Scott Bishop

Andrew Thomas

March 12, 2013

CS174A

Updated Early Project Report

**Integrity Constraints and how to fix them:**

* Customers: No two customer usernames can be the same 🡪If someone makes a new username that already exists, prompt them to choose a different one. Handled by relational schema.
* Accounts: No two customers can have the same id\_Num. Check new id\_Num against all others to make sure it is unique before issuing to customer. Handled by relational schema.
* Accounts: Account balances can never go below 0. Transactions fail if when performing a Market Transaction it makes the account go below 0 (handled by program).
* Accounts: Share balance can never go below 0. Transaction fails if a Stock transaction makes the share balance go below 0 (handled by program).
* Account: Must have initial deposit of $1000 when opening a market account. If no initial deposit, don't let the customer open a market account. (handled by program).
* Stocks: Each actor has only one stock associated with him/her denoted by a symbol. Illegal stock if a stock has two symbols. Handled by relational schema.
* Transactions: Every transaction must have a valid id\_Num associated with it. Make sure the id\_Num exists before making the transaction (handled by program).
* Stocks: Closing and Current Price of stocks can’t be below 0. When adding new stock to the system make sure they are not below 0. If they are, don’t let them be added to the system (handled by program).

CREATE TABLE Customer (username CHAR(10), password CHAR(10), name CHAR(20), state CHAR(2), phoneNum INTEGER, email CHAR(30), tax\_ID INTEGER, PRIMARY KEY (username));

CREATE TABLE Customer (username CHAR(10), password CHAR(10), name CHAR(20), state CHAR(2), phoneNum CHAR(10), email CHAR(30), tax\_ID INTEGER, ssn CHAR(9), account\_ID INTEGER, isAdmin CHAR(1), PRIMARY KEY (tax\_ID));

CREATE TABLE Actors (actor\_ID CHAR(3), current\_Price REAL, name CHAR(20), dob CHAR(20), movie\_Title CHAR(30), role CHAR(10), year INTEGER, contract REAL, PRIMARY KEY (actor\_ID));

CREATE TABLE Stock (tax\_ID INTEGER, shares INTEGER, actor\_ID CHAR(3), FOREIGN KEY (actor\_ID) REFERENCES Actors, FOREIGN KEY (tax\_ID) REFERENCES Customer);

CREATE TABLE Market (tax\_ID INTEGER, account\_ID INTEGER, balance REAL, PRIMARY KEY (tax\_ID), FOREIGN KEY (tax\_ID) REFERENCES Customer);

CREATE TABLE Market\_Transaction ( tax\_ID INTEGER, day CHAR(10), amount REAL, description CHAR(100));

CREATE TABLE Stock\_Transaction ( tax\_ID INTEGER, day CHAR(10), num\_Shares REAL, stock\_Price REAL , description CHAR(100));

CREATE TABLE theDate( aDate CHAR(10));

**Functional Architecture of System**

We designed multiple classes to handle the GUI. A Login GUI that has two check boxes, clicking the sign up box will send the user to enter more info to create an account. If a user is an administrator, he or she can click that button to enter to the AdminView interface. Without clicking either checkboxes, when you login it will send the user to the main GUI. There, one can deposit, withdraw, buy, sell, show balance, show transaction history, price of stock for specific actor, movie information, and movie reviews. Within the AdminView, one can add interest, generate monthly statement, list active customers, generate DTER, customer report, and delete transactions. We have a database class that contains the corresponding queries that are needed in each view.

**Task Divisions**

We pair programed and did everything together. Switching off coding and observer every so often so both of us got a full understanding of the database system.

insert into Customer values('scott', 'sb', 'Scott Bishop', 'CA', '6502244259', 'scott@ucsb.edu', 1111, 121212, 013, 'y');

insert into Customer values('andrew', 'at', 'Andrew Thomas', 'CA', '9166073739', 'andrew@ucsb.edu', 7777, 123456, 012, 'n');

insert into Market values (7777,012,10000);

insert into Market values (1111,12,10000);

insert into Actors values ('SMD', 71.00, 'Michael Douglas', '234 septemp', 'swag', 'actor', 1999, 50000);