Research Report Team 13 (ATOS Blockchain Team 1)

# Contents:

### Section 1: Blockchain

* 1. An Overview of Blockchain
  2. A comparison of the blockchain platforms

### Section 2: Ethereum

2.1 What are smart contracts

2.2 The Solidity language

Section 3: Technical Design Chocies

3.1 Web app vs Native app

3.2 Frameworks to use (front end)

3.3 Frameworks to use (back end)

3.4 System Architecture Diagram

3.5 Development Methodology

Section 4: Development Environment

4.1 Ethereum Set up

4.2 IDEs, Libraries

4.3 Configuration Scripts

Section 5: Testing and Continuous Integration

5.1 Test Driven Development

5.2 VCS and Deployment Process

5.3 Unit testing, Continuous Integration, User Acceptance Testing

Section 6: Construction contracts

6.1 What is a contract

6.2 Purpose of building contracts

6.3 Characteristics of a standard form

6.4 JCT Contracts

6.5 Roles

6.6 Issue of Certificates

6.7 Disputes

Section 7: Project Management Setup

7.1 Communication and Organisation

7.2 Gantt Chart and Outline of Plan

# Section 1: Blockchain

## 1.1 An overview of blockchain

Blockchain is the technology that our project is focused on, and it is a relatively new one. Bitcoin was the first technology to really make use of it, and is currently the most popular blockchain based platform. [1]

Bitcoin is a trustless system for pseudo-anonymous transactions of currency (the bitcoins themselves). The system works with public key cryptography at the heart, with the public key acting as the identifying information. The technology is entirely peer-to-peer without a central organisation to verify or arbiter, which is achieved by processing transactions in ‘blocks’. Each block contains a record of current transactions, which are signed by the entity that currently owns the coin and not the payee. When a large group of them is created, they along with the timestamp and hash of the previous block are hashed again to prove that the block was created directly after the previous one (to resist attack). To prevent further attack by a malicious entity, a ‘proof of work’ is required for each block, which is something that is computationally difficult to come up with, but very easy to verify. [1] Each block has something called a ‘nonce’ which is a 32 bit field that has no intrinsic meaning, but is chosen such that the rest of the block together with the nonce have a number of leading 0s (this can be set variably) when passed through the SHA-256 hash. [2] By adhering to the simple principle of the longest chain being the correct one, an attacker would need to generate correct blocks faster than the entirety of the ecosystem in order to succeed.

The consequences of such a technology is that ‘trust’ is not required from a central entity, as it relies on public key cryptography at the heart, pseudo-anonymity can be retained (transactions are only identifiable by a user’s public key) and also the system is secure from attackers as long as each cryptographic mechanism remains secure and the attackers do not gain a majority of the computational resources on the blockchain (a 51% attack). [3]

## 1.2 A comparison of the blockchain platforms

Blockchain is a system revolving around currency transactions, however other blockchain based systems have cropped up with different focuses in mind.

The other major player is Ethereum. Ethereum at its core still the same core blockchain ideas, but they are implemented differently and Ethereum is focused on using the blockchain to conduct computations, using a virtual machine (VM), with each ‘transaction’ corresponding to some computation on the VM. However, each computation is adjusted for relative complexity in terms of its price, and this is paid for with ‘gas’ which is given to the miner. Valid computations are carried out on all machines on the blockchain, so this means that each computation once in the blockchain can be verified as correct and error free (as the majority of computers have agreed upon it). Ethereum blocks are generated much faster than Bitcoin’s 10 minute average, and is done about every 12 seconds. Ethereum is now a maturing platform, and the computation model is based on ‘smart contracts’, which are ‘Turing Complete’. [4]

Since Bitcoin is not designed to do anything more complex than mere currency transactions, adapting it for general purpose computing would be highly inefficient, especially when coupled with the high block mining time which is useful for security, but prohibitive for more complex actions. In contrast, Ethereum allows execution of (nearly) arbitrarily large computations so long as the contract holder has enough ‘gas’ to complete it. After Bitcoin, Ethereum is the most popular blockchain based technology. [5] Blockchains are much more secure when they have a lot of users, and interest and development in Ethereum is high. This will mean that support will continue for a long time, but also documentation will be useful and plentiful. For these reasons, we have decided to go ahead with using Ethereum as the core of our application, as it will provide a secure but highly versatile technology to build our project on.

# Section 2: Ethereum

## 2.1 What are smart contracts

A smart contract is a term that can refer to either all computation code on the blockchain, or alternatively a type of legal contract that can automatically perform some action. The latter is something that uses programming code to define some of the rules and consequences of specific actions in certain contexts, and can include some automatically executing code. [6][7] E.g. the Ethereum ‘DAO’ (Digital Autonomous Organisation) is an idea to create an organisation that exists on the blockchain, whose actions are determined by smart contract code to perform certain functions, and an autonomous voting system. [8]

## 2.2 The Solidity Language

Solidity is one of the most popular languages used to write Ethereum VM (EVM) code. It is a high level languages, that is then compiled into a low level language for the VM to run directly. The language is statically typed with support for inheritance, libraries and user-defined data types so although it is not as versatile as other languages, it is still quite capable.The syntax is very similar to JavaScript, which we will also be using in our project so it will not be too difficult to learn. Furthermore, it has a great deal of interoperability with JavaScript through resources such as the web3.js API. [9]

# Section 3: Technical Design Choices

## 3.1 Web app vs Native app

The first consideration we must make is whether to use a web application or a native application.

Native applications allow for lower level hardware access, so can potentially support a richer feature set as compared to web applications. They also do not have to have a requirement of being connected to the internet, which is a minor consideration in their favour. It is easier to have granular control, in order to achieve the results that are necessary.

Web based applications do have the requirement of needing internet access, and although they do not have the same granular control as a native application, there are many mature web frameworks that make web development easier and more powerful.

Since the application pertains to potentially large sets of contracts, it is unlikely that any large percentage of these would be created on a mobile device, as this would be a cumbersome process. This reduces the need for native applications, which can make more difference on mobile platforms. Furthermore, since the application is required to be connected to the internet anyway due to it interacting with the Ethereum blockchain, the online requirement of a web app would not be any hindrance at all.

Another point in favour of web-applications is the web3.js API for Ethereum, which is designed to make interacting with the blockchain much easier, and is perfectly suited for use as part of a web app. Web applications are generally easier to create quick prototype versions of, and tools such as truffle help create template skeleton projects to make this process even faster. It is for these reasons, that we have decided to create a web based application.

## 3.2 Frameworks to use (frontend)

Since it is a web application, the use of HTML and CSS are mandatory. JavaScript is also a language that is paired with them to add more interactive functionality, and these 3 languages form the backbone of almost every single website in existence.

The bootstrap framework is useful for the creation of quick prototypes, and the grid layout is useful for creating responsive websites that can scale based on screen resolution and pixel density, without having to explicitly program much code for this functionality. There is a plethora of pre-built components to work from, which will let the focus of development be on the algorithms. Using these is also useful from a testing perspective, as it is a very popular and well tested framework. This means that each component does not need to be unit tested, we only need to worry about integration testing the composition of them rather than worrying about each individual piece. [10]

AngularJS is a JavaScript based framework that is very effective at speeding up the process of front end work, as well as providing an excellent structure to work from. It reduces the DOM (document object model) code needed to negligible amounts, and it enforces the MVC design pattern (model view controller) which is both efficient and maintainable while creating software. The data binding feature also increases the swiftness of writing interactive code by reducing boilerplate code for simple actions, automatically updating the view every time the model is altered. [11]

D3.js is a JavaScript library that is used primarily for data visualisation purposes. This will be useful for helping visualise contract code into a format that is more easily interpreted. Since it is a JS library, this will integrate with all of the other frameworks very smoothly, and it is a highly popular framework so has plenty of guides, resources and pre-made components to learn from and partially reuse. [12]

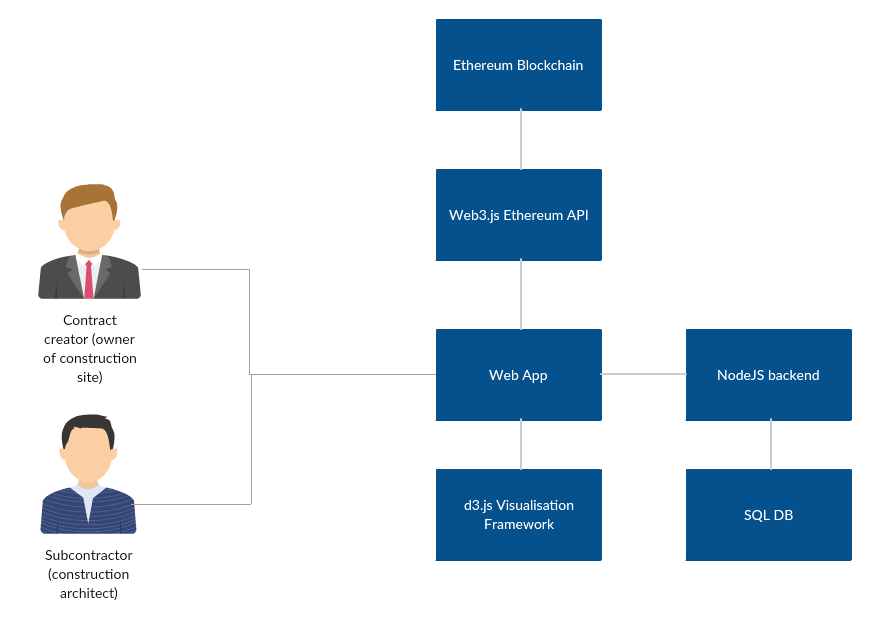
## 3.3 Frameworks to use (backend)

NodeJS is a very popular and well documented language to use for the backend. It is a JavaScript based language, which is convenient as this will require minimal further study to start developing. The package manager is one of the largest and best repositories for finding useful plugins, and handles problems such as project dependencies, updates, and conflicts thus greatly reducing the barrier to development rather than setup. [13]

Web3.js is an Ethereum compatible JavaScript API, that contains many highly useful functions for dealing with the Ethereum blockchain. This provides a ‘web3 object’ that can be used in JavaScript code to interact with the blockchain, such as contract execution, and reading the state of the blockchain. [14]

The application will need some persistent data storage for things such as basic account implementation/information, and perhaps storing private keys for blockchain transactions, as well as the addresses of blockchain contracts. An SQL based solution would be best, as it can store this type of information very easily. It is also widely supported and documented, meaning that it will be very easy to setup and maintain an SQL solution. Furthermore, all 3 of us have some experience using SQL from the recent module, so we will not have to expend too much effort learning a completely new technology.

## 3.4 System Architecture Diagram



## 3.5 Development Methodology

There are multiple potential approaches to development strategies. However, an Agile based approach would probably fit this project best. The plan for this project is to get a minimum working product finished as soon as possible, in order to get client feedback in further iterations. This will allow us to develop a product that is close to what they want, and is perfect for creating this proof of concept, as some features can be omitted initially whereas with Big Design up Front, this would not be possible and it would be very difficult and time consuming to make multiple iterations.

We will have sprints of 2-3 weeks long, so that each successive deadline encourages us to finish work early and often and not fall into bad habits of having a large workload near the end. These will be managed by having a ‘Trello’ board for all of the tasks that are due in that sprint, with each one being assigned to each person and assigned a deadline. This will help us to keep track of all of everything that goes on.

# Section 4: Development Environment

## 4.1 Ethereum Set up

There are many alternative possible solutions to setting up a development environment in Ethereum. An impractical and expensive one is to directly use the live Ethereum blockchain to conduct experiments, which would also take a non insignificant amount of time due to each block taking a small amount of time to mine. A better alternative would be to use the ‘testnet’, where transactions occur in 1 second, and the file size is much smaller. However the file size is still not insignificant.

Geth is a tool that allows a local blockchain network to be set up with accounts that are set up to already have funds, and only miners you permit can access the network. [15] This has the advantage of near instantaneous transactions, a tiny file size, and free execution of code while developing. Another similar tool is testRPC which does much the same thing as Geth, but is specifically designed for development and testing purposes. [16]

## 4.2 IDEs, libraries

The popular open source ‘Atom’ code editor has a ‘Solidity’ package, which has syntax highlighting for Solidity on top of its standard features. These features include directory browsing, and multiple panes of code in one window but also very close integration with Git and Github VCS (as Atom is developed by Github). [17]

Truffle is a very useful tool to quick start the development process for Ethereum web applications. On initialisation, it sets up a few test contracts, and creates a skeleton project to work on for contracts as well as a basic demo web application. It supports built in smart contract compilation, linking and deployment, and has a scriptable framework for deployment and migration of these. It supports automatic contract testing through Mocha and Chai frameworks for JavaScript. Truffle also has web server and web3 functionality built in, which is very useful for quick testing and developing more efficiently. [18]

To create the web application, JetBrains’ Webstorm IDE provides powerful functionality such as VCS integration, code autocompletion, and support for tools such as Grunt and npm directly through the IDE. It also provides a method of running a web server directly, which is far more convenient for quick testing and development than performing a manual start up. A debugger is included for testing client side and node.js applications, and it integrates with some unit test runners. There are several useful features for refactoring code quickly which helps maintain code quality. [19]

## 4.3 Configuration scripts

Truffle can automatically watch the filesystem for changes, recompile and redeploy and rebuild the application automatically, and serve the project to a localhost port for testing. This is a huge time saver not only in development, but also in production code, as any detected changes will be live very shortly (assuming that they have been thoroughly tested beforehand). [18]

Npm (node package manager) is a package manager that accompanies NodeJS, which supports a vast array of useful plugins to save time and effort to not reimplement functionality. It also handles problems such as updates, dependencies and conflicts which greatly frees up time to merely use them, and not have to worry about maintaining the careful balance and separation of lots of plugins. [13]

Task runners such as grunt [20] and gulp [21] help to automate common tasks that may have to be run at every update. This can include running code analysis, code compilation, stylesheet minification and other repetitive tasks. By fully automating as much as possible (especially repetitive tasks that are likely to always be the same), a lot of time can be saved and we no longer have to worry about tedious tasks or forgetting an important stage thus reducing errors.

# Section 5: Testing and Continuous Integration

## 5.1 Test Driven Development

Test-driven development (TDD) is a technique for developing software by focusing on testing first. By writing the tests first and the software later, the unit test coverage can be made more thorough, and the act of writing tests makes developers consider decoupling units more thoroughly (as they are easier to test this way), which is a core principle in good OOP. Another advantage of TDD is that once these tests are written, they can be automatically run again each time the code is changed and will provide full coverage of the system when done correctly. [22]

## 5.2 VCS and Deployment Process

Git is one of the most popular version control systems (VCSs) out there, and we have are already familiar with it so there will be little/no learning curve associated with it. We have set up 3 private repositories already, one for the submission website, one for experiments, and one for production/development code.

The development workflow that we will try to achieve is as follows:

1. Write unit tests, and develop new code in accordance with TDD
2. Push code to development branch of Github
3. Run integration tests on this, and if they pass:

Push code to master branch as needed, and run any tasks such as minification that may be useful.

Or if they fail:

Investigate and fix issue before pushing to master

The use of file watchers (such as truffle) can automatically configure and re-deploy the web application when a change in files (such as an update from Git) is detected.

## 5.3 Unit testing, Continuous Integration, User Acceptance Testing

Unit testing is the process of testing each small section of the code for expected behaviour in a vacuum, and these will be run automatically whenever source code is updated to ensure that the core code is still functioning as expected. Since we are going to be developing using a test driven approach, a strong testing framework with support for assertions will be practically mandatory.  MochaJS (for NodeJS and other JavaScript) is the main framework that we wish to use alongside ‘Chai’ (which adds support for assertions). Since the vast majority of our code will be JavaScript based (AngularJS and JS for front end, NodeJS for backend, web3.js, d3.js and even Solidity is very similar to JS code), this will be a perfect testing library for our development purposes. It is widely supported and popular, so learning and using it should not be a problem. [23] [24]

NodeJS supports scripting to create an automatic deployment script from Github. This would be able to accept a webhook from Github every time that an update occurs, to update remote servers when code is changed, and run bash scripts/start other processes to carry out integration tests. The successful completion of these would signal that our code is ready, and we would be able to move onto automatic deployment, which can be achieved with Truffle, gulp/grunt task runners, or even more nodeJS scripts. These integration tests will ensure that all of the code works together as required, and that there are no strange edge cases that occur when different sections are integrated together.

The use of Agile development processes, means that we have short development cycles. This means that we can get user feedback on each implemented feature, so as well as the regular testing on the code with unit and integration testing, we can take client feedback into account. This is perfect for use with agile, as it allows proper prioritisation of tasks in the next sprint, and quick changes to ensure that the product developed is the one that is actually desired.

# Section 6: Construction contracts

## 6.1 What is a contract?

A contract is an agreement of additional rights and duties (other than the law) between 2 or more parties involving an exchange of services/goods for money. So where there are agreed rights and duties on both sides we call it a contract.

Details of the contract are usually written on paper and signed by both parties to indicate agreement upon terms and conditions. The contracts are legally binding, which means that once contract is agreed on, neither party can change their minds without serious consequences.

If a party does something the contract doesn’t allow or fails to do something the contract requires, this is referred to as a ‘breach’ of contract. The person who is not in breach is referred to as an injured or innocent party. They are usually entitled to payment from the person in breach to make up for the breach (also called damages). The amount paid is normally calculated to put the injured party back in the same position as if the breach had not occurred. Payments to the injured party usually happen in court.

If the breach of contract is serious it is called a repudiation. This means that the breach is so serious that one party wants nothing more to do with the contract. For example, a contractor may walk of the site never to return halfway through a project, or an employer telling the contractor he is not going to be paid any money.

Faced with repudiation, the injured party has the choice of either accepting the repudiation and seeking damages through court, or saying the contract is still in place and carrying on with the project (also called affirmation) but still entitled to seek damages after affirmation.

In order for there to be a contract 3 things must be present:  An agreement where by both parties accept the exchanges of goods/services/money. An intention to create legal relations and something given by both parties. [25]

## 6.2 Purpose of building contracts

The purpose of the contract is to see the building come to life. The contracts also set out the rights and duties of the parties involved: what each may do and what each must do. It also sets out procedures for specific cases e.g. what the architecture should do if the contractor is not getting things done on time.

In the standard building contract (SBC), there will be an employer who employs the contractor who carries out the construction work (typically hiring sub-contractors to get specifics of the building done e.g. lights, electricity). There will also be a contractor administrator who is usually the architect who does things allocated in the contract, and a quantity surveyor who is concerned with valuing the work (checking to see if work is built to standard and done) [25]

## 6.3 Characteristics of a standard form

Bespoke contracts are designed to precisely meet the needs of the purchaser. Ideally this would be perfect for everyone however they are much more expensive typically costing thousands of pounds to create as opposed to using a standard contract costing £40-50. Making these contracts take a lot of time to allow for requirements to be thoroughly investigated and all terms carefully drafted to ensure all cases are covered.

Standard forms on the other hand are relatively inexpensive and requirements are based on what the majority of buildings require, which is the main disadvantage. JCT takes into account the fact that one form does not suit every case and so have produced various forms for particular categories such as Design and build, Prime Cost and Traditional.

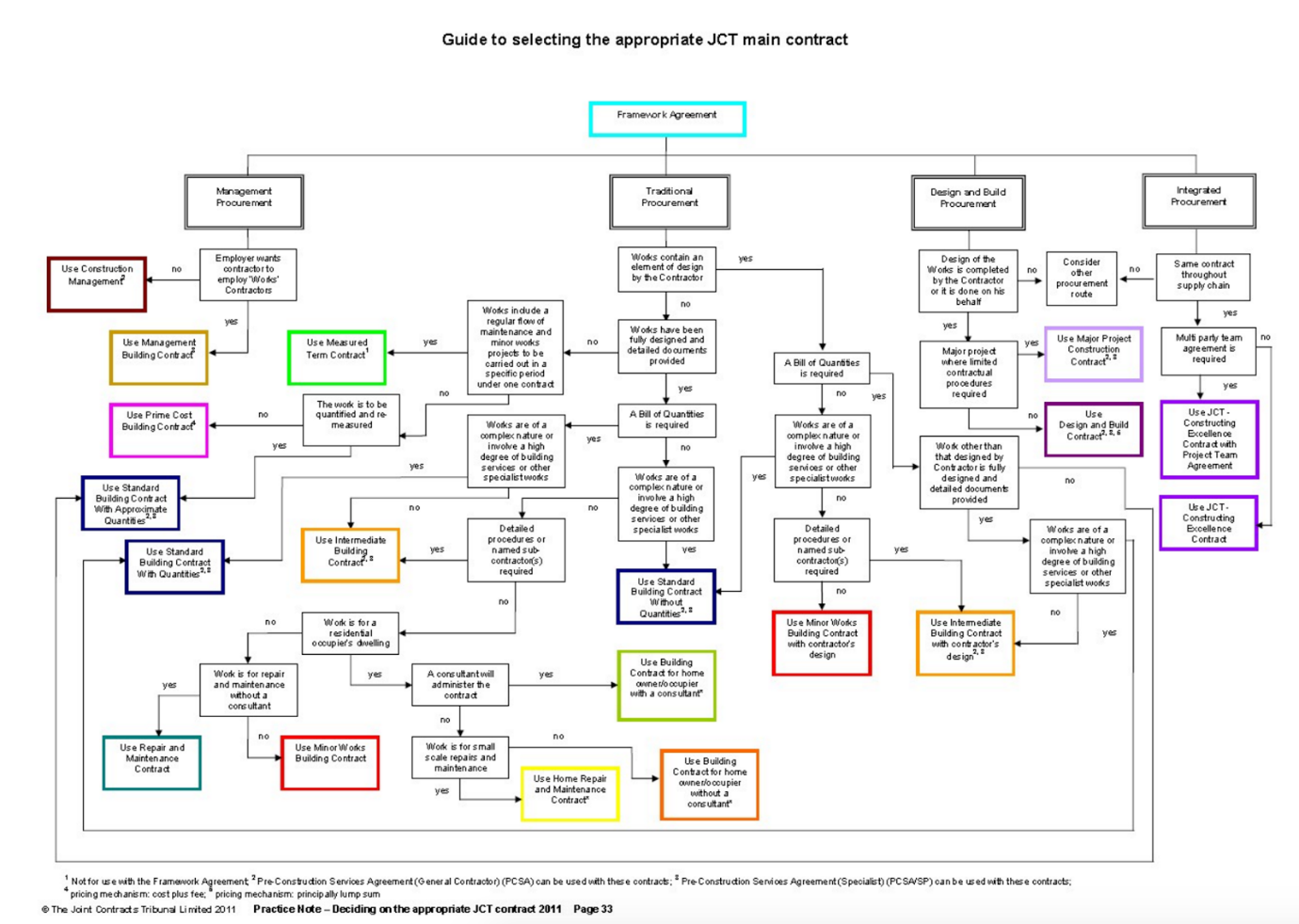
A big advantage of standard forms is that they are drafted by professionals and the forms are updated over the years to align with law and decisions made by courts, resulting in a refined form with few mistakes. Furthermore, architects and contractors become familiar with the forms and so deal the process much faster.

Disadvantages of the forms include having too many variations thus architects and quantity surveyors can’t really get to grips with the differences. Architects and project managers commonly use forms that may not fit the situation. Thus it may not be the most effective and efficient process. An analogy would be going from London to Birmingham in a tank as opposed to a car. The more complex the building the more unlikely that the standard form will be correct.

Hybrid forms may be used in this case and are formed in order to avoid paying large fees; this involves using the standard form with carefully edited parts to meet requirements. It is very risky to do as such, because it is difficult to work around complicated rules that may break the law. Thus they are hybrid forms are usually created by specialists. [25]

## 6.4 JCT Contracts

The most commonly used standard contracts in construction are the JCT (Joint Contracts Tribunal) contracts. The JCT contracts are made up of 8 main contracts compiled into a single standard contract, these include: Lump Sum, Measurement, Cost reimbursement, Design and Build, Management Contract, Partnering Contract, Pre-Construction Services Agreement and Consultancy Agreement. There are also variants of the 8 main contracts to suit specific needs, so JCT has provided a flowchart to help decide which ones to use.



[26]

## 6.5 Roles

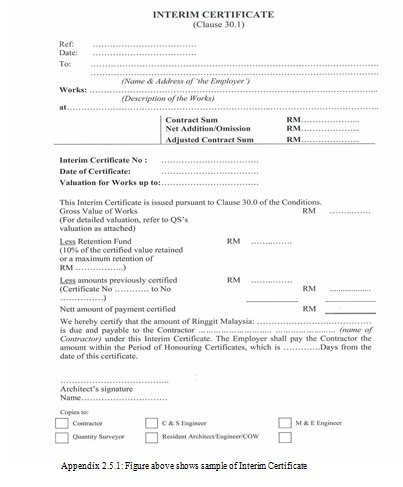
An architect's role is to do exactly as stated by the contract and no more. The contract usually includes a lot of detail about the architect's obligations and what the architect has the power to do but not necessarily do. [27]

The subcontractor typically hired by contractors to work on specific areas of the building, this typically happens as buildings become complicated and contractors don’t have the required skills to construct. Elements of works include: piling, roofing, steel work, plumbing, electrical services etc. If the work done by electrician for example is poor, the electrician is liable to the contractor but not the employer. The employer will blame contractor if electricians’ work is poor. [28]

The quantity surveyor (a consultant) responsible for creating the contract sum which involves figuring out the costs of work to be done and materials needed. This is done before contracts are signed. In the case of a large project architects will hire quantity surveyors to perform valuations before architects send out certificates. The valuations are usually done 7 days before the due date of the interim payment to avoid/reduce additional costs creeping up because of extra work done by contractors. [29]

## 6.6 Issue of certificates

The architect is responsible for the issuing of certificates; they are the most important document that an architect will give out. There are a number of certificates and once issued cannot be withdrawn. These include Non-completion certificate, practical completion certificate, section completion, certificate of making good, interim certificate, final certificate, certificate following employer’s termination of the contractor’s employment, certificates releasing insurance money.

Interim certificates confirm that the contractor has done work and payment to be made. Architect may hire quantity surveyor to check for valuation before issuing certificate. Here is an example of an Interim Certificate:

[30]

It is a requirement that the certificates are sent to both the employer and contractor at the same time by the architect via post e.g. using the same stamp class. Usually, the architect uses special delivery of the certificates to ensure the certificates are not lost and are sent to recipient. [31]

## 6.7 Disputes

Disputes happen when there is breach of contract. For example, payments are still due, completion date need to be extended, health and safety issues. There are levels to resolve the disputes, with each one getting more serious and costly. First one is mediation, where parties gather and come to agreement. Second is adjudication where 3rd party tries to resolve issue within 30 days. Third is when both parties go to private court so as to avoid the public to settle with a judge. Disputes usually happen 2-3 times a month.

# Section 7: Project Management Set up

## 7.1 Communication and Organisation

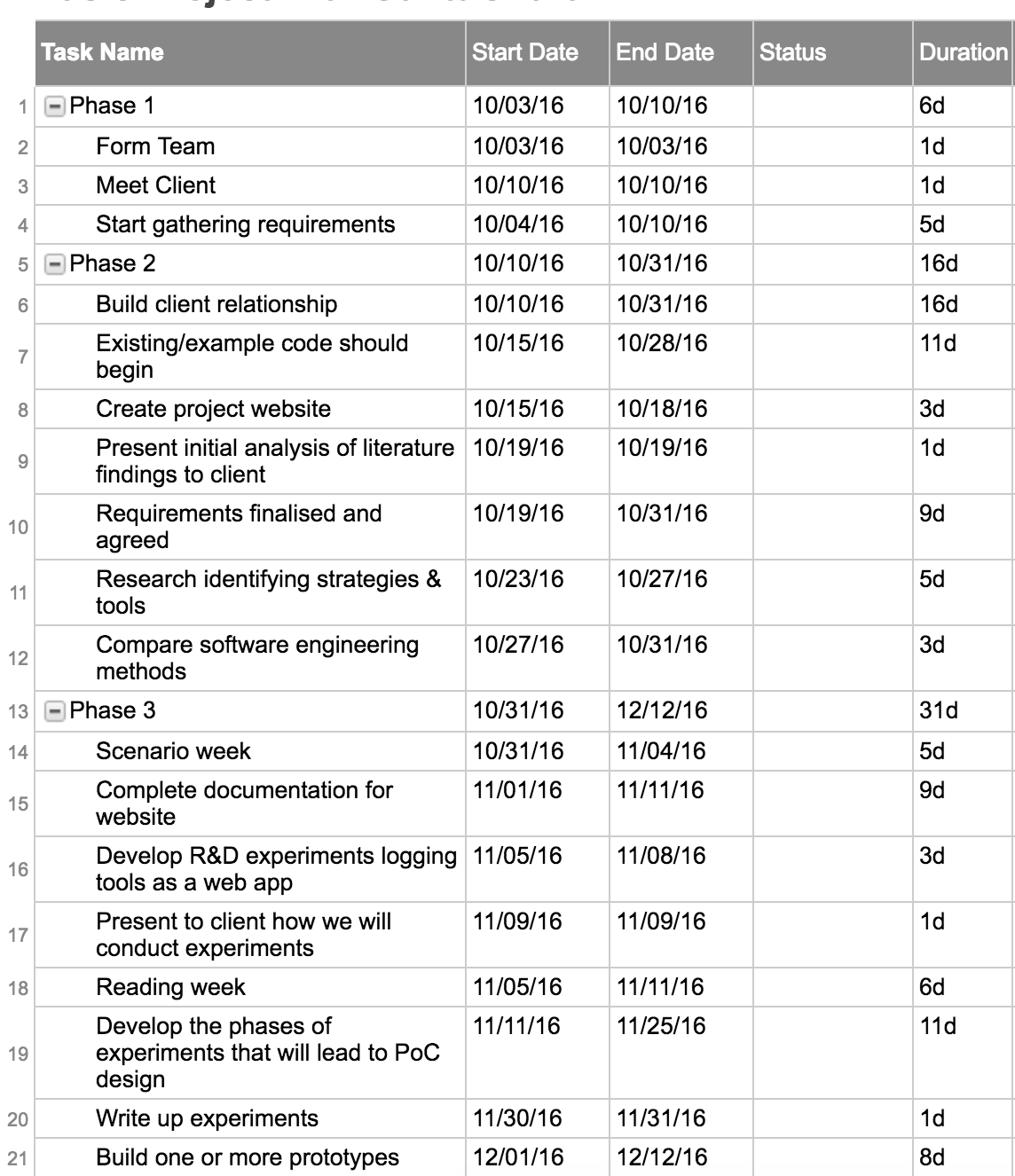
In order to effectively work as a team, communication is critical. We have set up multiple avenues of communication, for different purposes. In order to discuss more complex issues, and communicate with our client we have been using Skype, as meetings in this format are highly productive when directed at one topic, rather than having to type out messages.

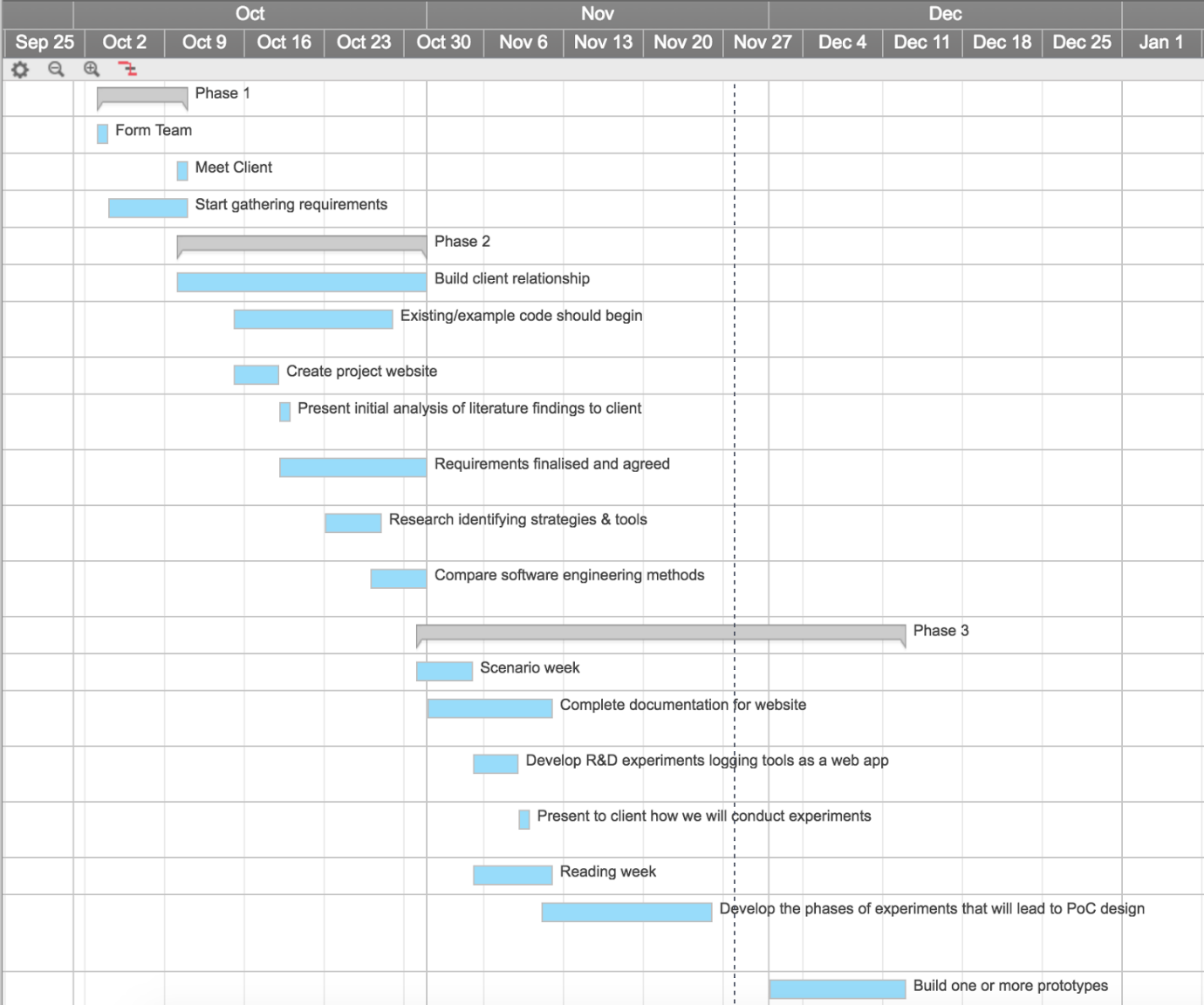
There are multiple teams working on blockchain based projects, and to communicate between ourselves we have set up a Slack group to quickly share messages, and useful file between us. Our team favours whatsapp as the primary communication method, as it is easy to set up and notifications are instantaneous allowing for quick response times.

Trello is the tool that we are using to organise roles and tasks between us, as it allows for easy creation and management of task lists, with member assignment and date setting. These features will allow us to handle Agile sprints too, with tasks being assigned to members for certain deadlines (sprint deadline or sooner), and completion can be marked upon them with colours making very easy to see progress.

To share software code, we will be using Git as outlined before, and most probably be using Github to store remote repositories as it is very popular and we have used it extensively before for previous projects. To share documentation and collaborate easily with each other, we are making extensive use of ‘Google Drive’, as it is very easy to share files and even work on the same files concurrently.

## 7.2 Gantt Chart and Outline of Plan





# Bibliography

1. Nakamoto S. Bitcoin: A Peer-to-Peer Electronic Cash System [Internet]. Bitcoin. 2008 [cited 26 November 2016]. Available from: <https://bitcoin.org/bitcoin.pdf>

2. Proof of work - Bitcoin Wiki [Internet]. En.bitcoin.it. [cited 26 November 2016]. Available from: <https://en.bitcoin.it/wiki/Proof_of_work>

3. Learn Cryptography - 51% Attack [Internet]. Learncryptography.com. [cited 26 November 2016]. Available from: <https://learncryptography.com/cryptocurrency/51-attack>

4. ethereum/wiki [Internet]. GitHub. 2014 [cited 26 November 2016]. Available from: <https://github.com/ethereum/wiki/wiki/White-Paper#ethereum-state-transition-function>

5. Crypto-Currency Market Capitalizations [Internet]. Coinmarketcap.com. 2016 [cited 26 November 2016]. Available from: https://coinmarketcap.com/

6. Stark J. Making Sense of Blockchain Smart Contracts - CoinDesk [Internet]. CoinDesk. 2016 [cited 27 November 2016]. Available from: <http://www.coindesk.com/making-sense-smart-contracts/>

7. Blockchain Explained - Distributed Ledgers and Blockchain Technology [Internet]. Blockchain Technologies. [cited 27 November 2016]. Available from: <http://www.blockchaintechnologies.com/blockchain-smart-contracts>

8. Create a Democracy contract in Ethereum [Internet]. Ethereum.org. 2016 [cited 27 November 2016]. Available from: <https://www.ethereum.org/dao>

9. Solidity — Solidity 0.4.7-develop documentation [Internet]. Solidity.readthedocs.io. 2016 [cited 27 November 2016]. Available from: <https://solidity.readthedocs.io/en/develop/>

10. Bootstrap · The world's most popular mobile-first and responsive front-end framework. [Internet]. Getbootstrap.com. [cited 27 November 2016]. Available from: <http://getbootstrap.com/>

11. AngularJS by Google [Internet]. AngularJS. [cited 27 November 2016]. Available from: <https://angularjs.org/>

12. Bostock M. D3.js - Data-Driven Documents [Internet]. D3js.org. 2015 [cited 27 November 2016]. Available from: <https://d3js.org/>

13. Foundation N. Node.js [Internet]. Nodejs.org. 2016 [cited 27 November 2016]. Available from: <https://nodejs.org/en/>

14. ethereum/web3.js [Internet]. GitHub. 2014 [cited 27 November 2016]. Available from: <https://github.com/ethereum/web3.js/>

15. ethereum/go-ethereum [Internet]. GitHub. 2015 [cited 27 November 2016]. Available from: <https://github.com/ethereum/go-ethereum/wiki/geth>

16. ethereumjs/testrpc [Internet]. GitHub. [cited 27 November 2016]. Available from: <https://github.com/ethereumjs/testrpc>

17. A hackable text editor for the 21st Century [Internet]. Atom. 2016 [cited 27 November 2016]. Available from: <https://atom.io/>

18. Truffle [Internet]. Truffle.readthedocs.io. 2016 [cited 27 November 2016]. Available from: <https://truffle.readthedocs.io/en/latest/>

19. WebStorm: The Smartest JavaScript IDE [Internet]. JetBrains. [cited 27 November 2016]. Available from: <https://www.jetbrains.com/webstorm/>

20. Grunt: The JavaScript Task Runner [Internet]. Gruntjs.com. [cited 27 November 2016]. Available from: <http://gruntjs.com/>

21. gulp.js - the streaming build system [Internet]. Gulpjs.com. [cited 27 November 2016]. Available from: http://gulpjs.com/

22. Palermo J. Guidelines for Test-Driven Development [Internet]. Msdn.microsoft.com. 2006 [cited 28 November 2016]. Available from: <https://msdn.microsoft.com/en-us/library/aa730844(v=vs.80).aspx>

23. Mocha - the fun, simple, flexible JavaScript test framework [Internet]. Mochajs.org. [cited 28 November 2016]. Available from: <http://mochajs.org/>

24. Chai Assertion Library [Internet]. Chaijs.com. [cited 28 November 2016]. Available from: <http://chaijs.com/>

25. The JCT Standard Building Contract 2011 by David Chappell [Book]. 1 Introduction. [cited November 2016]

26. Guide to selecting the appropriate JCT main contract [Internet]. 2011 <http://i.imgur.com/j0NfZ6x.jpg>

27. 5.1 What the architect can do or must do [Book]. [cited 28 November 2016] Available from: The JCT Standard Building Contract 2011 by David Chappell

28.10.1 General [Book]. [cited 28 November 2016] Available from: The JCT Standard Building Contract 2011 by David Chappell

29. 8.2 Quantity surveyors [Book]. [cited 28 November 2016] Available from: The JCT Standard Building Contract 2011 by David Chappell

30. Professional practice 2 [Internet]. Cheahsiokteng.blogspot.co.uk. [cited 28 November 2016] Available  from:[http://cheahsiokteng.blogspot.co.uk/2013/07/professional-practice-2](http://cheahsiokteng.blogspot.co.uk/2013/07/professional-practice-2.html)

31. 3.9 Certificates, notices and other communications [Book]. [cited 28 November 2016] Available from: The JCT Standard Building Contract 2011 by David Chappell