Bloomberg Functionality Replication in Quartz

A Major Qualifying Project

Submitted to the Faculty of
Worcester Polytechnic Institute
in partial fulfillment of the requirements for the
degree in Bachelor of Science in
Management Engineering
by
Julia Alvidrez
and in
Computer Science
by
Daniel Tocco

WPI

January 8, 2014
Bank of America
Project Advisors:
Professor Micha Hofri - Computer Science
Professor Kevin Sweeney - School of Business

Abstract

Bloomberg Professional is a third party application used by the trading staff, middle and back office users, and developers at Bank of America. It is used for many different functions which include viewing market data, conducting electronic trading with customers, and reviewing historical data. For this project, we will be developing an application that imitates the Bloomberg DES, YAS, and CDSW functions using Bank of America's in-house platform, Quartz. The application will allow Bank of America to validate the quality of in-house reference data and risk analytics as well as increase the number of employees who have access to this data due to licensing restrictions on Bloomberg Professional.

Acknowledgements

We would like to acknowledge the following people whose efforts helped us complete our project (in alphabetical order):

Professor Jon Abraham	Worcester Polytechnic Institute
Paul Ashby	BAML - Programmer Prof MKTS (Mgr)
William Carroll	BAML - Tech Project Sr Manager
Stefano Cattani	BAML - Programmer Prof MKTS
Andreas Clara	BAML - Programmer Prof MKTS
Michael Dalgado	BAML - Programmer Prof MKTS
Professor Arthur Gerstenfeld	Worcester Polytechnic Institute
Professor Micha Hofri	Worcester Polytechnic Institute
Andy Hudson	BAML - Service Delivery Consultant
Richard Jervis	BAML - Senior Trading Strategist I
Christopher Lawson	BAML - Service Delivery Consultant
Cristina Malenchino	BAML - Programmer Prof MKTS (Mgr)
Selina Pavan	BAML - Programme Manager
Professor Kevin Sweeney	Worcester Polytechnic Institute

Executive Summary

Bank of America is currently trying to replicate all of the third party applications and infrastructure used by the front office using their in-house platform, Quartz. These applications often have expensive subscriptions. Therefore, Bank of America has to limit the number of subscriptions they purchase. The replication of these applications will increase the number of employees who have access to the data displayed by these third party applications. The replication will also assist in validating the in-house calculations and analytics in Quartz by comparing the Quartz results to the results of the application being replicated.

The goal of this project is to replicate Bloomberg Professional Terminal (BBG) screens in Quartz. This application will only use Bank of America's database, known as Sandra, to obtain the market data - there will be no connection to any market data system.

Bloomberg Professional is a third party application used by Bank of America's trading staff, middle and back office, and developers. BBG provides analytic tools such as statistical & comparative analysis and pre- to post-trade analysis which allows users to watch trends, validate ideas, and generate value. Although the subscription to Bloomberg Profession is very expensive, $20,000 per year, traders often feel "out of the market" if they use another application such as Thomson Reuters.

For this reason, Bank of America is replicating the BBG screens in order to mimic the functionality of the screens without the subscription. The following are the BBG screens being replicated for this project:

- Bond Security Description Screen (Bond DES)
- Yield and Spread Analysis Screen (YAS)
- Credit Default Swap Security Description Screen (CDS DES)
- Credit Default Swap Valuation Screen (CDSW)

The DES screens provide description information of different securities including bonds and credit default swaps. The editable and auto-calculating fields of the YAS screen allow users to analyze the price of a bond. Users are able to create and value credit default swaps using the CDSW screen.

When completing the project, we followed an iterative development approach in order to continually receive feedback to improve our application. The following are the steps we followed (Bank of America, 2013):

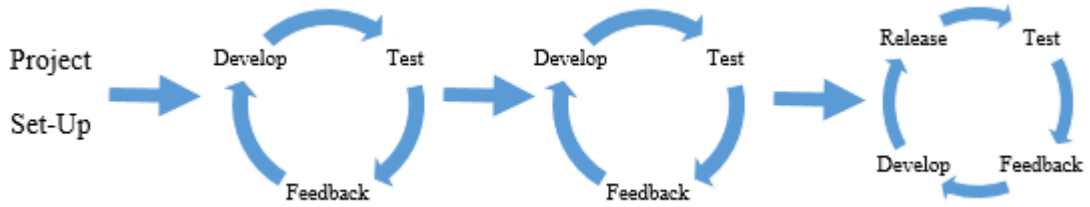

Figure 1. Iterative Development Approach

As part of the project set-up, we completed the tutorials provided by Bank of America to familiarize ourselves with Python and Quartz.

Once we felt comfortable using Quartz, we started developing the User Interface (UI) of the application. We decided to use the Model-View-Controller (MVC) design pattern based on other Quartz coding examples and past experiences. The UI framework we developed became known as the View. As is necessary when using the MVC design pattern, we created the View and Model simultaneously; whenever a field was created in the view its corresponding value was defined in the Model.

We tested the results of the Qzap Application by comparing them to the Bloomberg screens. If the Qzap Application and BBG matched we were confident that the Model was obtaining the correct data from the database and the analytics were performed correctly.

To receive feedback we would have code reviews with Stefano Cattani, Programmer Prof MKTS, after a major event, such as the completion of a screen. During these informal walkthroughs of the code, Stefano would make recommendations and suggestions for improvement. We also demoed all of the screens to Richard Jervis, Senior Trading Strategist I, to receive suggestions on improvement for the screens and feedback on the UI.

Once the code was complete, it went through a formal code review were the entire code was inspected. The reviewer ensured there was useful documentation, good coding practices were followed, and the code ran without any obvious bugs. Once the quality of the code was up to the expected standards, the reviewer pushed the code into production which meant the code was live. This meant that employees at Bank of America would now have access to the Bond DES, YAS, CDS DES, and CDSW replicated screens in Quartz.

Table of Contents

Abstract i
Acknowledgements ii
Executive Summary iii
Table of Figures viii
List of Acronyms and Key Terms x
1.0 Introduction 1
2.0 Motivation behind the Project 2
3.0 Literature Review 3
3.1 Financial Terms 3
3.1.1 Bonds 3
3.1.2 Credit Default Swaps 4
3.2 Bloomberg Professional 5
3.2.1 Bloomberg DES Screens 6
3.2.2 Bloomberg YAS Screen 8
3.2.3 Bloomberg CDSW Screen 9
3.3 Programming Practices and Terms 10
3.3.1 Development 10
3.3.2 Factory Methods 11
3.3.3 Design Pattern – Model-View-Controller 12
3.3.4 Directed Acyclic Graphs 12
3.4 About Quartz 14
3.4.1 Python 14
3.4.2 Quartz Development Process 14
3.4.3 QZDesktop 15
3.4.4 Sandra 19
4.0 Methodology 20
4.1 Project Set-up 21
4.1.1 The Quartz Project for Beginners – Getting Started 21
4.1.2 Meetings with Sponsor 21
4.2 Development 21
4.2.1 Development of the Overall Model 21
4.3 Testing 24
4.4 Feedback 24

4.4.1 Code Review with Stefano 24
4.4.2 Interview with Richard Jervis 24
4.5 Release 25
5.0 Results 25
5.1 Qzap Application 25
5.1.1 Bond DES Screen 26
5.1.2 YAS Screen 28
5.1.3 CDS DES Screen 29
5.1.4 CDSW Screen in the Qzap Application 31
5.2 Challenges 32
5.2.1 DES (Bond & CDS) Screen 32
5.2.2 YAS Screen 33
6.0 Discussion 35
6.1 Developing using the Agile Method 35
6.1.1 Developing and Overall Model 35
6.1.2 Build a Features List 35
6.1.3 Plan by Feature 36
6.1.4 Design by Feature 36
6.1.5 Build by Features 36
6.2 Qzap Application Features 37
6.2.1 Error Message 37
6.2.2 Switching between Screens 37
6.2.3 Qzap URL 38
7.0 Conclusion 39
8.0 Appendix i
I. Notes from Code Review with Stefano Cattani 2:30 PM 11/8/2013 i
II. Notes from Meeting with Christopher Lawson and Andy Hudson 9:00 AM 11/15/2013 ii
III. Notes from Code Review with Stefano Cattani 11/28/2013 iii
IV. Notes for Meeting with Richard Jervis 3:00 PM 12/3/2013 iv
V. To – Do List as of 12/6/2013 ix
VI. To – Do List of 12/16/2013 xiv
Bibliography 1

Table of Figures

Figure 1. Iterative Development Approach iv
Figure 2. Credit Default Swaps 5
Figure 3. Bloomberg Bond DES Screen 7
Figure 4. CDS DES Screen 8
Figure 5. YAS Screen shot 9
Figure 6. CDSW Screen Shot 10
Figure 7. Feature-Driven Development Process 11
Figure 8. Model-View-Controller Pattern 12
Figure 9. Car Transform Hierarchy 13
Figure 10. Car Transform Graph 13
Figure 11. Quartz Development Process 15
Figure 12. QZDesktop Screen Shot 16
Figure 13. QZDev Screen Shot 17
Figure 14. Bond DES Screen Shot 18
Figure 15. Powwow Screen Shot 19
Figure 16. Sandra Screen Shot 20
Figure 17. Iterative Development Approach 20
Figure 18. Qzap Application 26
Figure 19. Qzap Application Bond DES Screen 27
Figure 20. Qzap Application with ISIN USN9365BL23 DES Screen 28
Figure 21. Qzap Application YAS Screen 29
Figure 22. Qzap Application CDS DES Screen 30
Figure 23. Qzap Application CDSW Screen 32
Figure 24. Delegate Functions used in the YAS Screen 33
Figure 25. Delegate Functions used in the YAS Screen 34
Figure 26. "Bond Not Found" Error Message 37
Figure 27. Qzap Application YAS Screen 38
Figure 28. Qzap Application Screen 39
Figure 29. Bond DES Screen v

Figure 30. YAS Screen shot..... vi
Figure 31. CDS DES Screen Shot..... vii
Figure 32. CDSW Screen Shot viii

List of Acronyms and Key Terms

Acronym	Description of the Acronym
Accr	Accrued
Act/360	Day Count Convention
Amrt	Amortization
ASW	Asset Swap Spread
BBG	Bloomberg
BBID	Bloomberg ID
Bps	Basis Points
CDS	Credit Default Swap
CDSD	BBG Page Code for CDS Default Settings
CDSW	Credit Default Swap Valuation
CG	Capital Gain
CoCo	Contingent Convertible Bond
Cpn Freq	Coupon Frequency
Credit Dev	Credit Development
CUSIP	National Securities Identification Number for products issued from both the United States and Canada
Def Exposure	Default Exposure
DES	BBG Page Code for Security Description
Dsc Curv	Discount Curve
DV01	Dollar Duration
G-Spread	Interpolated Bong Spread to Government
GUI	Graphical User Interface
I-Spread	Interpolated Bond Spread to Swap Curve
IR	Interest Rate
ISDA	International Swaps and Derivatives Association
ISIN	International Securities Identification Number
Mmkt	Money MarkIT Equivalent Yield
Mty	Maturity
OAS	Option Adjusted Spread
Pay AI	Whether or not Accrued Interest is Paid
Pts Upf	Points Upfront
Rec Risk	Recovery Rate Risk
RED	Reference Entity Database - a product supplied by the market used throughout the industry
REF	Reference
Start Cnst	Started Constituents Count – number of constituents the index started with
TITIM	Telecom Italia ticker
UI	User Interface
YAS	BBG Page for Yield and Spread Analysis
Z-Spread	Zero-volatility spread

1.0 Introduction

Bank of America is a banking corporation which provides financial and investment services to consumers, small businesses, and corporations. Currently, Bank of America has approximately fifty-three million customers in more than forty countries throughout Europe, the Middle East and Africa, Asia Pacific and the Americas (About Bank of America, 2013).

Quartz is Bank of America's in-house platform. It is a Python based framework that was developed in order for programmers to create applications to rapidly respond to the needs of its users. Some of the uses of this platform are for market data, analytics, trades and risk measures. Quartz core components are QzDev, Powwow, and the UI Framework.

The trading staff, middle and back office staff, and developers at Bank of America use Bloomberg Professional - a computer system that provides financial software tools. Bloomberg Professional is the main product of Bloomberg L.P., a financial software, data and media company. It provides tools for analysis so the users can view market data, conduct electronic trading with customers, and review historical data.

Bank of America is currently rebuilding many of the applications used by the front office using only Quartz. The goal of our project is to develop an application that mimics the functionality of several of the Bloomberg Professional screens. The Bloomberg Professional screens that will be replicated are the Bond Security Description screen, the Yield and Spread Analysis screen, the Credit Default Swap Security Description screen, and the Credit Default Swap Valuation screen.

2.0 Motivation behind the Project

Bank of America will be able to complete two objectives with the completion of this project. The first objective is to validate the quality of in-house reference data and risk analytics by comparing the results of the application we developed with Bloomberg Professional. The second objective is to increase the number of employees who have access to real-time financial data without requiring access to Bloomberg Professional.

The application that we developed pulls financial data from Bank of America's in-house database. By comparing the results of the application to BBG, Bank of America will be able to validate the data they collected as well as the analytics preformed on the data. This application will display the data gathered by Bank of America in a similar format to Bloomberg Professional which will ease the comparison of the data.

As of October 2011, there were approximately 290,500 full-time associates employed at Bank of America. Bank of America currently has approximately 10,000 Bloomberg Professional terminals in use globally. If an employee at Bank of America requires access to real time data he/she will have to get a personal subscription to Bloomberg Professional, which are expensive and very restrictive. The goal of this application is to provide employees with the same type of analytics and quality of data that Bloomberg Professional would without the subscription (Lawson & Hudson, 2013).

3.0 Literature Review

3.1 Financial Terms

3.1.1 Bonds

A bond is a debt security that obligates the issuer to pay the bondholder a specified sum of money, called interest, periodically. Once the bond reaches maturity the issuer must then repay the face value of the bond known as the principle amount (What Is a Bond?, n.d.). Bonds are a type of fixed-income security due to the rate at which interest is paid and the amount is typically fixed at the time that the bond is put up for sale. A subordinated bond is one that will be paid once other loan obligations (of the issuer) have been met. Along the ones mentioned previously there are many other types of bonds, each with its own rules and regulations (Morris & Morris, 2012).

3.1.1.1 Callable Bonds

A callable bond is a type of bond in which the issuer is able to buy back the bond prior to the maturity date. The issuer must pay the holder of the bond the call price, which was determined in the legal agreement of the bond, if the issuer chooses to buy the bond prior to the determined maturity date (Marshall, 2000).

The risk that the bond holder has if the issuer buys back the bond before the maturity date is called the call risk. The issuer is more likely to buy back the bond if interest rates decline below the coupon rate of the bond. If the issuer does buy back its bond, the bond holder gets their money back sooner than expected and therefore has to reinvest on a bond (Marshall, 2000).

3.1.1.2 Floating Rate Note

A floating rate note is a type of bond that has a floating rate of interest rate. This interest rate is determined by the benchmark reference rate of interest and the spread for that benchmark.

Unlike a fixed interest rate which is constant over the life of the instrument, the floating interest rate fluctuates based on the market conditions (Marshall, 2000).

3.1.1.3 Benchmark Bond

Benchmark Bonds are government bonds which are used as points of references to measure the current state of the economy. The yield level of the United States Treasury long bond is directly related to the current state of US interest rates, public-sector debt, and economic growth. Investors use benchmark bonds to measure the value of the bond and compare the return on investment of the bond (Choudhry, 2006). Benchmark bonds are used in the YAS screen.

3.1.2 Credit Default Swaps

A credit default swap is a contract between a protection seller and a buyer that provides protection on a specified reference asset in the event of a default. The buyer pays a periodic fixed fee to the protection seller in return for protection on the asset. This means that in the event of a credit default, the protection seller pays the buyer the par value of the asset minus the recovery which is the investor's money that was not lost.

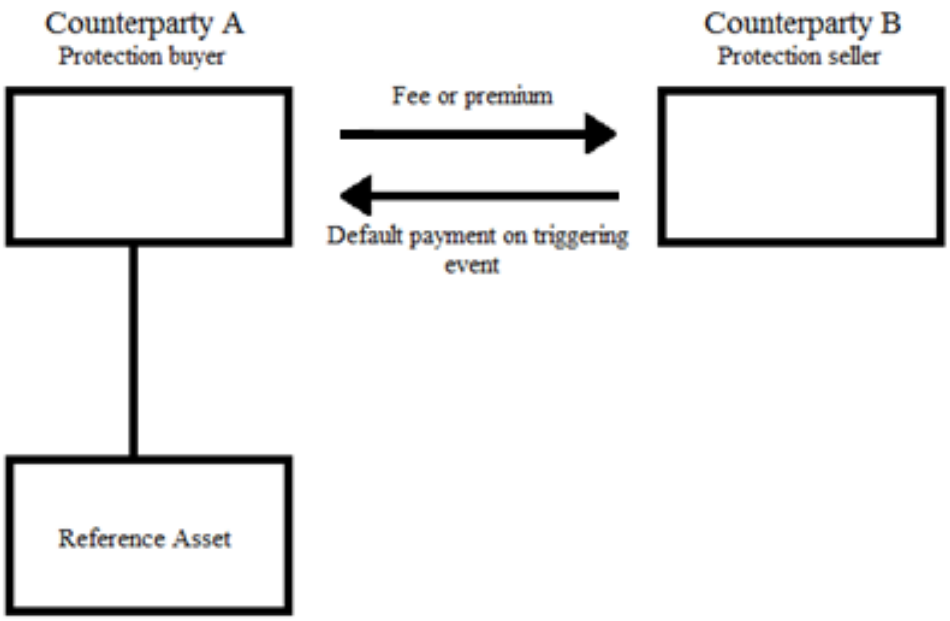

Figure 2. Credit Default Swaps

The credit default swap lowers the risk of the investment by transferring the risk to a counter party. The counter party is usually a larger institution therefore is able to take on a higher risk investment (Choudhry, 2006).

3.2 Bloomberg Professional

Bloomberg Professional (BBG) is a financial software tools used by the trading staff, middle and back office staff, and developers at Bank of America. It is the core product of Bloomberg L.P's, a financial software, data and media company. Currently, it is the market leader with about 315,000 subscribers as of May 2013 (Seward, 2013).

Bloomberg Professional provides tools for data analysis which allows users to validate ideas, watch trends, and generate value. The analytics tools provided included but are not limited to:

- Statistical & Comparative Analysis
- Volatility Monitors & Analysis

- Pre- to Post-Trade Analytics
- Portfolio & Risk Analytics

BBG offers its users two options for their subscriptions. The following are the access levels and a description of what they offer:

- Bloomberg Anywhere
  - Access available on any device such as any computer, terminal, or mobile device
- Bloomberg Open
  - Access available on only one terminal machine however it can it can have multiple logins

Bloomberg is very restrictive and closely monitors the users of the subscriptions, even including a biometric login on the Bloomberg Professional specific keyboard.

Although BBG does not publicize its prices, it is estimated that each subscription costs around $2,000 a month – making it the most expensive compared to rivals such as Thomson Reuters. They do offer a discounted price of $20,000 a year per subscription for institutions which have two or more subscriptions (Seward, 2013).

3.2.1 Bloomberg DES Screens

The Security Description (DES) Screen in Bloomberg provides descriptive information for different securities including corporate bonds, government bonds, syndicated loans, single-name credit default swaps, and credit default swap indexes.

3.2.1.1 Bloomberg Bond DES Screen

In order to obtain the bond's information the user can input different identifiers. The most common identifier is the International Securities Identification Number (ISIN) but the bond description or CUSIP can also be used. This identifier is the only user input for this screen.

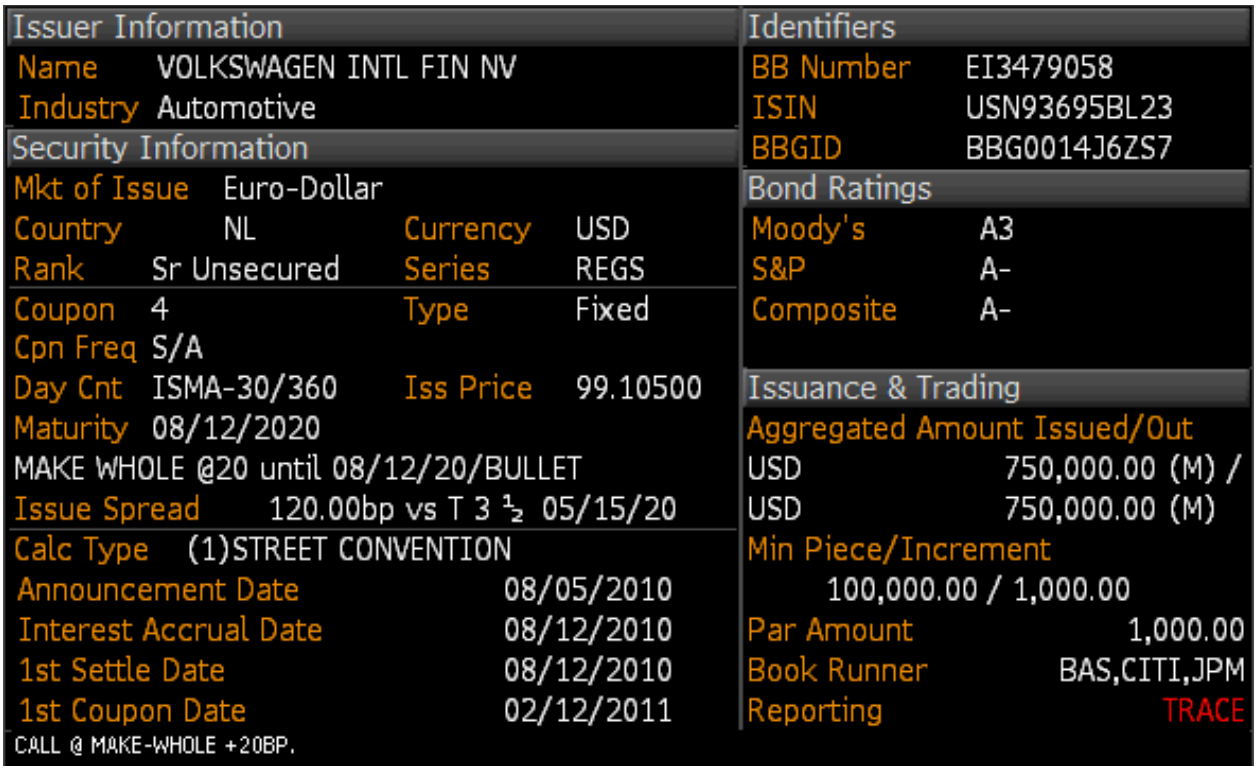

Figure 3. Bloomberg Bond DES Screen

3.2.1.2 Bloomberg CDS DES Screen

The user needs to input the ticker, currency, category of security, length of protection, and the yield of the credit default swap in order to obtain the security description information. Similar to the Bond DES Screen, there are no calculations and the values are obtained from the Bloomberg servers.

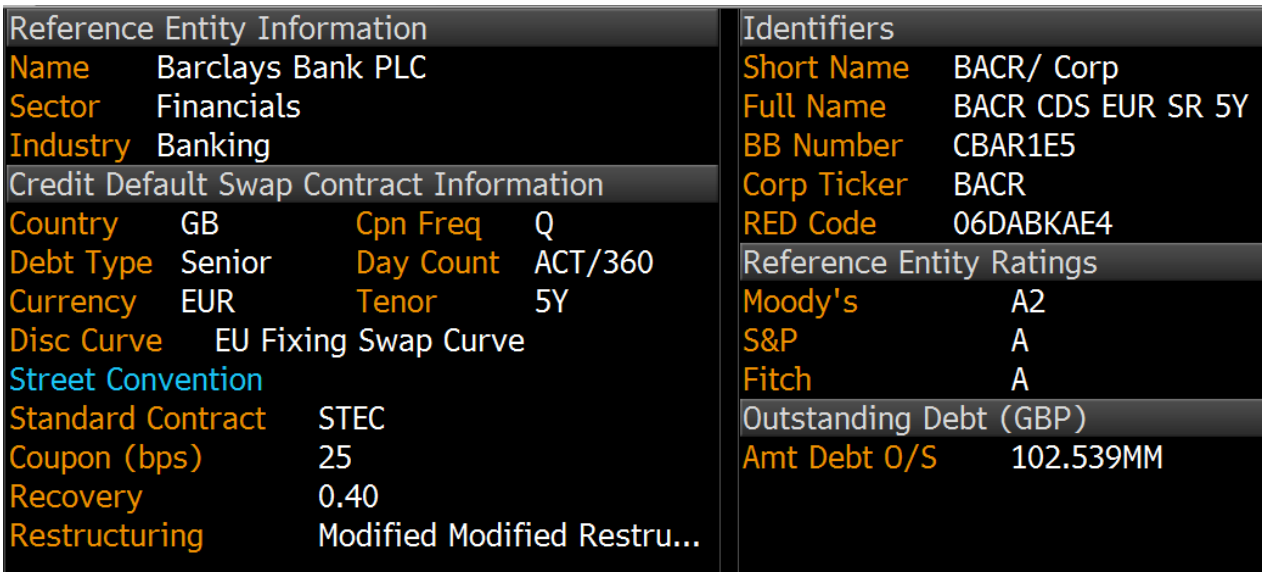

Figure 4. CDS DES Screen

3.2.2 Bloomberg YAS Screen

The Yield and Spread (YAS) screen allows for the analysis and pricing of a fixed income security based on user inputs. The information for the fixed income security is also retrieved with the same identifiers that the Bond DES screen accepts, including the ISIN number. The user also needs to input a reference security, usually a government security, which is used as a benchmark to compare the two securities. The yield and price of the reference bond are then auto-populated. The user needs to input the spread of the bond which auto-calculates the yield and price of the security. All of these fields are editable allowing the user to adjust the price, yield, and spread of both securities to analyze relationship between the different fields in different scenarios. When adjusting the price, yield, or spread any dependent fields are automatically-updated. All other fields are automatically populated once the ISIN is entered.

The following are the formulas for the calculations:

- Spread

  Yield of Fixed Income Security =
  Spread of Fixed Income Security + Yield of Reference Fixed Income Security

- Price of the Fixed Income Security

$$P^{full} = \frac{C_D / f_D}{(1 + y_D / f_D)^{f_D T_1}} + \frac{C_D / f_D}{(1 + y_D / f_D)^{f_D T_2}} + \frac{C_D / f_D}{(1 + y_D / f_D)^{f_D T_N}}$$

- P^{full} = price of the default bond
- C_D = the annualized coupon
- f_D = coupon frequency
- $T_1, \ldots, T_N$ = time to each of the cash flow payments in a year
- y_D = yield of defaultable bond

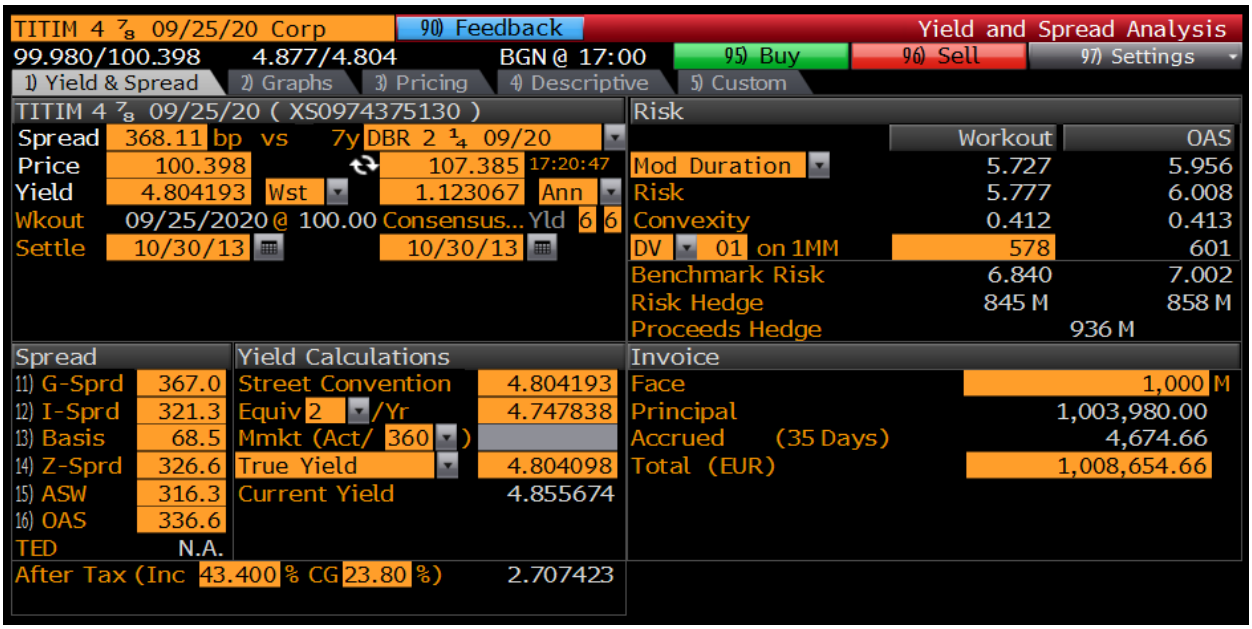

Figure 5. YAS Screen shot

3.2.3 Bloomberg CDSW Screen

The Credit Default Swap Valuation (CDSW) screen allows users to create and value credit default swaps. Issuers can be searched to create a CDS or an equity, government, municipal, preferred, corporate security, or CDS index can be chosen to be used as an issuer reference.

Like the YAS screen, most of the fields and values on this screen are editable, allowing the user to adjust the spread, points up front, and price just to name a few. When fields and values are adjusted any dependent fields are automatically updated and values are recalculated.

The following formulas are for the values in the screen:

- Cash Amount = Principal + Accrued
- Principal = Clean Price (calculated internally)
- Accrued = Accrued Interest (calculated internally)
- Price = 100 – Points Upfront
- Points Upfront = Principle / 100,000

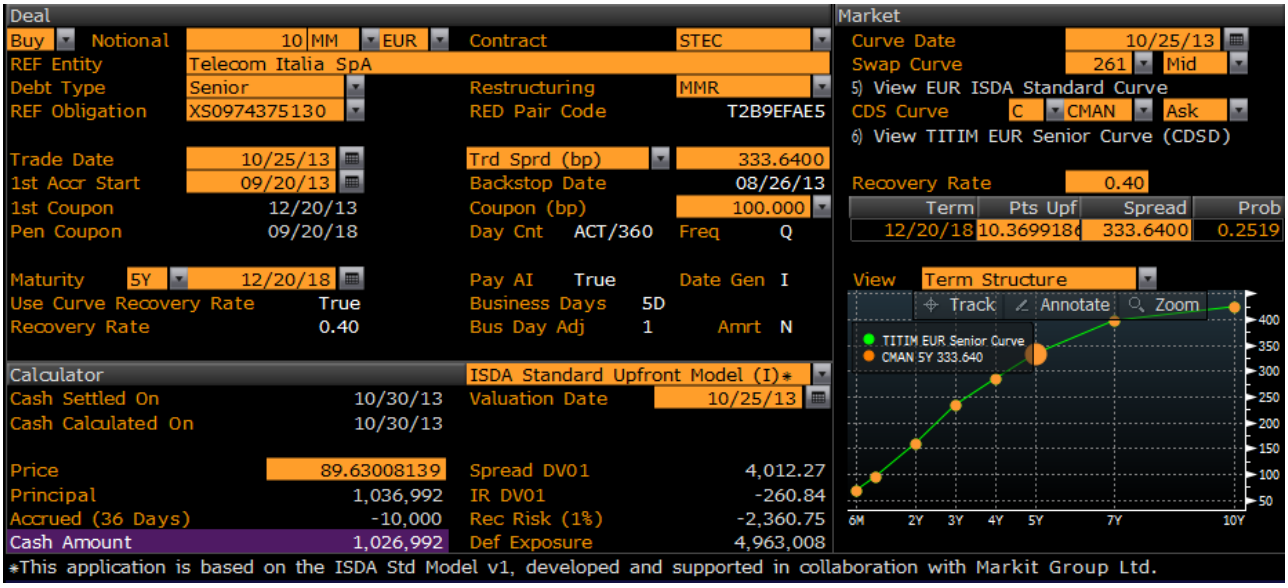

Figure 6. CDSW Screen Shot

3.3 Programming Practices and Terms

3.3.1 Development

For this project, we followed an agile method that is based on iterative and incremental development called the feature-driven development process. Agile methods promote evolutionary development and encourage rapid and flexible responses to change. The following diagram depicts the steps of the feature-driven development process (Luca, 2012):

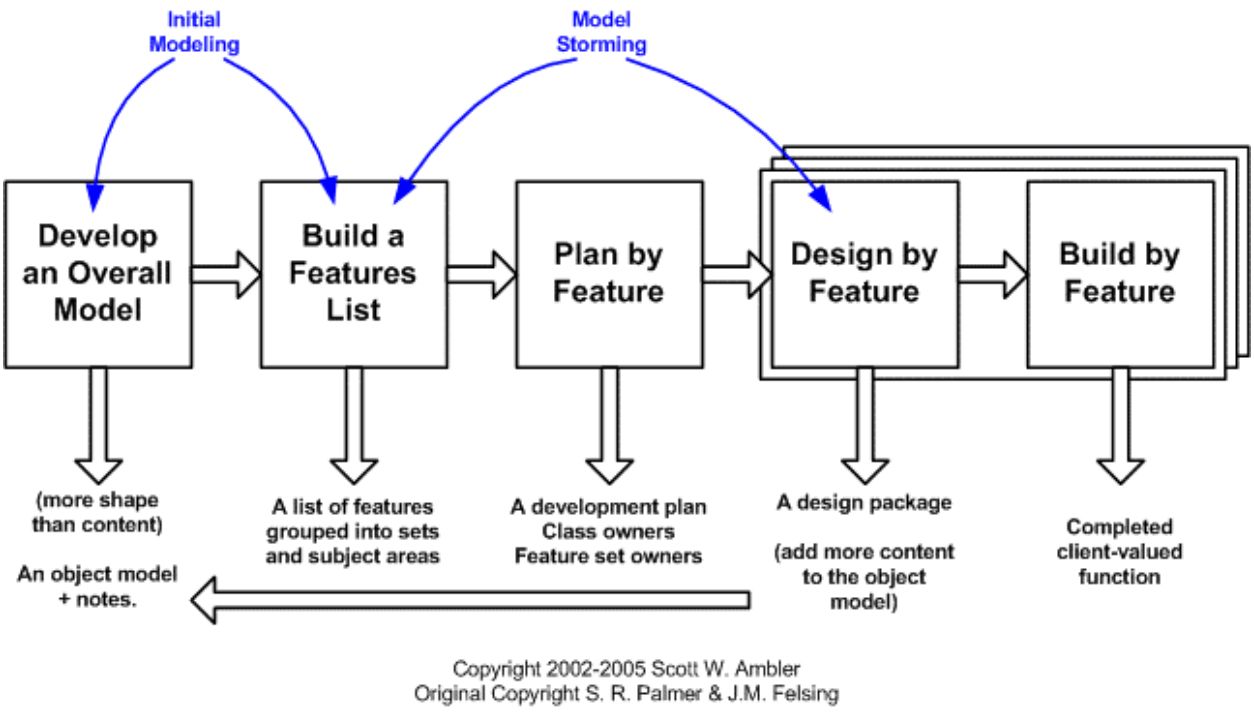

Figure 7. Feature-Driven Development Process

We chose an agile method because of the flexibility of the method. When compared to a traditional method, like the waterfall model which has strict steps, an agile method seemed most appropriate due to the limited seven week time span. The end result for each screen was left open and therefore could be updated throughout the project.

3.3.2 Factory Methods

The Graphical User Interface (GUI) was built by using the factory method design pattern for the fields and labels. These fields and labels were then used to separate each section of the screen into a new method. A factory method creates an object without specifying the exact details of the object created; instead, the details are passed. This allows for differently formatted objects to use the same method thus minimizing code. Another benefit of the factory method is it creates a single place where developers can make changes that apply to all fields, labels or similar attributes.

3.3.3 Design Pattern – Model-View-Controller

The Model – View – Controller pattern is a coding style which separates an application into three parts.

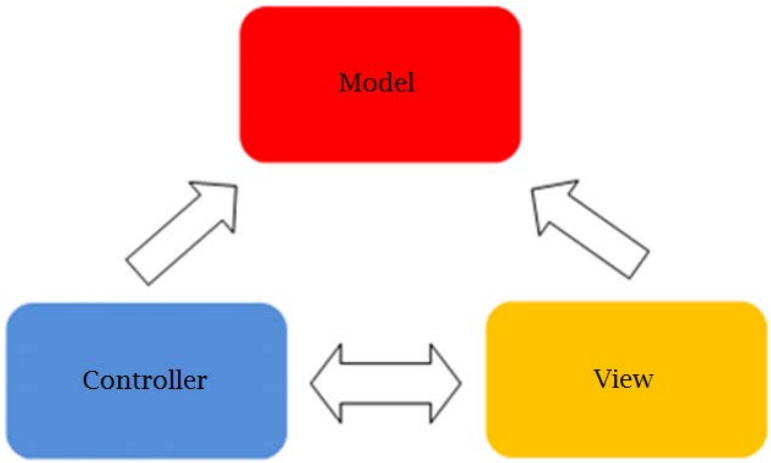

Figure 8. Model-View-Controller Pattern

Model
- contains the application data
- contains the business logic

View
- contains the code for how the is being displayed
- requests the data from the model

Controller
- manages how the view and model change based on the user inputs
- for this project the DAG cells take the place of most of the controller, the rest is integrated into the model

3.3.4 Directed Acyclic Graphs

Directed Acyclic Graph (DAG) is a hierarchy where objects are defined relative to the transformations on their parent objects. This hierarchy is also called a transformation hierarchy and the objects that implement it are called DAG objects or nodes. In a DAG, there are two types of nodes - transforms and shapes. In more simplistic terms, a DAG is a collection of nodes

and directed edges, each edge connecting one node to another. Transforms are parent nodes that must have at least one transform node as a parent. This means the shape nodes, being leaf nodes, cannot have any children beneath them. Therefore, there is no way to start at some node *N* and follow a sequence of edges that loops back to n again (Autodesk, n.d.). Normally the object-hierarchy is shaped like a tree, as seen below:

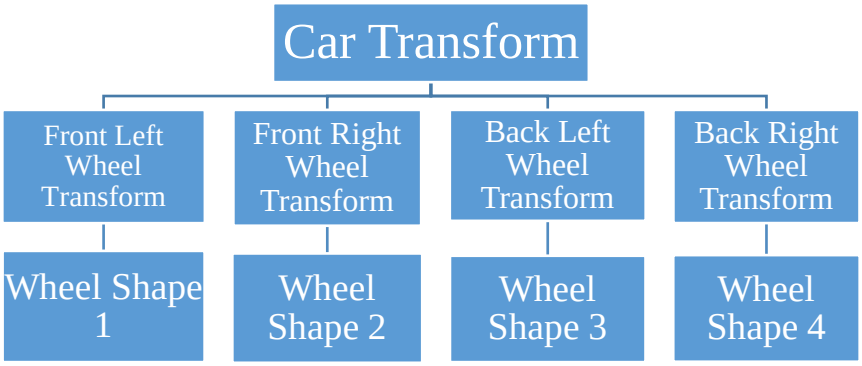

Figure 9. Car Transform Hierarchy

With a DAG hierarchy the structure is shaped like, as the name suggests, a graph. Using the same structure as seen above but the DAG objects, the hierarchy would look as such:

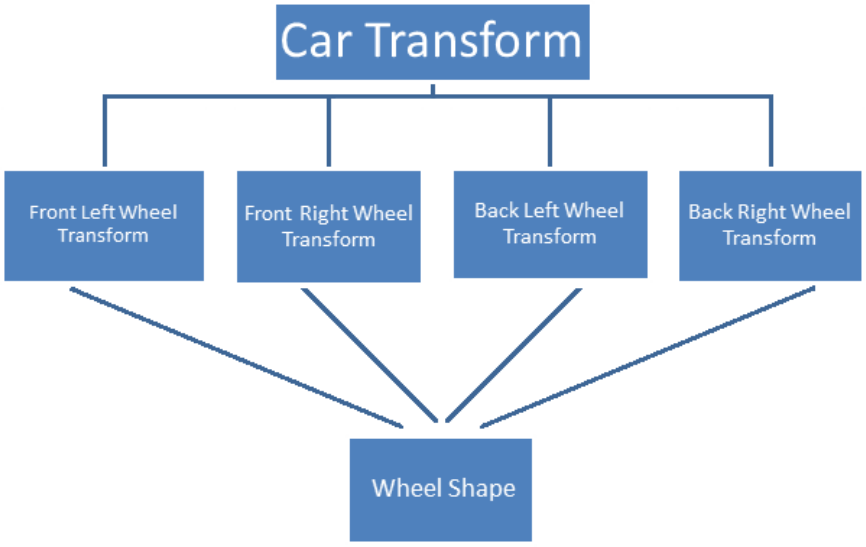

Figure 10. Car Transform Graph

3.4 About Quartz

Quartz is Bank of America's in-house integrated platform used by all asset groups for trades, market data, analytics, pricing and risk management. The primary programming language used is Python but it also includes C++ and C# / .NET. Quartz's core components are Sandra, the User Interface Framework demo, and Powwow.

3.4.1 Python

Python is a general-purpose high-level programming language developed by Guido van Rossum. It is known for being easy to read, learn, and modify. Python's philosophy, "there should be one – and preferably only one – obvious way to do it" is the complete opposite of Perl's "there is more than one way to do it" philosophy (Peters, 2004). When compared to other languages such as Ruby, Scheme, or Java, Python's distinguishing features are (About, 2013):

- Clear, readable syntax
- Exception-based error handling
- Extensive standard libraries and third party modules

3.4.2 Quartz Development Process

In order to promote Quartz's integrated development environment, there are four steps which team members are highly suggested to follow while developing in QZDev. The following figure displays the four steps along with a small description of the step:

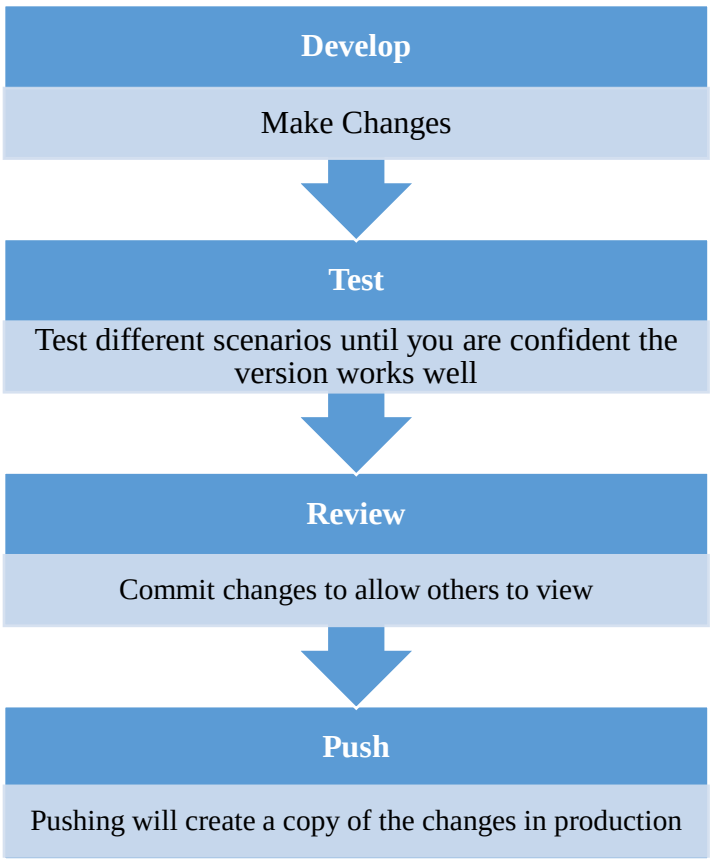

Figure 11. Quartz Development Process

3.4.3 QZDesktop

The QZDesktop is the collection of all the strategic business applications, which every employee at Bank of America has access to. The business applications we used while completing our project were Powwow, QZDev, and the UI Demo.

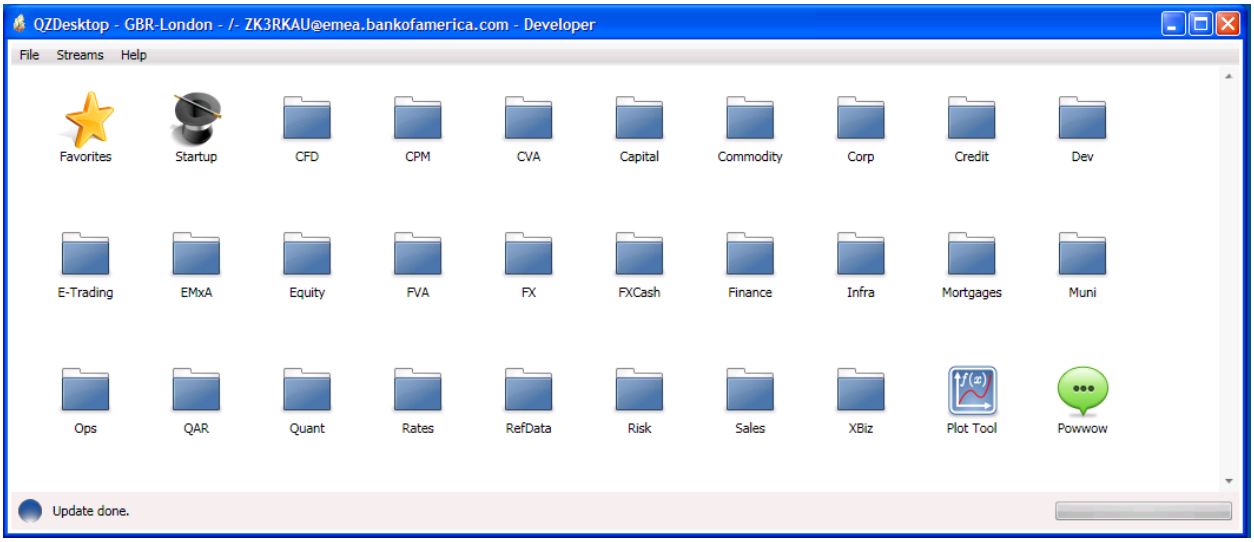

Figure 12. QZDesktop Screen Shot

3.4.3.1 Playground

Each developer at Bank of America has a playground which is a personal file where they can create code that does not affect any files or data outside of that file. The playground was used during the design and coding stages of each screen - before the code was deemed acceptable and moved into the working code base.

3.4.3.2 QZDev

QZDev is Quartz's integrated development environment (IDE) in which programmers develop and test their code. It consists of the Python shell, Python libraries, Quartz libraries, as well as other applications and tools.

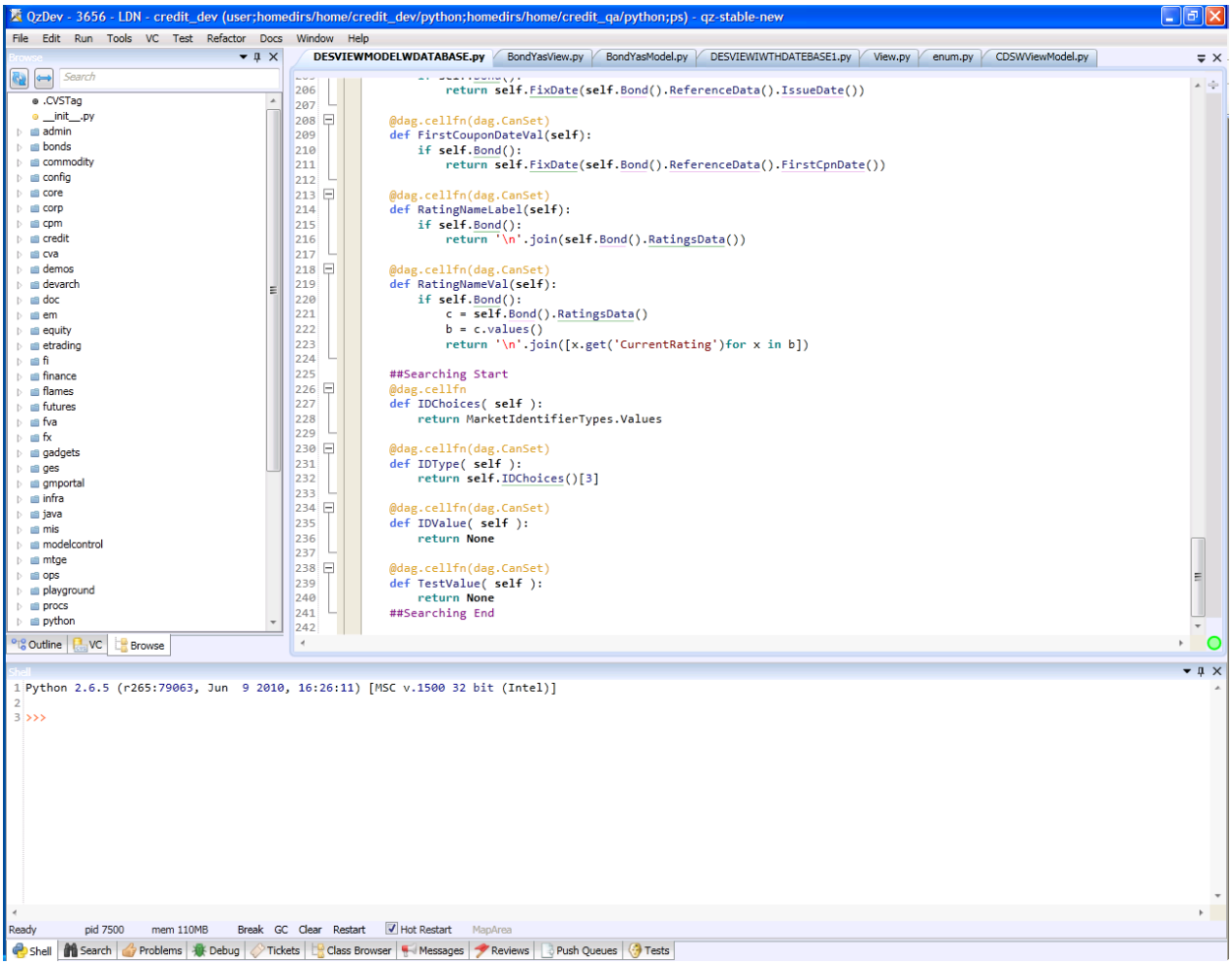

Figure 13. QZDev Screen Shot

3.4.3.3 User Interface Demo

The User Interface (UI) Demo application provides example code and displays the customizable features of the different UI objects in Quartz. During the first week of our project, we explored and examined the different UI objects to decide on the most appropriate for each aspect of the screens. This was also used as a source of references for using DAG objects with the UI objects which made up a majority of our screens.

Figure 14. Bond DES Screen Shot

3.4.3.4 Powwow

Powwow is an instant messaging application which allows employees to communicate with each other. It is designed for short messages with inquires about errors which occur while developing a code. Powwow also acts as a monitor for the code base.

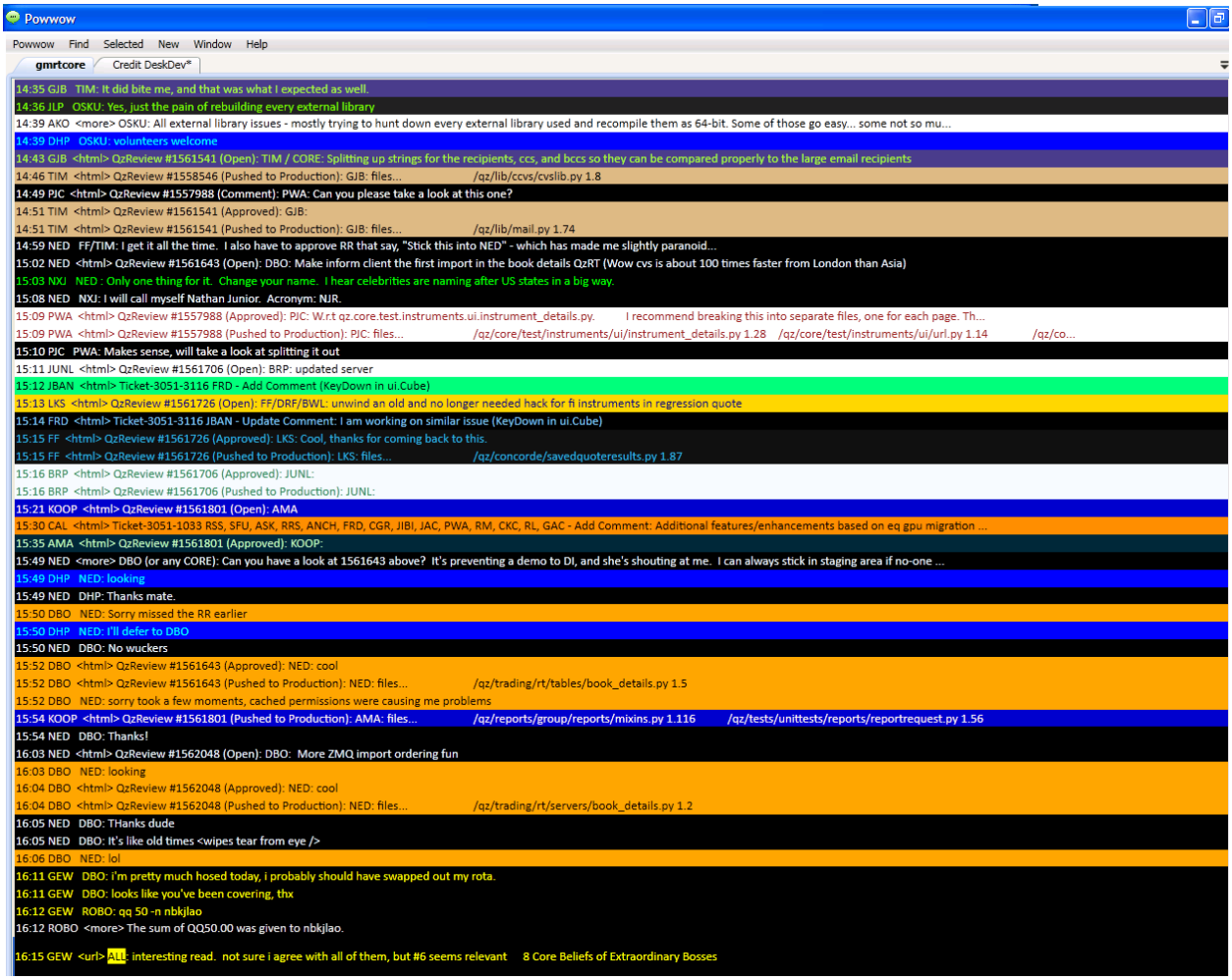

Figure 15. Powwow Screen Shot

3.4.4 Sandra

Sandra is Bank of America's in-house object database and is the source of all of the information our project accessed in Quartz. The contents of the Sandra database can be viewed in the Qzap Application DB Browser.

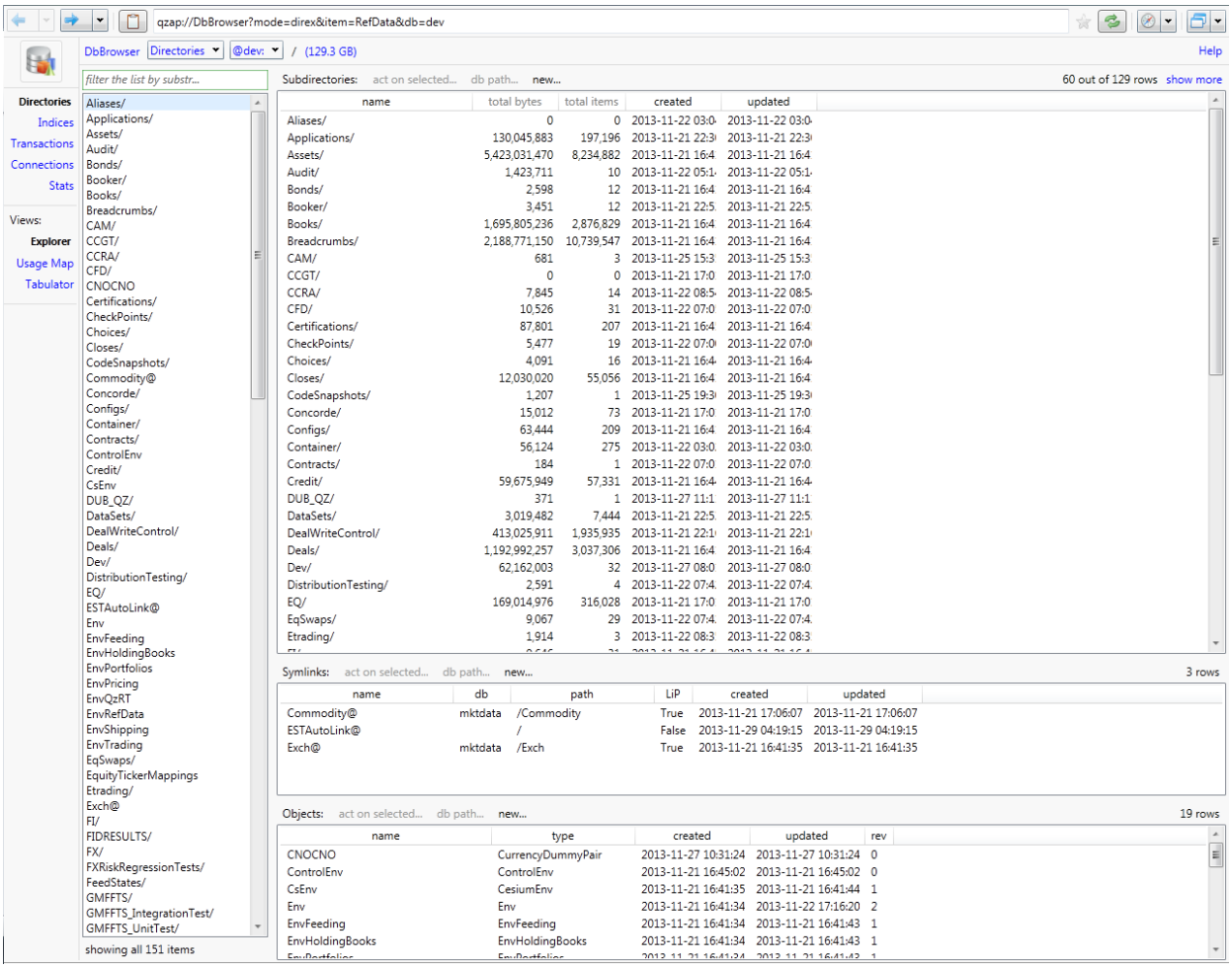

Figure 16. Sandra Screen Shot

4.0 Methodology

In order to continually receive feedback on the code we developed, we followed an iterative development approach while completing the project. The follow figure shows the approach we followed:

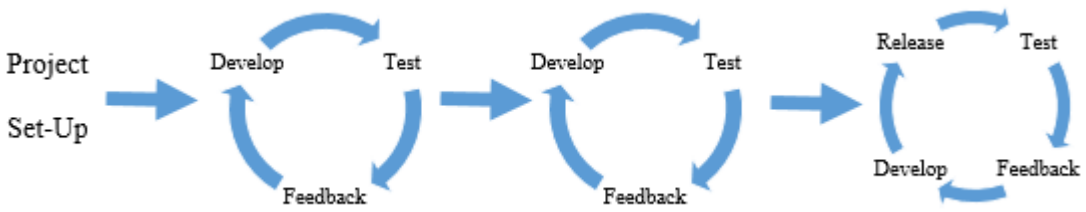

Figure 17. Iterative Development Approach

This chapter outlines the development, testing, and feedback steps we repeated until we felt the code was up to expected standards which was when the final step, the release of the code, occurred.

4.1 Project Set-up

4.1.1 The Quartz Project for Beginners – Getting Started

Bank of America provides new Quartz users with tutorials to familiarize themselves with the in-house software. These tutorials demonstrate the features of the Quartz functions and have step by step instructions on how to utilize these functions. We completed these tutorials during the first days of our project which allowed us to understand the features of Quartz and the programming language Python.

4.1.2 Meetings with Sponsor

During the first two weeks we met daily with our sponsor, Paul Ashby, to discuss our progress during the day. In these meetings, we would discuss any problems we came across, how to fix the problems, and our short term goals for the coming days. After the first two weeks we became more familiar using Quartz and the expectations of our project so we started to meet with our sponsor twice a week or whenever we had a major problem.

4.2 Development

4.2.1 Development of the Overall Model

This section describes the steps which we took when writing the code for the project. We determined the order of the steps based on dependencies of the features – we started with the features which had no dependencies and then proceeded to the more complex features.

4.2.1.1 Developing the User Interface

We first created a framework for the screens which would allow for easy removal or addition of different screen sections. These sections were divided into separate functions, or modules, and combined in the panel. The following is example code for the Bond DES screen's panel:

```
def panel(self):
    return ui.VL([
            self.bondSelector(),
            [[self.issuerInformation(),
                self.securityInformation(),
                self.column1Row3(),
                self.column1Row4(),],
              [self.identifiersInformation(),
                self.bondRatings(),
                self.issuanceAndTrading(),
                self.switchToYAS()]]],
                  attr=self.panelAttr(), scroll=True)
```

Each module contains the UI elements that make up that section. The code below is an example of the Issuer Information section:

```
def issuerInformation(self):
        issuerInfoLabel = self.makeDefaultHeader("Issuer Information", self.headerLabelAttr())

        nameLabel = self.makeDefaultLabel( "Name", self.defaultLabelAttr())
        industryLabel = self.makeDefaultLabel( "Industry", self.defaultLabelAttr())

        nameValueLabel = self.makeDefaultLabel(self.Binding().NameVal, self.defaultValueAttr())
        industryValueLabel = self.makeDefaultLabel(self.Binding().IndustryVal, self.defaultValueAttr())

            return self.addDefaultBorder(
                ui.VL([
                issuerInfoLabel,
                        [nameLabel,
                         nameValueLabel],
                          [industryLabel,
                        industryValueLabel]],
                        alignChildren=True))
```

All of the UI elements were created using a factory method. The factory method for creating labels can be seen below. Using factories to create the UI elements created a single

place to make changes in order to propagate any graphical changes to each element in the panel that used that factory method. The following is example code of the label creating factory:

```
def makeDefaultLabel(self, name, attributes):
    return ui.Label( name, attr=attributes, halign=ui.Align.LEFT, size=(ui.Size.STRETCH, ui.Size.STRETCH))
```

4.2.1.2 Developing the Model

Once the framework, also known as the view, for the screen was complete, the model was created - the controller was integrated into the view (see 6.1.1.2 Model – View – Controller for more information). The model was created by adding DAG elements for the corresponding value in the view. The following is an example of how the currency label in the view obtained its value from the model:

View:

```
currencyValueLabel = self.makeDefaultLabel(self.Binding().CurrencyVal, self.defaultValueAttr())
```

Model:

```
@DAG.cellfn(DAG.CanSet)
def CurrencyVal(self):
    if self.Bond():
        return self.Bond().ReferenceData().Currency()
```

4.2.1.3 Loading the Object

In order for the Model to load the bond/CDS object, the attributes of the desired object have to be entered onto the search field of the screen. For bonds, the Model communicates with Sandra directly and receives a bond object. The process is more complicated for credit default swaps. Instead of communicating with Sandra directly, the Model talks to an interface which handles communication with Sandra itself. Once the given attributes match a CDS in the database, the CDS object is returned.

4.2.1.4 Qzap Integration

Changing the QzDesktop application into a Qzap application did not require many changes. An interface was implemented which required functions that updated the screen,

returned the screen (as an object), and handled joining the URL's parameters to the search fields. Linking the URL's parameters to the search fields allowed the URL to automatically update itself whenever the values in the search field(s) were changed. It also allowed for the application to receive parameters from the URL and load the requested object, should the screen be accessed directly via a Qzap URL.

4.3 Testing

Our main method of testing the screens we created was comparing them to the Bloomberg Professional Screens. If the values matched the fields, then we were loading the correct object from the database or accessing the correct attribute. We were also able to use Bloomberg Professional to confirm that the algorithms that calculated certain fields were correct. If the numbers did not match, then we had to determine whether the error was caused by the code or by the algorithms that calculated it.

4.4 Feedback

4.4.1 Code Review with Stefano

After the completion of a major milestone, such as the completion of a screen, we would have a code review with Stefano Cattani – Programmer Prof MKTS. These code reviews were informal walkthroughs of the code in which Stefano would make recommendations and suggestions for improvement. Changes to the code were then made based on Stefano's suggestions and recommendations. Appendix I and III contain the notes from the code reviews with Stefano Cattani.

4.4.2 Interview with Richard Jervis

Once the User Interface was complete and about half of the functions of the screens were working correctly, we conducted an interview with Richard Jervis, a Senior Trading Strategist I,

to receive feedback on the screens. Richard is a frequent user of Bloomberg Professional and a potential user of our application. Appendix V contains our notes from this interview.

4.5 Release

Throughout the course of the project informal code reviews were held whenever a major point in the project was reached, typically when a screen reached a point of completion or a major bug was fixed. However, the code review for the final release was more formal, containing all of the new and changed files since the beginning of the project. After committing the final code, a code review was requested and sent to be approved. The reviewer looked for a number of things, mainly if there was useful documentation, good coding practices were followed (organization and naming conventions), and that the code ran without obvious bugs. If the code was up to the expected standards and free of obvious bugs, the review was approved and the code was ready to be pushed to production. Pushed to production meant that once QzDesktop was restarted, forcing it to update, the code was live. Any employee with access to QzDesktop would be able to view the updates, changes, or access the new application.

5.0 Results

5.1 Qzap Application

The Qzap Application can be accessed through the DB Browser in the QZDesktop. The user needs to input the respective URL to go to the designated screen. The following is a screen shot of the application:

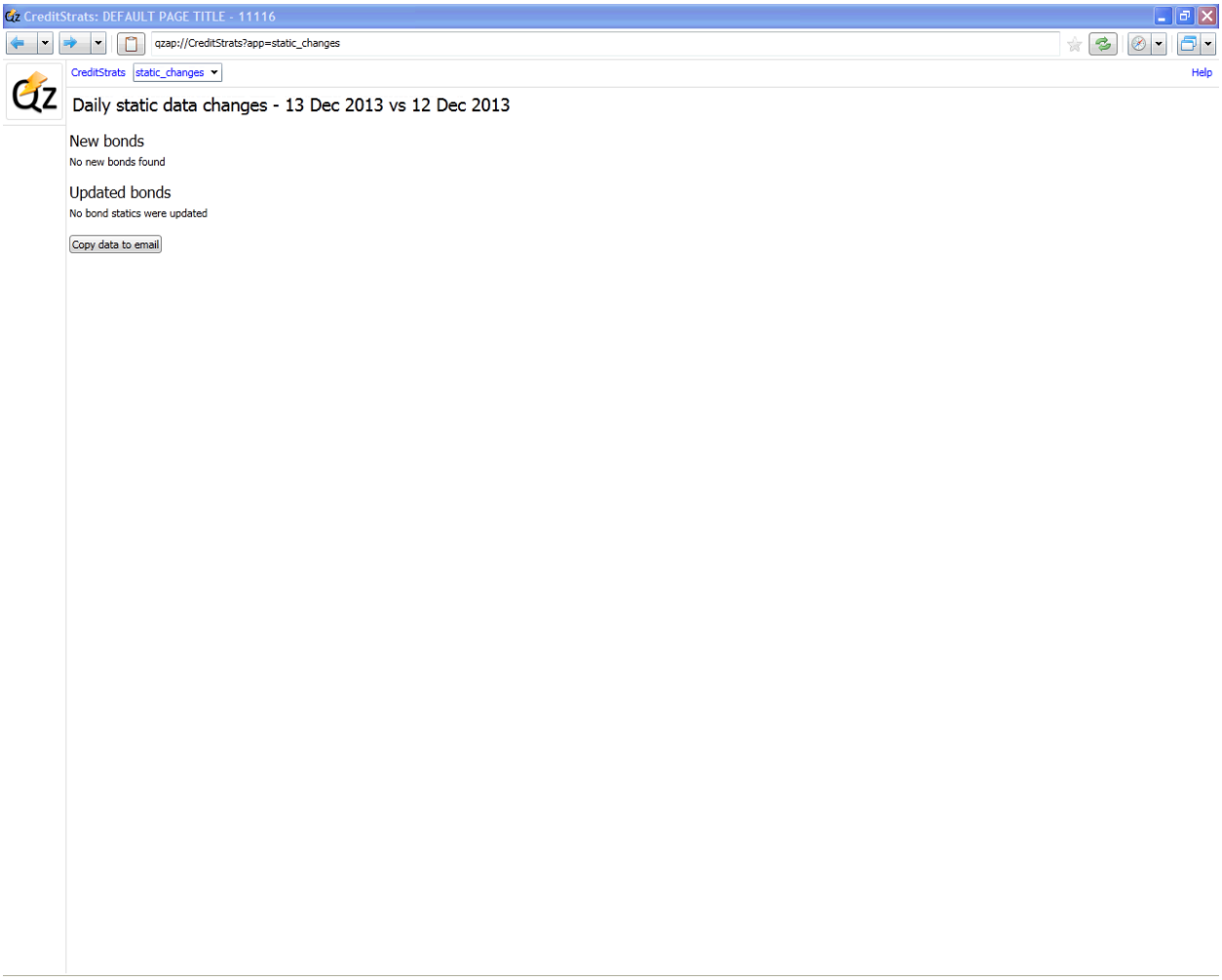

Figure 18. Qzap Application

5.1.1 Bond DES Screen

The URL to access the Bond DES screen is Qzap://CreditStrats?app=bond_des. The following is a screen shot of the Bond DES screen:

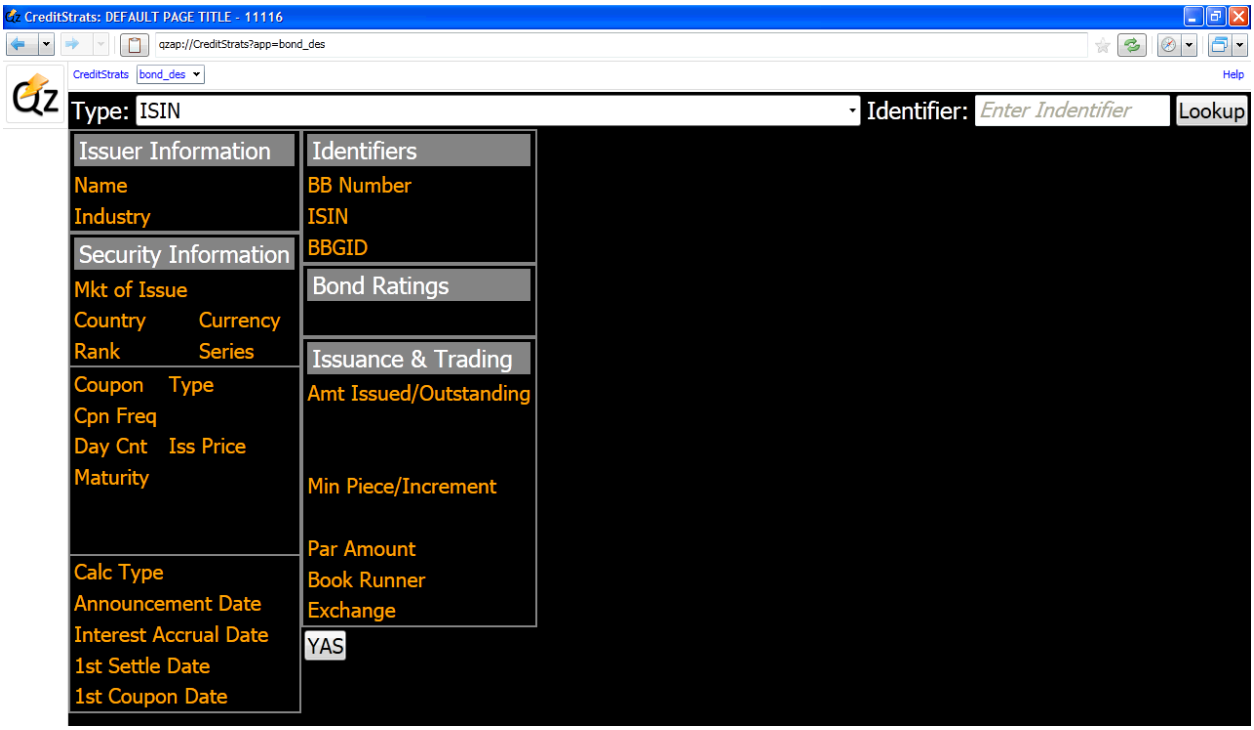

Figure 19. Qzap Application Bond DES Screen

When using the bond DES screen, the user has the option of entering the bond's CUSIP, ALICE, Bloomberg, ISIN, RED Code, or ticker identifier. The most common identifier used to look up a bond is the ISIN.

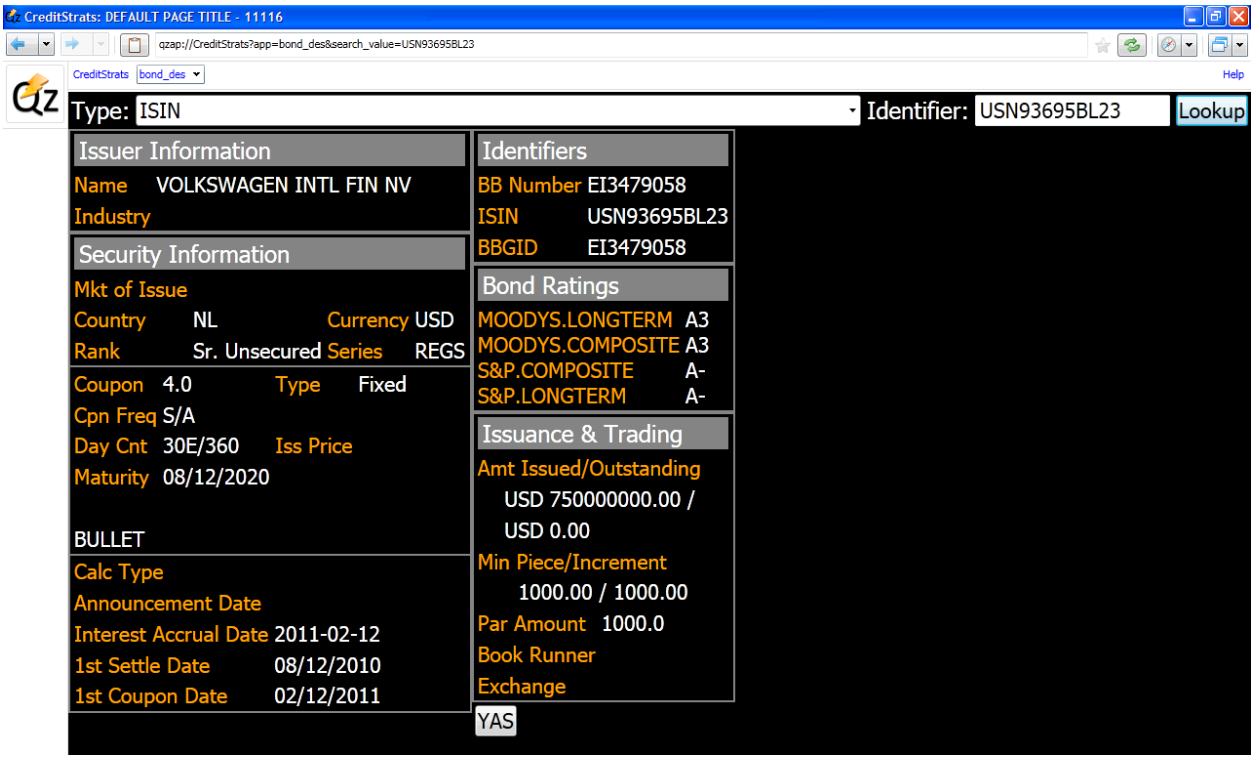

Figure 20. Qzap Application with ISIN USN9365BL23 DES Screen

5.1.2 YAS Screen

The URL to access the YAS screen is Qzap://CreditStrats?app=yas. The following is a screen shot of the YAS screen:

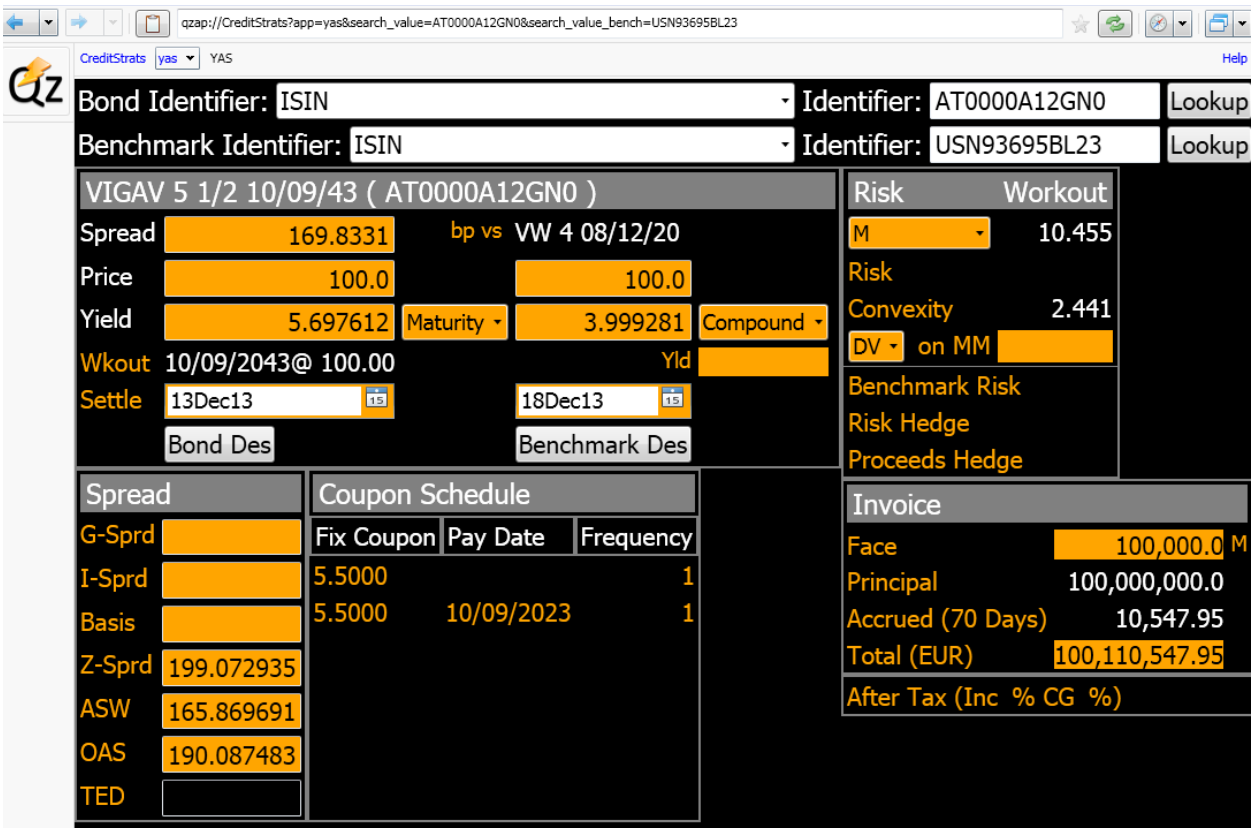

Figure 21. Qzap Application YAS Screen

For the YAS Screen, the user has to input the identifier for both the bond and the benchmark bond. Similar to the DES Screen the user can use the bond's CUSIP, ALICE, Bloomberg, ISIN, RED Code, or ticker identifier as the identifier. The user will also have to input the yield of the reference bond and the spread. The price and yield of the main bond as well as the settle dates will all be able to be adjusted by the user.

5.1.3 CDS DES Screen

The URL to access the CDS DES screen is Qzap://CreditStrats?app=cds_des. The following is a screen shot of the CDS DES screen:

Figure 22. Qzap Application CDS DES Screen

When using the DES Screen, the user has to input the CDS ticker and select the debt type, restructuring type, currency, payment frequency, and the duration of the payments.

The following are the options for Debt Type:

- Receivable
- Hybrid
- Loan Lien 2
- Senior Secured
- Loan Lien 1
- Loan Lien 3
- Preferred Jr Subordinated
- Subordinated
- Senior Secured

The following are the options for Restructuring:

- LCDS – CDS contract where the underlying is a syndicated loan
- MM – Modified Modified Restructuring
- SCN3
- SCN6
- MR – Modified Restructuring
- XR – No Restructuring
- CR – Complete Restructuring
- REC

The following are the options for the payment frequency:

- Continuous
- Simple
- Annual
- Semi-Annual
- Daily
- Monthly
- Bimonthly
- Quarterly
- Biweekly
- Weekly

The following are the options for the tenor:

- 0M
- 3M
- 6M
- 9M
- 1Y
- 2Y
- 3Y
- 4Y
- 5Y
- 7Y
- 10Y

5.1.4 CDSW Screen in the Qzap Application

The URL to access the CDSW screen is Qzap://CreditStrats?app=cdsw. The following is a screen shot of the CDSW screen:

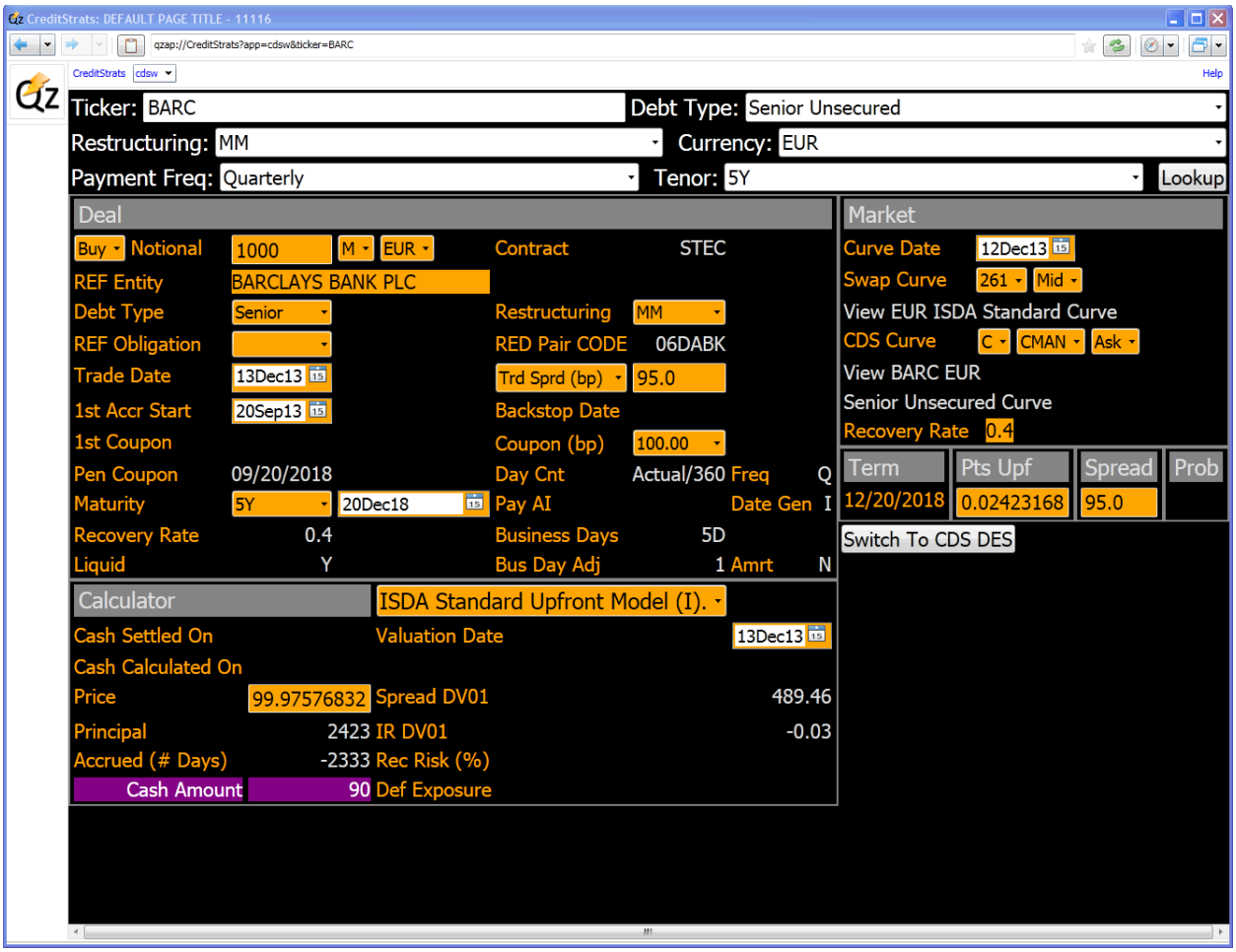

Figure 23. Qzap Application CDSW Screen

When using the CDSW Screen, the user has to enter and select the same inputs as the CDS DES Screen – the CDS ticker, the debt type, restructuring type, currency, payment frequency, and the duration of the payments.

5.2 Challenges

5.2.1 DES (Bond & CDS) Screen

When copying and pasting ISINs into the DES screen, the error "Bond Not Found" would occur due to trailing white space. A delegate function to handle changes to the identifier and ticker field was created that removes any leading or trailing white space, capitalizes all

letters, and updates the display of the screen. The *ForceUppercase* attribute was also added to handle only the removal of white space.

5.2.2 YAS Screen

Unlike the DES screens, this screen contained cyclic relationships which DAG objects are incapable of handling. When the spread was changed the yield changed and when the yield changed, the spread change. This meant that without a 3rd party to handle the updating of each variable, changing the spread or the yield would create an infinite loop of each variable updating the other. As can be seen below we used delegate functions to handle whenever a value was changed.

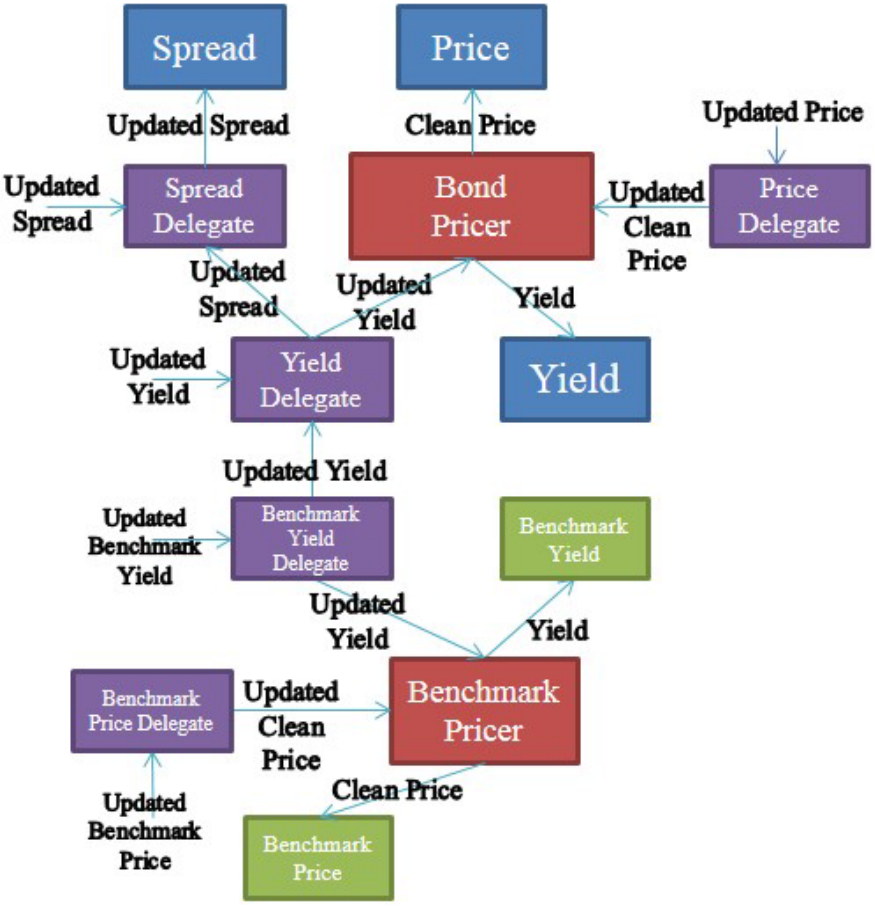

Figure 24. Delegate Functions used in the YAS Screen

A relationship graph without text, shown below, is also included to demonstration the dependencies more clearly.

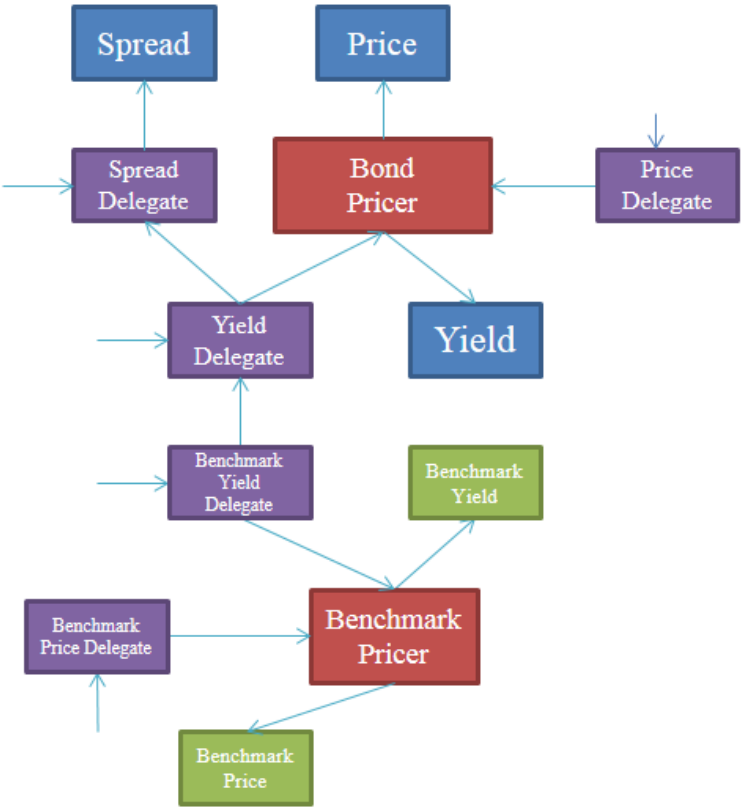

Figure 25. Delegate Functions used in the YAS Screen

One characteristic of the DAG delegate function which was very useful was its ability to ignore the delegate functions of other objects. For example, the spread of the bond could update the yield without triggering the yield's delegate function, thus avoiding the infinite loop.

6.0 Discussion

6.1 Developing using the Agile Method

6.1.1 Developing and Overall Model

6.1.1.1 User Interface

We were given the ability to pick the design and layout of our application; the only requirements we were given were the specific functionality of the screens. The idea behind the replication of the appearance of Bloomberg Professional for users who have interacted with Bloomberg to feel comfortable while using the replicated screens. Therefore, the users do not have to learn how to use a new piece of software, especially if the replicated screens and Bloomberg are used interchangeably.

6.1.1.2 Model – View – Controller

We decided to use a modified Model – View – Controller pattern based on past experience and examples of code we reviewed. Since we knew the final appearance of the screen but not the complete functionality, this pattern allowed us to fully create the view and model separately without having to wait for the completion of the other. Due to the way Qzap URLs functioned, we decided to integrate the majority of what would be found in the controller into the view. Since DAG objects update their children when their value has been changed, they handled the remaining aspect of what would normally be found in the controller (Goyal, 2008).

6.1.2 Build a Features List

At the beginning of the project, we came up with a minimum set of functionality for each screen. The other features that needed to be added would be built upon the original functions. After each screen was created, or a new set of features were added, we would sit down with the

Credit Dev team to determine which features would be replicated and established a new features list.

6.1.3 Plan by Feature

The implementation of each feature was based on the complexity of it and its dependencies. For the YAS and CDSW screens, we took into account the user-editable auto-updating fields and prioritized features based on the dependencies.

6.1.4 Design by Feature

Once we prioritized the features, we looked at the relationships between each, the dependencies, and created a sequence diagram. This step was mainly needed of the YAS screen as the yield, price, benchmark yield, benchmark price, and spread all had dependencies among themselves and needed to maintain those relationships when any of those values were changed. We knew that all values in the view had to have a corresponding value in the model but there were also some value labels that retrieved their values from the model as well. At the completion of this step, we moved onto building the actual view and model.

6.1.5 Build by Features

The different sections of the screens were created as modules with subsections that allowed developers to easily add, remove, or reorder the appearance of a screen. Once the GUI framework was built, we created the model and added DAG elements for the corresponding value in the view. Once the DAG elements pulled the correct information from the database and the UI elements displayed their corresponding DAG element, we added the bond/CDS loading feature. The Qzap integration feature was added once all of the features functioned correctly.

6.2 Qzap Application Features

6.2.1 Error Message

The following screen shot displays the error message, “Bond Not Found”, that appears in order to inform the user that an incorrect identifier number has been entered.

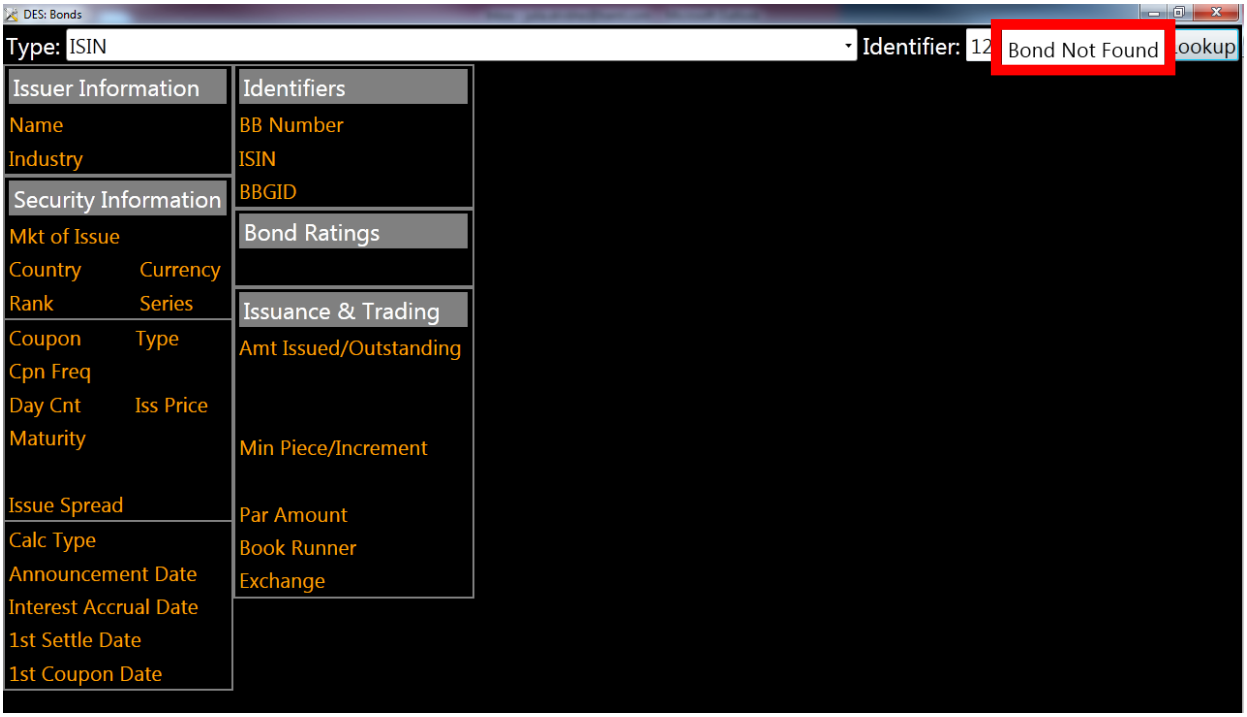

Figure 26. "Bond Not Found" Error Message

6.2.2 Switching between Screens

A button was added in order to facilitate the switching of screens. The following are the screens which can be switched between:

- Bond DES Screen → YAS Screen
- YAS Screen → Bond DES Screen
- CDS DES Screen → CDSW
- CDSW → CDS DES Screen

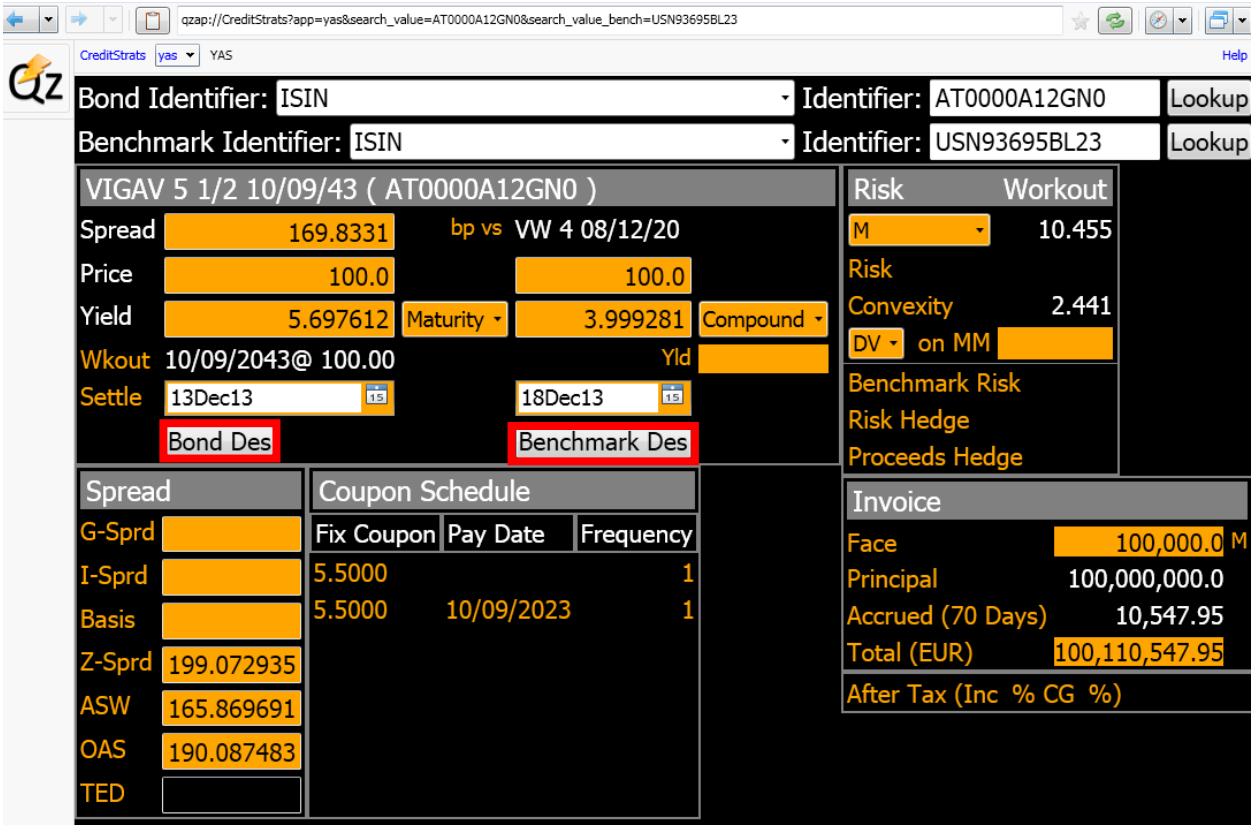

Figure 27. Qzap Application YAS Screen

6.2.3 Qzap URL

In the Qzap Browser, a variety of different applications can be accessed via a URL. Similar to a URL used in a web browser, it contains parameter values for the application. In regards to our project, the benefit of the Qzap URL is that if a bond or CDS is loaded into one of the screens, the link automatically updates itself. This means that displaying the screen with that specific bond or CDS on another computer or on another tab is easy, the Qzap URL just has to be copied. An example URL for the Bond DES screen would look as such:

Qzap://CreditStrats?search_type=ISIN&app=bond_des&search_value=AT0000A12GN0

- Qzap:// indicates that the link should be opened in the Qzap Browser
- CreditStrats is the Directory
- App is the desired application
- Search_type and search_value are the parameters for the Bond DES application

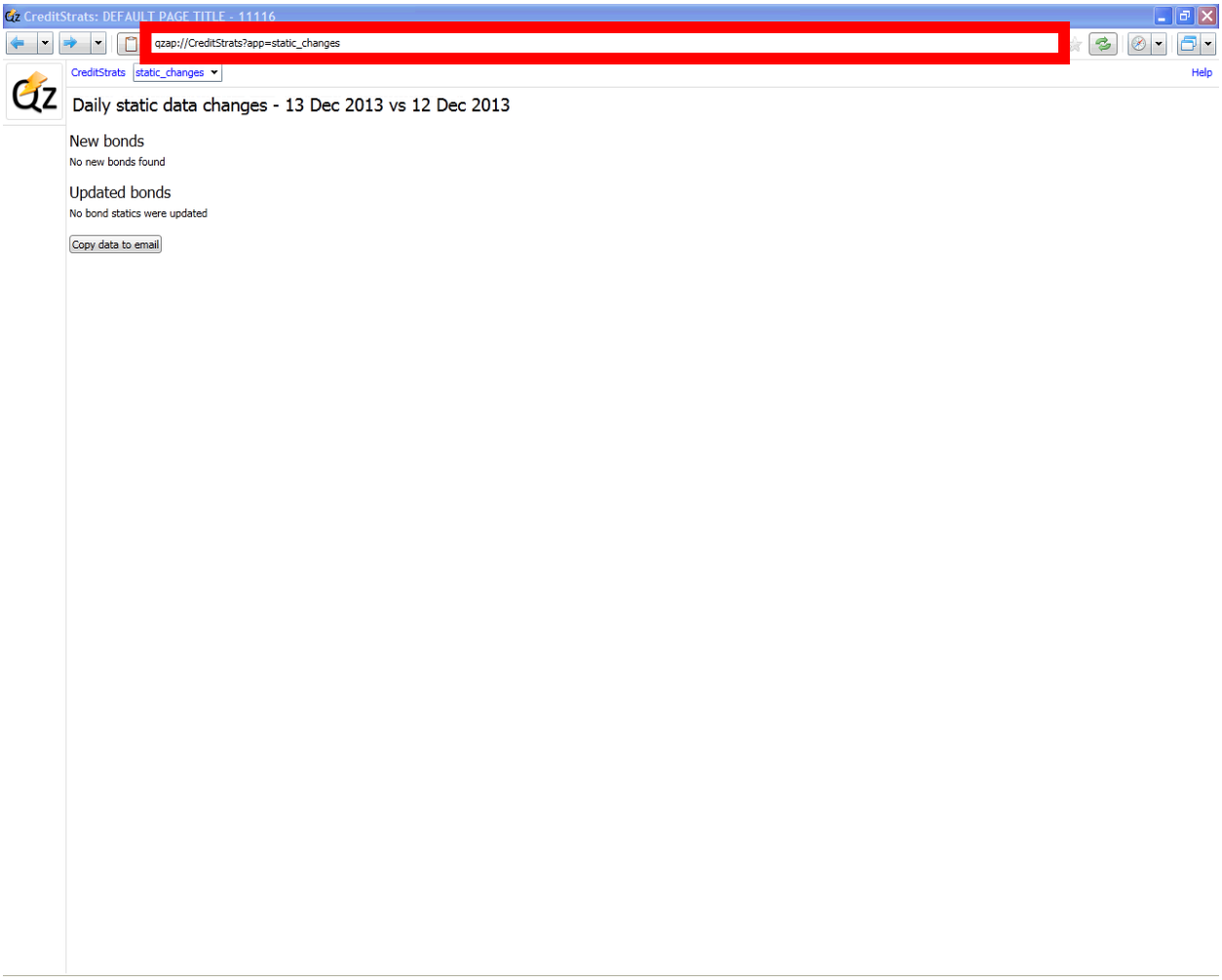

Figure 28. Qzap Application Screen

7.0 Conclusion

In conclusion, we were able to replicate the Bloomberg Professional bond and credit default swap Security Description Screen and the Yield and Spread screen, as well as create the foundation for the Credit Default Swap Valuation screen.

By retrieving the information we needed from the object database, Sandra, we were able to successfully replicate three out of the four screens. Both DES screens are able to pull data from Sandra about the selected bond or CDS and display it to the user. The YAS screen allows for users to edit fields such as the price, spread, yield, and benchmark yield, to name a few. Any

fields dependent on the changed fields will recalculate themselves, thus fields automatically updating themselves based on any changes made to the bond or benchmark bond. The CDSW screen is the only screen that is not fully functional - the calculations are off. Despite the incorrect calculations, the rest of the data displayed is correctly retrieved from Sandra, CDSs can be loaded, and fields automatically recalculate themselves when details of the CDS are changed such as the notional or spread values.

Despite running into some challenges along the way such as the cyclic dependencies among the fields of the YAS screen all four applications were developed and tested to completion. Upon completion, the code was reviewed by the credit dev team, accepted, and pushed into production. This means that all Bank of America employees that have access to the Qzap Browser, now have access to the four applications that were produced and can use them at their leisure.

Due to the seven week time constraint, we were not able to complete all aspects of the screens. We have included a [INSERT APPENDIX] “To - Do” list which includes the next steps for this project along with what needs to be improved upon had we been given more than seven weeks. With the completion of this project, Bank of America will be able to validate their in-house date by comparing their results to the results of Bloomberg in addition to enabling more employees at BAML to have access to this type of data analysis.

8.0 Appendix

I. Notes from Code Review with Stefano Cattani 2:30 PM 11/8/2013

Bond DES Screen & CDS DES Screen:

- Make attributes functions so there is a single place to make changes
- Clean up some of the code to better define the Model & View
- Clean up the ratings function, use a lambda statement for it
- Set up try/catch when loading a bond to catch when the loading fails or the bond isn't found

II. Notes from Meeting with Christopher Lawson and Andy Hudson 9:00 AM 11/15/2013

Our Project Is For:

- Research Team
- Trader Assistance
- Prove Quartz Analytics

Bloomberg:

- For anyone who wants live data via a GUI
- Very restrictive & watched
- Different Access Levels:
  - Bloomberg Anywhere: Can be used to access Bloomberg on any device (any computer, terminal, mobile, etc…)
  - Bloomberg Open: terminal machine, can only be used on a specific machine (can have multiple logins), cannot use remote desktop access
- Login Security
  - Biometric Logic
- Cost: $20,000 per person per year
- Bloomberg has the fixed income market, without Bloomberg a company/trader is "out of the market"

Thompson Reuters:

- Competitor of Bloomberg
- More relaxed security, can distribute data more
- Previously a news agency, uses that network to get its data
- Thompson has a deal with Libor to get its data before others

Thompson versus Bloomberg:

- Some data different
- Uses different markets to get data from so some spreads & analytics different
- Depends on market working in
- Small (insignificant) differences

Traders:

- Prefer Bloomberg over Thompson
- Chat feature is significant for Bloomberg
- Users are increasing for Bloomberg

III. Notes from Code Review with Stefano Cattani 11/28/2013

YAS Screen

- Make the identifier field more prominent
- Show description of benchmark bond instead of dropdown
- Tweak the market data date
- When changing the spread or benchmark bond yield, the bond in focus' yield should change (previously changing the benchmark bond's yield would trigger the spread to change)
- Changing the bond in focus' price should change its yield, not the spread
- Only make Z-Spread editable
- Remove OAS values and other value not being used (e.g. Yield Calculations section)
- Minor UI Tweaks

Bond DES:

- Remove Bloomberg prefix on ratings label
- Only display ratings with Bloomberg prefix

All:

Allow switching to related screen (ex: Bond DES to YAS) via buttons or Qzap links

IV. Notes for Meeting with Richard Jervis 3:00 PM 12/3/2013

We interviewed Richard Jervis, a Senior Trading Strategist I, in order to receive feedback on the application we developed. During this interview, we demoed the four screens of the application. When asked about the user interface, he agreed with our decision that making the application appear similar to Bloomberg will facilitate the switching between Bloomberg and the Qzap Application (Jervis, 2013).

The following sections contain screen shots of the screens during the time of the interview and bullet points of Richard Jervis's comments and suggestions for improvement for each screen. His time-permitting/long-term goal suggestions are denoted by '[Additional]'. These were additions that he thought would be helpful for users but are not needed to use any of the four screens.

Bond DES Screen

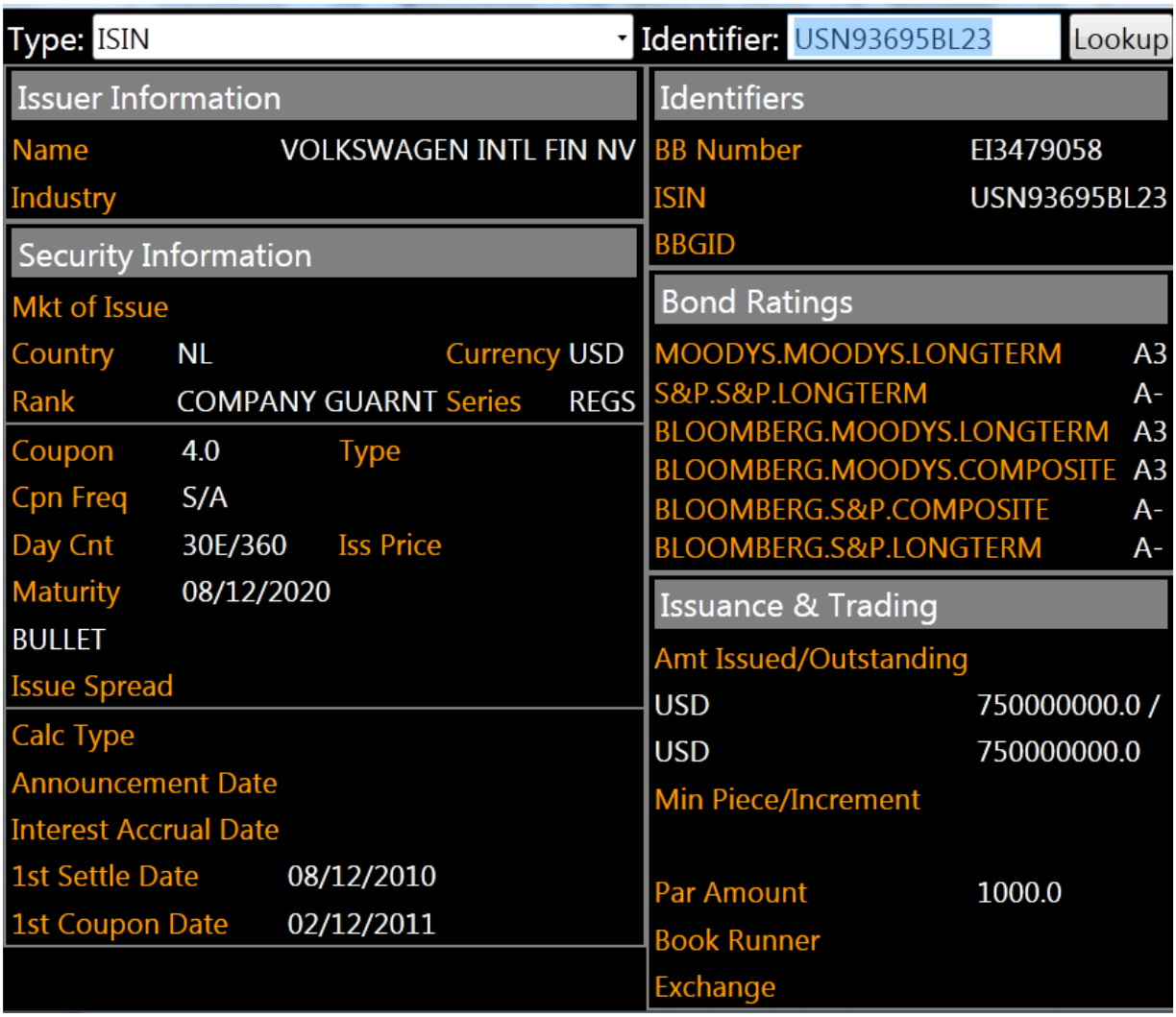

Figure 29. Bond DES Screen

- Add a field that gives a more detailed description of the rank
- Add call data for callable bonds
- Change static “BULLET” label
- Display only the Bond Ratings which are prefaced with ‘Bloomberg’
- [Additional] Link to a page displaying all children for the parent of the currently selected bond

YAS Screen

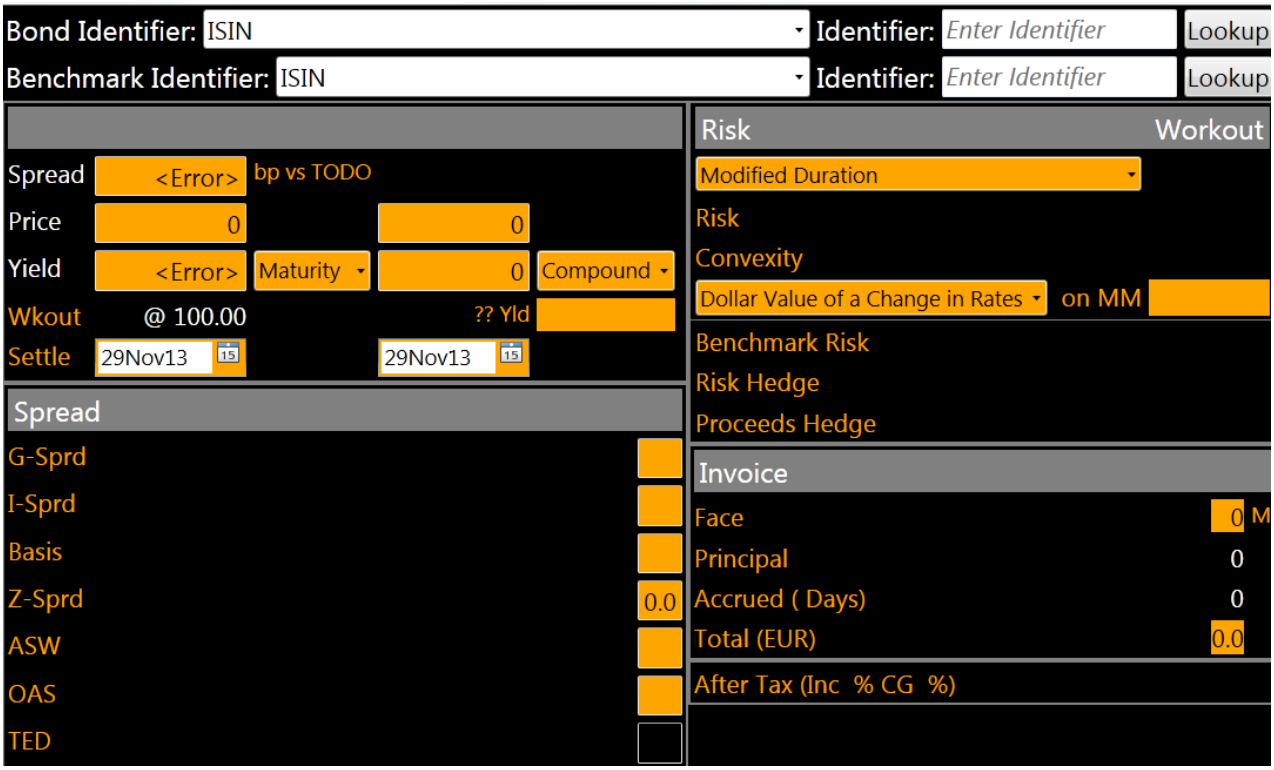

Figure 30. YAS Screen shot

- Capability to handle both callable and floating rate bonds

CDS DES Screen

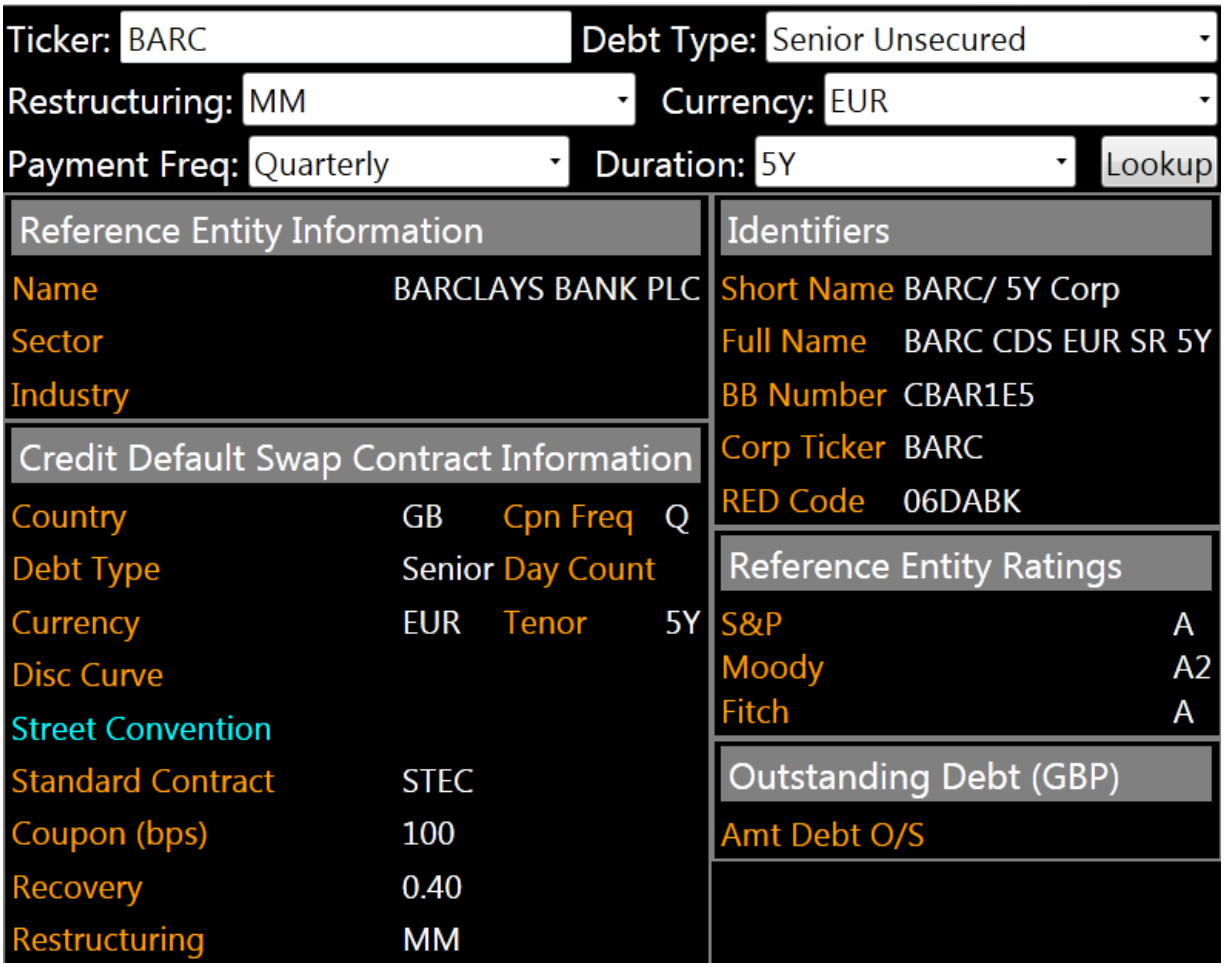

Figure 31. CDS DES Screen Shot

- Ensure that the CDS is liquid – Don't display non-liquid CDS
- [Additional] Link to a page displaying all children for the parent of the currently selected bond

CDSW and CDS DES Screen

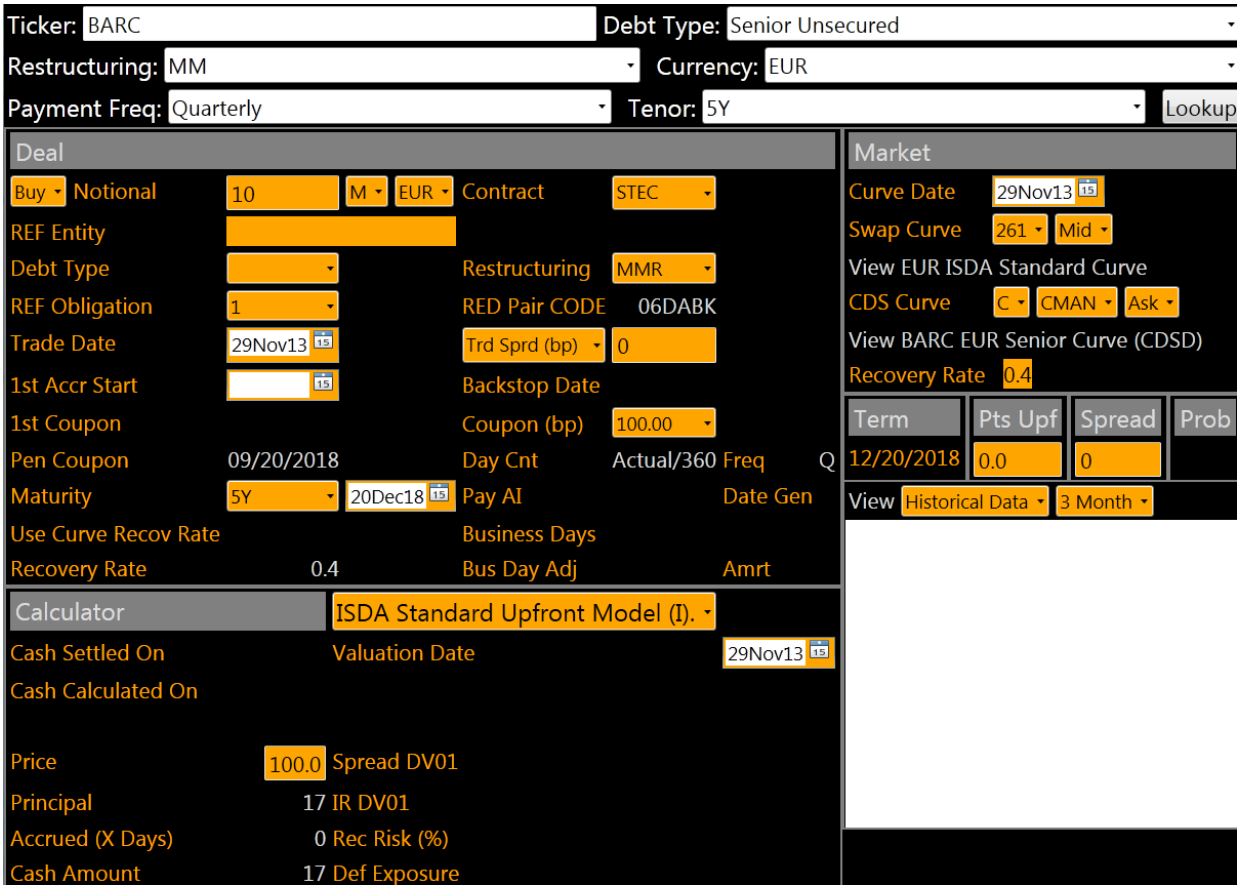

Figure 32. CDSW Screen Shot

- [Additional] New method of selecting the credit default swap properties (typing options instead of selecting from a drop down menu)

All Screens

- [Additional] Ability to Toggle between Bloomberg and internal terminology

The final screen shots of each of the screens are shown in chapter 5.0 Results. The specific changes made in regards to Mr. Jervis's recommendations are stated. Due to the time constraints of our project, we were not able to implement all of Mr. Jervis's recommendations.

V. To – Do List as of 12/6/2013

Working Functionality

Bond DES:

- Load Bond using different identifiers
  - If no bond found message is shown
- Display basic details about bond
- Can use a standalone app or in Qzap (Qzap://CreditStrats?app=bond_des)
- Can see selected Bond in YAS
- Sanitizes the identifier put before using it to search for a bond

CDS DES:

- Load CDS
- Display basic details about cds
- Can use a standalone app or in Qzap (Qzap://CreditStrats?app=cds_des)
- Can see selected in the CDS in CDSW

YAS:

- Load Bond
- Load Benchmark Bond
- Display spread, yield, price, and other values related to the bond
- Allow user to change values and have related fields automatically update
- Can use a standalone app or in Qzap (Qzap://CreditStrats?app=yas)
- Can see the DES screen for either of the two selected Bonds
- Sanitizes the identifier input before using it to search for a bond

CDSW:

- Load CDS
- See trading and calculated values for CDS
- User can change fields such as notional and spread and related fields will automatically be recalculated
- Can use a standalone app or in Qzap (Qzap://CreditStrats?app=cdsw)
- Can see DES screen for selected CDS
- Sanitizes the Ticker input before using it to search for a CDS

All:

- When values in search fields are changed the url (in Qzap) is automatically updated & can be used to bring up the same screen in another tab/window/instance of the Qzap browser
- When switching from one screen to another using the supplied buttons
  - Standalone apps will open a standalone instance

- Qzap apps will open a new Qzap tab with the app loaded

Note: for [x], x is the variable name in the Model

Errors, Bugs, Issues, & Needs Improvement

Bond DES:

- Bond Ratings: Shows long names directly from database
- Bullet [*BulletLabelVal*]: Is currently a static label, doesn't pull anything from the database
- Rank [*RankVal*] from DB/Bond doesn't match Bloomberg
- Announcement Date [*AnnouncementDateVal*] isn't getting any date value from the database (pulling BondReferenceData().AnnounceDate())

CDS DES:

- The selected tenor is not used

YAS:

- Workout Risk [*RiskWkoVal]* produces error rather than producing value
- OAS fields and yield calculations have been removed (but the code is still there)
- A description of the benchmark bond has been added to replace the benchmark bond dropdown [*benchmarkBondComboBox* in View, *BenchmarkBondType* in Model]

CDSW:

- Debt Type combo box [*FirstDebtType*] needs a dictionary to link different subordinations to one of the three possible choices
- ISDA Standard Upfront Model [*FirstISDA*] is only choice for dropdown
- No difference between "Buy" or "Sell" choices [*FisrtBuy*]
- None of the "Market" combo boxes are functional
- Pts Upf can't be changed
- Price can't be changed
- Pen Coupon [*PenCouponVal*] calculator assumes that the date for a:
  - BiMonthly is maturity date – 15 days
  - BiWeekly is 3 days before the maturity date
- Many of the combo boxes have a singular value or wrong choices as the selected value does not affect anything
- The value for DV IR01 [*IRVal*] is incorrect

All:

- User editable fields only force numeral values
  - Does not specify max input value
  - Does not specify min input value

Unimplemented/Not Completed

Bond DES:

Issuer Information:

- Industry [*IndustryVal*]

Security Information:

- Iss Price [*IssPriceVal*]
- Calc Type [*CalcTypeVal*]

Identifiers:

- BBGID [*BbgidVal*]

Issuance & Trading:

- Book Runner [*BookRunnerVal*]
- Exchange [*ExchangeVal*]

YAS:

Bond Details:

- List of Benchmark Bonds [*BenchmarkBondChoices*]
- Coupon Schedule [*YieldDurationChoices*]
- Type of Yield [*durationYieldValue1* & *durationYieldValue2*]
- Callable vs Maturity Dropdown? [*YieldDurationVal*]

Yield Calculations Section:

- Equiv
  - ComboBox [*EquivDurationVal*]
  - Value [*Equiv2Val*]
- Mmkt ComboBox [*MmktDurationChoices*]
- True Yield ComboBox [*ListofYieldCalc*]
- Current Yield [*CurrentYieldVal*]

Spread Section:

- G-Sprd [*GSpreadVal*]
- I-Sprd [*ISpreadVal*]
- Basis [*BasisVal*]
- TED [*TEDVal*]

Risk Selection:

- OAS Values (All):
  - Modified Duration [*ModOasVal*]
  - Risk [*RiskOasVal*]

    - Convexity [*ConvexOasVal*]
    - Dollar Value of a Change in Rates [*DvOasVal*]
    - Benchmark Risk [*BenchOasVal*]
    - Risk Hedge [*HedgeOasVal*]
- Workout
    - *Workout Modified Duration [*ModWkoVal*]
    - *Workout Convexity [*ConvexWkoVal*]
    - *Workout Dollar Value of a Change in Rates [*DVWkoVal*]
    - Workout Benchmark Risk [*BenchWkoVal*]
    - Workout Risk Hedge [*BenchWkoVal*]
    - Workout Proceeds Hedge [*ProceedsRiskWkoVal*]

Invoice:

- After Tax
    - Inc Percentage [*TaxIncPercentVal*]
    - CG Percentage [*TaxCGPercentVal*]
    - Value [*AfterTaxVal*]

CDS DES:

Credit Default Swap Contract Information:

- Day Count [*DayCountVal*]
- Disc Curve [*DisCurveVal*]

Outstanding Debt

- Amt Debt O/S [*AmtDebtOSVal*]

CDSW:

Deal

- REF Obligation [*FirstREF*]
- Use Curve Recov Rate [*CurveRecoveryRateVal*]
- Restructuring [*RestructuringVal*]
- Backstop Date [*BackstopDateVal*]
- Pay Al [*PayAlVal*]

Calculator

- Cash Settled On [*SettledOnVal*]
- Cash Calculated On [*CalculatedOnVal*]
- Accrued
    - # days [*AccruedLabel*]

ISDA Standard Upfront Model

- Rec Risk
  - Percentage [*RecRiskLabel*]
  - Value [R*ecRiskVal*]
- Def Exposure [*DefExposureVal*]

Market

- Swap Curve
  - Label [*StdCurveLabel*]
  - ComboBox 1 [*FirstSwap*]
  - ComboBox 2 [*FirstSwapTwo*]
- CDS Curve
  - Label [*SeniorCurveLabel*]
  - Combo 1 [*FirstCDSCurve*]
  - Combo 2 [*FirstCDSCurveTwo*]
  - Combo 3 [*FirstCDSCurveThree*]
- Prob [*ProbVal*]
- Type of Data [*GraphDataTypeVal*]
- Data Length Span [*GraphTimeFrameVal*]
- Graph
  - Data [*GraphData*] (Not connected in View, look for graphData)
  - Spec [*ChartSpec*] (In View, *graphSpec*)

VI. To – Do List of 12/16/2013

Working Functionality

Bond DES:

- Load Bond using different identifiers
  - If no bond found message is shown
- Display basic details about bond
- Can use a standalone app or in Qzap (Qzap://CreditStrats?app=bond_des)
- Can see selected Bond in YAS
- Sanitizes the identifier put before using it to search for a bond

CDS DES:

- Load CDS
- Display basic details about cds
- Can use a standalone app or in Qzap (Qzap://CreditStrats?app=cds_des)
- Can see selected in the CDS in CDSW
- Sanitizes the ticket input before using it to search for a CDS

YAS:

- Load Bond
- Load Benchmark Bond
- Display spread, yield, price, and other values related to the bond
- Allow user to change values and have related fields automatically update
- Can use a standalone app or in Qzap (Qzap://CreditStrats?app=yas)
- Can see the DES screen for either of the two selected Bonds
- Sanitizes the identifier input before using it to search for a bond
- Any date before today cannot be selected for the settle date for either Bond or Benchmark

CDSW:

- Load CDS
- See trading and calculated values for CDS
- User can change fields such as notional and spread and related fields will automatically be recalculated
- Can use a standalone app or in Qzap (Qzap://CreditStrats?app=cdsw)
- Can see DES screen for selected CDS
- Sanitizes the Ticker input before using it to search for a CDS

All:

- When values in search fields are changed the url (in Qzap) is automatically updated & can be used to bring up the same screen in another tab/window/instance of the Qzap browser

- When switching from one screen to another using the supplied buttons
  - Standalone apps will open a standalone instance
  - Qzap apps will open a new Qzap tab with the app loaded

Note: for [x], x is the variable name in the Model

Errors, Bugs, Issues, & Needs Improvement

Bond DES:

- Bond Ratings: Shows long names directly from database
- Bullet [*BulletLabelVal*]: Is currently a static label, doesn't pull anything from the database
- Rank [*RankVal*] from DB/Bond doesn't match Bloomberg
- Announcement Date [*AnnouncementDateVal*] isn't getting any date value from the database (pulling BondReferenceData().AnnounceDate())

CDS DES:

- The selected tenor is not used

YAS:

- Workout Risk [*RiskWkoVal]* produces error rather than producing value
- OAS fields and yield calculations have been removed (but the code is still there)
- A description of the benchmark bond has been added to replace the benchmark bond dropdown [*benchmarkBondComboBox* in View, *BenchmarkBondType* in Model]
- Settle Dates ([*SearchBenchSettleDate*] and [*SearchSettleDate*]), if changed from default, are not reflected in the Qzap URL

CDSW:

- Debt Type combo box [*FirstDebtType*] needs a dictionary to link different subordinations to one of the three possible choices
- ISDA Standard Upfront Model [*FirstISDA*] is only choice for dropdown
- No difference between "Buy" or "Sell" choices [*FisrtBuy*]
- None of the "Market" combo boxes are functional
- Pts Upf can't be changed
- Price can't be changed
- Pen Coupon [*PenCouponVal*] calculator assumes that the date for a:
  - BiMonthly is maturity date – 15 days
  - BiWeekly is 3 days before the maturity date
- Many of the combo boxes have a singular value or wrong choices as the selected value does not affect anything
- The value for DV IR01 [*IRVal*] is incorrect
- CDSI.FirstCouponDate always returns "None" [*StCouponVal*]

All:

- User editable fields only force numeral values
  - Does not specify max input value
  - Does not specify min input value

Unimplemented/Not Completed

Bond DES:

Issuer Information:

- Industry [*IndustryVal*]

Security Information:

- Iss Price [*IssPriceVal*]
- Calc Type [*CalcTypeVal*]

Identifiers:

- BBGID [*BbgidVal*]

Issuance & Trading:

- Book Runner [*BookRunnerVal*]
- Exchange [*ExchangeVal*]

YAS:

Bond Details:

- List of Benchmark Bonds [*BenchmarkBondChoices*]
- Coupon Schedule [*YieldDurationChoices*]
- Type of Yield [*durationYieldValue1* & *durationYieldValue2*]

Yield Calculations Section:

- Equiv
  - ComboBox [*EquivDurationVal*]
  - Value [*Equiv2Val*]
- Mmkt ComboBox [*MmktDurationChoices*]
- True Yield ComboBox [*ListofYieldCalc*]
- Current Yield [*CurrentYieldVal*]

Spread Section:

- G-Sprd [*GSpreadVal*]
- I-Sprd [*ISpreadVal*]
- Basis [*BasisVal*]
- TED [*TEDVal*]

Risk Selection:

- OAS Values (All):
  - Modified Duration [*ModOasVal*]
  - Risk [*RiskOasVal*]
  - Convexity [*ConvexOasVal*]
  - Dollar Value of a Change in Rates [*DvOasVal*]
  - Benchmark Risk [*BenchOasVal*]
  - Risk Hedge [*HedgeOasVal*]
- Workout
  - *Workout Modified Duration [*ModWkoVal*]
  - *Workout Convexity [*ConvexWkoVal*]
  - *Workout Dollar Value of a Change in Rates [*DVWkoVal*]
  - Workout Benchmark Risk [*BenchWkoVal*]
  - Workout Risk Hedge [*BenchWkoVal*]
  - Workout Proceeds Hedge [*ProceedsRiskWkoVal*]

Invoice:

- After Tax
  - Inc Percentage [*TaxIncPercentVal*]
  - CG Percentage [*TaxCGPercentVal*]
  - Value [*AfterTaxVal*]

CDS DES:

Credit Default Swap Contract Information:

- Day Count [*DayCountVal*]
- Disc Curve [*DisCurveVal*]

Outstanding Debt

- Amt Debt O/S [*AmtDebtOSVal*]

CDSW:

Deal

- REF Obligation [*FirstREF*]
- Use Curve Recov Rate [*CurveRecoveryRateVal*]
- Restructuring [*RestructuringVal*]
- Backstop Date [*BackstopDateVal*]
- Pay Al [*PayAlVal*]

Calculator

- Cash Settled On [*SettledOnVal*]
- Cash Calculated On [*CalculatedOnVal*]

- Accrued
  - # days [*AccruedLabel*]

ISDA Standard Upfront Model

- Rec Risk
  - Percentage [*RecRiskLabel*]
  - Value [R*ecRiskVal*]
- Def Exposure [*DefExposureVal*]

Market

- Swap Curve
  - Label [*StdCurveLabel*]
  - ComboBox 1 [*FirstSwap*]
  - ComboBox 2 [*FirstSwapTwo*]
- CDS Curve
  - Label [*SeniorCurveLabel*]
  - Combo 1 [*FirstCDSCurve*]
  - Combo 2 [*FirstCDSCurveTwo*]
  - Combo 3 [*FirstCDSCurveThree*]
- Prob [*ProbVal*]
- Type of Data [*GraphDataTypeVal*]
- Data Length Span [*GraphTimeFrameVal*]
- Graph
  - Data [*GraphData*] (Not connected in View, look for graphData)
  - Spec [*ChartSpec*] (In View, *graphSpec*

Bibliography

About. (2013, Oct. 15). Retrieved from Python.org: http://www.python.org/about

About Bank of America. (2013). Retrieved November 13, 2013, from Bank of America: http://about.bankofamerica.com/en-us/index.html

Autodesk. (n.d.). Retrieved December 5, 2013, from DAGNodes: http://download.autodesk.com/us/maya/2010help/Nodes/DAGNode.html

Choudhry, M. (2006). The CDS Basis I: The relationship Between Cash and Synthetic Credit Markets. In *The Credit Default Swap Basis.* Bloomberg Press.

Goyal, S. (2008). *Major Seminar On Feature Driven Development.* Technical University Munich. Retrieved from http://csis.pace.edu/~marchese/CS616/Agile/FDD/fdd.pdf

Jervis, R. (2013, December 3). Director - Senior Trading Strategist I. (J. Alvidrez, & D. Tocco, Interviewers)

Lawson, C., & Hudson, A. (2013, November 15). Assistant Vice President - Service Delivery Consultant. (D. Tocco, & J. Alvidrez, Interviewers)

Marshall, J. F. (2000). *Dictionary of Financial Engineering.* New York: Whiley.

Morris, V. B., & Morris, K. M. (2012). *Guide to Money & Investing.* Lightbulb Press.

Peters, T. (2004, August 19). *The Zen of Python*. Retrieved from Python: http://www.python.org/dev/peps/pep-0020/

Seward, Z. M. (2013, May 15). *This is How Much a Bloomberg Terminal Costs*. Retrieved November 15, 2013, from Yahoo! Quartz: http://finance.yahoo.com/news/much-bloomberg-terminal-costs-181855473.html

What Is a Bond? (n.d.). Retrieved from The Wall Street Journal: http://guides.wsj.com/personal-finance/investing/what-is-a-bond/