# Introduction

Blockchain technology has made significant strides in the past couple of years with several companies and government organizations actively seeking to capitalize on its benefits.  And yet we also understand that enterprise systems demand significantly higher standards than everyday applications. Security, scalability and resilience while vital in enterprise systems are not readily in the vernacular of most non-enterprise communities; indeed, a bug which is a nuisance in a personal app can be the death knell of an enterprise.  In the context of blockchain the above issues are accentuated by the fact that it is a new technology with many vulnerabilities yet to be discovered.

BlockSIM is a simulation software for blockchain networks that help blockchain architects design and plan resilient blockchain networks. It is based on a python simulator “Simpy” which is used to model the dynamic behavior of the blockchain system.

## Project Aims and Objectives

**Aims**

The aim of this project is to create a flexible prototype of blockchain network simulator that can simulate node behavior, network topology and event of a blockchain. It must be capable of testing multiple consensus at ease.

**Objective**

* Modelling the blockchain network topology.
* Modeling the node properties of blockchain.
* Modelling a complete system-based transaction on blockchain.

## Project Development Methodology

BlockSIM follows incremental software development lifecycle. Following are the reason for the suitability of the choice in accordance to the nature and scope of the system.

* Different prototype can be developed in the earlier stage (phase) of the application development.
* Prototypes can even be tested in the earlier stage and instant changes could be made to the application taking addressing the core requirements of the client and hence deliverables can be made within the designated timeframe.

The phases of Incremental Software Development Life Cycle are:

1. Software Requirements Engineering and Solution Specification

Requirement engineering includes problem elicitation, analysis, requirement specification and design analysis.

1. Problem Elicitation

Problem domain research can be done by reviewing the existing applications similar to the proposed application. By knowing the pros and cons of the compared systems during the research process, the proposed product can be developed by considering the limitations of the compared application. n this way the researcher’s product will be one step ahead than the comparable systems. Development relevant legislation can also guide the path of development inconsideration of other problem domains.

1. System Design and Analysis

The author has chosen Object Oriented Analysis for this development. This technique provides object-oriented modelling description. It describes the problem domain as well as the system at once. (Jackson,1995). The chosen object-oriented analysis techniques are: Textual Analysis, Significant Event Analysis, Command, Queries and Constraints and BON Diagram.

Classes are used to identify the design of the system in depth. Significant event analysis will be used to determine the functionalities that will be available to the end users in whom users can manipulate given data available within their access. Business Object Notation (BON) diagram will be designed to determine the user interaction with the system. Command, Query and Constraint will be used to define the classes required for the system. This table contains the commands, queries and constraints of the respective class. BON cluster chart defines the clusters in which the classes are involved. Entity Relationship Diagram (ERD) will be used to create the structure of the database that will be used for the application. Series of wireframe will be drafted for the layout of the user interface.

1. System Implementation

The core system and backend will be developed using Python programming language under object- oriented programming (OOP) principle. SQL Database will be used for database.

The implementation of the application can be done after the completion of the development. Before implementation, complete testing must be done. Through proper testing, all the bugs and possible errors can be determined. The application can be access with user’s viewpoint.

During the implementation, to store the data in the database, data will be entered on the application. Initially, dummy data will be inserted for testing and with the successful result, real time data shall be entered.

For evaluation of the system, testing shall be done by allowing blockchain architects to execute the system. Through this, developers can analyze the missing features or the drawback of the system and improve it. These instantaneous feedback from the possible users will help to improve.

1. System Testing

Strategies like black-box testing and user testing shall be done for testing the application. Black Box testing examines the functionalities of the solution. It makes sure that the output meets the requirements specified. The system will also be tested throughout the development of the system to minimize the rise of bugs.

# Requirements Engineering

### 2.1. Problem Domain Description

1. System Complexity: The network topology of Blockchain network contains numerous nodes, either computers and other hardware components or cloud instances. Due to the massive deployment, it can be hard for researchers to study the dynamic behavior of a large and alive Blockchain network.
2. Inflexible Structure

As modelling a Blockchain network in real world requires large sources, it is difficult to test the network specification.

1. Time Consuming:

Foremost, modelling of blockchain network require lots of time. Various aspects shall be tested during the development process. Different test strategies like stress testing, logical testing, etc. shall be used on the system which may consume time depending on the size of the network.

1. Expensive:

With the larger hardware requirement comes high running cost. The system can turn out to be expensive even to run a prototype system or for testing purpose.

### 2.2. Functional Requirements

The author has listed out the functionalities to be carried out in the new system based on the research and analysis. The system will have single access level i.e. all the registered user will have the privilege to use all the functionalities available in the system. The user must log in to access the system. Only the administrator can add new user to the system. All the user will have following functionalities:

* Creating new simulation
* Access to past simulations created by the user respectively.
* Viewing the simulator created.
* Downloading the simulation report.
* Updating the user’s personal information.

#### 2.2.1. User Registration

### **Use Case**

|  |  |
| --- | --- |
| **Use Case ID** | **Use Case Name** |
| UC001 | User Registration |

|  |  |
| --- | --- |
| **Goal** | To register the user |
| **Primary Actor** | User |
| **Pre-Condition** | User must follow the subscription plan and contact the system admin. |
| **Post-Conditions** | User is successfully registered to the application and an email with user’s username and password is sent. |

#### **Basic Flow**

|  |  |  |
| --- | --- | --- |
| BASIC FLOW | | |
| Step | Action | Reference |
| 1 | User contacts the system admin. |  |
| 2 | User receives their credentials through email. |  |

#### 2.2.2 User Login

### **Use Case**

|  |  |
| --- | --- |
| **Use Case ID** | **Use Case Name** |
| UC002 | User Login |

|  |  |
| --- | --- |
| **Goal** | To login the user. |
| **Primary Actor** | User |
| **Pre-Condition** | User must be registered into the system. |
| **Post-Conditions** | User is successfully logged in to the system. |

#### **Basic Flow**

|  |  |  |
| --- | --- | --- |
| BASIC FLOW | | |
| Step | Action | Reference |
| 1 | User enters their credential. | User must use the username and password provided. |

#### 2.2.3 Create New Simulation

### **Use Case**

|  |  |
| --- | --- |
| **Use Case ID** | **Use Case Name** |
| UC003 | Create New Simulation |

|  |  |
| --- | --- |
| **Goal** | New Simulation |
| **Primary Actor** | User |
| **Pre-Condition** | User must be login to the system. |
| **Post-Conditions** | A new simulation according to the user’s preference is successfully created. |

#### **Basic Flow**

|  |  |  |
| --- | --- | --- |
| BASIC FLOW | | |
| Step | Action | Reference |
| 1 | The user must fill multiple forms. |  |
| 2 | Either CSV file containing data must be uploaded or user must choose to auto generate the data given by system. | All the field must be entered in the ‘Auto generate Data’ Section. |

#### 2.2.4 Simulation Screen

### **Use Case**

|  |  |
| --- | --- |
| **Use Case ID** | **Use Case Name** |
| UC004 | Simulation Screen |

|  |  |
| --- | --- |
| **Goal** | Monitoring simulation process. |
| **Primary Actor** | User |
| **Pre-Condition** | Either a new simulator must be created, or a previously created simulator must be selected. |
| **Post-Conditions** | Viewing the report on the respective simulation. |

#### **Basic Flow**

|  |  |  |
| --- | --- | --- |
| BASIC FLOW | | |
| Step | Action | Reference |
| 1 | The user creates a new simulation or selects a previous simulation. |  |
| 2 | The simulator displays the blockchain network status in form of visual graph nodes. The information on network stability is represented in form on linear graph. |  |
| 3 | The user is also able to download the full report of the simulation generated. |  |

#### 2.2.5 Updating user profile

### **Use Case**

|  |  |
| --- | --- |
| **Use Case ID** | **Use Case Name** |
| UC005 | Updating user profile |

|  |  |
| --- | --- |
| **Goal** | Allowing the user to update personal information stored in the system. |
| **Primary Actor** | User |
| **Pre-Condition** | User must be logged in. |
| **Post-Conditions** | The data must be updated according to user’s change. |

#### **Basic Flow**

|  |  |  |
| --- | --- | --- |
| BASIC FLOW | | |
| Step | Action | Reference |
| 1 | The user enters the “Edit Profile” page. |  |
| 2 | The user’s information is visible to the user. |  |
| 3 | User can change the desired information and save. |  |

#### 2.2.6 Notification

### **Use Case**

|  |  |
| --- | --- |
| **Use Case ID** | **Use Case Name** |
| UC005 | Notification |

|  |  |
| --- | --- |
| **Goal** | Displaying the notification about completion or error of the simulation. |
| **Primary Actor** | System |
| **Pre-Condition** | A simulation process must be initiated. |
| **Post-Conditions** | User gets notification about the process. |

#### **Basic Flow**

|  |  |  |
| --- | --- | --- |
| BASIC FLOW | | |
| Step | Action | Reference |
| 1 | The user runs a new simulation or selects a previous simulation. |  |
| 2 | Notification is prompt if a user runs a simulation process with larger timeframe or schedules a simulation |  |

### 2.2.3 Performance Requirements

This requirement section discusses about the performance of the system on the basis of speed, capacity, reliability and usability.

#### 2.2.3.1 Speed

The user of the system expects to access or retrieve data in fraction of seconds. The user can perform various tasks quickly if the speed of the system is high. Different users must be able to access same resource at the same time without any delay in either of the accounts. So, the system must be able to deal with heavy file load and high number of users. The factor affecting the speed of the system are the user’s chosen timeframe to run the simulation and the numbers of user running the system simultaneously.

#### 2.2.3.2 Capacity

The memory space of the system or the database organized for the system to store the entire data and information entered by all type of users is capacity. Higher the capacity, larger the number of files can be stored. This system requires large capacity to manipulate and store data of all the users. If proper management of the storage isn’t considered on time, then it can slow down the entire system when one accesses a larger file.

#### 2.2.3.3 Reliability

The reliability of a system is its capability of smooth functioning without any crashes and less interruption. The ability to handle large number of users and data simultaneously without crashing down or slowing the system is reliability. Different tests must be conducted to prevent the system from such disturbances and to solve any arising bugs beforehand.

#### 2.2.3.4 Usability

The usability of the system depends on GUI of the system. The GUI must be uncomplicated, interesting and user-friendly. It must contain all the requirements which will make it easy for user to use. Having nominal approach to the GUI design will make better communication between users and the application.

### 2.2.4 Design Constraints

Design constraints are the components or factor that needs to be considered throughout the system’s interface design and code convention.

* The web application must have clarity, simplicity and reliability.
* The GUI must be designed in such a way that the user doesn’t have to think much to do any process.
* Before developing the system, future expansion of the application must be considered.
* The design will be based on a chosen color combination that is pleasing, effective and suitable.
* Python’s framework Django will be used as the backend, Javascript for the frontend and python programming language for developing the core system.

### 2.2.5 Commercial Constraints

**Time:** The deadline of the application is at the end of April 2018

**Programming Language**: With the consideration of future employability, this system has beendeveloped as a web application using python programming language as its core language.

|  |  |  |  |
| --- | --- | --- | --- |
| Professional | | | |
| System Reputation | | As this type of application is not commercially available, the development of this application can boost the reputation of the application as well as the developer’s company. | |
| Social | | | |
| Improved user satisfaction | With the use of this application, other developers and researchers will find it beneficial for their work. | | |
| Economic | | | |
| Investment for the real time development of a blockchain network | | As this application can simulate a blockchain network, it can be used as an alternative to the need of develop a fully working blockchain network that has to be done even during research process or testing of a prototype system. | |
| Legal | | | |
| Legal Acts | | | Copyright or licensing of the document produced as part of the project is solely entitled to the development team. |

# System Analysis & Design

System analysis and design explain about the development planning of the system. By understanding and specifying the details of the system requirement, author can identify the required components of the system that must be implemented.

## Preliminary Design Stages

### Textual Analysis

The classes of object have been advocated from the requirement specification. This analysis determines entities in form of nouns and verbs.

|  |  |
| --- | --- |
| Candidate Classes | Routine |
| User | * Login * Logout * Access to all the information linked to user’s account. * Edit user’s personal information. * Change the data used for simulation. |
| Create simulation | * Add * Simulate * Generate Report * Download Report |
| Previous simulation | * View * Search * Simulate * Generate Report * Download Report |
| Notification Management | * Display * Acknowledge * Disable |
| User personal Information | * View * Update * Save |

### Significant Event Analysis

|  |  |  |
| --- | --- | --- |
| Events | Performers | Attributes |
| Add User | Admin  Database | Username  Password  Email |
| Login | User  Database | Username  Password  User ID  User Status |
| Create Simulation | User  Database | User ID  Status  Simulation Information (Name, Description, Topology Latency, No. of Nodes, Network node proximity, consensus type, Max Throughput) |
| Add, Update Notification | System  Database | User ID  Notification ID  Simulation ID  Status |
| Update User Information | User  Database | User ID  User Information (Name, Email, Contact No, Payment Status, Organization) |

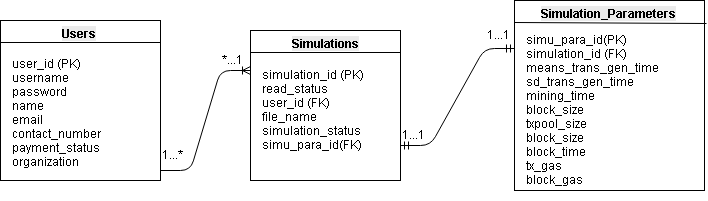
## Detailed Static System Designs

### 3.2.1 First Draft BON System Architecture Diagram

### 3.2.2 BON Class Charts

# System Database Design

## E-R Model



## Attribute Listing

|  |  |  |  |
| --- | --- | --- | --- |
| Table name | Attributes | Data type | constraints |
|  |  |  |  |
| users | Id | INT(11) | Primary key |
|  |  |  |  |
|  | user\_id | VARCHAR(25 | NOT NULL |
|  |  | 5) |  |
|  |  |  |  |
|  | username | VARCHAR(25 | NOT NULL |
|  |  | 5) |  |
|  |  |  |  |
|  | password | VARCHAR(25 | NOT NULL |
|  |  | 5) |  |
|  |  |  |  |
|  | name | VARCHAR(25 |  |
|  |  | 5) |  |

# System Interface Designs

## Draft Interface Designs

### 4.1.1 Wireframes

### 4.1.2 System Navigation Diagram

### 4.1.3 System Screen mock-ups

## Design Revisions

## Final System Build

# Test Strategy

# Report Conclusion