

**Capstone Project 1**

**DEMARK - DECENTRALIZED MARKET**

Database Design Document

Version: 1.1

CODE: DEMARK-1.0

**Mentor:** Man Nguyen Duc

**Team member:**

Ha Truong Van

Dat Ngo Ha Van

Nhan Vo Hoang Quoc

Phuong Tran Nhat

**INTERNATIONAL SCHOOL**

# **Project Information**

|  |  |  |  |
| --- | --- | --- | --- |
| **Project acronym** | DEMARK | | |
| **Project Title** | Demark - DECENTRALIZED MARKET | | |
| **Start Date** | Aug 31, 2018 | **End Date** | Dec 10, 2018 |
| **Lead Institution** | International School, Duy Tan University | | |
| **Project Mentor & contact details** | Man Nguyen Duc  Email: mannd@duytan.edu.vn  Tel: 0904 235 945 | | |
| **Product Owner** | Ha Truong Van | | |
| **Scrum Master** | Phuong Tran Nhat | | |
| **Team members** | **Name** | **Email** | **Tel** |
|  | Ha Truong Van | vanha30111997@gmail.com | 0969356097 |
|  | Dat Ngo Ha Van | ngohavandat93ndc@gmail.com | 01288446176 |
|  | Nhan Vo Hoang Quoc | quocnhan810@gmail.com | 01674559527 |
|  | Phuong Tran Nhat | nhatphuongb1@gmail.com | 01692502010 |
|  |  |  |  |

# **Proposal Document**

|  |  |  |  |
| --- | --- | --- | --- |
| **Document Title** | Database Design Document | | |
| **Reporting Period** | October, 2018 | | |
| **Team Information** | **Name** | **Role** |  |
|  | Nhan Vo Hoang Quoc | Team member |  |
|  | Ha Truong Van | Product owner & team member |  |
|  | Dat Ngo Ha Van | Team member |  |
|  | Phuong Tran Nhat | Scrum master & Team member |  |
| **Date** | September 10, 2018 | **Filename** | DEMARK\_DATABASE\_1.1.docx |
| **Access** | Project and Duy Tan University Program | |  |

# **Document History**

|  |  |  |
| --- | --- | --- |
| Document History | | |
| **Version** | **Date** | **Comments** |
| V1.0 | September 10, 2018 | * Draft of document |
| V1.1 | September 20, 2018 | * Official document |

# **Document Approvals**

​The following signatures are required for approval of this document.

|  |  |  |
| --- | --- | --- |
| Man Nguyen Duc  *Mentor* |  | Date: |
| Ha Truong Van  *Product Owner* |  | Date: |
| Phuong Tran Nhat  *Scrum Master* |  | Date: |

TABLE OF CONTENTS

[**Project Information**](#_3znysh7)

[**Proposal Document**](#_tyjcwt)

[**Document History**](#_4d34og8)

[**Document Approvals**](#_17dp8vu)

[**1.** **Introduction**](#_26in1rg)

[**1.1.** **Purpose**](#_35nkun2)

[**1.2.** **Goal**](#_44sinio)

[**1.3.** **Scope**](#_z337ya)

[**1.4.** **Data storage platforms:**](#_3j2qqm3)

[**1.5.** **Definitions, Acronyms and Abbreviations**](#_1y810tw)

[**2.** **Database Design:**](#_2xcytpi)

[**2.1.** **Table Overview:**](#_2bn6wsx)

[**2.2.** **Entity Relationship Diagram**](#_3as4poj)

[**2.3.** **Table Relationship Diagram:**](#_49x2ik5)

[**2.4.** **Detail**](#_147n2zr)

[**2.5.** **Tokens**](#_23ckvvd)

[**2.6.** **Users**](#_32hioqz)

[**3.** **References**](#_1hmsyys)

# **Introduction**

## **Purpose**

Place information system’s database design document describes structure of database and file structure of system. Database Design document will introduce all attribute the System that will help developer and tester base on this design to implement and test.

## **Goal**

To create database tables most accurate.

## **Scope**

This Database Design Document provides the basic for “Demark – Decentralized Market” Database design.

It defines the database that will support the “Business Service Management” Data Model.

It describes both logical and physical definition, non-functional issues, and the database interfaces; storage aspects are defined in the physical database design sections.

The tables performance considerations requirements. The following topics are covered in this document:

* Assumptions and decisions on database design.
* Entity-mapping.
* Table column definitions.
* Primary, unique and foreign key definitions.
* Column and row level validation rules (check constraints).
* Rule for populating specific columns (sequences, derivations, demoralized (column)).
* Interfaces and dependencies with other components.
* Data access description.

The database design for “Demark – Decentralized Market” is composed of definitions for database objects derived by mapping entities to tables attributes to columns, unique identifiers to unique keys and relationship to foreign keys.

During design, these initial definitions are enhanced to support the functionality described in the functional specification/ user stories and defined in the primary and supporting modules of the application high level design.

## **Data storage platforms:**

- Data of application is stored in Firebase Realtime Database and Ethereum network.

- Data structure is stored as Json Tree includes: Node root & node leaf

- Data is stored as “Key”: “Value” Definitions, Acronyms and Abbreviations

## **Definitions, Acronyms and Abbreviations**

|  |  |  |
| --- | --- | --- |
| **Abbreviations** | **Description** | **Comment** |
| PK/FK | Primary/ Foreign Key | Use to indicate a file is a Primary or Foreign key in a table |
| ERD | Entity Relationship Diagram | Show the relationship between entities in the system |
| AC | Auto created | Auto |

1. **Database Design:**

* 1. **Table Overview:**

|  |  |  |
| --- | --- | --- |
| No. | Table name | Short Description |
| 1 | Tokens | This table contains all tokens in the system. |
| 2 | Users | This table contains all users and their address in Blockchain |

Ethereum blockchain overview [2]:



As with all other blockchains, the Ethereum blockchain begins life at its own genesis block. From this point (genesis state at block 0) onward, activities such as transactions, contracts and mining will continually change the state of the Ethereum blockchain. In Ethereum, an example of this would be an account balance (stored in the state trie) which changes every time a transaction, in relation to that account, takes place. [1]

Importantly, data such as account balances are not stored directly in the blocks of the Ethereum blockchain. Only the root node hashes of the transaction trie, state trie and receipts trie are stored directly in the blockchain. [1]

There are two vastly different types of data in Ethereum; permanent data and ephemeral data. An example of permanent data would be a transaction. Once a transaction has been fully confirmed, it is recorded in the transaction trie; it is never altered. An example of ephemeral data would be the balance of a particular Ethereum account address. The balance of an account address is stored in the state trie and is altered whenever transactions against that particular account occur. It makes sense that permanent data, like mined transactions, and ephemeral data, like account balances, should be stored separately. Ethereum uses trie data structures to manage data. [1]

* 1. **Entity Relationship Diagram**

****

State trie [2]:

There is one, and one only, global state trie in Ethereum. [1]

This global state trie is constantly updated. [1]

The state trie contains a key and value pair for every account which exists on the Ethereum network. [1]

A storage trie is where all of the contract data lives. Each Ethereum account has its own storage trie. [1]

Transaction trie [2]



Each Ethereum block has its own separate transaction trie. A block contains many transactions. The order of the transactions in a block are of course decided by the miner who assembles the block. The path to a specific transaction in the transaction trie, is via (the RLP encoding of) the index of where the transaction sits in the block. Mined blocks are never updated; the position of the transaction in a block is never changed. This means that once you locate a transaction in a block’s transaction trie, you can return to the same path over and over to retrieve the same result. [1]

* 1. **Table Relationship Diagram:**

****





* 1. **Detail**

This includes the attribute:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Attributes | Datatype | Null | Key | Default | Extra |
| ID | string | not | PK |  | AC |
| name | string | not |  |  |  |
| symbol | string | not |  |  |  |
| address | string |  |  | 0x0 |  |
| cashier | string |  |  |  |  |
| decimal | number | not |  |  |  |
| rating | number | not |  |  |  |
| description | string | not |  |  |  |
| approve | boolean | not |  | false |  |

* 1. **Tokens**

****

* 1. **Users**

This includes the attribute:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Attributes | Datatype | Null | Key | Default | Extra |
| dtuID | string | not |  |  |  |
| walletID | string | not | PK |  | AC |
| isAdmin | boolean | not |  | false |  |



**DTU Token (Smart contract):**

****

1. **References**

[1] <https://hackernoon.com/getting-deep-into-ethereum-how-data-is-stored-in-ethereum-e3f669d96033>

Last accessed 7/12/2018

[2] <https://i.stack.imgur.com/afWDt.jpg>

Last accessed 7/12/2018