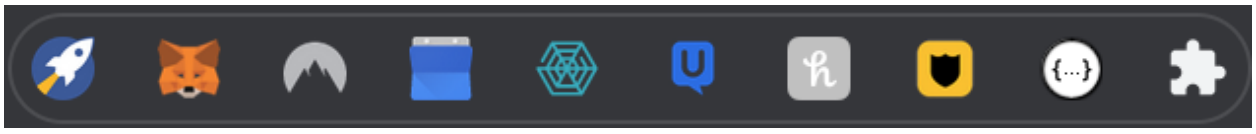
Ways To Access:

Website

The website is ground zero for this project. On the website will be full documentation and walkthrough on how to use our products. The website will have full documentation to our other channels of obtaining our product as well. The other channels are the plug in app and integrating into a client's softwares.

Plug In Application

Our plug in application will be a core component of our platform. The plug in app will follow users everywhere and will get them comfortable with us. Having a plug in app makes the users life easier and will further engrain us into the app utility set of users.



Client Integration Packages

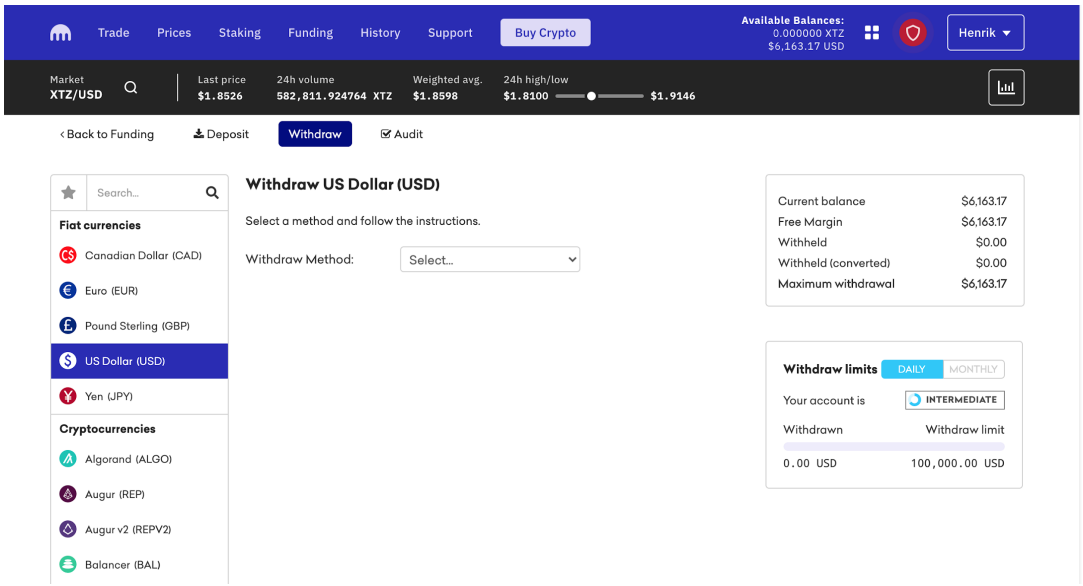
We will offer our client integration packages to exchanges and provide their tech teams with technical support for no cost to the client - boosting the credibility and activation of the tezos economy.

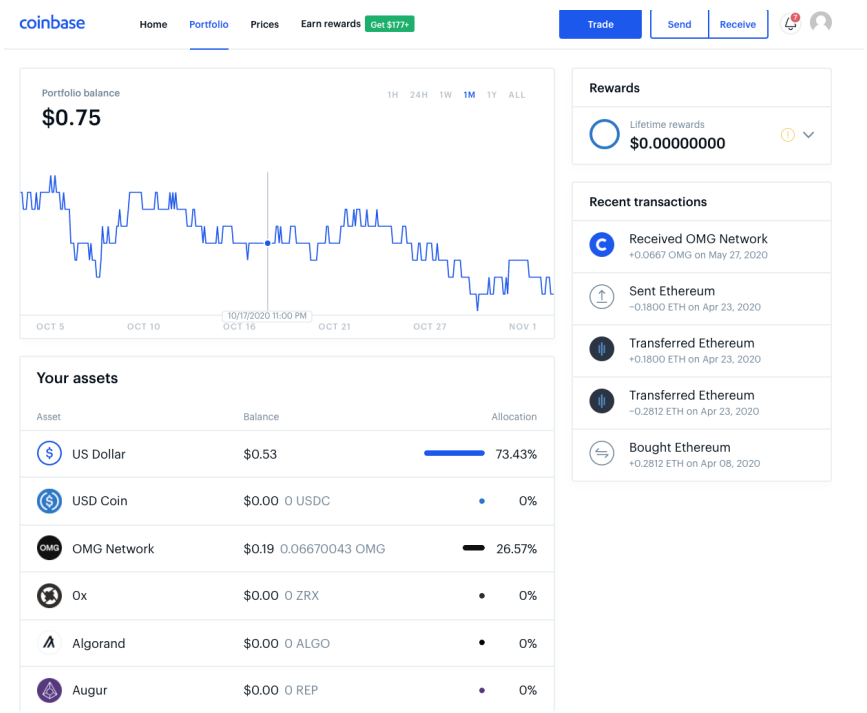
Our unique ux dialogue box will stay constant through all the partnerships. There will be a signature tag on the ux box that links to our website.

The contents of the script tag are the socket dialogue (or the restful dialogue) to the server.

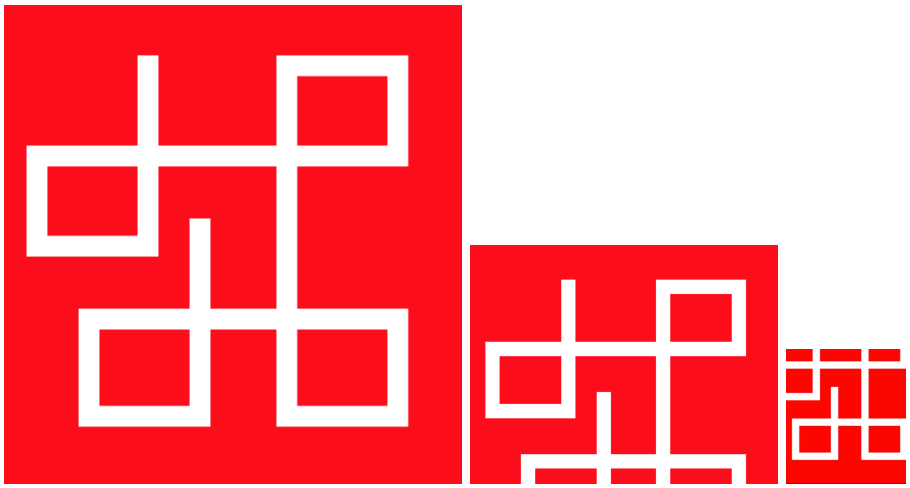
Client requests can be configured to the webpage through our optional bootstrap html/css layout that can be added to the webpage by its canvas id. Its sass/css file will be supplied for custom modding as well. Developers can also omit our front end products and just hardwire their way into our api. In this event, we will assist developers and request our signature page element with the hyperlink be displayed in the client DOM.

Two example client DOMs:





CryptoCount V2 Project Appraisal



Economic Value Plan

Economic value projection - directly to the market base and indirectly through strategic to market plans increasing ecosystem awareness.

PoS Tax V1 has an average economic impact when fully realized as a software of 645,000 XTZ.

With CryptoCount, the market for our products increases dramatically through the Tezos ecosystem. We are now targeting all delegators to give them a complete tax accounting software solution.

I estimate that the market impact will be increased by a magnitude of at least 6. Pushing our value in tax savings to Tezos stakers to over 3.5 million XTZ.

I plan on taking this project to the front of blockchain news. Indirect impact on the price of XTZ will also raise our value to the community.

Using the PoS Tax V1 as a reference 27,000USD for 645,000XTZ impact.

CryptoCount project appraisal: 180,000 USD.