Blockchain VPP POC Quick Start Guide

**Purpose**: Build a Blockchain based application to handle transmitting PHI across various actors in a typical healthcare scenario.

**Knowledge**

* Be able to explain what a Blockchain is and what it is used for.
* Be able to explain the Ethereum framework and what smart contracts are.
* Have a basic understanding of how the below software (solidity, JavaScript, etc.) work together.
* Have a basic understanding of the steps of typical healthcare scenarios (e.g. making an appointment as a patient)
* Have a basic understanding of the U.S healthcare industry in regards to the relationships being payers, providers, etc.

**Infrastructure**

We have two repositories for this project, the main one in GitHub and a mirror in Team Foundation Server. They house all the existing work items (documents, code, etc.). You can use to see how far we are right now.

The GitHub link is below:

<https://github.com/DonNull0ck/SDLC_Blockchain_Demo>

You will need a GitHub account to access this repository. Reach out to Collin Donaldson ([cdonaldson@sdlcparnters.com](mailto:cdonaldson@sdlcparnters.com)) to be added as a contributor to this project.

This is the login for TFS:

http://sdlcalm01.sdlcpartners.com:8080/tfs/Delivery/VPP

You can login in using your SDLC email and password. If you are not prompted for a username and password or receive an error, contact IT, in particular John Vorchak. Likewise if you do not have access to the TFS directory (VPP 🡪 Blockchain POC) reach out to John or someone else in IT about access. Note: you must be connected to SDLC’s network to utilize the TFS.

The TFS also has a task list breakdown for specific components of the project.

**Stack:**

Disclaimer: Some elements of the stack are evolving and may change from the time of writing. Note that elements not mentioned below, such as what editor/IDE you use to write JavaScript, are up to individual preference.

Front End: HTML, CSS (derived from UI/UX software Zeplin, InVision, etc.), JavaScript (React, Web3, Drizzle).

The Blockchain itself: Ganache (for testing at least)

Backend: Solidity, JavaScript (React, Node.js config, etc.)

Build Automation: Truffle, Node.js (runtime), NPM (package manager)

Other: Remix IDE, Personal IDEs (Atom, Visual Studio Code, etc.), MetaMask, MyEtherWallet

Potential Future Software: Swarm, Whisper

Note: May add diagram showing how this software relates here

**Helpful Links:**

**General Environment Background, Documentation, Setup (Node.JS, NPM, React, etc.)**

The node.js website (and installation):

<https://nodejs.org/en/>

For installing NPM’s node-gyp and its dependencies (READ THIS!)

<https://www.npmjs.com/package/node-gyp>

Python v.2.7:

<https://www.python.org/downloads/>

XCode (Mac only): <https://idmsa.apple.com/IDMSWebAuth/signin?appIdKey=891bd3417a7776362562d2197f89480a8547b108fd934911bcbea0110d07f757&path=%2Fdownload%2F&rv=1>

Windows Easy Option

npm install --global --production windows-build-tools from an elevated PowerShell or CMD.exe (run as Administrator).

Windows Hard Option

* Install Visual C++ Build Environment: [Visual Studio Build Tools](https://visualstudio.microsoft.com/thank-you-downloading-visual-studio/?sku=BuildTools) (using "Visual C++ build tools" workload) or [Visual Studio 2017 Community](https://visualstudio.microsoft.com/pl/thank-you-downloading-visual-studio/?sku=Community) (using the "Desktop development with C++" workload)
* Install [Python 2.7](https://www.python.org/downloads/) (v3.x.x is not supported), and run npm config set python python2.7 (or see below for further instructions on specifying the proper Python version and path.)
* Launch cmd, npm config set msvs\_version 2017

Windows troubleshooting:

<https://github.com/Microsoft/nodejs-guidelines/blob/master/windows-environment.md#compiling-native-addon-modules>

On windows: Helpful software to create the symbolic links:

<http://schinagl.priv.at/nt/hardlinkshellext/linkshellextension.html#download>

A Git Bash download (Windows) for installation, terminal commands, etc. (HIGHLY RECOMMENDED):

<https://git-scm.com/downloads>

The link to download Web.JS:

<https://github.com/ethereum/web3.js/#installation>

Link to install and compile solidity on Windows using the solc compiler (optional):

<https://www.codeooze.com/blockchain/solc-hello-world/>

**Blockchain(dApp) Environment Background, Documentation, Setup (Ethereum, Solidity, Truffle, etc.):**

The Ethereum project homepage:

<https://ethereum.org/>

The online IDE for Solidity

<http://remix.ethereum.org/#optimize=false&version=soljson-v0.5.2+commit.1df8f40c.js>

The documentation for the Solidity language:

<https://solidity.readthedocs.io/en/v0.5.2/>

The documentation for the Web 3 JS framework:

<https://web3js.readthedocs.io/en/1.0/web3-eth.html>

Ethereum wallet plugin MetaMask:

[https://MetaMask.io/](https://metamask.io/)

The Truffle Framework homepage:

<https://www.truffleframework.com/>

The download page for the Truffle development environment itself:

<https://www.truffleframework.com/truffle>

The download for Ganache to visualize and deploy the Blockchain:

<https://truffleframework.com/ganache>

The download page for Drizzle to ease UI development:

<https://www.truffleframework.com/drizzle>

**Roles:**

Disclaimer: These roles are examples and not formal titles; their objective is to help you understand this project’s work using terms familiar to employees who have worked on client assignments or other such work experience. There is a strong possibility you will wear multiple hats working on this project.

Application Developer (Front End, Back End, etc.): Write JavaScript to build the front end (e.g. web pages), convert UI/UX design to webpage using CSS, write Solidity smart contracts, etc.

Business/Project Analyst: Perform requirements gathering/analysis (functional, non-functional, etc.), create and maintain documentation (e.g. user stories, use cases, flowcharts, etc.), assist leadership with project navigation, etc.

Project/Resource Manager: Obtain appropriate company assets for project, build and maintain the project team, manage team and project fluctuations (team growth/loss, deadlines, etc.)

Quality Assurance (QA) Analyst/Engineer: Write test cases, perform manual and automated testing (unit, regression, AB, etc.), help troubleshoot issues as they arise, etc.

Subject Matter Expert/Specialist: Provide expert knowledge, enforce best practices and solid methodologies, consult on the feasibility of the project as well as specific components, etc.

Systems/DevOps Analyst/Administrator/Engineer: Manage files stores and packages (source code, diagrams, documents, etc.), document, manage, and perform with application deployments, ensure compatibility across platforms (e.g. different operating systems).

User Interface/Experience (UI/UX) Designer: Develop and maintain website wireframes, conduct usability tests on different webpages aimed at different users, coordinate with application developers on UI design items like dimensions, navigation, page routing, etc.

**Glossary (Terms, Concepts, etc.). Main source is Wikipedia:**

**Blockchain**: A growing list of records, known as blocks, which are linked using cryptography. Each block contains a cryptographic hash of the previous block, a timestamp, and transaction data. Typically used as a decentralized transaction ledger. May be public (hosted on the internet) or private (hosted locally). A goal of Blockchain is security by design, in particularly high Byzantine fault tolerance. Data structure wise is usually modeled as a Merkle tree root hash, though there are other implementations (a specially designed linked list, a specially designed database like BigchainDB, etc.).

**Cryptocurrency**: A digital asset designed to work as a medium of exchange that utilizes strong cryptography to secure financial transactions, control the creation of additional units, and verify the transfer of assets. The most famous use case of Blockchain. This project is not currency-centric, but as cryptocurrency is baked into Ethereum’s fabric, it is important to understand the concept. Bitcoin is #1 in popularity, but we only need concern ourselves with #2, Ether.

**Decentralized App (dApp):** A software application that is not managed by a centralized organization (corporations, governments, etc.). Blockchain apps are intended to be dApps, at least regarding public blockchains. The Ethereum white paper defines three types of dApps: 1. Apps that manage money 2. Apps where money is involved and 3. Other non-monetary applications (e.g. voting). As of writing, this project fits type 3: eventually it will at least overlap with types 1 and 2 if not become one of them. Note: these terms are loose.

**Decentralized Autonomous Organization (DAO)**: The (largely hypothetical) idea to replace the current model where centralized organizations (the U.S Federal Government, Google, etc.) control applications (Medicaid.gov, Gmail, etc.). The idea is that a leaderless organization forms, creates program rules at the beginning about how members can vote, use organization funds, features, etc. and then deploys their dApp on to the internet and lets it go. The concept is like how open source software organizations/licenses. How SDLC will eventually manage this product (a DAO, a private chain, a platform, etc.) is TBD.

**Drizzle**: A collection of front-end libraries to make writing front-ends for dApps easier. Part of the Truffle suite. Reference Blockchain when searching for info on it to avoid food-based results.

**Ethereum**: A decentralized platform to build Blockchain applications. It is the most popular platform for Blockchain development and is a sort of Platform as a Service (PaaS) a la Microsoft Azure. Run by the Ethereum foundation, which is essentially a DAO.

**Ethereum Virtual Machine (EVM)**: The platform-agnostic runtime environment for smart contacts based on Ethereum. It is isolated from Ethereum itself (the network). It is comparable to the Java Virtual Machine (JVM) in that it translates high-level code (e.g. Solidity) into bytecode, assembly and vice versa.

**Ganache**: Software that creates a locally-stored volatile Blockchain used for developing applications, deploying smart contracts, and running tests. There is both a GUI and command line version, we mostly use the command line version. Part of the Truffle suite. Reference Blockchain when searching for info on it to avoid food-based results.

**Gas (Ethereum):** Ethereum’s internal pricing gauge for measuring computational use for running a transaction or contract on the Ethereum main network. To run a contract or transaction on Ethereum you need to spend two things: Ether and Gas. If you go to a concert, you need money to gain admission and you need to pay for transportation to get there (e.g. gasoline for your car.). The concert is Ethereum, the money is Ether, the car is your computer, and the transportation cost (e.g. gasoline) is Gas. This model allows people to make money and pay infrastructure costs while limiting how much clout any given person has over the whole network.

**Git:** A version control system for tracking changes in computer files and coordinating work on those files among multiple people.

**GitHub**: A web-hosting service for version control using Git that expands its functionality to include things like bug tracking, task management, etc. Our main repository including source code, documentation, etc. is in a GitHub project.

**Git Bash (Git for Windows)**: A windows program that emulates the Bourne Again Shell (BASH) native to many \*NIX operating systems (Mac, Linux, etc.) and integrates with Git. There is also a GIT GUI. We use Git Bash as our windows terminal in place of cmd/PowerShell for this project to achieve better consistency with other operating systems and for general ease of use.

**MetaMask**: An Ethereum wallet that lets you connect to the Ethereum blockchain. Acts a bridge between the traditional internet and the decentralized Ethereum network. Can be installed as a browser plug in (Google Chrome). There are alternatives like Mist. We intend to connect this to MEW.

**MyEtherWallet** **(MEW):** A client-side interface for generating Ethereum wallets. We intend to connect this to MetaMask.

**npm**: A package manager for JavaScript programming. Comes with Node.js. Technically not an acronym hence why it is written in lowercase, though you can mentally think of it as “Node Package Manager”. Contains a command line client as well as an online database of public and private (premium) packages called the npm registry. We use npm to manage the dependencies inherent in our project and to work with Node.js.

**Node.js**: A server-side JavaScript run-time environment. Works with npm and the Truffle suite to compile and run our application.

**Proof of Work/Stake/etc**.: The main algorithm types cryptocurrency blockchain networks utilized to achieve distributed consensus (e.g. majority rule). Ethereum used to use Proof of Work and is moving (or has moved to) Proof of Stake.

**React (React.js):** A JavaScript library and framework for building use interfaces. Maintained by Facebook and a community of independent developers and companies. Chosen since it works well with Blockchain development, particularly the Truffle suite and Web3.js. It along with Angular.js and Vue.js are very popular JavaScript frameworks.

**Remix**: A browser specific Solidity IDE and Runtime Environment. The official environment to deploy smart contracts to the main Ethereum network as well as various test networks (including homemade networks). Will eventually use to test and deploy this project, right now we mainly use it for unit testing individual smart contracts.

**Smart Contract:** In simplest terms, the code that is stored, verified, and executed on a blockchain. Ethereum has a smart contract framework, protocol, and dedicated language. In IBM/Hyperledger parlance this is called “Chaincode” (not to be confused with “Chain code”). They are called “Smart Contracts” because they are a “smart” (secure, automated) method of enforcing the terms of a contract in the legal sense.

**Solidity:** A [smart] contract-oriented programming language. Solidity is the language we write smart contracts in. It is native to Ethereum but also targets some other platforms. It is influenced by languages like Java, Python, and JavaScript, and is the successor to a language called serpent. Smart contracts do not have to be written in solidity, it just makes sense to do so. Note that since solidity support is inconsistent (being a domain specific language) we internally copy and convert solidity files (.sol) into other formats such as JSON and regular JS files.

**Swarm:** A decentralized service that provides APIs to upload and download content to the cloud without web servers. Could potentially provide the heavier data storage, ETL, etc. capabilities that Blockchain itself does not excel at. May be implemented in the future, not currently pursuing.

**Team Foundation Server (TFS, recently renamed Azure DevOps Server)**: A version control system and hosting platform like Git and GitHub. SDLC uses TFS to host projects. Due to some of its constraints as well as strengths, we use TFS as a backup repository and for task management.

**Truffle**: The Truffle framework’s development environment, testing framework, and asset pipeline. Part of the truffle suite. Reference Blockchain when searching for info on it to avoid food-based results. Works with Node.js and npm to compile and run our application.

**Truffle Framework**: The overarching ecosystem behind the Truffle Suite. Compare with Microsoft’s .NET framework or Java frameworks like Hibernate.

**Truffle Suite:** A collection of software used to build dApps targeting Ethereum. Includes Truffle, Drizzle, and Ganache.

**Web3.js**: The Ethereum JavaScript API. Contain specific functionality for the Ethereum ecosystem. Used to connect JavaScript and Ethereum.

**Whisper:** A messaging protocol for DApps to communicate with each other. Think like RabbitMQ but for dApps. May be implemented in the future, not currently pursuing.