## **Smarta Tools Research**

## 

## **Final Tech-Stack**

* Core → HTML, CSS, JS
* Front end → React + some framework (material-ui, material kit, etc.)
* Backend → Nodejs
* Blockchain → Ethereum
* Database → MySQL
* Server → AWS
* Testing → Selenium

## **Suggestions**

* We should try to utilize existing front-end frameworks rather than pure HTML, CSS, JS for our front end. Robust frameworks allows us to:
  + Produce a shared aesthetic among all sections of the app
  + Streamlines development process by providing already configured components, to edit as we please using HTML/CSS
  + Keep up to date on new releases
  + Harness extensive documentation for troubleshooting and development
* We need to iron out the role of blockchain in our current app so we can build accordingly

## **Notes**

* Ethereum and React are commonly paired together for web development
* Can use reactnodeGUI to convert our web app into a desktop app
* Research suggests the application we are building will require traditional development tech stacks, with Ethereum added as the main back end for security sensitive operations

## **Criteria**

* Extensibility
  + Core of the product we are building is blockchain
  + Must be able to extend desktop application to support blockchain features
* Open source
  + Huge developer community to support needs
  + Evolves over time through open source contributions
  + Free to use! Fantastic for building a highly effective MVP
  + Great for prototyping
* Aesthetics
  + We’re presenting an MVP for funding and marketing - it has to look beautiful!
* Popularity
  + Popular tools/frameworks became so due to efficiency + usability
  + Relevant tools/frameworks are market relevant → can hire other engineers specializing in these tools

## **Desktop UI**

**Electron**

* Allows development of desktop GUI apps using web technologies
* The Electron framework lets you write cross-platform desktop applications using JavaScript, HTML and CSS
* Can use any web app framework and it will work with electron
* Resources:
  + [What Is Electron and Why Should We Use it?](https://dzone.com/articles/what-is-electron-amp-why-should-we-use-it)
  + [Learn Electron in Less than 60 Minutes - Free Beginner's Course](https://www.youtube.com/watch?v=2RxHQoiDctI)
  + [What Framework you are using to style your Electron app](https://discuss.atom.io/t/what-framework-you-are-using-to-style-your-electron-app/60199)

## **Front End**

**React**

* Created by Facebook → very popular front end framework
* Huge developer community → robust UI component libraries/frameworks
* Fantastic developer tools, plenty of documentation
* Faster rendering → speed
* Can be paired with React Node GUI to build a desktop application!
* Resources:
  + [Multi Step Form With React & Material UI](https://www.youtube.com/watch?v=zT62eVxShsY&t=321s)
  + [React JS Crash Course](https://www.youtube.com/watch?v=sBws8MSXN7A)

## **Back End**

**Nodejs → Web Dev**

* Runtime environment for executing JS code outside of a browser
* Use Node to build back end services (API)
* Pros:
  + Good for building scalable, data-intensive, real time apps
  + Very fast - built on Google Chromes V8 JS engine and all API’s are asynchronous (ie. non-blocking)
  + Great for prototyping and agile development
  + Node uses JS for the back end too → don’t need a separate language for front end and back end
  + Contains Node Package Manager (NPM) → publicly available, reusable components that can easily be installed and its version/dependencies maintained
* Resources:
  + [What is Node js?](https://www.youtube.com/watch?v=uVwtVBpw7RQ)
  + [Connect to the Ethereum Blockchain with Node.js and Vonage](https://www.nexmo.com/blog/2020/04/24/connect-to-the-ethereum-blockchain-with-node-js-and-vonage-dr) → potential way of connecting nodejs to the Ethereum blockchain
  + [Ethereum Tutorial: Sending Transaction via NodeJS Backend](https://medium.com/coinmonks/ethereum-tutorial-sending-transaction-via-nodejs-backend-7b623b885707) → Sending transactions on Ethereum using Node.js

**Ethereum → Blockchain**

* Used for making decentralized applications (dapp) → blockchain is used as a backend for programming logic and storage
* Main language: Solidity
  + High-level language for implementing smart contracts
  + What are smart contracts?
    - Where all the business logic of the application lives
    - Similar to microservices on the web
    - In charge of reading and writing data to blockchain as well as executing business logic
  + Influenced by C++, Python and JS + designed to target Ethereum VM
  + <https://solidity.readthedocs.io/en/v0.4.24/>
* Affiliated technologies:
  + Remix → Ethereum IDE
  + Solidity
  + VS Code Extension
* Resources:
  + [Ethereum: Building Blockchain Decentralized Apps (DApps) - LinkedIn Learning](https://www.linkedin.com/learning/ethereum-building-blockchain-decentralized-apps-dapps/welcome?u=55034593)
  + [Blockchain Basics - LinkedIn Learning](https://www.linkedin.com/learning/blockchain-basics/welcome-and-introduction?u=55034593)

## **Database**

[SQLite vs MySQL vs PostgreSQL: A Comparison Of Relational Database Management Systems](https://www.digitalocean.com/community/tutorials/sqlite-vs-mysql-vs-postgresql-a-comparison-of-relational-database-management-systems)

**SQLite**

* Supported data types:
  + Null
  + integer → signed ints 1,2,3,4,6,8 bytes depending on magnitude
  + Real → floating point values
  + Text → strings stored in DB encoding (UTF-8,16BE,16LE)
  + Blob
* Pros:
  + Small footprint
    - Fully self contained → no need to install external dependencies on system for SQLite to work
    - <600 KiB of space
  + User friendly
    - Ready to use right out of the box
    - Serverless process
  + Portable → store entire database in a single file
  + Great for testing
* Cons:
  + Limited concurrency
    - Only one process can make changes at a time
    - Multiple processes can access and query a SQLite DB
    - Not as good as MySQL or PostgreSQL
  + Bad security → server provides some security, this is serverless
* Use:
  + Testing
* Don’t use:
  + Working with lots of data
  + High write volumes
    - Only one write application at a time
  + Network access applications

**MySQL**

* Supported data types:
  + Numeric types
  + Data and time types
  + String types
* Pros:
  + Popularity and ease of use
    - Plenty of resources to learn + professionals with experience
  + Security
    - Supports user management and access privileges
    - Provides DB installation password protection
    - Removes anonymous accounts
  + Speed
    - Fastest DB solution (PostgreSQL comes close)
  + Replication
    - Shares info across two or more hosts to improve reliability, availability and fault tolerance
* Cons:
  + Designed for speed and ease of use → cost of certain functional limitations
  + Dual licensed → free + paid licenses; only some features available under paid
* Use:
  + Distributed operations like primary-secondary or primary-primary architectures (read more into this)
  + Website and web apps → powers many websites and apps across the net. This is in large part, because it’s easy to install and set up MySQL along with speed and scalability
  + Expected future growth → helps facilitate horizontal scaling. Easy to upgrade to a commercial MySQL product (like MySQL Cluster)
* Don’t Use:
  + When SQL compliance is necessary → MySQL does not fully adhere to SQL standard
  + Concurrency and large data volumes → performs well with read-heavy operations, but not concurrent read-writes

**PostgreSQL**

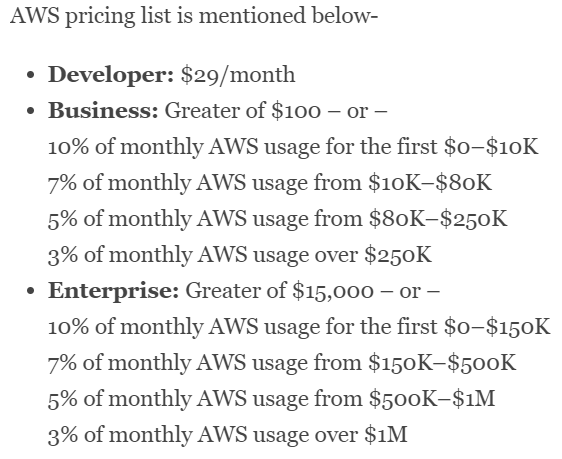
* Plethora of supported data types: numeric, string, and date and time data types like MySQL. In addition, it supports data types for geometric shapes, network addresses, bit strings, text searches, and JSON entries, as well as several idiosyncratic data types.
* Pros:
  + SQL compliance
    - Closely adheres to SQL standards → supports 160/179 features
  + Open-source
    - Source code developed by a large community
  + Extensible
* Cons:
  + Memory performance → forks a new process for every new client connection, which is ~10 MB of memory. This can add up quickly for DB with lots of connections
  + Apparently MySQL is more popular → fewer third-party tools and experts that can help manage a PostgreSQL DB
* Use:
  + When data integrity is important → fully ACID compliant since 2001 and implements multoversion currency control
  + Integration with other tools (easy migration)
    - Compatible with a wide array of programming languages and platforms
  + Complex operations
    - can support query plans which leverage multiple CPUs in order to answer queries with greater speed
    - Great for complex operations like online transaction processing and data warehousing
* Don’t Use:
  + Speed is imperative
    - Designed with extensibility and compatibility in mind, so it is slower
  + Complicated setup
  + Complex replication

## **Server**

**Azure**

* Pros:
  + App friendly
    - Works well with other MS products
  + High availability
    - Only ~4.2 hours of downtime over entire year (~43.2 seconds a day)
    - 99.5% of the time, Azure will be available for use
    - Same up-time for AWS and Google cloud too
  + Security
    - DADSC → detect, assess, diagnose, stabilize, and close method
    - Multi-factor authentication
    - Multiple data centers → if one of them is down for maintenance, other data centers still have your info
  + Scalability
    - Easy to upgrade for times in the month where data use is high
    - No need to purchase data packets, just upgrade when you need and remove when you don’t
  + Tool pairings
    - Pairs well with VSCode, Nodejs, JS, etc.
    - Can use any tool or framework well with this
  + AI services
    - Bots which naturally interact with consumers
    - Data analytics → faster predictions
* Cons:
  + Requires MS expertise to ensure cooperation of moving parts
    - Easy to over-provision cloud services → waste of money
    - Need to have people managing the data properly → must hire Azure experts for using Azure, monitoring server and patching
  + Security
    - Apparently weaker than AWS
  + Poor documentation
    - Not very developer friendly
  + Bill shock
    - Might be using more than you need → must make sure you manage your cloud expenses well
    - Tools to help prevent bill shock:
      * Azure advisor
      * Role based access control
  + Easy to overuse when not careful due to ease of access
    - Might end up spending more money than you thought
  + Difficult migration from other competitors

**AWS**

* Pros:
  + Ease of use
    - Well documented API’s
    - Apparently more developer friendly
    - Plenty of tutorials
  + Diverse tool set
    - Offers 70 additional services besides cloud storage + computing
    - Database, software, mobile, analytics, networking
    - Centralizes all cloud computing needs
  + Unlimited server capacity
    - Main reason why AWS is on top
    - Prevents business disruption due to virus/malware
  + Encryption and Security
    - 90% of all WordPress sites were hacked last year
  + Managed IT services
    - Different tiers available
    - They will manage all IT infrastructure based on our tier
    - Can change services needed as business scales
    - Types of services:
      * Security
      * Cloud Management
      * IT infrastructure and networking
      * Technical support
  + Flexibility and affordability
    - Allows us to load any service/software we want in a virtual ecosystem
    - Eases migration from different platforms or deployment of new ones
* Cons:
  + Bill shock
    - Might be using more than you need → must make sure you manage your cloud expenses well
  + Complicated billing
    - Might be better to go with a reseller who understands the invoice/billing → though we must make sure they don’t extort us
  + Limits resources according to region
    - As a new user, AWS prevents you from using too much resource and spending a lot of money → protective measure to prevent overflow attacks
    - Limited info on how much resources we can really use → preventative measures for security (can mitigate this however)
  + Technical support fees
    - 
  + Risk of data leakage, privacy issues, security, downtime, limited control
    - Same as any cloud platform
    - AWS is top tier though
* Resources:
  + [The 5 Benefits of AWS (And 3 Drawbacks) for Your Business – SADOS](https://sados.com/blog/aws-benefits-and-drawbacks/)

## **Questions**

1. To use web app to desktop converters, do we need the website to be live? Or can it just be a local development copy?
2. Which OS are we targeting? Windows or Mac?
   1. **Prefer to target Windows - standard OS**
3. Should we make a desktop app using traditional means (ie. C++, C#, VB for Windows) or use web tech and convert that into a desktop app?
   1. **Use web tech → has incredible UI/UX packages used by top companies in the world. Aesthetics are important for our app!**
4. Can mobile apps connect to and utilize the blockchain?
   1. **Probably can → blockchain is just on top of internet, should be fine**
5. Do we need a full blockchain network for all aspects of our application? How exactly will we be utilizing the blockchain for the property management side of the product?
   1. **Use blockchain for payments and document storage only - don’t need it for everything**
6. Can we combine a traditional tech stack with a blockchain network for security sensitive operations?
   1. **Yes**
7. Can we extend an existing full stack application into a blockchain based one?
   1. **TBD**
8. Are we purely focusing on the property management for now, then extending it to also manage smart devices/services later?
   1. **Yes**
9. Do we envision our management platform to manage any smart device from any company? Or just Smarta specific smart devices?
   1. **TBD**

## **Blockchain Research**

* What is blockchain?
  + A time-stamped series of immutable records of data that is managed by a cluster of computers not owned by any single entity → decentralized, democratic, governed by all
  + Information on blockchain is open for anyone and everyone to see
  + Really useful for eliminating third party entities taking a cut of profits during financial transactions → free, secure way to make direct payments for free!
    - Blockchain stores a record of transactions that are immutable and independently verifiable as the “block” is verified by thousands/millions of computers before being added to the “chain”
  + Numerous applications possible → relies on “encoding” a service such that a transaction can access that service and pay the author of that service directly
* What are smart contracts?
  + Defined as a computerized transaction protocol that executes the terms of a contract
  + Applicable due to distributed ledger technology
  + In blockchain context, smart contracts are stored scripts on the blockchain with a unique address
  + Basically, the contract will execute autonomously and exactly as agreed upon by all involved parties at the required time. Users’ private data is protected
* Downsides?
  + Challenging to develop
  + Efficiency needs to be improved
* Research papers of blockchain:
  + [Improving IoT Services in Smart-home Using Blockchain Smart Contract](http://cse.stfx.ca/~cybermatics/2018/Proceedings/pdfs/iThings!GreenCom!CPSCom!SmartData!Blockchain!CIT!Cybermatics2018-1Q1rrimpxFNyCxx89cHrQN/6E3nqxXUMU56R8gkNA81oX/59osLxUloHjYqOnzVIq6vA.pdf)
    - Proposes an IoT architecture in smart home environments which are based primarily on blockchain and smart home environments
    - Suggests private blockchain network, public blockchain network and a smart contract
    - Benefits of blockchain in smart home environment:
      * Transparency → all transactions are immutable and placed in the distributed ledger
      * Security → network security level is robust due to cryptographic and decentralized nature of blockchain
      * Privacy → blockchain avoids information leakage. People have control over which information they want to hand out, and which information they wish to keep private
    - Challenges applying blockchain in IoT settings:
      * High resource consumption, scalability, processing time
      * To send anything to the blockchain, new blocks must be created. Creating new blocks requires lots of computational power to solve, resulting in higher processing times
      * IoT devices don’t have high computation power and adequate storage space. As every transaction process in blockchain requires proof of work, this results in latency for device communication
  + [(PDF) Blockchain for Smart Homes: Review of Current Trends and Research Challenges](https://www.researchgate.net/publication/338081465_Blockchain_for_Smart_Homes_Review_of_Current_Trends_and_Research_Challenges)