**Project Report:** VirtualStockExchange

This is a game that simulates a stock exchange.

**Required Installs:**

(run these commands on your linux terminal)

-install python

-install pip

*sudo apt-get install python3-pip*

-install flask

*pip install flask*

-install mysql

*sudo apt-get install mysql-server libmysqlclient-dev*

-miscellaneous

*pip install flask-mysqldb*

*pip install Flask-WTF*

*pip install passlib*

**Database setup:**

(run these commands on your mysql terminal)

*CREATE DATABSE VSE;*

*USE VSE;*

-create users table

*CREATE TABLE users(id INT(11) AUTO\_INCREMENT PRIMARY KEY, name VARCHAR(100), lastname VARCHAR(100), email VARCHAR(100), username VARCHAR(100), password VARCHAR(100), cash INT(11) DEFAULT 1000);*

-create stocks table

*CREATE TABLE stocks(stock\_id INT(11), ticker VARCHAR(5) PRIMARY KEY, company VARCHAR(100), avail\_volume INT(11), price INT(11));*

-create transactions table

*CREATE TABLE transactions(trans\_id INT(11) AUTO\_INCREMENT PRIMARY KEY, ticker VARCHAR(5), volume INT(11), type VARCHAR(10), current\_price INT(11), trans\_date TIMESTAMP DEFAULT CURRENT\_TIMESTAMP, id INT(11));*

-insert values into all stocks table

*INSERT INTO stocks VALUES(1, 'AZN', 'Amazon', 500, 20),(2, 'ALE', 'Apple', 300, 15),(3, 'FBK', 'Facebook', 400, 15), (4, 'GLE', 'Google', 500, 15), (5, 'MST', 'Microsoft', 400, 10), (6, 'NFX', 'Netflix', 200, 5), (7, 'NKE', 'Nike', 100, 5), (8, 'PSE', 'Porsche', 100, 5), (9, 'SBS', 'Starbucks', 100, 5), (10, 'ZRA', 'Zara', 200, 10);*

-add trigger to keep stock prices positive

*DELIMITER //*

*CREATE TRIGGER keepPricePositive BEFORE UPDATE ON stocks FOR EACH ROW BEGIN IF NEW.price <= 0 THEN SET NEW.price = OLD.price; END IF; END//*

*DELIMITER ;*

-event scheduler to change stock prices

*SET GLOBAL event\_scheduler = ON;*

*CREATE EVENT increasePrice ON SCHEDULE EVERY 5 SECOND DO UPDATE stocks SET price = price + rand()\*(0.1\*price);*

*CREATE EVENT decreasePrice ON SCHEDULE EVERY 5 SECOND DO UPDATE stocks SET price = price - rand()\*(0.1\*price);*

*SET GLOBAL event\_scheduler = OFF;*

**Features:**

**REGISTER/LOGIN:**

-User can register to the system if not already registered. The password is encrypted using SHA1.

*INSERT INTO users(name , lastname, email, username, password) VALUES(%s, %s, %s, %s, %s)", (name, lastname, email, username, password)*

-The user can then login using his/her registered username and password.

*SELECT \* FROM users WHERE username = %s", [username]*

**VIEW / BUY FROM ALL AVAILABLE STOCKS:**

-The user can view all the available stocks along with their current costs and available volume under the "All stocks" tab.

*SELECT \* FROM stocks ORDER BY company*

-The user can enter the number of stocks(> 0) he/she wants to buy and hit "Buy".

-if the user doesn't have enough cash(the user's total available cash is displayed at the top right corner of the page)

-the transaction is blocked.

-else

-the user's cash is correspondingly reduced.

-the volume of stocks bought is subtracted from the all stocks table.

-the transcation is recorded.

*UPDATE users SET cash=cash-%s WHERE username=%s", (bill, session['username'])*

*UPDATE stocks SET avail\_volume=avail\_volume-%s WHERE ticker=%s", (volume, ticker)*

*INSERT INTO transactions(ticker, volume, type, current\_price, id) VALUES(%s, %s, %s, %s, %s)", (ticker, volume, 'Buy', price, session['id'])*

**VIEW / SELL OWN STOCKS:**

-The user can view all of his/her own stocks along with their current costs and volume under the "My stocks" tab.

*SELECT transactions.ticker, SUM(volume) as myvolume, price as current\_price FROM transactions LEFT JOIN stocks ON transactions.ticker=stocks.ticker WHERE id=%s GROUP BY ticker HAVING myvolume>0 ORDER BY ticker", (session['id'],)*

-The user can enter the number of stocks(> 0) he/she wants to sell and hit "Sell".

-the volume of stocks sold is added to the all stocks table.

-the user's cash is correspondingly increased.

-the transcation is recorded.

*UPDATE users SET cash=cash+%s WHERE username=%s", (credit, session['username'])*

*UPDATE stocks SET avail\_volume=avail\_volume+%s WHERE ticker=%s", (volume, ticker)*

*INSERT INTO transactions(ticker, volume, type, current\_price, id) VALUES(%s, %s, %s, %s, %s)", (ticker, -volume, 'Sell', current\_price, session['id'])*

-The user can also view all his/her own transactions (in reverse chronological order) in the "My transactions" table

*SELECT \* FROM transactions WHERE id=%s ORDER BY trans\_date DESC", (session['id'],)*

**Open Source technologies used:**

**Python-Flask (backend framework):**

Python is an interpreted high-level programming language for general-purpose programming. Here Python is used to run the backend of the website. Python's Flask framework is used to support the development of web applications including web services, web resources, and web APIs. Flask is a micro web framework written in Python and based on the Werkzeug toolkit and Jinja2 template engine. It provide a standard way to build and deploy web applications and aims to automate the overhead associated with common activities performed in web development. It provides libraries for database access, templating frameworks, and session management, and promotes code reuse. Although it targets development of dynamic web sites, it can also be applicable to static websites.

**Bootstrap (frontend framework):**

Bootstrap is a free and open-source front-end library to build responsive, mobile-first projects like websites and web applications. It contains HTML- and CSS-based design templates for typography, forms, buttons, navigation and other interface components, as well as optional JavaScript extensions.

**MySQL (DBMS):**

MySQL is an open-source relational database management system based on Structured Query Language(SQL). MySQL runs on virtually all platforms, including Linux, UNIX, and Windows. Although it can be used in a wide range of applications, MySQL is most often associated with web-based applications and online publishing.