Early Warning System

for

Commerzbank

Version 1.2

Prepared by Klaudia Kloczewiak

31st July 2017

Table of Contents

Table of Contents ii

Revision History ii

1. Introduction 1

1.1 Purpose 1

1.2 Intended Audience and Reading Suggestions 1

1.3 Project Scope 1

2. Overall Description 2

2.1 Product Perspective 2

2.2 Product Features 2

2.3 User Classes and Characteristics 3

2.4 Operating Environment 3

2.5 Design and Implementation Constraints 3

2.6 User Documentation 3

2.7 Assumptions and Dependencies 3

3. System Features 4

3.1 Data Connectivity 4

3.2 Alert Conditions 4

3.3 Calculation Types 5

3.4 Email Notification 5

4. External Interface Requirements 6

4.1 User Interfaces 6

4.2 Software Interfaces 10

4.3 Communications Interfaces 10

5. Other Nonfunctional Requirements 11

5.1 Performance Requirements 11

5.2 Safety Requirements 11

5.3 Security Requirements 11

Appendix A: Proof Of Concept (PoC) 12

Appendix B: Data Cost Analysis 16

Appendix C: Next Steps 17

Revision History

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Date** | **Reason For Changes** | **Version** |
|  |  |  |  |
|  |  |  |  |

# Introduction

## Purpose

Credit Portfolio Management (CPM) team within Commerzbank requested a build for an in-house system that will alert credit analysts of any significant market and corporate alerts. This document describes the first release of the system and its initial functionality based on a MarkIT data feed.

## Intended Audience and Reading Suggestions

This document is intended for project managers, developers and users.

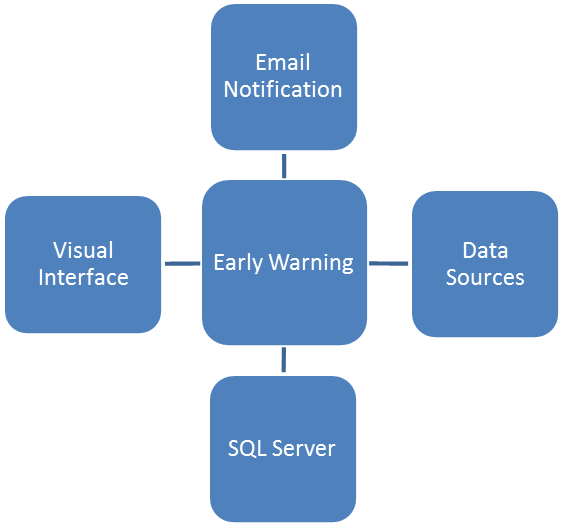
## Project Scope

This system will send notifications to users on market movements they track. The objective of the software is to enhance the processing time within CPM in response to market and corporate movements and news. With successful implementation of this tool, team will be able to adapt a quick response to changes that can affect their portfolio and adjust ratings as and when required.

# Overall Description

## Product Perspective

This is a new product that will be integrated with existing system Loan Flow Manager (Keats). The Early Warning system will share Keats’ database and user interface. New software will be connected to data feeds starting with MarkIT data for CDS and CDS index. In future releases this would be complemented for data such as shares, ratings, indices, financials and news.



Share Price

Equity Index

External

Rating

News

Financials

CDS

CDS Index

**MarkIt (EOD)**

**Potentially Bloomberg (EOD)**

**Potentially Lexus Nexus**

**Potentially Bureau Van Dijk**

**Loan Flow Manager   
(Keats)**

## Product Features

Users and their managers will be able to configure different alert conditions they would like to receive notifications for. The interface will be web based and accessible with a login and a password via Keats. Upon accessing it, users will be able to find a data point in relation to a corporate they are interested in and configure a condition upon which they would like to be notified of a change. Managers will be approving changes in tracking requests. Upon a breach of an alert condition, user tracking the alert will receive an email notification.

## User Classes and Characteristics

There would be 3 user classes within the system:

* Administrators
  + Full access to change any configuration and pages.
* Users
  + Able to create new alert conditions.
  + Able to request changes to existing alert conditions.
  + Able to request changes to their tracking configuration.
  + Able to leave comments for their alert notifications.
  + Able to change the configuration for corporates in their assigned sector. For example a user may be linked to a food sector and therefore is permissioned to change configuration of corporates in that industry.
  + Able to add news to each corporate obtained through their daily news monitoring and research.
  + Able to edit the company launch pad page for their corporates. For example user that owns food sector will be able to administer configuration for companies in that industry.
* Managers
  + Linked to users; managers should be able to do everything that users can with extra privilege to approve requests from users and bypass approval requirement for their changes.
  + Managers should also be able to assign also other users to alert conditions.

## Operating Environment

System will operate on a server with a data stored in MS SQL database. The code will be written in python and C# with use of ASP.NET framework. The system will have to co-exist with existing functionalities of Keats system that stores pdf summary of each of the corporate in CPM’s portfolio and is both accessible by Commerzbank and their clients.

## Design and Implementation Constraints

Software development will have to be constrained to usage of C# and MS SQL Server where possible as these are bank’s standards.

## User Documentation

The only documentation that should be delivered would be a manual for end users.

## Assumptions and Dependencies

1. Availability to build user interface within existing Keats application.

# System Features

## Data Connectivity

3.1.1 Description and Priority

Connection to data sources for data points mentioned in table 1. Upon reception of last data system should test it against price history in relation to specified alert condition.

Table 1: Required Market Data Points

|  |  |  |  |
| --- | --- | --- | --- |
| Data Point | Type | Source | Priority |
| CDS | End Of Day | MarkIT | Medium |
| CDS Indices | End Of Day | MarkIT | Medium |
| Share Price | End Of Day or Hourly Snapshot | To be defined | Low |
| External Ratings | End Of Day or Hourly Snapshot | To be defined | Low |
| Equity Indices | End Of Day or Hourly Snapshot | To be defined | Low |
| Financials | To be defined | To be defined | Medium |
| News | To be defined | To be defined | High |

3.1.2 Stimulus/Response Sequences

Once the data has been received (real-time feed) or scheduled (flat file) it then should be analysed against alert conditions specified by users.

## Alert Conditions

3.1.1 Description and Priority

System should be developed so that it stores relevant conditions for each alert. These should have options such as:

* Company it relates to
  + Dropdown of companies that CPM can monitor
* Price Type
  + What is the price type that the user is interested in monitoring in relation to the company
    - Share Price
    - CDS
    - External Rating
* Benchmark
  + Users should be able to choose how high or low their benchmark should be, for example benchmark 5 for calculation id 1 would be 5% movement.
* Over Period
  + Historic period against which the calculation should be applied to, for example 5 day over period for calculation id 1 would translate to % movement compared against the price from 5 days ago.
* Calculation ID Type
  + Which calculation the user would like to choose (e.g. 1 for % movement, 2 for standard rolling deviation)
* Maximum Frequency
  + What is the maximum time frequency at which the user would like to receive the alert. For example if the user would have 7 days maximum frequency that would mean the user would only receive the alert once a week if it would be occurring frequently.

## Calculation Types

3.1.1 Description and Priority

System should be flexible over its calculation types. In the first release the calculation types should cover the following:

* + - 1. Rolling standard deviation (over time)
      2. Rolling standard deviation relative to index (over time) \*
      3. % movement (over time)
      4. % movement relative to index (over time) \*

Future release should cover calculations such as:

* Calculation leveraging machine learning models based on market data, fundamentals and news.
* Change in external rating (any time)

\* Movement relative to index would be triggered if a market value would differ from its index at a specified benchmark for a specified period of time. Indices should be mapped as per the following to the corporates:

1. CDS:
   1. Itraxx Main for Europe & Africa Investment Grade
   2. Itraxx Cross Over for Europe & Africa Sub-Investment Grade
   3. CDX Investment Grade for US & Asia Investment Grade
   4. CDX High Yield for US & Asia Sub-Investment Grade

Where Investment Grade means a company with a rating equal or higher than BBB.

1. Share Price
   1. S&P 500 for America (All)
   2. FTSE350 for UK
   3. EuroStoxx 50 for Europe
   4. Nikkei 225 for Asia
   5. JALSH for Africa

## Email Notification

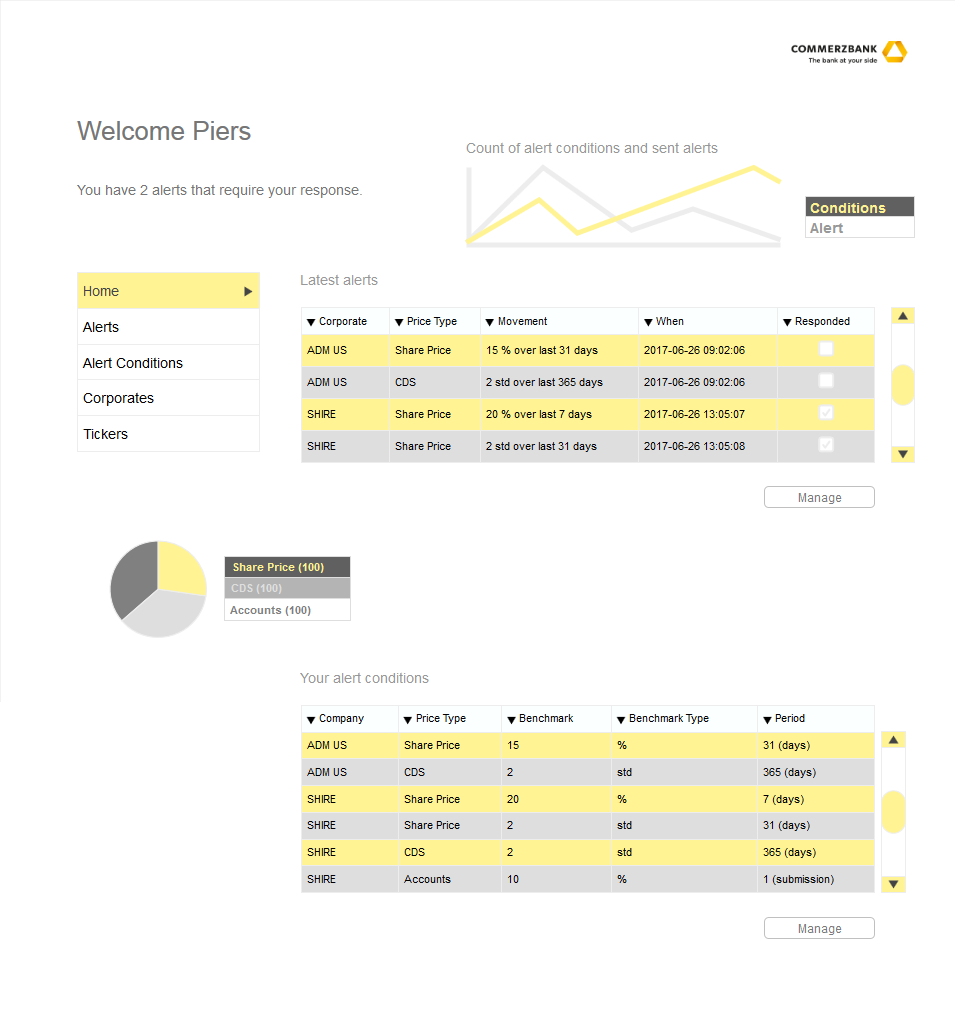
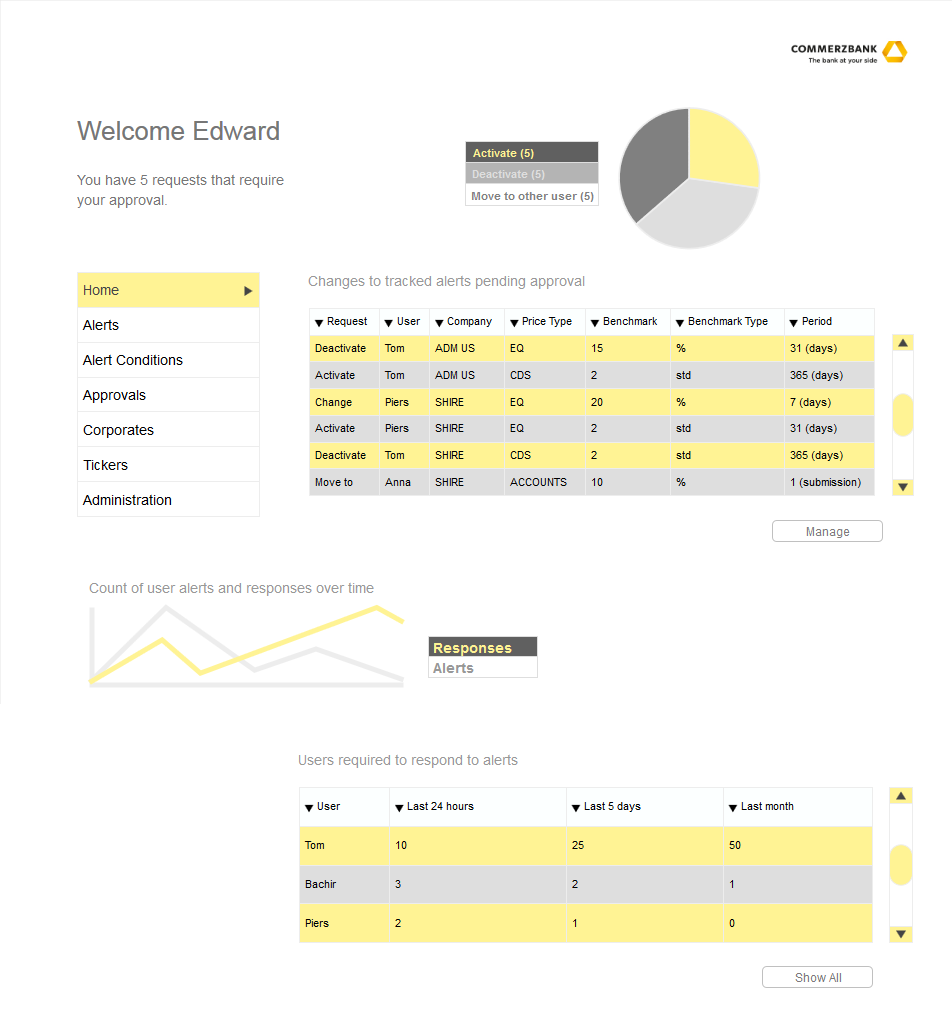
3.1.1 Description and Priority

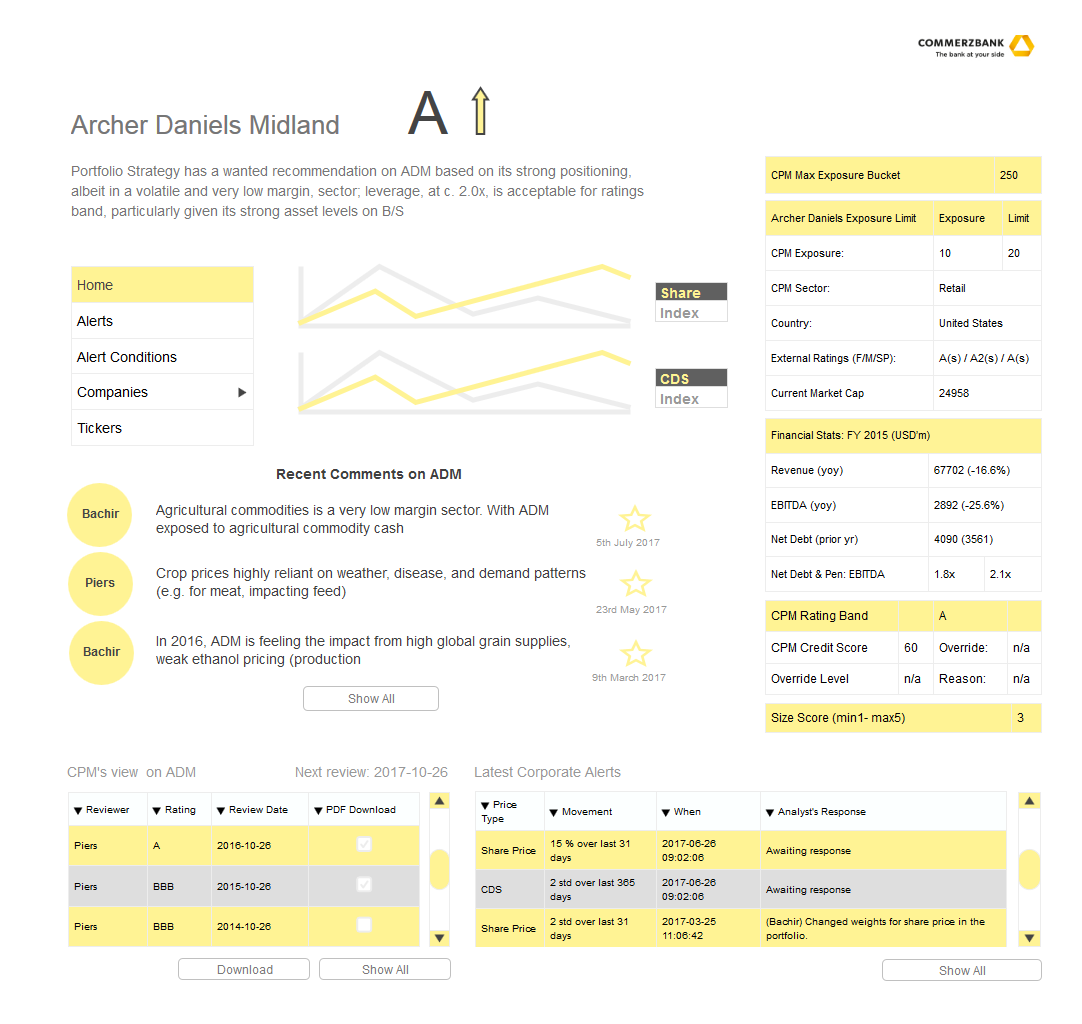
Upon breached alert condition users that track that condition should receive an email stating which alert has been breached, for example…

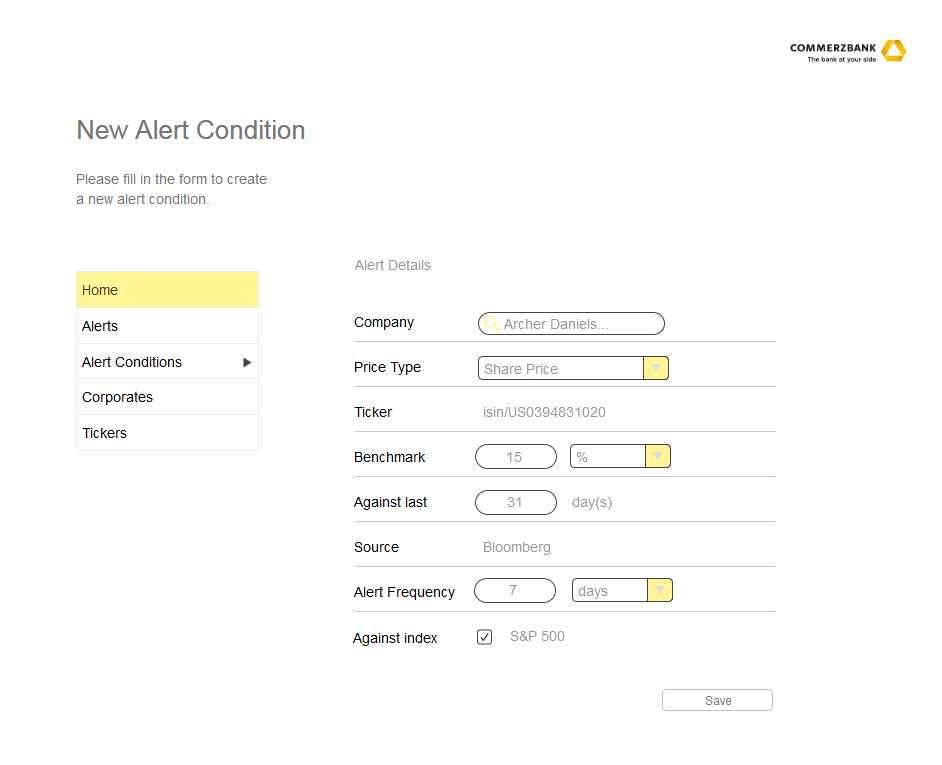
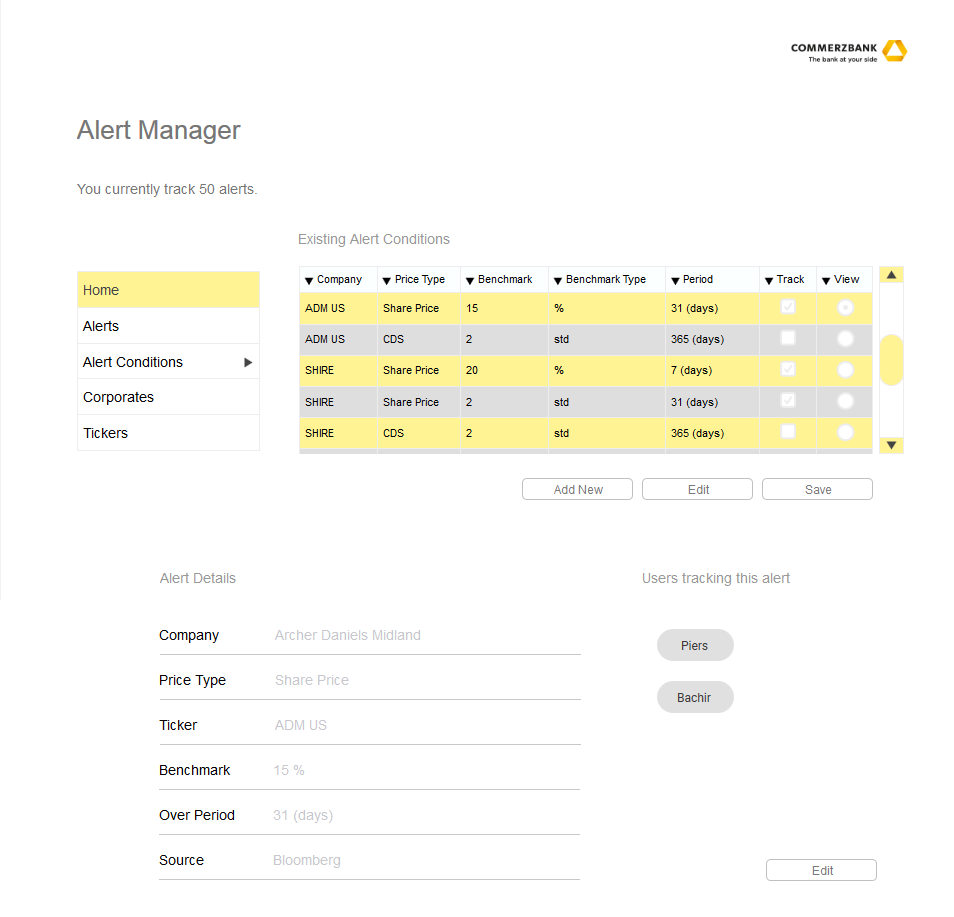
# External Interface Requirements

## User Interfaces

User interface will be developed within existing Loan flow manager / Keats application and will consist of the following example pages:

* Welcome Page (User)
  + Page outlining alerts that user should respond to and user’ alert conditions. Page could also show the graph of number of active alert conditions and email notifications over time.
* Welcome Page (Manager)
  + Page outlining alerts that manager’s reports should respond to.
* Company Launch Pad
  + Page for each of the company’s that CPM has in their portfolio or/and can monitor. It should consist of high level fundamental analysis including latest market cap, graph of share price vs equity index over time and graph of cds vs cds index over time.



* Alert Condition Manager
  + Page for managing alert conditions. This should also include a button that would open up a New Alert Condition Pop-Up Screen if user would like to create a new alert condition.

## Software Interfaces

This software will require access to:

* CPM SQL Server for Loan Flow Manager

Server: cpm-legacy-live\PRU\_P1

Databases:

* KEATS (read and write)
  + Tables: All
* PRUCVA (read only)
  + Tables: DCL.MarkIT\_CDS\_Composite\_Index, MarkIT\_CDS\_Composite
* TFS Server for Loan Flow Manager / Keats development environment
* Data Connectivity to source for CDS. With time it will also require connectivity to News, Fundamentals, Share Price, Equity Indices and External Ratings

## Communications Interfaces

This software will send automatic internal emails and therefore require access to:

Fixed IP

<https://serviceportal.intranet.commerzbank.com/src9/secure/main.jsp#search/service/%22Static%20IP%20Request%20for%20Win7%20CoBaClient%22>

SMTP MRelay Access Request

<https://serviceportal.intranet.commerzbank.com/src9/secure/main.jsp#search/service/%22Mrelay%20access%22>

# Other Nonfunctional Requirements

## Performance Requirements

System should be optimized so that the analysis of incoming data is timely. Delays should not extend beyond a couple of minutes.

## Safety Requirements

Loss of alert conditions and users that track them could result in slower performance of the CPM team. Data should be stored on a SQL server backed up and maintained by IT to stop that from happening.

## Security Requirements

System will need different types of users and authentication. Login page already exists within Keats, some of that functionality may have to be expanded. Depending on the data source used, system may have to store different details about each of the system users. For example what data they receive it and in what form (can they view a row data point?).

Keats system has access to both public and private side of the bank and has the necessary controls around that. When expanding the systems functionality to early warning report, development team should take extra steps to ensure that the private information is still secured.

Appendix A: Proof Of Concept (PoC)

This appendix outlines work completed during the proof of concept. The development has been created based on a real-time Bpipe data feed.

**Pre-requisite**

Firstly, a new application has been registered with the Market Data team within Commerzbank under the name “MarketAlert”. For testing purpose, system has been permissioned access to data for New York Stock Exchange Level 1 – Delayed (EID: 39491). This has been done to test development of system functionalities against an incoming share prices for ADM US (one of the corporates in CPM’s portfolio). In order to successfully connect to the data stream, Information Technology (IT) has created a firewall rule for the IP that application was tested from. Access via firewall has been tested in cmd with the following text “telnet bbg-gre-uat-mbpipe.lon.ib.commerzbank.com 8194”.

Secondly, IT fixed the IP that the application was accessed from. After that the SMTP MRelay access has been requested and granted for the IP. This has been completed so that the application would be permissioned to send automatic emails internally within Commerzbank.

Thirdly, the following python libraries were installed to deliver the PoC: blpapi, pandas, sqlite3, pypyodbc, datetime, smtplib.

**Development**

Code developed can be found in the attached “Proof Of Concept” zip file. Code has been developed in python 3.4 initially with sqlite3 database later moved to sql server.



The code has been developed with Bloomberg to subscribe to prices “subRequestBpipe.py” via Bpipe. The code subscribes to an isin that belongs to “ADM US” for which data is published by New York Stock Exchange. Upon reception of a price from Bloomberg, code takes the price to analyse. The following authentication was used in command line:

**--host=bbg-gre-uat-mbpipe.lon.ib.commerzbank.com --auth-type=APPLICATION --auth-name=CMZTest:MarketAlert  --auth-dirSvc=mail --port=8194**

The authentication details have changed since the PoC as per the following:

***The application name changed to "CMZTest:MarketAlert Test and Dev". Instead of name you can simply add the application ID instead of the application name. Input app ID "184626" which is the APP ID associated with "CMZTest:MarketAlert Test and Dev".***

The analysis is carried out in “benchmarkCalc.py” file. The code first imports the alert conditions stored in the “Alert\_Condition” table linked the incoming price. After that the price history is queried for maximum required lookback period, so that enough historical data is imported from the Price\_History table. Historical data is calculated against the incoming price and its configured benchmark. If there are any triggered alerts, the system queries users that track to these alerts from Users\_Alert table and then their email addresses from Users table. The next step is checking if the alert should be sent out in line with user alert configuration on alert frequency. If the alert has not been sent out lately to the user within the specified time limit, the alert is logged in Alert\_History table and an email is sent out to that user with “emails.py” file.

Description of tables created with test data under “MarketAlert” schema for the PoC:

* Corporate
  + Storing names and unique identifiers for all the corporates that CPM would be interested in monitoring.
* Security
  + Mapping table for Corporate tickers/name conventions and source to query data.
* Users
  + Place for storing user names, emails and unique identifiers.
* Alert\_Condition
  + Table for saving different alert conditions and their associated data point.
* Price\_History
  + Place for storing historical data for each data point that has active alert condition.
* Users\_Alert
  + Mapping table to assign users to alert conditions they need to monitor.
* Alert\_History
  + Table for storing all sent out alerts to users and storing their responses to these alerts.
* Corporate\_Details
  + Table created to store results of fundamental analysis by CPM. This table would cover some of the points published on Executive Summary on each of the corporates by CPM to Loan Flow Manager (Keats) system. This would then be displayed on Company Launch Pad.
* Corporate\_CPM\_Comments
  + Table created to store all the comments CPM analysts have on each of the corporates. This would then be displayed on Company Launch Pad.
* Users\_Manager
  + Table created for assisting the request process in the future development. Table stores each of the user’s assigned manager able to approve their requests and have extra rights within the application.

Description of table valued functions created under “MarketAlert” schema for user interface:

* Func\_Corporate\_Alerts
  + View that shows alerts for a specified corporate. To be displayed on the company launch pad.
* Func\_Corporate\_Comments
  + View that shows CPM comments on news for a specified corporate. To be displayed on the company launch pad.
* Func\_Corporate\_Details
  + View that shows CPM fundamental analysis for a specified corporate. To be displayed on the company launch pad.
* Func\_Manager\_Not\_Responded\_User\_Alerts
  + View that shows the number of alerts that has not been responded per user. To be displayed on the manager’s welcome page.
* Func\_User\_Alerts
  + View that shows alerts for a specified user. To be displayed on the user’s welcome page.
* Func\_User\_Alerts\_Conditions
  + View that shows alert conditions tracked by a specified user. To be displayed on the user’s welcome page.

**Other**

The code has been developed with Bloomberg to query historical data via Bpipe “historyRequestBpipe.py” which has not been utilized. The code can also be found in the “Proof Of Concept” zip file.

**List of completed items**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Phase** | **Order** | **Priority** | **Type** | **Description** | **Details** |
| 0 - PoC | 1 | High | Access | Registration of new application with Market Data team | Registered new application under the name of MarketAlert with below details to connect to Bloomberg:  '--host=bbg-gre-uat-mbpipe.lon.ib.commerzbank.com --auth-type=APPLICATION --auth-name=COMMERZBANK:MarketAlert  --auth-dirSvc=mail --port=8194' |
| 0 - PoC | 2 | High | Access | Registration of application for real-time market data with Market Data team | Requested access to the real-time market data feed via Bpipe. |
| 0 - PoC | 3 | High | Access | Setup for machine to have access to send automated emails | Requested and received fixed IP and access to SMTP relay in order to send automated emails within Commerzbank from the application. |
| 0 - PoC | 4 | High | Access | Setup for machine to pass Firewall in order to receive data | Registered machine's IP with IT to firewall rules. |
| 0 - PoC | 5 | High | Access | Real-Time Data Connection to Bloomberg | Established successful connection in python via Bpipe service for market data (delayed 20 minutes). |
| 0 - PoC | 6 | High | Access | Permission to a bundle of Bloomberg market data | Worked with Market Data team to access any data for testing. Received permission for NY stock exchange data. |
| 0 - PoC | 7 | High | Code (Logic) | Tested Reception of market data from Bloomberg | Run the code to query price for ADM US on NY stock exchanges and received latest values |
| 0 - PoC | 8 | High | Access | Registration of application for historical market data with Market Data team | Requested access to the historical market data feed via Bpipe. |
| 0 - PoC | 9 | High | Data | Infrastructure for alert system built | Build tables in sqlite3 with test data. The following has been set up as configurable: 1) Alert conditions (ticker, calculation type, benchmark, period of monitoring history) 2) List of users tracking alert conditions 3) Assigned manager to the user |
| 0 - PoC | 10 | High | Data | Test data created for alert system | Created test data in sqlite3 tables to work with python code for testing alert logic. |
| 0 - PoC | 11 | High | Code (Logic) | Logic for alert system built | Developed logic in python that triggers alert logic when a price is received from Bloomberg following below steps: 1) Checks what alert conditions has been setup for that price 2) Tests if the conditions has been breached 3) Looks up users that follow that alert condition 4) Checks in the history if that alert condition sent out an automated email in the last 48 hours 5) If it hasn't an email is sent out to the users and the alert is stored in the history table |
| 0 - PoC | 12 | Medium | Access | Setup for machine with development rights and required applications | Requested and received development rights including access to installation of Visual Studio for C# and installed MS SQL Server Mgmt Studio |
| 0 - PoC | 13 | Medium | BRD | GUI Design and Business Requirements Gathering | Designed a GUI displaying configuration of the alert system and data from the Loan Flow Manager |
| 0 - PoC | 14 | Medium | Access | Setup for machine to access CPM's SQL Server | Requested creation of tables for alert system on existing database that feeds the Loan Flow Manager and access to it. |
| 1 | 1 | Medium | Data | Migration of test data to CPM's SQL server | Move all data from current local database to CPM's database. |
| 1 | 2 | Medium | Code (Logic) | Change static data source from Sqlite3 to CPM's SQL server | Once data has been moved incorporate changes in the code. |
| 1 | 3 | Medium | Data | Additional infrastructure for alert system to make alert time frequency messaging intervals configurable | Currently the system will allow a specific alert condition to send a message only once every 48 hours. This period should be configurable per subscription. A new field needs to be created for this. |
| 1 | 4 | Medium | Code (Logic) | Additional infrastructure for alert system to make alert time frequency messaging intervals configurable | Once the field for message interval has been implemented, code needs to be adjusted to incorporate this. |

The following types mean:

1. Access – Obtaining access for user or application in order to proceed with development.
2. BRD – Scoping business requirements for application development.
3. Data – Creating data architecture; Creating and migrating data points.
4. Code (Logic) – Developing script in python/C# for back end functionality.

Appendix B: Data Cost Analysis

This appendix outlines production data costs analysed for Share Price, External Ratings and Equity indices. The results of this analysis can be found in the attached “CPM data costs” pdf file.



**Share Price**

Share Price costs were based on the attached “CPM Tickers” excel file and their relating Exchanges (“CPM EIDs” excel file”) in order to estimate their costs for delayed real-time data feed (Bpipe and Electron).



**Equity Index**

Index costs were based on the scoped requirement covering top level prices for the following 5 indices: S$P 500, FTSE 350, EuroStoxx 50, Nikkei 225, JALSH.

**External Ratings**

Ratings costs were based on the assumption that only S&P, Moody’s and Fitch would be required.

Appendix C: Next Steps

This appendix outlines next steps to be carried out for the development of the Early Warning system.

**List of items to be delivered**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Phase** | **Order** | **Priority** | **Type** | **Description** | **Details** |
| 2 | 0 | High | Code (Logic) | Logic for alert system (CDS) | Add script to run on a daily schedule analysis for CDS data from MarkIT stored in Keats database. |
| 3 | 3 | Low | Code (Logic) | Additional Logic for alert system on data compared to an industry index (CDS) | Add calculation module that will allow to set an alert condition on a price against its index (e.g. if a corporate’ was 5% higher/lower than its industry index for a week OR a corporate was 10% higher/lower every day for the past month) |
| 2 | 1 | High | Test | Historical alert test (CDS) | Run the code historically for a number of dates to see the number of alerts created. |
| 2 | 2 | Medium | Access | Setup for machine to access development environment of Loan Flow Manager | Requested access to TFS server and a Loan Flow Manager copy to work on GUI |
| 2 | 3 | Medium | Code (GUI) | User Landing Page | Create a page for a user to see all their outstanding alerts and alert conditions |
| 2 | 4 | High | Code (GUI) | Alert condition configuration | Create user interface for configuring user alert conditions that will require logins and write / edit functionality to database depending on their permissions. Managers should be required to approve request for changes in configuration. |
| 2 | 5 | Medium | Code (GUI) | Manager Landing Page | Create a page for a manager to see all unanswered alerts by users and requests for change in alert conditions |
| 2 | 6 | Medium | Code (GUI) | Manager request approval area | Create an area where managers will be able to approve / reject requests to change in alert conditions |
| 2 | 7 | Medium | Code (Logic) | Daily schedule to refresh alert condition status (active, deactivated). | Everyday code should check which alert conditions are still active (have users tracking them) and update them |
| 2 | 8 | Low | Code (GUI) | User Alert response area | Create a page that will allow users to leave response / comments to each alert they received |
| 2 | 9 | Low | Code (GUI) | Corporate Landing Page | Create user interface displaying all relevant information with regards to a corporate within CPM's portfolio (fundamentals, latest new collected by CPM, latest alerts, relevant prices and indices) |
| 2 | 10 | Low | Code (GUI) | Ticker configuration | Create a page that will store all tickers / prices / ratings that a user would be able to track and their relating corporate and data source |
| 2 | 12 | Medium | Code (GUI) | Corporate information configuration | Create a page for administrators to configure data for a corporate (fundamentals, rating, latest pdf, tickers, latest CPM news, etc.) |
| 2 | 13 | High | Data | Collect static data from business and import to the system | Collect all related data from CPM (corporate information, users and their managers, corporate owners within CPM, ticker data, required alert conditions) |
| 2 | 20 | High | Test | Test the system against a big number of tickers | Test if the system is optimised for speed on all population of all tickers under alert conditions. |
| 2 | 21 | High | Test | Develop unit tests | Develop unit tests for different parts of the code that would run overnight and with every commit |
| 2 | 25 | High | Test | End User testing | Hand over the system to users and ask for feedback. |
| 2 | 26 | Medium | Code (Logic) | Productionise the code - Release 1 | Release the system |
| 3 | 0 | High | BRD | Choose Data Source (News) | Analyse which data source would be best for News and choose one. |
| 3 | 1 | Medium | Access | Registration of application / gain access to news data source with Market Data team | Request and test access to new data feed. |
| 3 | 2 | Medium | BRD | Collect different scenarios for relevant news data | Speak to business about different news they would be interested in being alerted about. |
| 3 | 3 | High | Code (Logic) | Machine Learning on News | Develop a model that could find the most relevant data points. |
| 3 | 4 | Medium | Code (Logic) | Additional Logic for alert system on Corporate news | Develop logic for monitoring corporate news data that would be available as an alert condition |
| 3 | 5 | Low | Test | Historical alert test (News) | Run the code historically for a number of dates to see the number of alerts created. |
| 3 | 6 | Medium | Code (Logic) | Productionise the code - Release 2 | Release the system |
| 4 | 0 | High | BRD | Choose Data Source (Fundamental Financials) | Analyse which data source would be best for Fundamental data and choose one. |
| 4 | 1 | Medium | BRD | Collect different scenarios for corporate data and their solutions | Speak to business about different formats of financial data and how it could be managed. |
| 4 | 2 | Medium | Access | Registration of application / gain access to fundamental / financial data source with Market Data team | Request and test access to new data feed. |
| 4 | 3 | Medium | Code (Logic) | Additional Logic for alert system on Corporate Financials | Develop logic for monitoring corporate financial data that would be available as an alert condition |
| 4 | 4 | Low | Test | Historical alert test (Fundamentals) | Run the code historically for a number of dates to see the number of alerts created. |
| 4 | 5 | Medium | Code (Logic) | Productionise the code - Release 3 | Release the system |
| 5 | 0 | Low | BRD | Choose Data Source (Share Price, External Ratings and Equity Indices) | Analyse which data source would be best and choose one. |
| 5 | 1 | Low | Access | Registration of application for real-time news data with Market Data team | Request and test access to new data source. |
| 5 | 2 | Low | Code (Logic) | Data Source Connectivity (Share Price, External Ratings and Equity Indices) | Adjust the code for new data source connectivity. |
| 5 | 3 | Low | Code (Logic) | Additional Logic for alert system on data compared to an industry index (Equity Indices) | Add calculation module that will allow to set an alert condition on a price against its index (e.g. if a corporate was 5% higher/lower than its industry index for a week OR a corporate was 10% higher/lower every day for the past month) |
| 5 | 4 | Low | Code (Logic) | Additional Logic for alert system to monitor external ratings (Moody's, S&P, Fitch) | Add calculation module to monitor movements in external ratings. Any movement should trigger an alert. |
| 5 | 5 | Low | Test | Historical alert test (Share Price, External Ratings and Equity Indices) | Run the code historically for a number of dates to see the number of alerts created. |
| 5 | 6 | Low | Code (Logic) | Productionise the code - Release 4 | Release the system |

The following types mean:

1. Access – Obtaining access for user or application in order to proceed with development.
2. BRD – Scoping business requirements for application development.
3. Data – Creating data architecture; Creating and migrating data points.
4. Code (Logic) – Developing script in python/C# for back end functionality.
5. Code (GUI) – Developing script in python/C# for front end functionality, specifically user interface.
6. Test – Establish success criteria and test developed items against these criteria.