IOTA Visualizer
— Report One —

Ao Shen, Yuxiang Wu, Ruikun Cao, Jiangbo Yu, Xiao Luo, Dongxiao Huang
{as5017, yw1117, rc2917, jy3016, xl1716, dh4317}@imperial.ac.uk

Supervisor: Prof. William J Knottenbelt, Dominik L Harz
Course: CO530, Imperial College London

5th March, 2018

1 Introduction

Some IOTA investors and enthusiasts could care less about the underlying technology underpinning their investments, but others do. Our mission is to help explain in layman's terms to those who do care, the mechanics of IOTA. To this end, we have decided to develop a website featuring 3D visualization and informative transaction data. The requirements necessary to make this possible is categorized by their essentiality, functionality and feasibility.

2 Project Specification

2.1 Essential/Non-essential

A Tangle Visualizer illustrating a map of milestones, confirmed transactions and new tips joining the network is essential as this will be the key medium to which users can interface with IOTA's distributed ledger technology. As for the back-end, our initial goal is to set up a stable server to query the IOTA API and send HTTP requests so we can provide the essential functionalities to a select group of users.

Secondarily, We have also considered useful, but non-essential features for our projects if our time and skills permit. An example would be a global heat map revealing the geographic distribution of transactions. This would help investors visualize the location and volume of transactions to best take advantage of the situation. We could also try to provide useful computations such as mean confirmation time, value transfer per second, and information on each block/node.

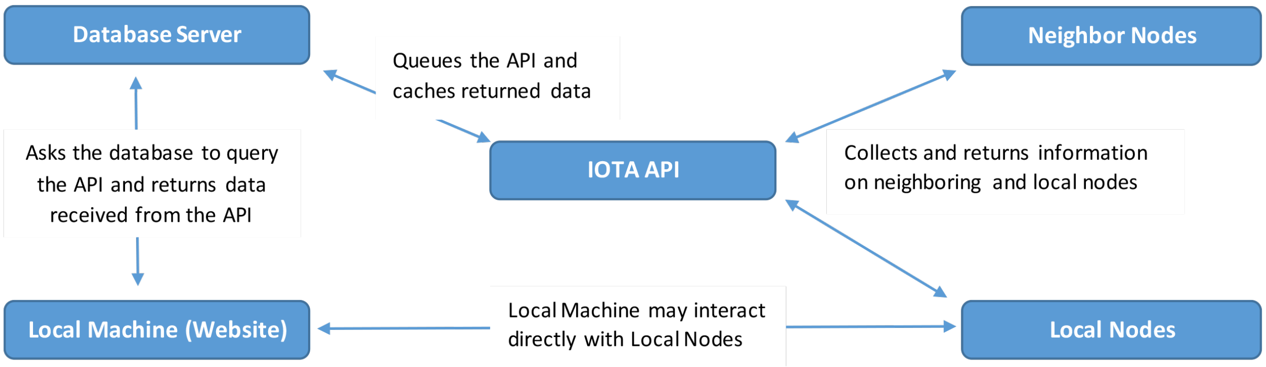

Database Server
Queues the API and caches returned data
Neighbor Nodes
Asks the database to query the API and returns data received from the API
IOTA API
Collects and returns information on neighboring and local nodes
Local Machine (Website)
Local Machine may interact directly with Local Nodes
Local Nodes

Figure 1: Initial conceptualization of the network model

Essential	Non-essential
Tangle visualizer	Dynamic 3D animation
Website with real-time transaction information	Useful secondary computed data
Database for caching IOTA API data	Global activity heat map

Table 1: Essential and Non-essential requirements

2.2 Functional/Non-functional

The functional requirements are consistent with our essential features. However, many non-functional aspects including visual fidelity and speed are subjective. Portability, performance, extendibility, and maintainability are again difficult to measure without a benchmark, but will be factored into our internal discussions when we consider trade-offs between different libraries and implementation methods.

Functional	Non-functional
Tangle visualization map	Visually pleasing
Website conveying real-time scraped and computed info	portable, and fast
A database to cache API data before being sent to the website	Extensible and maintainable

Table 2: Functional/Non-functional requirements

2.3 Risk and Feasibility

Due to time and manpower limitations, we need to assess the feasibility (and risk) of each feature and how we can prioritize them to maximize user satisfaction. We divided these into front-end and back-end components.

2.3.1 Front-end

1. High Priority: The front-end is the most essential and is therefore also the minimum requirement for this project. This will allow users to interface with our work. Without it, no one will be able to see even the most sophisticated software. Additionally, as the GUI will be used in visual testing, delayed completion of the front-end will hinder the overall progression of the project.

2. High Feasibility, Low Risk: Although starting out, few members in the group have prior knowledge in internet languages and computer networks, we are confident that with a strong foundation in Java and C++, every member can eventually contribute as they acquire relevant knowledge throughout the term. A big risk is the looming final exams, which may impede our progress from March to May. Another would be the application/processing layer of the program. We are unsure of the ramifications of implementing analytics in JavaScript, and its impact on browser performance.

2.3.2 Back-end

1. Medium Priority: Near the early stages of the project, a server is not necessary as a web page can directly queue the IOTA API. However, this is inefficient and not scalable. A well-designed server side service can help improve overall performance and reduce redundant bandwidth usage.

2. Medium Feasibility, High Risk: Due to the abundance of database libraries, and our participation in the database and distributed systems courses, we are confident in our ability to implement the database server. The risk arises from the fact that none of us have had previous exposure to database management. We have no idea as to how much data we need to process, potentially creating a data flow problem.

2.4 Extensions beyond the scope of this project

We believe that with the graphical user interface and information terminal completed, the project can be easily extended by, or integrated into a full-fledged trading platform in the future.

2.5 Stakeholders

2.5.1 Users

The likely users and target audience of this product will be those who wish to learn more about the IOTA tangle, the transactional features of IOTA and its directed acyclic graph technology. Accessibility and understandability are imperative, hence our choice to develop a web application with visualization tools.

2.5.2 Core Developers

The developers and maintainers of the project are the 6 members of our group.

2.5.3 Developer Community

As the repository is hosted publicly on GitHub, we welcome feedback and incorporation of our code into other projects given sufficient credit.

2.5.4 Support

Our supervisors support us in providing resources such as server access, specification guidance, implementation assistance, and communication facilitation.

2.5.5 Assessors

Whether we meet our deliverables in an adequate standard will be determined by independent judges from the department of computing, and industry.

3 Development Strategy

3.1 Agile Method (XP programming)

As our core development team is quite small, XP programming seems to most suit us. In particular, our group of 6 is frequently divided into groups of 2 or 3 and undergo peer programming. An experienced programmer is often coupled with 2 less experienced ones to share knowledge, spot mistakes and bounce ideas. This has improved our teamwork and computing skills. Additionally, We have decided not to draw an UML diagram as our actual website veered so far from the website design template (See Appendix Figure 1 and 2) that we believe it will be a waste of time to develop another diagram for our project.

3.2 Communication

We document bugs and write our reports using Overleaf, a cloud-based document editor that supports LaTeX. The program compiles in real-time and allows for version control and concurrent contributions.

We communicate through WeChat – a multi-functional messaging system. While we do a significant amount of scheduling and information sharing on the platform, we prefer to debug and test our programs in-person, where communication is more precise. Outside of our formal weekly meetings, we share an almost entirely identical schedule. This allows to be constantly in touch physically and test new features as they are pushed.

We also use Slack and emails to communicate with our supervisors and core IOTA developers, to enlist their help, and to facilitate meetings and discussions.

When we are coding apart, we make use of extensive commenting on our code to inform each member of our intentions.

3.3 Team Management and Division of Work

Members tend to reach consensus as to what we need to do and select their own tasks. We have yet to encounter interpersonal conflicts, but should it rise, we intend to resolve it electorally. Each group member's contribution will be logged separately to encourage participation.

Staying true to Agile and XP we strongly prefer working software, over models and documentation, and refactoring over lengthy design discussions. Extensive documentation of bugs for a relatively small project often takes longer than choosing to fix them quickly.

3.4 Test-driven development

Our testing strategy is divided into visual and unit testing. We intend to write tests before we produce the code to eliminate any biases, and incorporate them into our specification. A more elaborate scheme will be introduced in Report Two. A bug management database may also be introduced.

4 System Boundaries

4.1 Hardware Dependencies

This project will be browser-based and requires no emulation. The processing requirements are also expected to be within the bounds of most web applications. Therefore, there will not be any local hardware requirements. However, we do depend on an external database server that has reasonable uptime. This will greatly enhance the performance of our platform as another server, instead of our local machines will be doing most of the redundant queuing. In our case, server access has been granted by Imperial's Department of Computing through Microsoft Azure.

4.2 Software Dependencies

Our website (See Appendix Figure 2) will be developed using JavaScript and HTML5 for maximum comparability with popular modern browsers such as Chrome, Internet Explorer, Firefox, and Safari. While our program will not depend on the operating system (Windows, MacOS, Linux), as the differences are expected to be simulated away by the browser, we do recognize the performance and dimensional requirements of mobile browsers and OS (IOS, Android). However, as mobile usage is not our key concern at this point, development will be focused on the average PC. For lower bandwidth and processor capacity scenarios, we will consider reducing the refresh rates of our visualizer in exchange for reduced visual fidelity.

4.2.1 Languages

- JavaScript for advanced styling and the queuing of the API
- HTML5 for modern web browser compatibility
- CSS for basic styling of web pages
- NodeJS for back-end integration of the web server to the browser
- SQL for database management

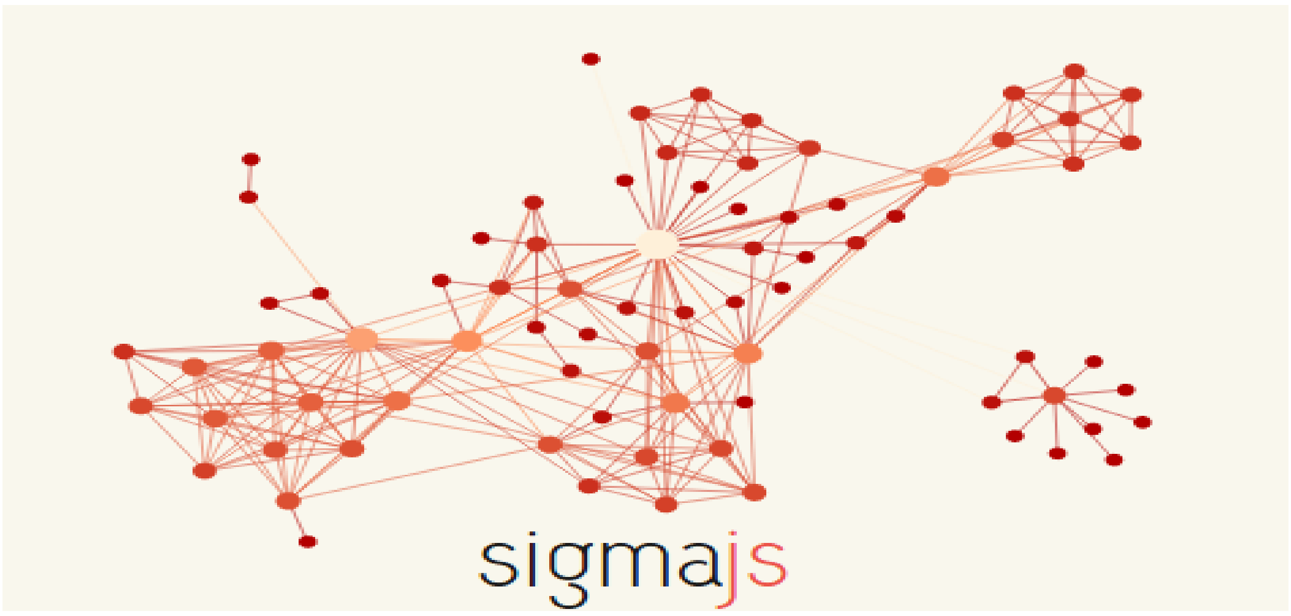


Figure 2: Potential visualization library we will use from Sigma.JS

4.2.2 Package and service dependencies

- IOTA API [1] which enables the project to fetch data from IOTA nodes.
- External visualization libraries will be used to visualize the Tangle.
- Microsoft Azure will be used to test and deploy our back-end server.

5 Version-control System

A Public Github Repository [2] with collaborators will be the project's Version-control System. We do not mind if our project is forked. In fact, we welcome it.

Our project follows an iterative model, meaning that our group is in a constant feedback loop regarding changes to the requirement, implementation and testing. Each new iteration will have an informative commit log and version number. When members work individually off-line for long periods of time, branching is encouraged to reduce merge conflicts and allow concurrent work to be done.

6 Schedule

Deadline	Description	Progress
Jan 15	Gathering Requirements	Completed
Jan 22	Website (See Appendix Figure 2)	Completed
Jan 29	Report One	Completed
Feb 5	Queuing the API using JavaScript/JQuery	In Progress
Feb 12	Choosing and Implementing a Visualization Library	Not Started
Feb 19	Finalizing the Visualization Library	Not Started
Feb 26	Building the back-end database	Not Started
Mar 5	Report Two	Not Started
Mar 12	Finalizing the database	Not Started
May 16	Extensions, Final Report, Presentation Preparation	Not Started

Table 3: Adopted concrete and timed deliverables to combat slippage

References

[1] IOTA Foundation. *IOTA API*. 2018. URL: `http://github.com/iotaledger`.

[2] IOTA Team. *Project Repository*. 2018. URL: `https://github.com/ao333/IOTA-Visualizer`.

7 Appendix

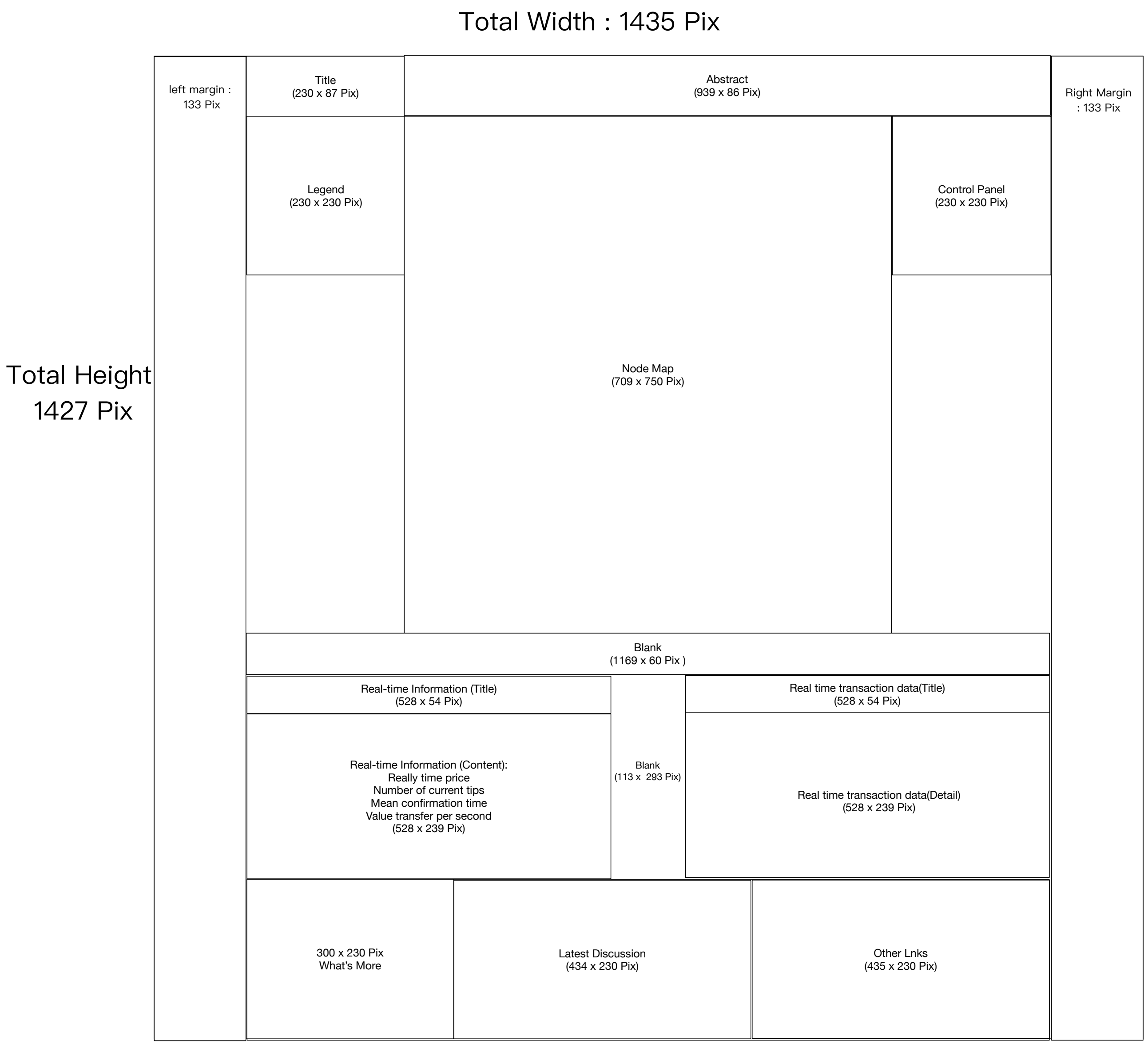
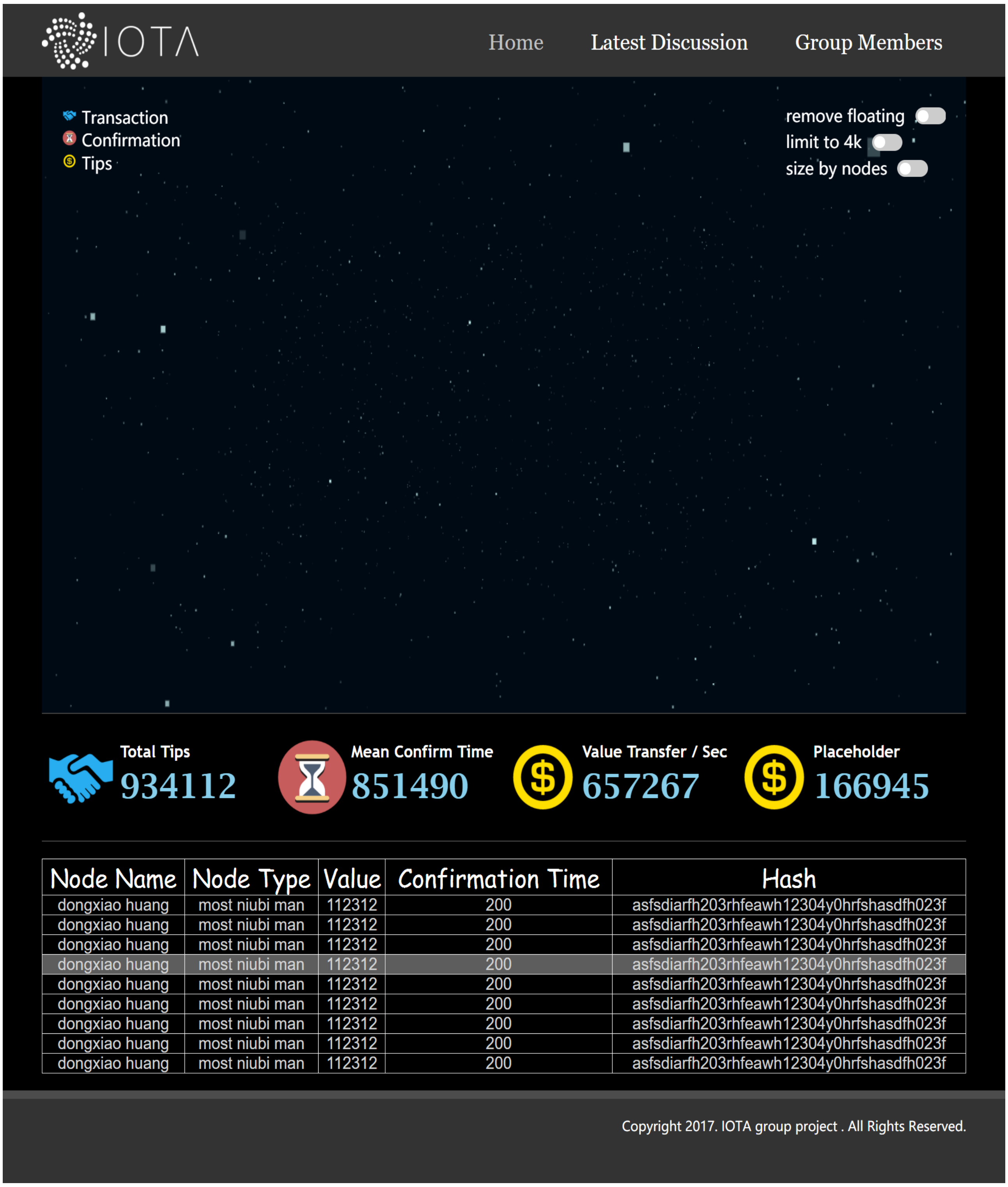

Total Width : 1435 Pix
Total Height 1427 Pix
left margin : 133 Pix
Title (230 x 87 Pix)
Abstract (939 x 86 Pix)
Right Margin : 133 Pix
Legend (230 x 230 Pix)
Control Panel (230 x 230 Pix)
Node Map (709 x 750 Pix)
Blank (1169 x 60 Pix)
Real-time Information (Title) (528 x 54 Pix)
Real time transaction data(Title) (528 x 54 Pix)
Real-time Information (Content): Really time price Number of current tips Mean confirmation time Value transfer per second (528 x 239 Pix)
Blank (113 x 293 Pix)
Real time transaction data(Detail) (528 x 239 Pix)
300 x 230 Pix What's More
Latest Discussion (434 x 230 Pix)
Other Lnks (435 x 230 Pix)

Figure 1: Initial Website Spec



Total Tips 934112

Mean Confirm Time 851490

Value Transfer / Sec 657267

Placeholder 166945

Node Name	Node Type	Value	Confirmation Time	Hash
dongxiao huang	most niubi man	112312	200	asfsdiarfh203rhfeawh12304y0hrfshasdfh023f
dongxiao huang	most niubi man	112312	200	asfsdiarfh203rhfeawh12304y0hrfshasdfh023f
dongxiao huang	most niubi man	112312	200	asfsdiarfh203rhfeawh12304y0hrfshasdfh023f
dongxiao huang	most niubi man	112312	200	asfsdiarfh203rhfeawh12304y0hrfshasdfh023f
dongxiao huang	most niubi man	112312	200	asfsdiarfh203rhfeawh12304y0hrfshasdfh023f
dongxiao huang	most niubi man	112312	200	asfsdiarfh203rhfeawh12304y0hrfshasdfh023f
dongxiao huang	most niubi man	112312	200	asfsdiarfh203rhfeawh12304y0hrfshasdfh023f
dongxiao huang	most niubi man	112312	200	asfsdiarfh203rhfeawh12304y0hrfshasdfh023f
dongxiao huang	most niubi man	112312	200	asfsdiarfh203rhfeawh12304y0hrfshasdfh023f

Copyright 2017. IOTA group project . All Rights Reserved.

Figure 2: Screen shot of our Website