VENDA: ELECTRONIC ASSET TRADING PLATFORM

CAB302 Semester 1, 2021

Jeremy Chang, Nicole Truong, Natalie Smith – Group 21

This document includes the Detailed Design documents for Milestone #2

Table of Contents

Detailed Design

Classes

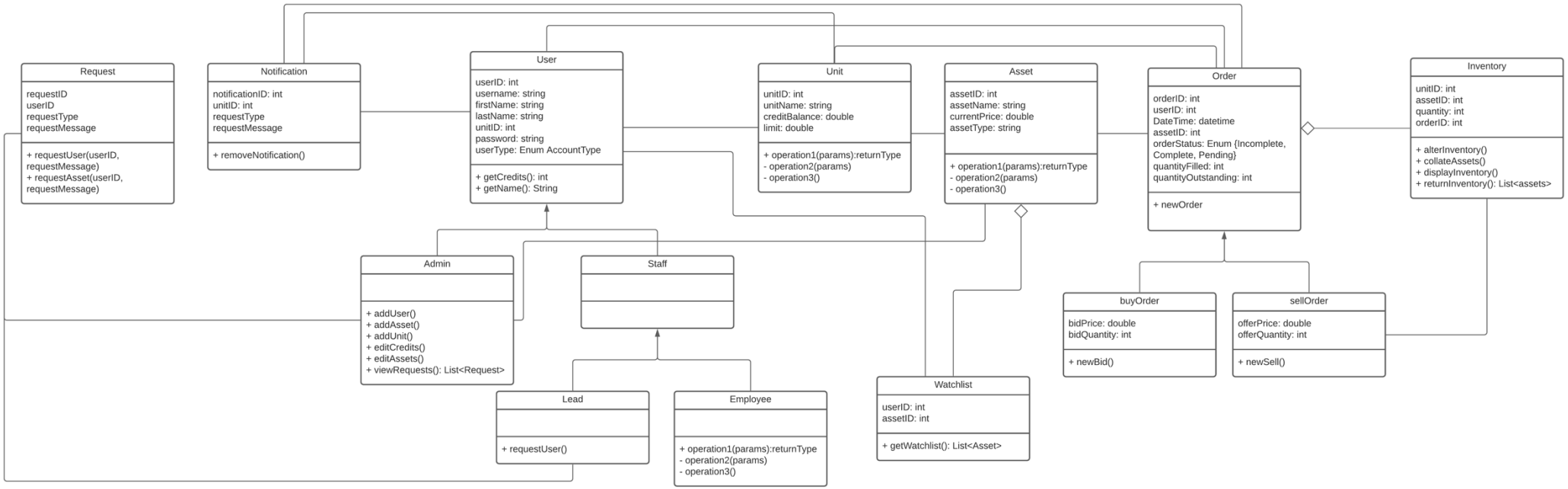
Classes:

- User
  - This class is used for manipulating and creating users in the trading platform storing their identification details and passwords encrypted. Different types of users will be subclasses of users who have different authority in the system - admin and staff → lead and employee.
  - Relevant information, most importantly, their username and password are required to access the system. Therefore, in adding a new user or accessing information regarding the user, they must input username, password, unitID, account type, first name, last name. The User class works alongside the unit, where of which the user would be able to perform various tasks that directly impact the unit – such as make orders on behalf of the unit.
- Asset
  - Different assets can be traded on the platform and must hold similar variables to be traded on the marketplace. Unit's can create new units to be traded within the organisation.
- Unit
  - This class is responsible for identifying an organisation unit and how a unit may interact with other units/ with other users. The units are the basis in the trades as all trades are executed through a unit.
- Order
  - Performing a trade on the platform will be executed through the order class. It will identify whether the order has been executed or is still pending. Trades require mandatory fields to align the needs of buyers and sellers.
- Notification
  - A simple class used to create notifications to improve user experience. It will store information like 'order executed', 'new user created', etc
- Request
  - Organisational unit leads will be able to request the addition of new users, assets and other requests that will be sent to the systems admin.
- Inventory
  - Inventory class responsible for manipulating inventory figures and will interact dynamically with trades that occur. Each organisational unit will have an individual inventory.

- Watchlist
    - Each organisational unit will be able to track assets they are interested in and this class will store that information.

Classes	Method	Purpose	Parameters	Output
User	getCredits()	To retrieve the credits of the user based on the unitID they are assigned to		int
Admin	addUser()	The admin are the only users able to create new users. Therefore, the addUser function would only be addressed to create a new user if and only when the user is an admin	Username string, first_Name varchar, last_Name varchar, unitID, Password string, userType AccountType enum,	
Asset				
Unit				
Order				

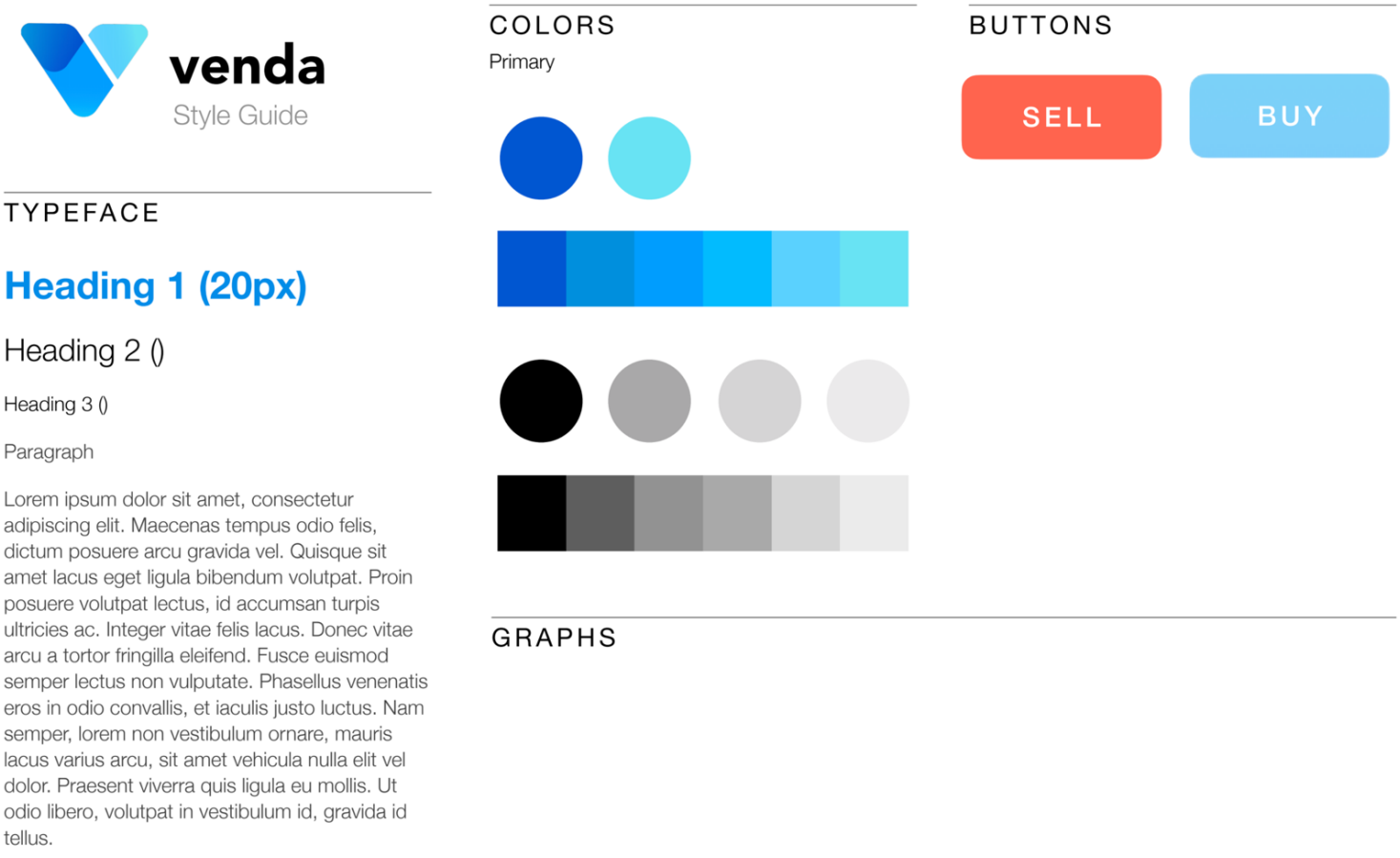
UML Class Diagram



GUI Designs

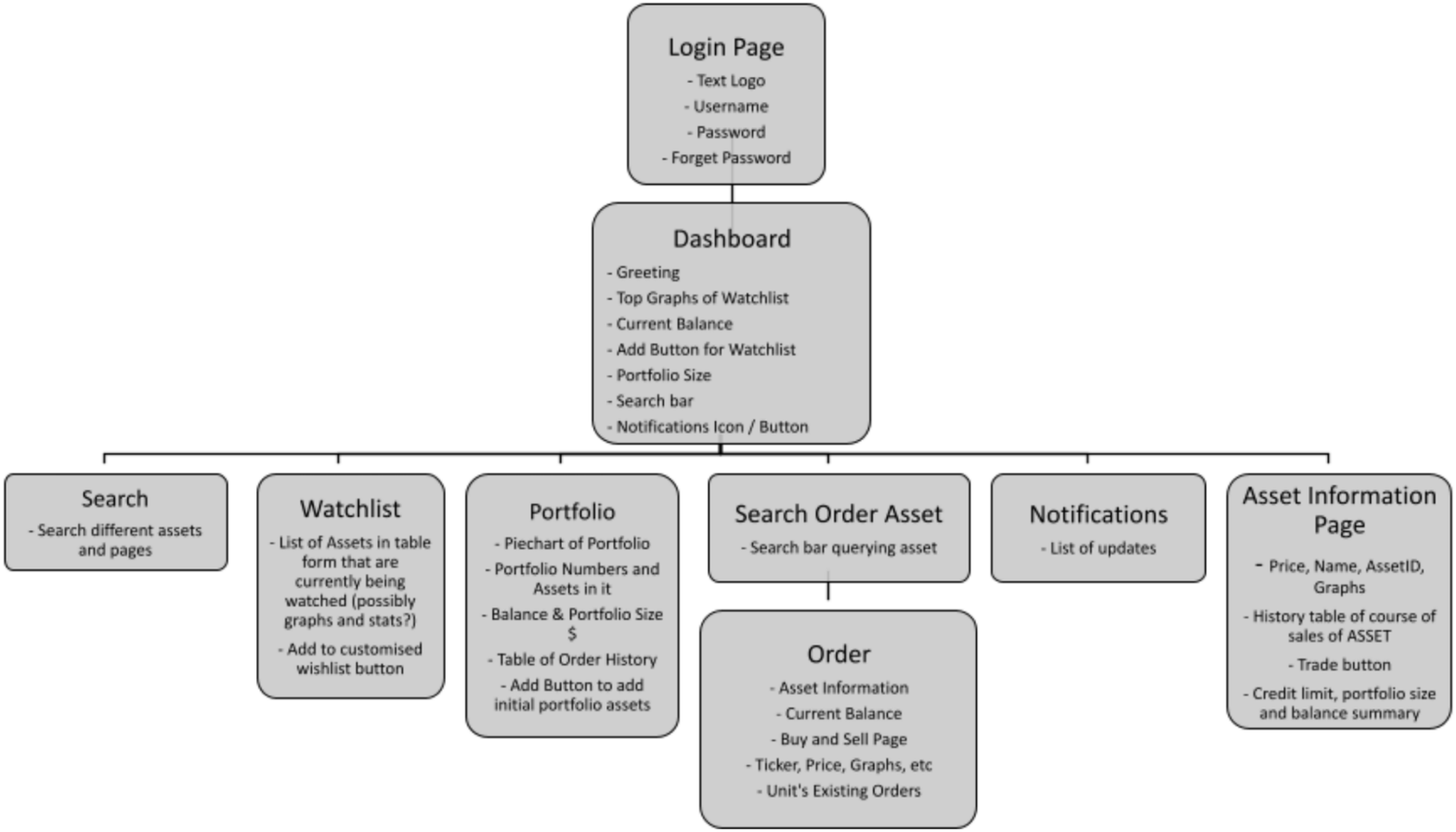
Style Guide

For the design of the website, we have chosen a simple, minimalist blue colour palette to evoke a responsible, loyal and trustworthy feel to the platform. As shown below, we have also decided on a Logo for the platform.



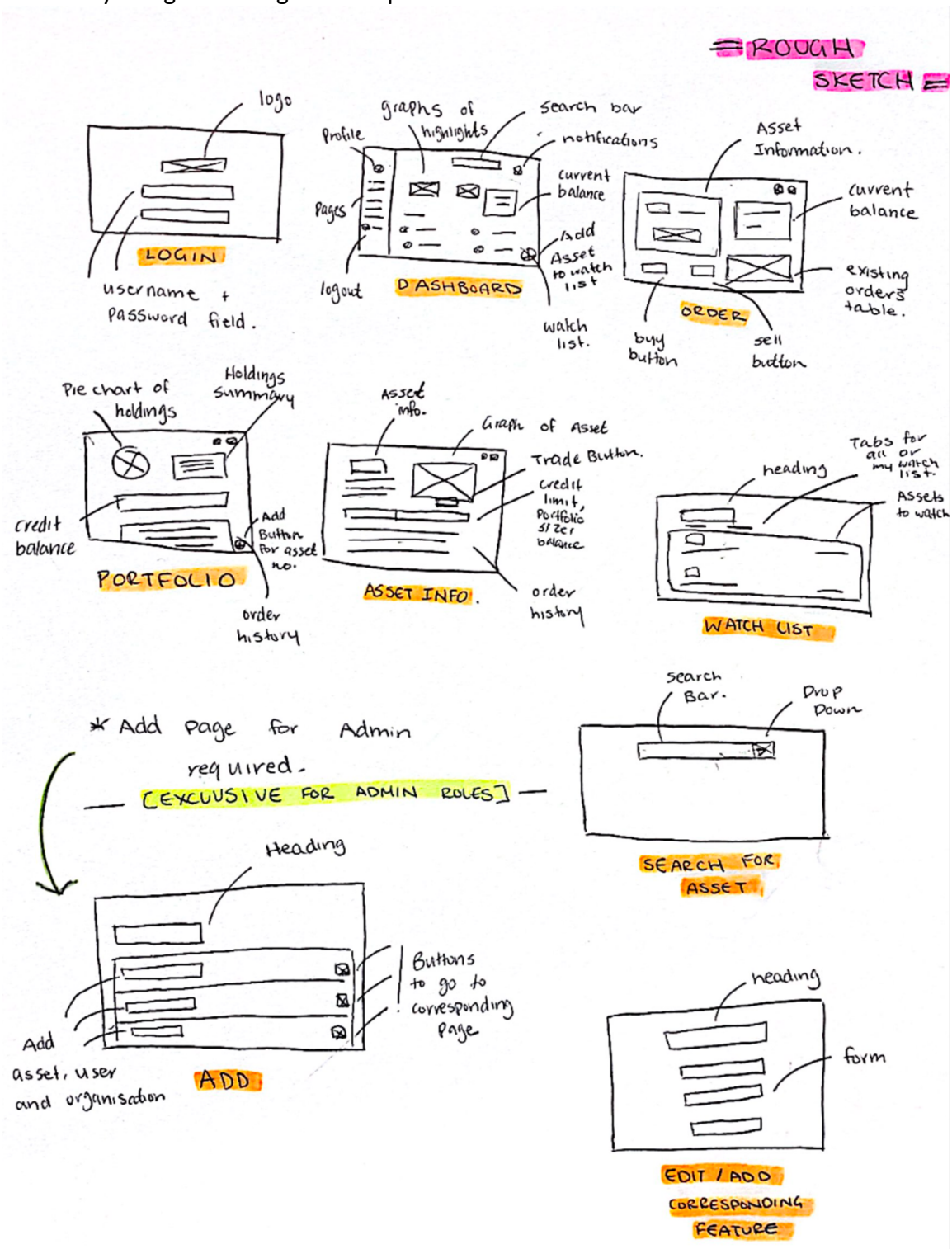
Hierarchy of Pages

The hierarchy of the pages was developed to establish the different elements required for each of the pages within the GUI. This would then be used in designing the wireframes.

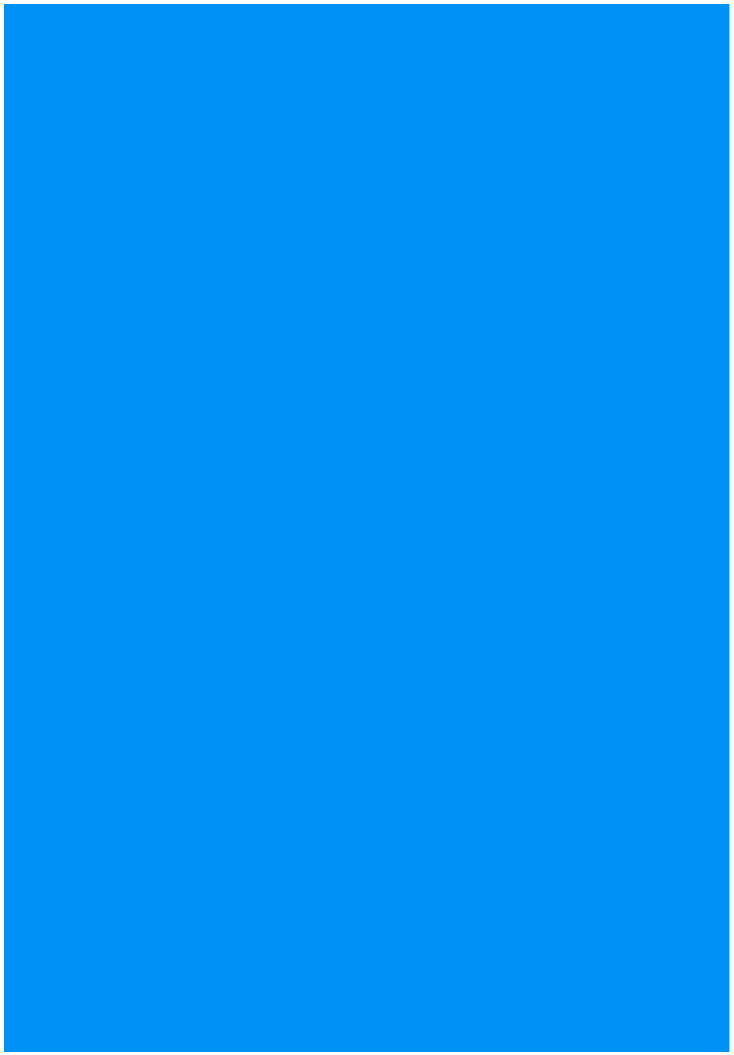


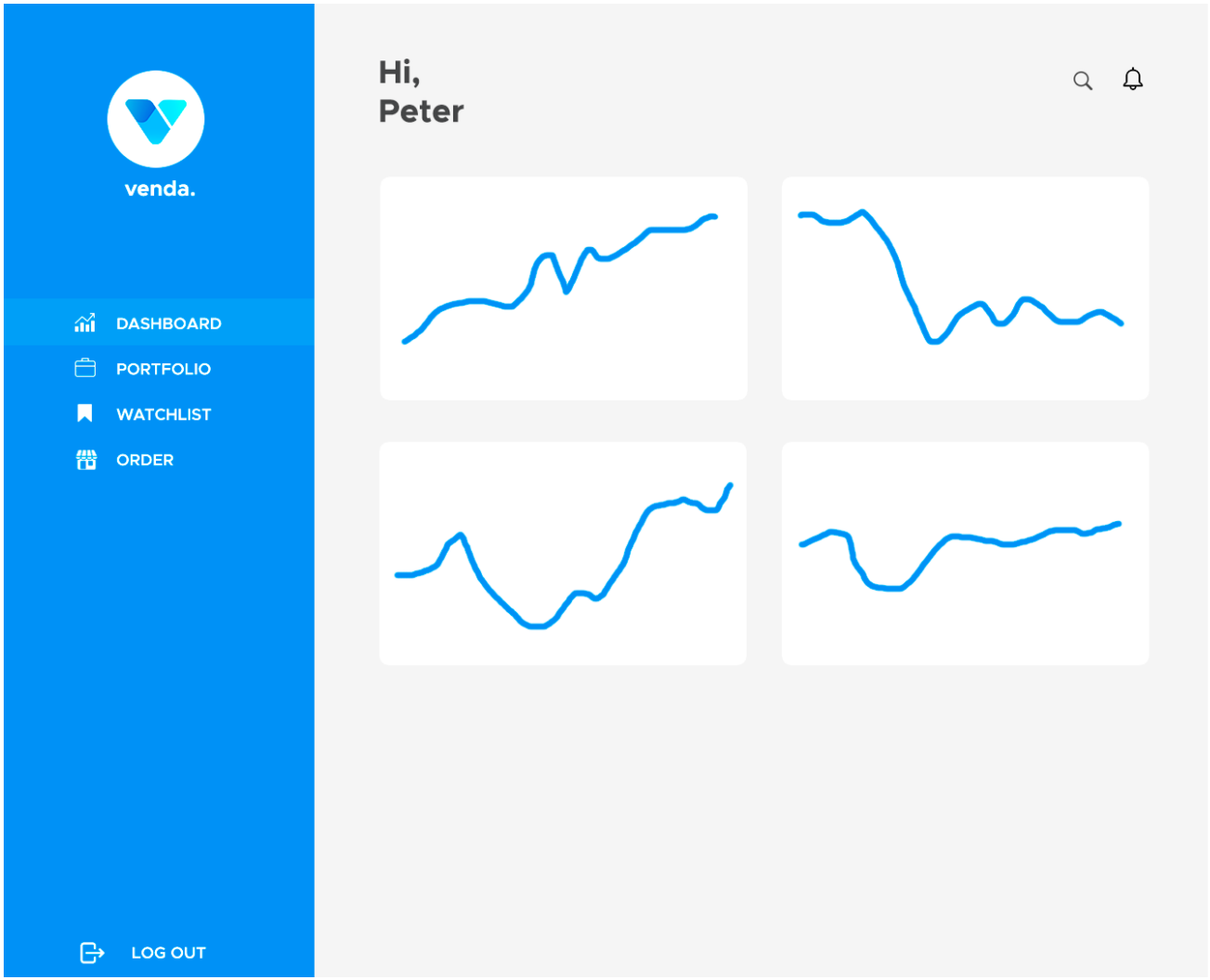
Wireframes

Below are icons of the different pages we expect to be included in the electronic trading system. Consistent design elements were applied to ensure the user can easily and effectively navigate throughout the platform.



Medium Fidelity



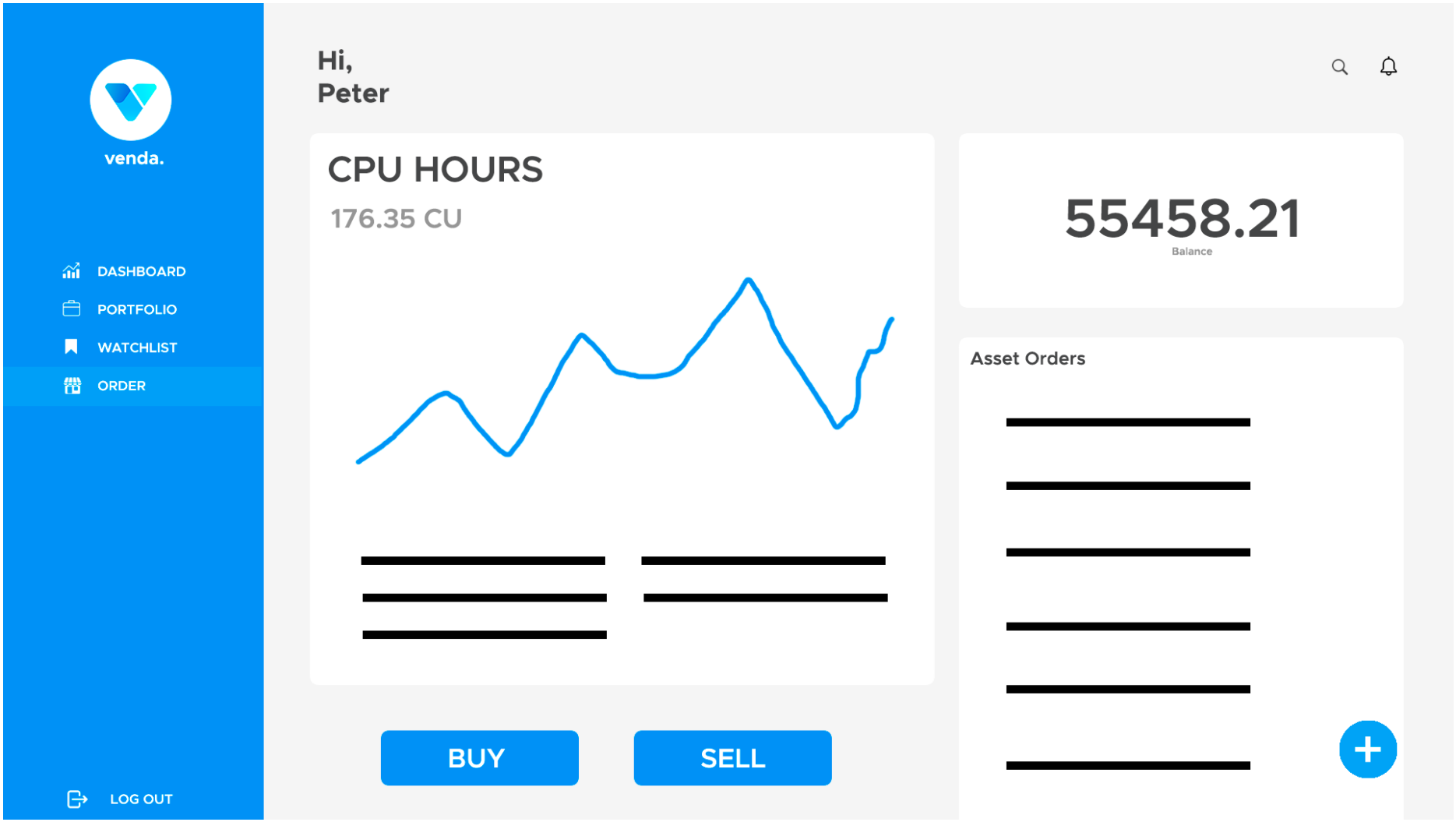


8742

Credit Units

5

Outstanding Orders

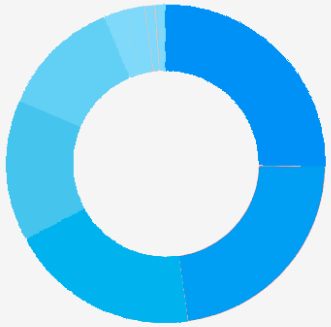

venda.
DASHBOARD
PORTFOLIO
WATCHLIST
ORDER
LOG OUT
Hi,
Peter
CPU HOURS
176.35 CU
55458.21
Balance
Asset Orders
BUY
SELL

venda.

DASHBOARD
PORTFOLIO
WATCHLIST
ORDER

LOG OUT

Hi,
Peter

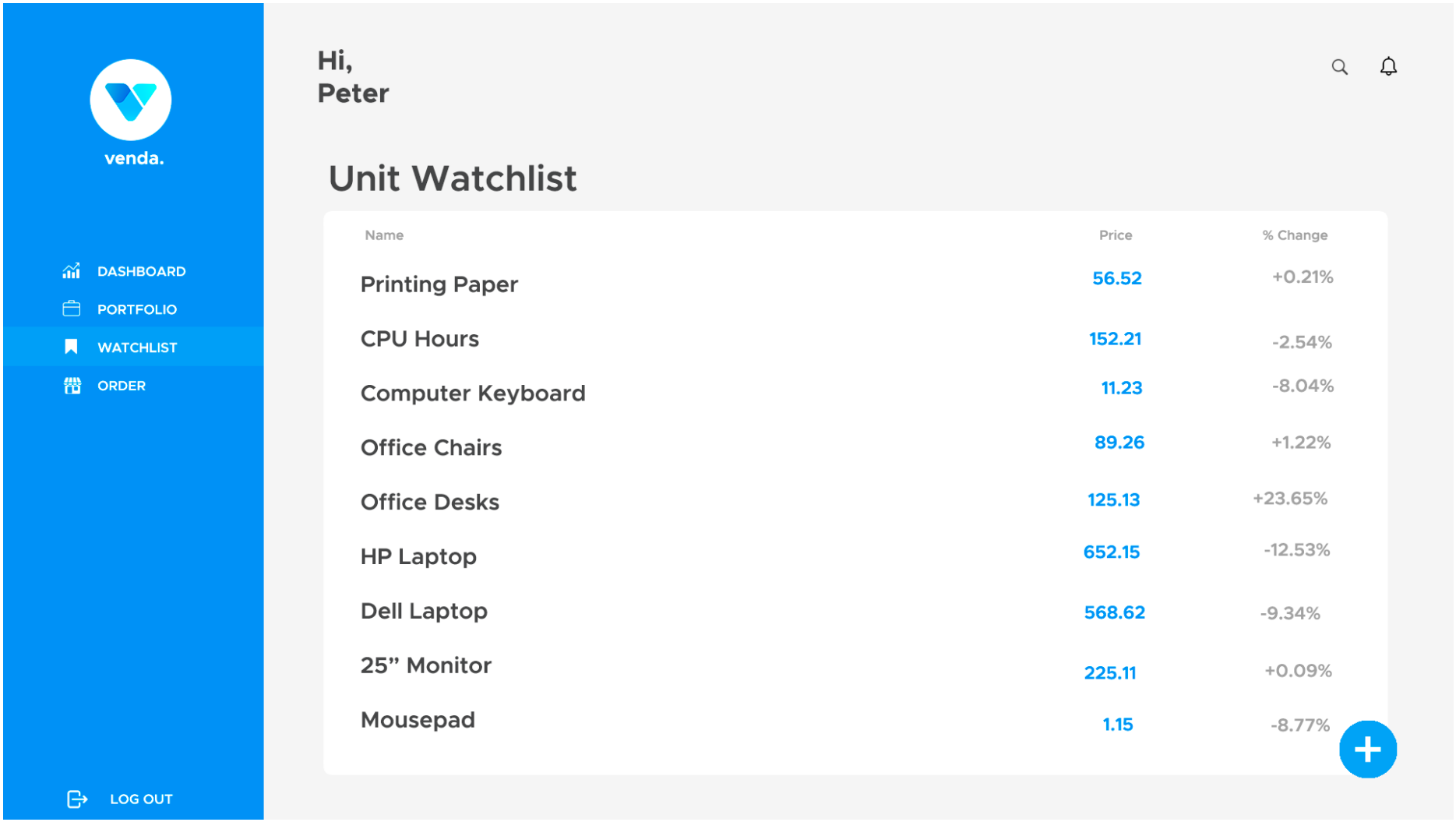


26%	CPU Hours	5000 CU
23%	Printing	4000 CU
15%	Keyboards	2000 CU
7%	Computer Mice	1000 CU
6%	Other	500 CU

8742
Credit Units

5
Outstanding Orders

Order History


venda.
DASHBOARD
PORTFOLIO
WATCHLIST
ORDER
LOG OUT
Hi,
Peter
Unit Watchlist
Name
Price
% Change
Printing Paper
56.52
+0.21%
CPU Hours
152.21
-2.54%
Computer Keyboard
11.23
-8.04%
Office Chairs
89.26
+1.22%
Office Desks
125.13
+23.65%
HP Laptop
652.15
-12.53%
Dell Laptop
568.62
-9.34%
25” Monitor
225.11
+0.09%
Mousepad
1.15
-8.77%

Database Schema

Tables

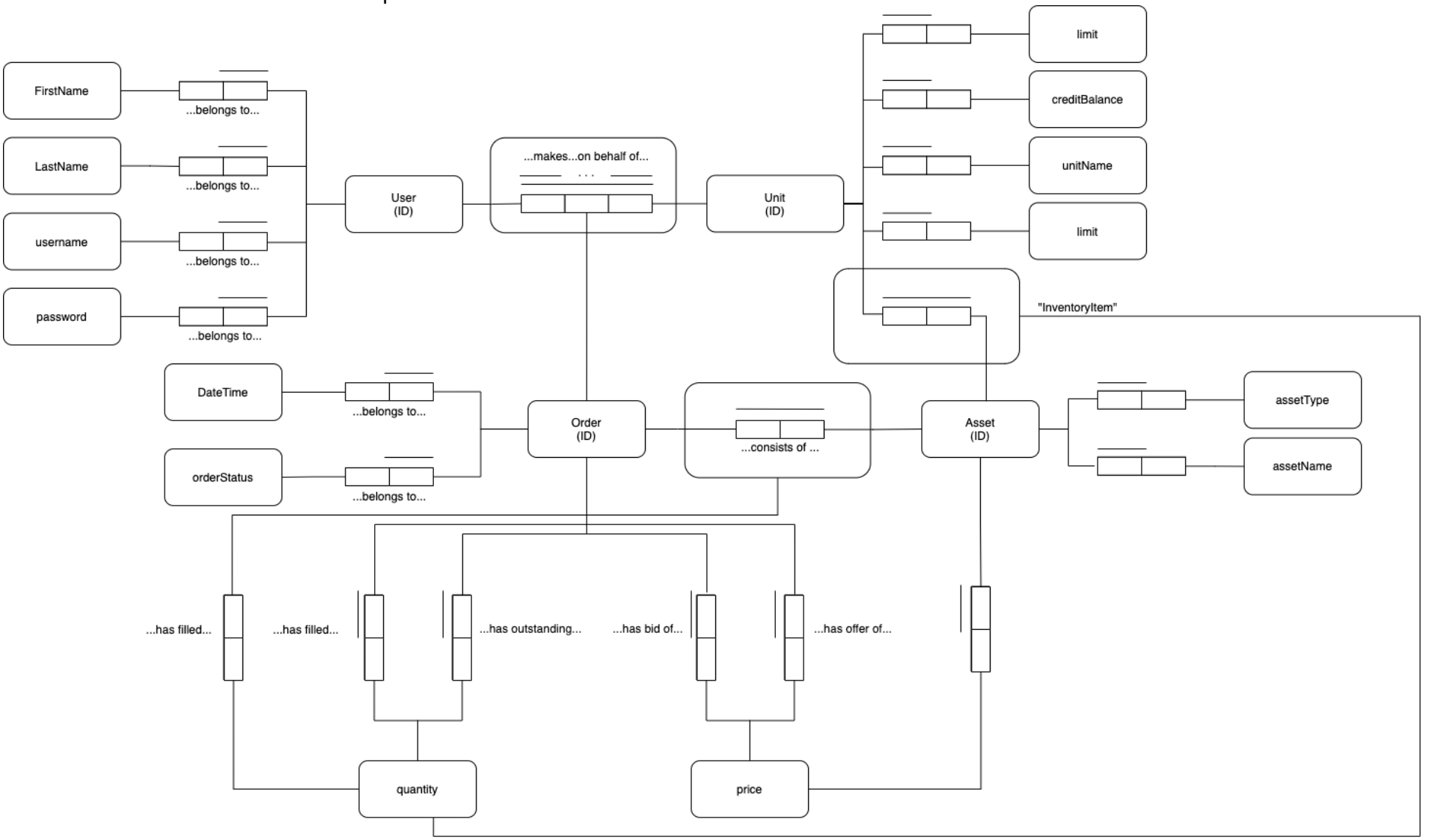
Below is a table of information stored within each of the tables within the database. It also shows the relevant constraints of each of the variables used.

Table	*Description*	*Variable*	*Data Type*	*Constraints*
Users	*This table is used to store the users within the system. Each user is provided with a unique username and password. Along with this, it is required that the user be associated with a specific unitID, to ensure operations regarding orders and buying and selling are effective.*	userID	Int	Primary Key, unique, mandatory
		username	string	AlternateKey, Mandatory, unique, mandatory
		firstName	String	
		lastName	String	
		unitID	Int	Foreign Key, mandatory
		Password	string	Mandatory, encrypted
Units	*The purpose of the units table is to collate the information of each unit within the organisation, and provide relative information required for the system – this includes assigning the unitID with a unit name, credit balance and limit to the credits a unit can use.*	unitID	Int	Primary Key, unique, mandatory
		unitName	string	Mandatory
		creditBalance	Double	Mandatory
		limit	Double	mandatory
Assets	*Assets are used in various functions within the system.*	assetID	Int	Primary Key, mandatory
		assetName	String	Mandatory
		currentPrice	Double	
		assetType	string	mandatory
Orders	*Each order that a user submits will be added to the order table. In doing so, it would also assign an order ID to it. For each order, as they are processed before being fulfilled therefore requires the use of the variables order status, quantity filled and outstanding. This assumes an order could go through*	orderID	Int	Primary Key, mandatory
		userID	Int	Foreign Key, mandatory
		dateTime	datetime	Mandatory
		assetID	int	Foreign Key, mandatory
		orderStatus	string	Mandatory
		orderType	Enum {BUY, SELL}	mandatory
		quantityFilled	Int	Mandatory
		quantityOutstanding	Int	mandatory
		orderID	Int	Primary Key, mandatory

Trade History	***Each completed and successful trade that is conducted would be stored in a trade history table. Each unit would have different past trade orders, and these should be outputted to the user based on their associated userID.***	userID	Int	Foreign Key, mandatory
		dateTime	datetime	Mandatory
		assetID	int	Foreign Key, mandatory
		orderStatus	string	Mandatory
		orderType	Enum {BUY, SELL}	mandatory
		quantityFilled	Int	Mandatory
		quantityOutstanding	Int	mandatory

ORM Diagram

A first iteration of the ORM was developed to visualise the connections between the tables.



Database Design Diagram

As shown in the diagram below, the design of the database is displayed. As shown, it highlights the relationships between each of the variables within the tables.

Insert the many to many etc. relationships

Relational Schema

Users {userID, username, firstName, lastName, unitID, password}

Units {unitID, unitName, creditBalance, limit}

Assets {assetID, assetName, currentPrice, assetType}

Order {orderID, userID, dateTime, assetID, orderStatus, quantityFilled, quantityOutstanding}