Market Data Extractor (MDX): A System to Download Market Data

Raja H. Singh*, Nolan Burfield*, Frederick Harris, Jr.*
*Department of Computer Science and Engineering, University of Nevada, Reno
email: rajas@nevada.unr.edu, nburf@nevada.unr.edu, Fred.Harris@cse.unr.edu

***Abstract*—Stock market data is an important component to any person or entity interested in financial markets. The issue with attaining the information is dependent on cost and/or time. One could browse through a data vendor such as Finance.Yahoo for the necessary data, however this is a very time consuming and a tedious process. Another option would be to purchase the data from a large data vendor at high costs. Both of these options are not cost effective in terms of time and/or money. The means for attaining market data should be both time and cost effective. In this paper we present such a system: Market Data Extractor (MDX).**

MDX is a system that is designed to extract market data and present it both graphically and numerically to the user. The user is not expected to program any part of the system to extract the data. It follows the Model View Controller architecture and keeps the complexity of the system hidden from the user. The user utilizes the system via a graphical interface. The system is user friendly, robust, secure and cross-platform.

***Index Terms*—Technical Analysis; Historical Data; Computational Finance; Stock Market; MySQL; Java;**

I. Introduction

According to the WorldBank, [1] the market capitalization of the United States is close to 19 trillion dollars, and according to [2] over 54 percent of Americans own stocks. Due to the advancement of both technology and emergence of the tech savvy generation we now live in an era where an average user can set up an electronic brokerage account and execute trades by themselves. Even with advancement in the exchange markets, there is a large lack of information accessibility for an average user. Most users either have to have a background in programming or pull data directly from online sources such as Yahoo Finance; the latter being a very tedious and time consuming process.

In this paper we present a system, Market Data Extractor (MDX), that we designed and developed, which allows a person to both visualize and download market data into CSV format. The system is designed to be user friendly and requires no programming from the user, past setting up MySQL on their system. Most users only rely on market data to make trading decisions, while the veterans recommend considering both market and fundamental data prior to making trading decision. Our system only provides market data.

II. Prior Works

Due to the nature of the stock market, information is a well guarded commodity. There have not been many papers published on systems that provide a simplistic means for an average person to pull market data. Most books such as Chan, [3] point towards various databases, but don't provide the means to download the data. For example, for the Historical Databases Table, under the Daily Stock Data, he lists five sources with only Finance.yahoo.com being free; he doesn't provide the code to download the data. Instead he states to download it from the website in the CSV format.

Hilpisch [4] actually provides the code in his book to download historical stock data from Finance.yahoo into a Pandas data-frame. However, to understand or utilize the code, the user must have some background in programming in Python. The Pandas data-frame does allow the user to download the data into a CSV file, but once again the user must know how to program.

There are also websites such as [5] and [6] that provide tutorials and docs to write code to extract historical data free of charge from providers such as Yahoo. However, their target audience are people with programming background. An average user trying to learn from these sources will not be able to do so without a prior understanding of the language(s) these websites are utilizing.

III. System Overview

A. Concept

The system was created with the Model View Control (MVC) architecture in mind. Model does most of the heavy lifting in the back end of the system. It connects with the data source and makes request for historical data, and current data. It also handles the connection to the database and all query executions. The View is the front end (graphical user interface) the user will see and use to interact with the system. The user never sees the model itself, only the view. The controller is the missing link that connects the view (what the user sees and interacts with) to the model (the back end that does all of the heavy lifting). It takes the actions and events from the view and calls/communicates the desired actions to the model.

B. Data Source

All of the data comes from Finance.Yahoo. It is important to note that the data has a 20 minute delay; meaning what it pulls from the source might not be the most current information about the stock. Unfortunately to get an up-to-date information about the market, most data providers charge money.

There are several online sources that provide code to pull market data, such as Markit on Demand, Quandl and Quant

Connect to name a few. However they are geared towards an audience with programming background and provide code that the user can download and execute. As of yet we have not come across a source that provides code for both a back-end system and front end user interface. The systems that do provide up-to-date information with an interface for the user such as the Bloomberg Terminal have subscriptions cost in the thousands [7].

IV. Hardware and Language

This system is written in Java due to it being a cross platform language. For the underlying database, we used MySQL database for the same reasons we used Java. We utilized Eclipse IDE on a Linux system to program the system. All of the libraries (.jars) utilized in the production of this system are open source.

The program was tested on a MacBook Pro with OSX Yosemite. It has a 2.3 GHz Intel Core i7 processor with 8 GB memory and 1600 MHz DDR3. Running on a NVIDIA GeForce GT 650M with 1024 MB memory.

V. Model

A. System Requirements

As stated in section I the user is required to have a MySQL Database installed. The method for installation of MySQL is to use a package installation software such as Homebrew, apt-get, or direct source from MySQL website. Additionally the next step to a MySQL installation is to set a root password for the MySQL database. This root password is a required input from the user when initialing the database for MDX. If a database is not detected by the system a root MySQL password is required to build the database and tables associated to the database. The method of the database build process is furtherer discussed in section V-E.

B. Model Overview

As mentioned earlier, the model or back-end of the system does majority of the heavy lifting. When MDX is initialized, the back-end first opens up the ticker file and downloads all of the companies and ticker symbols in a specified file. We refer to this as the pre-processing step. If the file doesn't exist an error is thrown. However, for the prototype we provided the system with a file so that that user doesn't have to deal with any file related exceptions/errors. The next phase of the system is to connect with the MySQL database and confirm that the historical database exists.

If the database does not exist then it enters the database creation phase where it first creates a historical database, a default user and a password pertaining to the default user. Next it grants the user with privileges to access and modify the database. After creating the database, the system creates the appropriate entities (tables) in the historical database. The next phase after the creation of the historical database and tables is to populate the historical database with all the stocks provided to the system at the booting of the system. After the update, the system launches the graphical user interface and waits for a signal (from the controller) of user action.

However, if the database does exist after the pre-processing step the system enters the compute difference phase. During this phase it checks to see the last stock date in the database and computes the difference between the last stock date and the current day's date. During the computation of the difference, the system does not include the days the market was closed. However if there is a lapse in the historical data (meaning it is missing days of historical data between the last storage date and current date), the system will update the historical data by pulling the missing days' data and storing it in the database. Next the system launches the graphical user interface and then enters the waiting phase as described in the alternate scenario.

Once it receives a command it executes the specified command. If the command it receives is to shutdown, the system terminates. Otherwise it continues on to the data visualization phase. The system executes commands based on the specifications of the command, either related to historical data, current data or current ratios. Once the system is informed which information category the user wants, it moves on to the display phase.

During the display phase, the system has two options either visualize the data or display the numerical data. Please note we refer to displaying the numerical data as "Raw Console Output." The reason behind this choice is that the "view" or user interface chosen to display numerical data can be anything, for MDX it is a JTable. Once the data has been displayed the system moves back to the wait for user action. Please refer to Figure 1 for the flow chart of the model overview.

C. Technical Package

Each component of the MDX is broken down into a separate package. The model classes are all part of a package we refer to as Technical. The following section describes the types of classes in this package and their functionality. The primary classes for each group type are explained, the minor/helper classes are left out due to space constraints.

1) Intialization Classes :

The model is controled through one class we named Technical. At the launch of the system, the main driver calls the technical class by creating a technical object. Upon its instantiation, the class goes through the pre-processing step of pulling all the company names and tickers and storing them for use by the system. Then it instantiates two other classes: the MySQLStocks class and the LoginCredentials class. Both of these classes are subclasses of MySQL class described later in the paper.

When Technical instantiates the MySQLStocks class, it connects with the MYSQL database through JDBC [13] and executes several MySQL queries in a series of try and catch blocks. First it tries to create a pre-specified user and password. In case the user and password already exist, the preceding

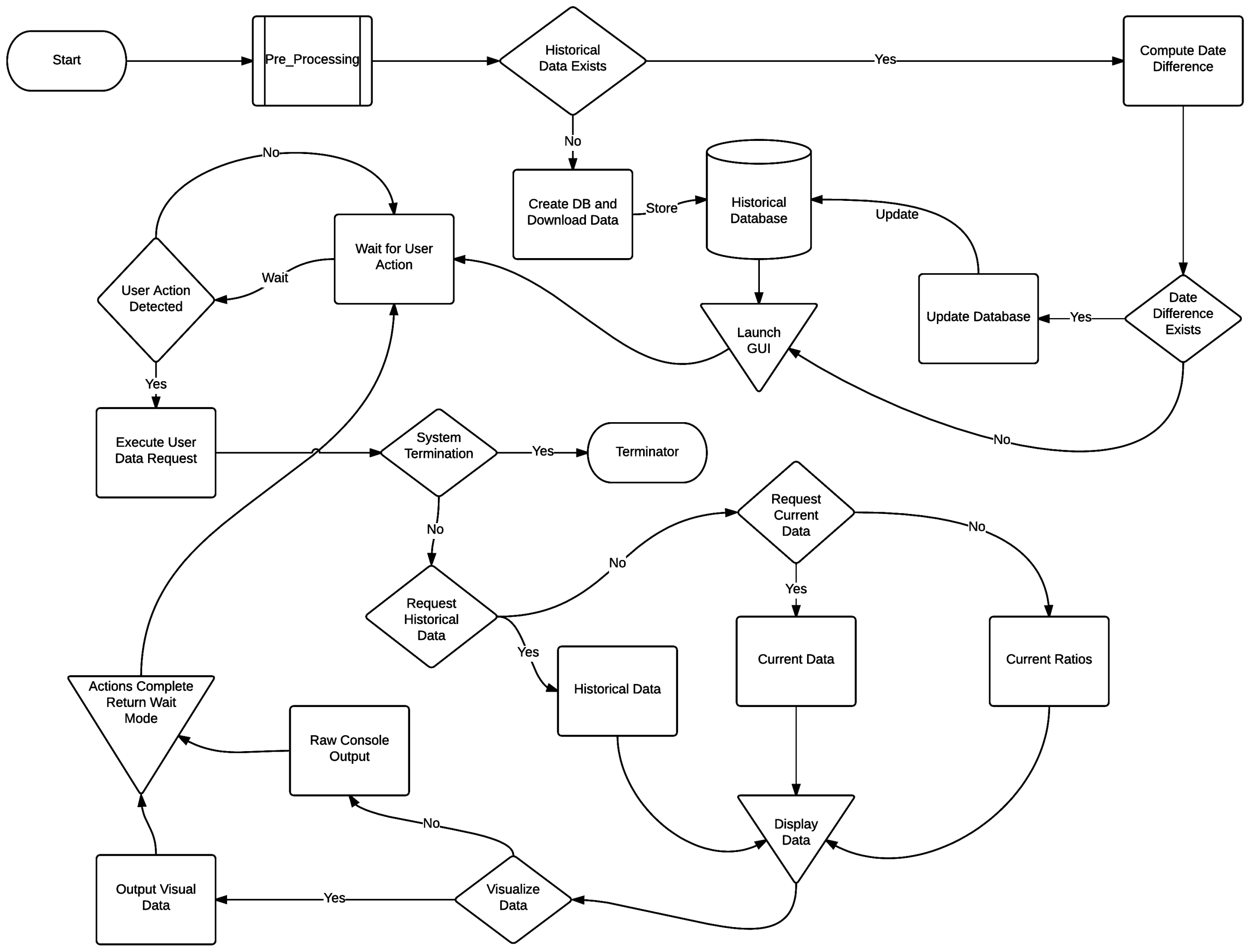


Figure 1. Model Flow Diagram for the Model(backend) component of the MDX system

catch block logs that the user already exists and continues on, otherwise it completes the creation instructions. Next the class tries to create a pre-specified historical database, once again the try/catch ensures that the system continues on and a database does exist before it reaches the next try block in which it attempts to create specified tables based on the design shown in the ERD diagram in Figure 2. Finally the class tries to grant the authority to the created user on the database. The try/catch make sure the system continues on and logs any issues caught. This class also is responsible for all the historical data manipulation in the database.

Next class the Technical class instantiates is the LoginCredentials class. This class, similar to MySQLStocks class, is responsible for the connection to the database and creation of tables. However, the focus of this class is to make sure that there is a users table inside the database. This class also executes a series of try/catche blocks. The main difference is that the purpose of this class is security. This class first creates a specified user and password to access the database (this user is granted the authority to execute MySQL queries). Then it creates the users table that contains a list of users and their salt encrypted passwords (these users are used to validate the login for MDX). Due to the try and catch set up if the table already exists it just logs the SQL exceptions otherwise it creates the user and users table. Next it populates the table with the base, “admin user” that has the ability to create, delete or upgrade privileges for the MDX system. This class uses the PasswordEncryptionService class which is part of the Security package for all of the password encryption, decryption and comparisons. PasswordEncryptionService class is explained later in the paper.

2) Data Collection Classes :

The data collections classes are responsible for pulling and parsing all of the data pulled from Finance.Yahoo. The three main classes are HttpRequest, the RequestStockInfo, RatioSymbolBuilder. All of the data collection classes and their inner classes are static. We decided to go this route so that an object of the class type didn't have to be instantiated. All of the functions can be invoked through the syntax allowed by Java [10] and [11].

The HttpRequest is the class that communicates with the data source and makes the put and get commands. It communicates via the Java.net.URL package and uses the languages containers and various input and buffer streams to extract the data incoming from Finance.Yahoo. This class does not know which website it is connecting with, it just makes the http request under the assumption that the class invoking it handles all of the details such as correct formatting.

The RequestStockInfo class controls the formatting of all the http requests based on the parameters passed to it. This class uses the HttpRequest class to make requests for the data. Within this class are other nested classes written so that the format of the data requested can be either CSV or JSON. The RequestStockInfo class also deals with the type of data being requested based on the function signatures. This class was used to pull historical data, current stock information, and the stock ratios.

The RatioSymbolBuilder class has the sole purpose of passing the appropriate parameters to the RequestStockInfo class based on information it receives from the invoking object. RatioSymbolBuilder class passes the appropriate function parameters to the RequestStockInfo class based on what the invoking object is requesting from the class.

3) Housekeeping Classes :

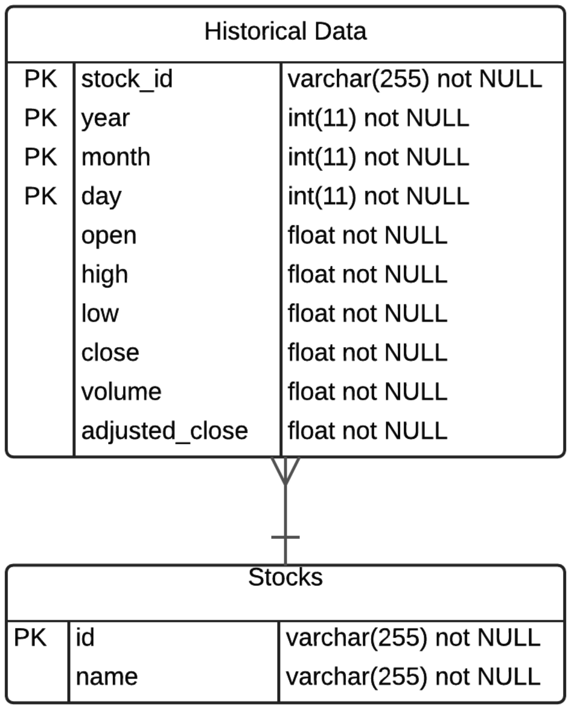
Housekeeping classes are the classes that we wrote for maintaining a log of all the events that take place in the model. These classes essentially log all exceptions thrown by the data provider, database, and any java collection. They also log all of the console outputs related to the stock downloads. For example the log class creates a Log.txt that tracks all of the tickers that have been pulled from Finance.Yahoo. These classes are there so that the user has the ability to look at files to determine download transactions and any other errors that might result while operating MDX.

D. MySQL

MySQL class is part of the SQL package, and is the super class for both MySQLStocks and LoginCredentials classes. The primary function of this class is the communication with the database. This class utilizes the JDBC [13] to send commands to the installed MySQL database. When the class is instantiated, its default constructor establishes the connection with the database. This class contains the functions to execute all MySQL queries. This class's functions are used by both MySQLStocks and LoginCredentials classes for all modifications of the underlying database.

E. Database Schema

The database contains three tables. The first table is stocks, which is made up of two items; the ticker and the company name. The other table stores the historical data. This is made up of the integer fields year, month, and day, separated in order to handle the date without parsing overhead. The float fields in the historical table hold the values of open, high, low, close, volume, and adjusted close, which are discussed in section VI-A. The last pieces of the table establishes the relationship of the database. The stock id will be a string of the stock ticker in order to belong to a stock in the stocks table. Therefore the stocks tab has many historical data rows, and the historical data belongs to only one stock. In order to validate information stored in the database a primary key is fixed to the ticker value in the stocks tables. The historical tab has primary keys fixed to the stock id, day, month, and year, which ensures a stock will not have two entries on the same day. A third table not shown in the ERD is the users that will store the user name, authority clearance, and password related fields mention in section V-F. This table is not shown since it does not relate to storing stock information, but is used for system use validation and security. There are no other tables that interact with this entity.



Historical Data		
PK	stock_id	varchar(255) not NULL
PK	year	int(11) not NULL
PK	month	int(11) not NULL
PK	day	int(11) not NULL
	open	float not NULL
	high	float not NULL
	low	float not NULL
	close	float not NULL
	volume	float not NULL
	adjusted_close	float not NULL

Stocks		
PK	id	varchar(255) not NULL
	name	varchar(255) not NULL

Figure 2. ERD: Relationship between Stocks and Historical Data tables

F. Security

To ensure that only authorized persons can access MDX, we utilized PBKDF2 Password-Based Key Derivation Function 2 (PBKDF2) with SHA-1 hashing algorithm as suggested by NIST [8] and outlined by Orr [9]. The system also generates random sequence of bits (salt) and adds them to the SHA-1 hashed password. This helps protect the system from rainbow table attacks and results in avoidance of password collisions.

Please refer to [8] for more explicit information. PasswordEncryptionService class (located in the authorization package), handles the encryption and decryption of all passwords using the mentioned security measures.

VI. View

The graphical user interface segments the viewable technical market data into three tabs. These tabs are historical data, current data, and stock ratios. The historical data shows a graph of selected companies with different selectable data to be presented. Current data allows the selection of companies, and shows the market data at the time displayed. Stock ratios will display multiple selectable tabs showing various tables containing technical data. Each view tab is designed using Java GridBagLayout due to the information displayed to the user the gridded style layout was the best option for setting up the views. Please refer to [10] for furtherer details on GridBagLayout.

A. Historical Data

The historical data tab was designed to show the user a visual representation of the historical data stored on the database discussed in section V-D. In figure 3 the historical data tab is displayed while graphing three companies based on time and company stock value. The upper left of the figure is a pop up menu to select the companies to graph; currently this value is restricted to only three companies to be graphed at one time. Knowing from the data values pulled about each company there is the option to select what values to show out of open, high, low, close, adjusted close, and volume. These values can be selected with the radio buttons located along the bottom of the screen. The user is also able to adjust the X and Y ranges on the graph by clicking and dragging inside the graph to select a given area. A beginning and ending date for the graphed values are displayed so the user knows exactly what date range is being displayed. The date fields are also selectable and can be adjusted to show more precisely selected date ranges. MDX provides an option for the user to download the historical data from the selected companies into a CSV form to the users desktop with the click of a button.

B. Current Data

The Current Data tab allows the user to select stocks (from a pop-up checklist) and displays the most current stock information. The view shows the most current information for multiple stocks. The data is displayed in a table format via JTable and looks similar to the Current Ratios tab shown in Figure 4. The columns that are shown in the view are listed below with the left most tab listed first and the following tabs moving from left to right:

- Name (Specifies the name of the company)
- Ticker (Specifies the ticker symbol of the company)
- Close (Specifies the closing price of current day, NA otherwise)
- Gain (Specifies the gain for the day (opening minus current price)
- Open (Specifies the price of the ticker at start of day)
- High (Specifies the highest price reached for ticker)
- Low (Specifies the lowest price reached for ticker)
- Volume (Specifies the volume of the trade for the day)

The current data view shows the most current information available, this changes based on the source of the data. Since we used Finance.Yahoo, the data could have a delay of about 20 minutes.

C. Stock Ratios

The Stock Ratios tab is similar to the Current Data tab in appearance. However it has multiple sub-tabs that correspond to the various ratio information pertaining to stocks. Similar to the Current Data tab, the Stock Ratios tab also contains a stock selection button that produces a stock checklist from which the user is able to select the stocks they want to pull the information about. Once the user selects the stock(s) they want to get the information about, the system pulls all of the ratios for the specified stock(s) and populates all of the Stock Ratios sub-tabs with the appropriate information. Please refer to Figure 4 for the appearance of the Stock Ratios Tab. Each sub-tab is described below with the order from left to right as it appears in the Stock Ratios view:

- Dividends (Provides information on Dividends such as Dividend Yield — Dividend Per Share — Dividend Pay Date)
- Averages (Provides information on volume such as 1 Year Target, 200 Day Moving Average, % Change from 200 Day Moving Average)
- Misc Stock Information (Provides miscellaneous information about Stocks such as Order Book, Revenue, Holdings Value)
- 52 Week Pricing (Provides information on 52 Week Pricing such as 52 Week High, 52 Week Low, 52 Week Range)
- Symbol Information (Provides information on Stocks such as Market Capitalization, Float Shares, Shares Owned)
- Volume (Provides information on volume such as Volume, Ask Size, Bid Size)
- Ratios (Provides information on ratios such as EPS, EBITA, PEG)

VII. Controller

The controller segment of the model view controller architecture is designed to interact with the technical model from section V and the views in VI. The controller will grab the data from the technical class and hand it to the graphs or tables in the model. The structure is of the same type as the views; there is the historical data class, current data class, and ratios class. An additional class utilized by the controller and used in all the tabs is the company selection class. This class will handle a button click event to pop up and display a

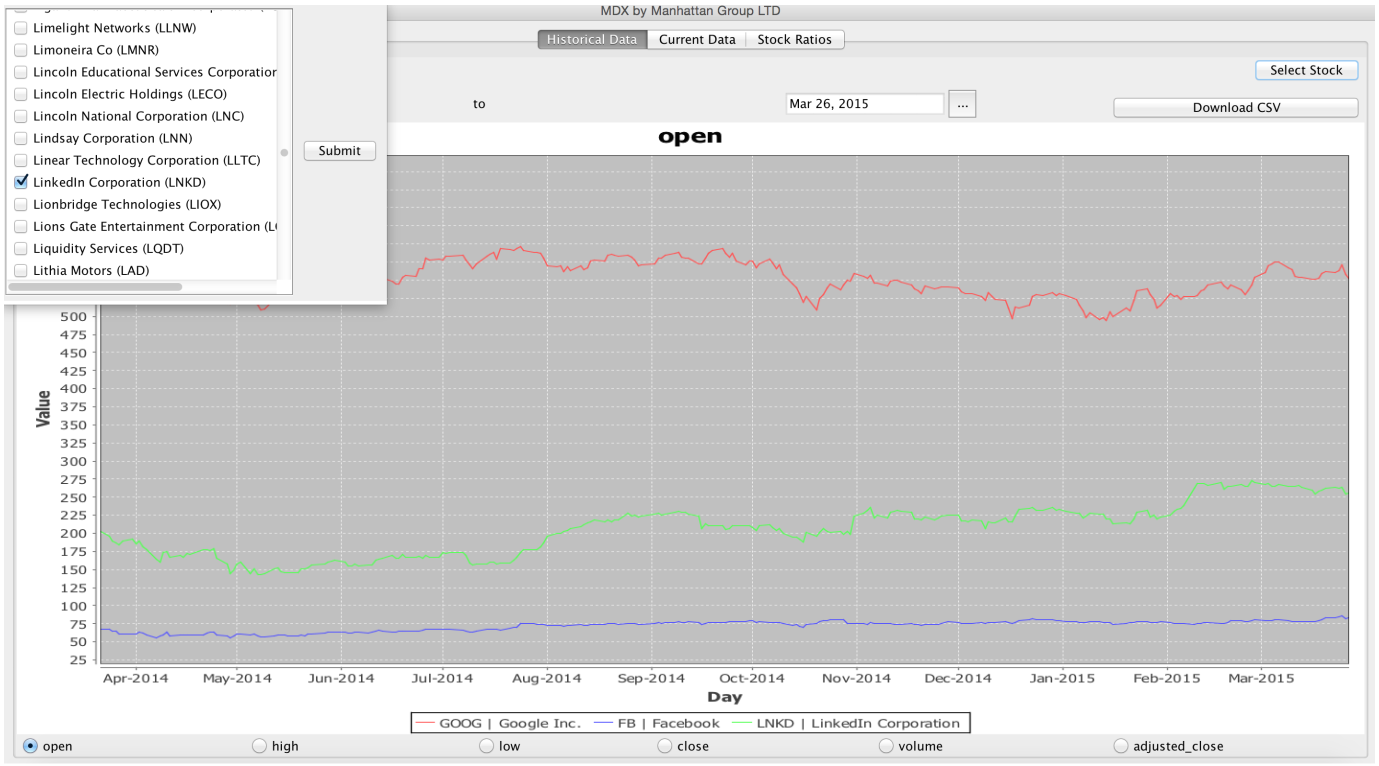

MDX by Manhattan Group LTD
Historical Data
Current Data
Stock Ratios
Select Stock
to
Mar 26, 2015
Download CSV
Submit
Limelight Networks (LLNW)
Limoneira Co (LMNR)
Lincoln Educational Services Corporation
Lincoln Electric Holdings (LECO)
Lincoln National Corporation (LNC)
Lindsay Corporation (LNN)
Linear Technology Corporation (LLTC)
LinkedIn Corporation (LNKD)
Lionbridge Technologies (LIOX)
Lions Gate Entertainment Corporation (L
Liquidity Services (LQDT)
Lithia Motors (LAD)
open
Value
Day
GOOG | Google Inc.
FB | Facebook
LNKD | LinkedIn Corporation
open
high
low
close
volume
adjusted_close

Figure 3. This figure shows the historical data tab graphing three companies' open price, company selection field, and different graphing options.

check-box list of companies given to it. The submission of the pop up window triggers a call back that each utilizing class will setup, and are then given the list of companies that are selected. This controller object is editable by the user since they could upload their own CSV file containing companies to populate into the list. The uploaded CSV is required to be in the format of [ticker, company, name] for the controller to parse correctly.

A. *Historical Data*

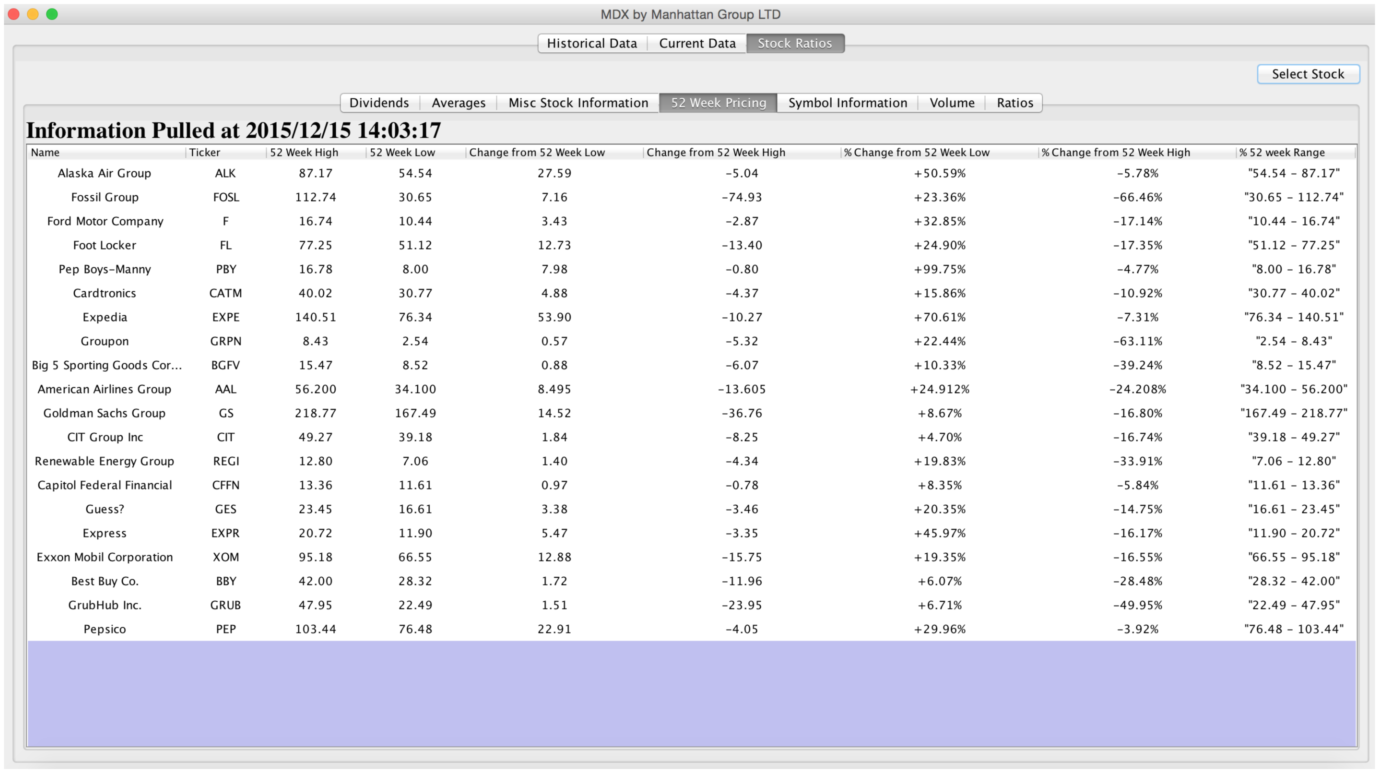
The historical data controller handles the tab of the historical data in section VI-A. It is required for the controller to handle the proper calls to the model to receive the historical data for the selected companies. The controller builds the graph with the data based on the proper data value selected through the radio buttons. Boundary checks are implemented to prevent the user from selecting improper date ranges such as nonexistent dates or a start after end range. These date values are handled in an object of the controller for date selections. The graph is editable by selecting sections of the graph, so the controller has a listener event to grab any graph change and push updates of the date to the displayed date objects. The values provided in historical data are selectable, and therefore must be handled in the controller on updates. All data for the selected companies are not stored when graphing, only the graphed value is saved so another call to the database must be made in order to pull the new values to be shown on the graph. The last key feature the historical data controller provides to the MDX system is the CSV download. This is done with a listener on the download button, and just as the changing of data to be displayed is handled, a call to the database must be made again. The controller will format the CSV with oldest date to current date, comma delimited showing each stored values: open, low, high, close, adjusted close, and volume.

B. *Current Data and Stock Ratios*

Current data controller and the stock ratios controller have similar functionalities. They acquiring the list of selected companies, interact with the model to pull the correct information, and generating the JTables with the given information. Refer to [10] for more on JTables. The controller sends the new table to the view, and calls for a repaint of the screen. The difference between the current data and stock ratios is the need to handle multiple table updates inside stock ratios. In order to make the ratios tab user friendly the controller establishes multiple updates across each of the tabs in the stock ratios to update each table upon company selection.

VIII. Potential Uses

This system has potential uses by three target groups. The first group is composed of the researchers in the computational finance and/or computer science field. The system can be broken down by them and they can interface their own systems directly with the back end (model) and use it to pull market data for their research. The other potential group of users are from the finance field. Most researchers in the finance side don't have much, if any programming experience and



Information Pulled at 2015/12/15 14:03:17

Name	Ticker	52 Week High	52 Week Low	Change from 52 Week Low	Change from 52 Week High	% Change from 52 Week Low	% Change from 52 Week High	% 52 week Range
Alaska Air Group	ALK	87.17	54.54	27.59	-5.04	+50.59%	-5.78%	"54.54 - 87.17"
Fossil Group	FOSL	112.74	30.65	7.16	-74.93	+23.36%	-66.46%	"30.65 - 112.74"
Ford Motor Company	F	16.74	10.44	3.43	-2.87	+32.85%	-17.14%	"10.44 - 16.74"
Foot Locker	FL	77.25	51.12	12.73	-13.40	+24.90%	-17.35%	"51.12 - 77.25"
Pep Boys-Manny	PBY	16.78	8.00	7.98	-0.80	+99.75%	-4.77%	"8.00 - 16.78"
Cardtronics	CATM	40.02	30.77	4.88	-4.37	+15.86%	-10.92%	"30.77 - 40.02"
Expedia	EXPE	140.51	76.34	53.90	-10.27	+70.61%	-7.31%	"76.34 - 140.51"
Groupon	GRPN	8.43	2.54	0.57	-5.32	+22.44%	-63.11%	"2.54 - 8.43"
Big 5 Sporting Goods Cor...	BGFV	15.47	8.52	0.88	-6.07	+10.33%	-39.24%	"8.52 - 15.47"
American Airlines Group	AAL	56.200	34.100	8.495	-13.605	+24.912%	-24.208%	"34.100 - 56.200"
Goldman Sachs Group	GS	218.77	167.49	14.52	-36.76	+8.67%	-16.80%	"167.49 - 218.77"
CIT Group Inc	CIT	49.27	39.18	1.84	-8.25	+4.70%	-16.74%	"39.18 - 49.27"
Renewable Energy Group	REGI	12.80	7.06	1.40	-4.34	+19.83%	-33.91%	"7.06 - 12.80"
Capitol Federal Financial	CFFN	13.36	11.61	0.97	-0.78	+8.35%	-5.84%	"11.61 - 13.36"
Guess?	GES	23.45	16.61	3.38	-3.46	+20.35%	-14.75%	"16.61 - 23.45"
Express	EXPR	20.72	11.90	5.47	-3.35	+45.97%	-16.17%	"11.90 - 20.72"
Exxon Mobil Corporation	XOM	95.18	66.55	12.88	-15.75	+19.35%	-16.55%	"66.55 - 95.18"
Best Buy Co.	BBY	42.00	28.32	1.72	-11.96	+6.07%	-28.48%	"28.32 - 42.00"
GrubHub Inc.	GRUB	47.95	22.49	1.51	-23.95	+6.71%	-49.95%	"22.49 - 47.95"
Pepsico	PEP	103.44	76.48	22.91	-4.05	+29.96%	-3.92%	"76.48 - 103.44"

Figure 4. This figure shows the stock ratios tab with the fifty-two week pricing information for multiple selected companies.

usually rely on systems such as SAS or VBA for their statistical/financial research. For them the system could be used as is. They will be able to download the historical data using the "Download CSV" button on the graphical user interface. The third group of users are the average consumers who can use the system to visualize and compare the stock performances using the graphical user interface (view component).

IX. Conclusion and Future Work

In this paper we presented MDX, a system that provides its users with a graphical user interface to download market data from Finance.Yahoo in an efficient and user friendly manner. The system follows the MVC architecture and is created so that the user never has to type any code to pull the data. There is a lot of potential to further enhance the system.

One of the possible areas to improve the system on is to thread the download process so that multiple companies' information can be pulled simultaneously thus decreasing the system lag. Thread safety was one of the key concerns for us we when we were choosing the the various components of the system. For example, we chose java collections that are thread safe, the database which is thread safe and so on...

There is also potential for alternate types of charts for the historical data instead of just time-series, for example candlestick charts would be very useful for data visualization. However, after asking faculty in the financial department, we were informed that candlestick charts have a very niche user base, it is more useful to day traders. However we can add other charts that have the ability to visualize stock correlations.

Acknowledgment

This material is based in part upon work supported by the National Science Foundation under grant no. IIA-1301726. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the National Science Foundation.

References

[1] "Market Capitalization of Listed Companies (current US$)." Market Capitalization of Listed Companies (current US$). N.p., n.d. Web. 14 Dec. 2015. http://data.worldbank.org/indicator/CM.MKT.LCAP.CD.

[2] "In U.S., 54% Have Stock Market Investments, Lowest Since 1999." Gallup.com. N.p., n.d. Web. 14 Dec. 2015. http://www.gallup.com/poll/147206/stock-market-investments-lowest-1999.aspx.

[3] Chan, Ernest P. "Backtesting." Quantitative Trading: How to Build Your Own Algorithmic Trading Business. Hoboken, NJ: Wiley, 2008. 30-67. Print.

[4] Hilpisch, Yves. "Financial Time Series." Python for Finance. Beijing: O'Reilly, 2014. 137-71. Print.

[5] Elias, Kelly. "Download Stock Ticker Symbols." Jarloo. N.p., 03 May 2015. Web. 02 Oct. 2015. http://www.jarloo.com/download-stock-ticker-symbols/.

[6] "Yahoo Query Language (YQL)." YQL. N.p., n.d. Web. 2 Sept. 2015. https://developer.yahoo.com/yql/.

[7] Seward, Zachary M. "This Is How Much a Bloomberg Terminal Costs." Quartz. N.p., 15 May 2013. Web. 15 Dec. 2015. http://qz.com/84961/this-is-how-much-a-bloomberg-terminal-costs/.

[8] Turan, Meltem S., Elaine Barker, William Burr, and Lily Chen. "Recommendation for Password-Based Key Derivation Part 1: Storage Applications." NIST Special Publication 800-132 (2010): 1-14. NIST Computer Security Resource Center. National Institute of Standards and Technology, Dec. 2012. Web. 05 Dec. 2015. http://csrc.nist.gov/publications/nistpubs/800-132/nist-sp800-132.pdf.

[9] Orr, Jerry. “Jerry on Java.” : Secure Password Storage. N.p., 16 May 2012. Web. 05 Dec. 2015. http://blog.jerryorr.com/2012/05/secure-password-storage-lots-of-donts.html.

[10] “The Java Tutorials.” The Java Tutorials. Oracle, n.d. Web. Fall 2015. https://docs.oracle.com/javase/tutorial/.

[11] Sierra, Kathy, and Bert Bates. Head First Java. Sebastopol, CA: O’Reilly, 2005. Print.

[12] Bodnar, Jan. “MySQL Java Tutorial.” MySQL Java Tutorial. Zetcode, n.d. Web. 4 Oct. 2015. http://zetcode.com/db/mysqljava/.

[13] “JDBC Tutorial.” Www.tutorialspoint.com. N.p., n.d. Web. 13 Oct. 2015. http://www.tutorialspoint.com/jdbc/.

[14] Koirala, Shivprasad, and Marla Sukesh. “Learn MVC (Model View Controller) Step by Step in 7 Days Day 1.” CodeProject. N.p., 13 Feb. 2015. Web. 03 Nov. 2015. http://www.codeproject.com/Articles/207797/Learn-MVC-Model-View-Controller-step-by-step-in.

[15] “SWING Tutorial.” Www.tutorialspoint.com. N.p., n.d. Web. 26 Oct. 2015. http://www.tutorialspoint.com/swing/.

[16] Zukowski, John. The Definitive Guide to Java Swing. England: APress, 2005. Print.