**XAVIER UNIVERSITY ATENEO DE CAGAYAN**

**COLLEGE OF COMPUTER STUDIES**

**DEPARTMENT OF COMPUTER SCIENCE**

**“Game Cosmetics Trading System”**

**In partial fulfilment of the requirements for the course**

**CSCC 20 - Object Oriented Programming**

**Submitted by:**

**Francis James Dela Cruz**

**Richard Graceel Roxas**

**Submitted to:**

**Jessie Christopher Lagrosas**

**October 16, 2019**

**TABLE OF CONTENTS**

**COVER PAGE ……….……….……….……….……….……….……….……….………. i**

**TABLE OF CONTENTS ……….……….……….……….……….……….………. ii**

**INTRODUCTION ……….……….……….……….……….……….……….………. 1**

**ANALYSIS ……….……….……….……….……….……….……….……….………. 2**

REQUIREMENTS ANALYSIS **……….……….……….……….……….………. 2**

USE-CASE ANALYSIS **……….……….……….……….……….……….………. 3**

USE-CASE DIAGRAM **……….……….……….……….……….……….………. 5**

**OOP DESIGN ……….……….……….……….……….……….……….……….………. 6**

CLASS DIAGRAM **……….……….……….……….……….……….………. 6**

SEQUENCE DIAGRAMS **……….……….……….……….……….……….………. 7**

**APPENDICES ……….……….……….……….……….……….……….……….………. 12**

ERD DIAGRAM **……….……….……….……….……….……….……….………. 12**

**INTRODUCTION**

Online Game Cosmetics’ Trading has become rampant in the current age of online gaming. Players that seek to make some profit, want to achieve the cosmetics they like at a lower price, or desire to collect every cosmetic customization a game has to offer actively participate in trading groups and communities online. They spend time searching for the items that they want and try to negotiate with the seller to drag the prices to the lowest possible rate.

One of the games popular among gamers that trade cosmetic items online is DOTA 2. It is currently the most played game on Steam, Valve Corporation’s video game digital distribution service platform. Hundreds of thousands of players worldwide enter battle as one of over a hundred DOTA heroes everyday, and the cosmetics that the game offers enhance the players’ experience and the game’s overall beauty.

The goal of our project is to create a mock inventory system that enables users to trade their DOTA 2 items into balance. Each user can then use their balance to buy the cosmetics that they want from the list of DOTA 2 items that our system offers. Users can buy DOTA 2 items from the system or sell DOTA 2 items into the system guided by the prices that the system provides.

The prices that the system has set is based on the steam community market prices to ensure the legitimacy of our items’ values. The system sets the values of the items being sold to users a little bit higher, and the values of the items sold by the users a little bit lower. This way, the admins of the system can ensure profitable trades everytime and guarantee that the system’s economy doesn’t crumble.

The program just offers a mock trading system. To completely utilize the system for profitable use, pairing it with an online site and a number of bot steam accounts is recommended.

**ANALYSIS**

**Requirements Analysis**

Game Cosmetics Trading System involves scenarios that are related with trading in particular. Thus, it has a number of functional and non-functional requirements.

FUNCTIONAL REQUIREMENTS:

* Users can Login
* Users can View their Inventories
* Users can View the Market
* Users can View Item Prices
* Users can Choose item quality
* Users can Buy items from the market
* Users can Sell items from their inventory

NON-FUNCTIONAL REQUIREMENTS:

* User Balance is deducted when buying items
* User Balance is added when selling items
* Items from User Inventory is removed when sold
* Items from Market Inventory is removed when sold
* Items sold by the Users are added to Market Inventory
* Items bought by the Users are added to User Inventory

**Use-Case Analysis**

Our system only has a single type of actor, the user.

USE CASES:

1. Login

* Actor ~ User
* Pre-condition ~ Account Username and Password Recorded in the Database
* Success Scenario ~ Successful Login

~ Login GUI closes

~ User is redirected to Main GUI

1. Buy Items

* Actor ~ User
* Pre-condition ~ Login

~ Main GUI

~ Sufficient Balance

* Success Scenario ~ Bought Item added to User Inventory

~ Bought Item removed from System Inventory

~ User Balance is deducted

1. Sell Items

* Actor ~ User
* Pre-condition ~ Login

~ Inventory has Items

* Success Scenario ~ Sold Item removed from User Inventory

~ Sold Item added to System Inventory

~ User Balance is added

1. View Balance

* Actor ~ User
* Pre-condition ~ Login

~ Balance loaded from the server

* Success Scenario ~ Upon start-up of the Main GUI, balance is displayed on the upper middle part of the Main GUI

1. View Market

* Actor ~ User
* Pre-condition ~ Login
* Success Scenario ~ Items sold in the system are displayed in the Main GUI with corresponding qualities and quantities

~ User can pick item quality

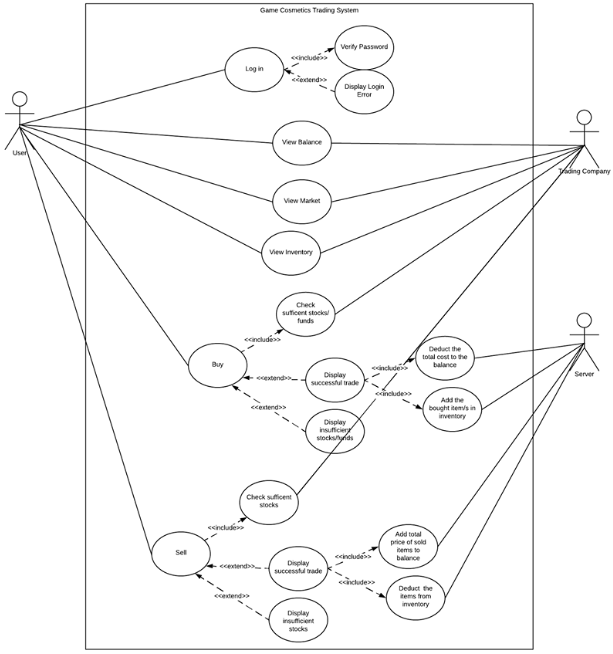
~ Item prices are displayed

1. View Inventory

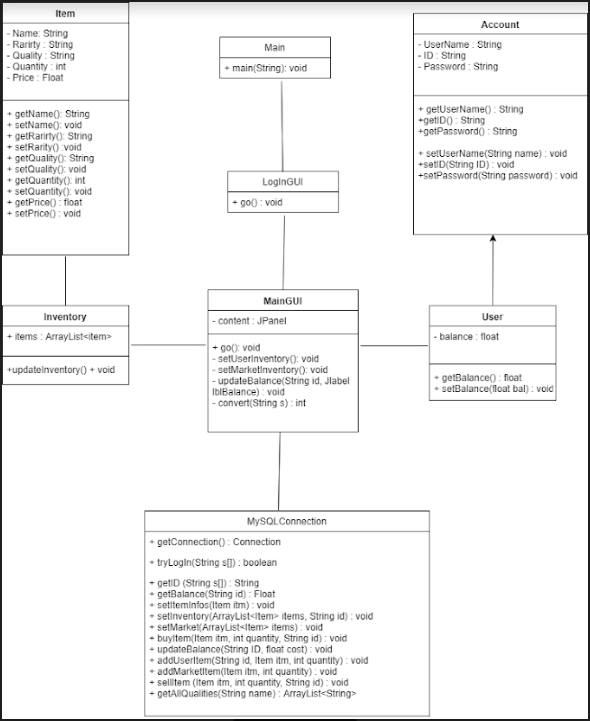
* Actor ~ User
* Pre-condition ~ Login
* Success Scenario ~ Upon successful login, User Inventory is displayed in the Main GUI

~ Items are displayed with corresponding qualities and quantities

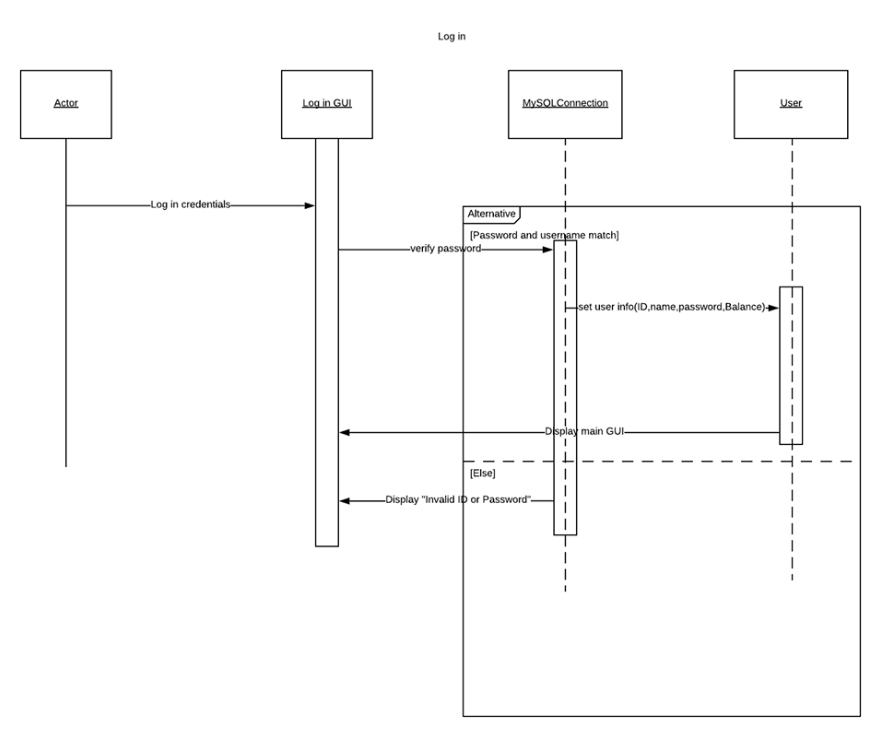
**Use-Case Diagram**

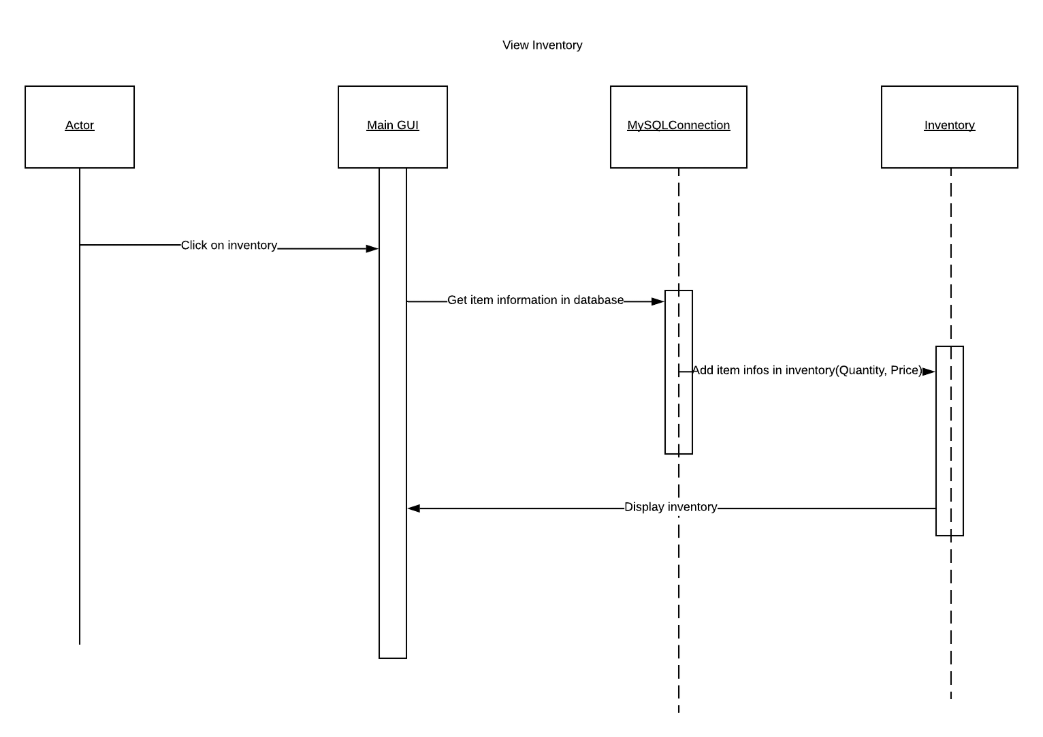
****

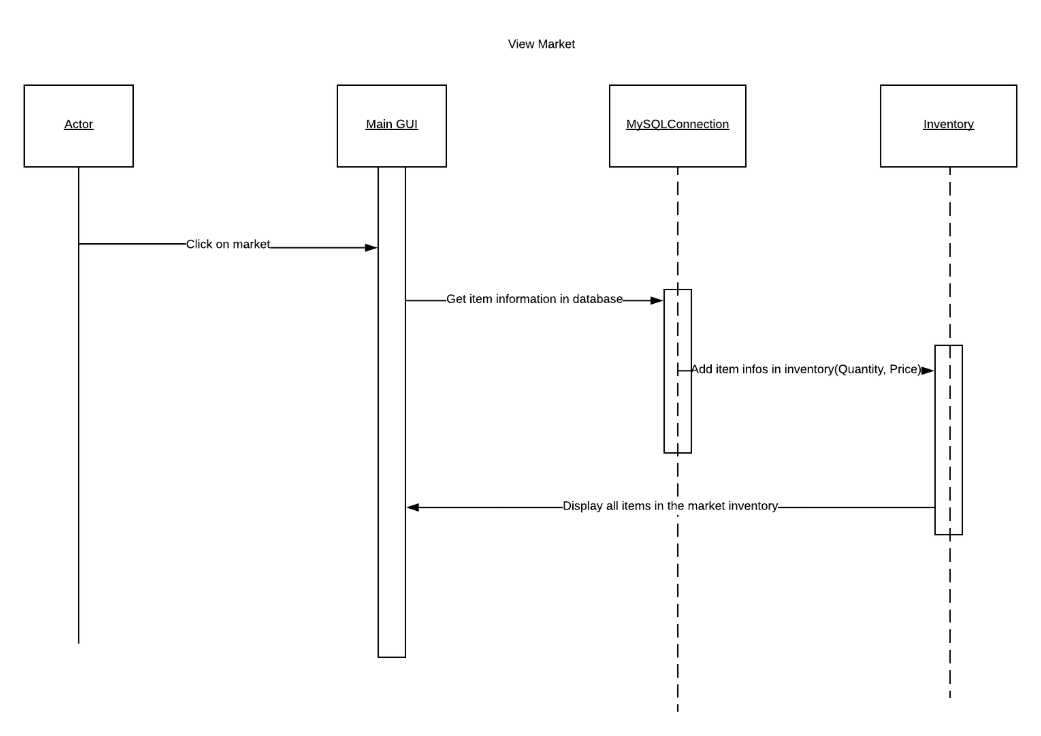
**OOP DESIGN**

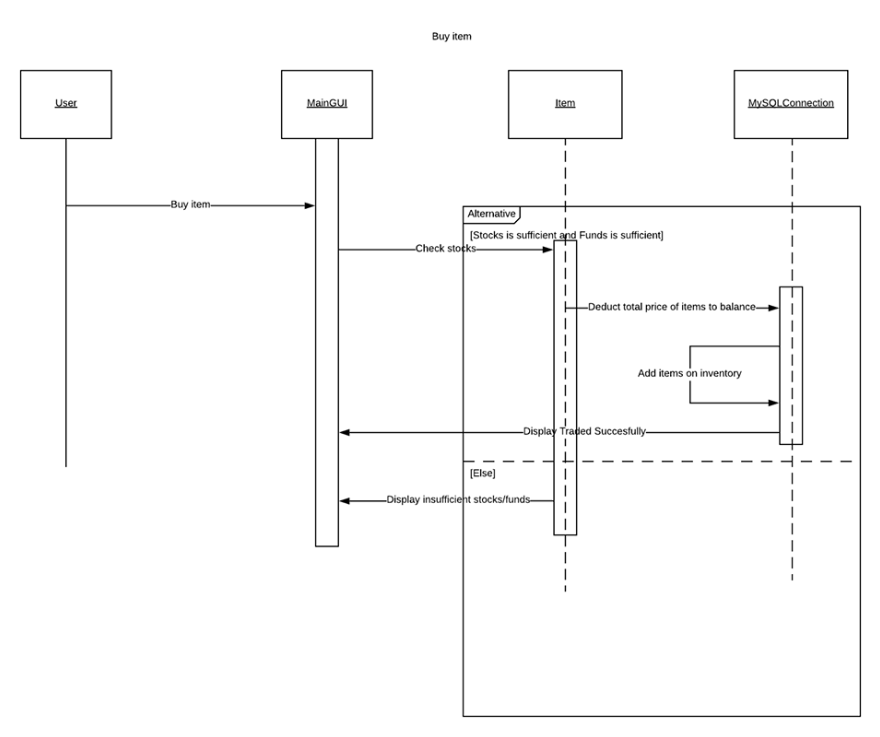
**Class Diagram**

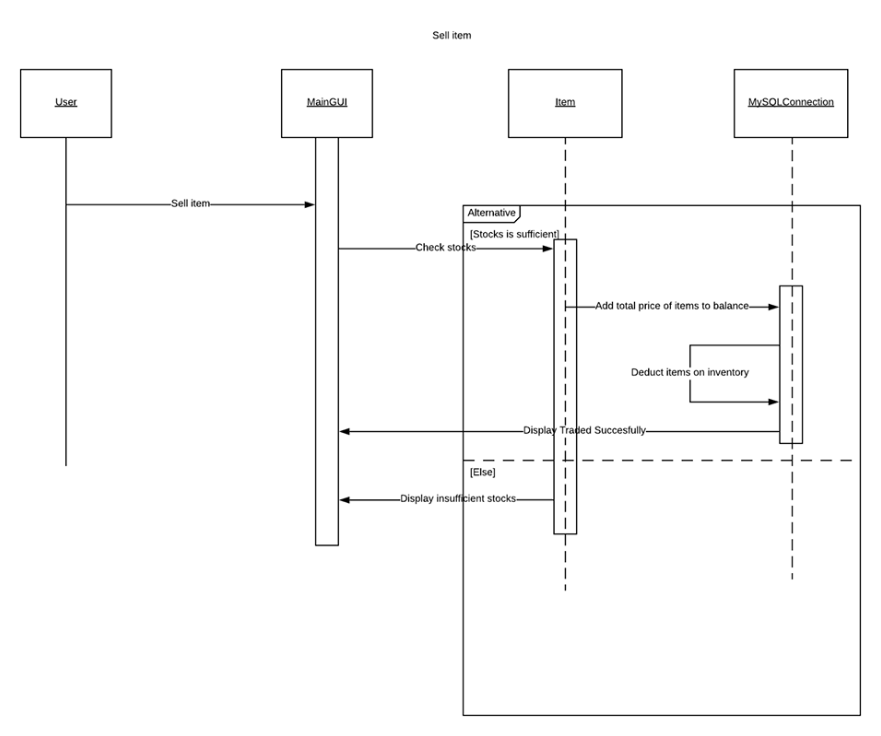
**Sequence Diagrams**











**APPENDICES**

**ERD Diagram**

