DIT029 - Final Documentation

HT2013 - Project: Software Architecture for Distributed Systems
Group Blue

By: Carl Nätterdal, Ale Lotström, Lídia Nyman, Georgij Marchulija, Ylva Fonselius, Daniel Hosseini, Odzaya Batsaikhan.
Instructor: Björn Olsson
Supervisor: Khashayar Abdoli
Scrum Master/Project Manager: Carl Nätterdal(carl.natterdal@gmail.com)
Software Engineering Department
Gothenburg University

Table of Contents

Introduction

This document is the final report of the course for Software Architecture for Distributed Systems of Gothenburg University. The course is part of the Software Engineering & Management Bachelor Program Term 3.

It aims to provide a detailed description of the project with sprint plan, software architecture, software design specification, system requirement specification, test plan, test report, UI design report and technology report and time report.

The objective of this project is to let the students be divided into different project roles and handle their specific tasks for the roles as well as organizing and dividing the work based on the Scrum Methodology, taking responsibility during the development of different applications.

1. Sprint Plan

1.1 Scope

The scope of this project is to create a multi-platform business intelligence system with a backend data storage that serves three front-end applications (website, desktop and android). which allows end users to keep a track of stock market and to have an easy and quick access to basic information such as news and stock quotes for the specific stock. To be able to archive this we started off looking for sources able to return the info we needed.

- Google Finance
- Yahoo Finance
- Markitondemand

The sources above are being used to extract the stock related information we need in order to archive this goal. the data extracted will be parsed through our backend, written in Erlang and then proceeded to be loaded onto our database.

1.2 Website

To archive a minimalistic and user friendly website, we used the latest HTML5, CSS3 technologies. The goal here is to make a website that anyone can easily used and not to be confused by the website and its functions. The search functions and RSS are created in Ruby and JavaScript.

1.3 Desktop

The desktop application is made in Java, while we know that there are many restrictions with java's GUI. This is something that we had to choose for the best of the team due technical knowledge. The design approach will be making a similar design of the website for a user friendly layout.

1.4 Android

For the mobile application we wanted it to be user friendly and taken into account of various sizes of a mobile phone. This means we can't design an application that works fine on one resolution and not another. Everything should be visible and clickable in all different resolutions. The application itself it's programmed in Java.

1.5 Process Organization

1.5.1 Process Model

For this project we will be using Scrum as our process model. Scrum fits perfectly for our project because of the uncertain nature of the project, what I mean is that the fact that our group is new to most of the requirements of the project. We will be trying and learning a lot new things along the way, this also means that there are bound to be changes in the project. By working in iterations (Sprints, a period of 2 weeks in our project) we will be able to better assign more fitting assignments and improve on our project plan over time.

1.5.2 Project Roles

After the introduction of project, we sat down as a group and started divide the work between the group. First off we decided to assign different roles suitable for each member. It was very straightforward because most of us have been working together since term 1. This means we knew eachothers qualities and flaws with this we could divide the roles accordingly. We let the new member take the roles they previously had or similar ones.

- Carl Nätterdal - Project Manager/Scrum Master & Programmer.
- Ylva Fonselius - User Interface Designer & Programmer.
- Lidía Nyman - Quality Manager & Programmer.
- Georgij Marchulija - Designer & Programmer.
- Ale Iotström - Backend Designer & Programmer.
- Daniel Hosseini - Technology Manager & Programmer.
- Odzaya Batsaikhan - Software Architect and Programmer.

Project manager/Scrum Master

The job is to develop a project plan and make sure that everything during the project proceed as planned, make sure the schedules are followed and members registering their time and provide necessary tools for the members or help the project proceed when problems arise, enforcing change management. The project manager also has to hold daily meetings with the group, make sure everyone is proceeding as planned.

User Interface Designer

The user interface designer is responsible for gathering information about different technologies and choosing the most appropriate one to the project. He/she creates the designer interface for the system and mockups.

Quality Manager

The quality manager is responsible for creating test cases according to the project requirements, and running those tests in early stages and/or regularly through the project development. He/She is responsible for assure the quality of the system.

Backend Designer

The role of the backend designer is to define the technologies and data behind the front-end system. For instance choose the server, structure the database, the ETL process (extract, transform and load).

Designer

The designer is responsible for the overall design of the system. He will be working together with the backend designer for the overall design of the whole project.

Technology Expert

The role of the technology expert is to choose the most suitable tools for the project and its members. He/she should also give pro's and con's for the different programming language, tools.

Software Architect

The software architect is responsible for creating the software architecture of the system based

on BI related architecture. He/She is responsible for drawing the system components

1.6 Management Methods, tools and techniques

1.6.1 Time management

To keep a better track of the progress, every member of the development team were required to log their individual working time during the sprints. It should have a brief description of what they have done and how many hours. The tool we used for time tracking is called OnTime (http://www.ontimenow.com/). OnTime is an agile scrum based project management tool and is therefore fit for this terms project. The result of this projects working hours is included in the in the final hand-in folder(See Time report.csv)

1.6.2 Risk Management

The biggest risk during this project is the fact that it is a big project, where we have to early on divide the work between different people with different knowledge. Obviously we try our best to give people the most suitable work so that everyone can handle their part. But this doesn't eliminate the fact that it won't be any risk.

Example on this is the backend, for everyone in our group, erlang is a totally new language, it's functional and not objective oriented like java. There is a big risk that the people assigned to backend won't be able to handle it, this means we have to assign people good at erlang to do the backend (see change management). Not only to handle the risk but also make sure the backend is as good as it can be. This applies to the overall project and it's a major risk we are prepared for and thanks to the iterative process of scrum, we will be able to embrace changes.

This term we picked up new people for the project, this is also a risk where we are working in a different environment from their previous groups as well as communication between the members. To assign this risk we paired the new people up with older members to let them experience how we work in our group and interact with the members.

Another risk is related to availability of the files and also the security of it. Same as previous terms, this is something that must be considered. In case of someones hard drives fails, someone quitting the program or anything else that can ruin the files. This risk is handle by using tools that we used previous terms a shared dropbox for all programming related files(source codes), the other documentations are shared on our google drive.

1.6.3 Change Management

Change management has played a major role in our group's success and accomplish everything we set out to do. Without the changes, we might not have finished everything that we wanted or on time. Starting from our group, while a majority of the members have been together since we started the program we have over time taken new people into the group. Older members can get too comfortable with the way things are being done and the new members can have trouble matching the new groups way of working. All these things are important to taken into account and if something is not working, it's time for a change.

This term's project is by far the most complex project yet. The project itself can be divided into many categories and even with careful planned and assigned tasks. The person/people assigned to the task might not be able to handle it. I think it's important as the project manager to make sure the project moves forward.

When something like this happens, I follow a few steps in order to fix the problem. First off when a tasks isn't moving along for a while (Excluding very busy weeks with assignments/exams for everyone) I try to look at the task and provide information that I can find. If that doesn't work I'll look into alternative ways of solving it by adding extra help (other people who has finished their part). Due to the fact that we are using Scrum as our process model, it makes it much easier to add people after the sprint or switch people around who has completed their part.

Following the change it's important to communicate with the team, I usually takes it up after the daily meeting when change happens. This happens also when the requirement changes for a task. First off explaining the consequence of not changing, how it affects the group and the project, secondly involving the team in the decisions to change and last but not least explain the changes and making sure everyone understands why we have to change and how we are going to do it.

1.7 Sprint summary

Sprint 1
Starting up the project, setting up OnTime(Scrum management tool), make sure everyone knows their roles and what needs to be done in upcoming sprints in a general sense.

Sprint 2
Due to most parts of the project is new to us, we had to do a lot of research and overall reading for the parts. The first mockups were done during this sprint. People had trouble remembering registering time and I had to talk with the members and make sure they are registering after working. Erlang has been giving trouble for a few members in the group.

Sprint 3
The sprint where 2 members were not able to work much, one of the member was inactive due

to surgery and the other one's computer broke down. But this was not a big problem, because we knew the member was going to have surgery so it was planned with this sprint. Other members could take over and do her part and move the project forward. But the computer breaking down was unexpected. This didn't cause a big issue either due to risk management where we knew something like this could happen and everything was stored on dropbox so it didn't affect the progress of the project. This sprint also had the erlang assignment and erlang lectures from 9-17 which definitely slowed down the group.

Sprint 4
This sprint was predicted to be a tough one for the team, we had re-submission of erlang and the erlang exam. Members tried to work as much as possible, rss were added and diagrams was implemented. ETL was parsing symbol data for different stock markets in both csv/txt.

Sprint 5
Erlang course is now over and we started focusing on the project. The ETL encountered problem and we have re-think the way we wanted to structure the backend. We managed however to extract the data from different market after requesting it via ticker. The desktop having problems implementing the charts on the same page. Website added diagrams and charts. The basic android app is in the making.

Sprint 6
Finally the group is functioning very well, we are more at school together. Helping eachother with problems. Mentioned in previous java were having problems and we had to remake the application to better structure it and we had to implement the charts in a different way than we wanted initially. The website is being uploaded online. The android has found diagram libraries and started on search. All three applications are connecting to the local db to test search function. The new designed ETL is made and functioning, but there were minor bugs and the database updating/creating was not working.

Sprint 7
Due to working together efficiently in school everything was finished during this sprint and we started doing documentation drafts. ETL fixed the database updating/creating. The desktop application connected to online DB, parsing returned data, specific rss news and displaying everything on the diagrams. The website connected to the online DB, parsing returned data and display everything on the diagrams. Android application put everything together same as the desktop application.

Sprint 8
Minor bug fixes in the application, uploading daily stock data for the database. Overall all members are working on their part of the final documentation and also the final presentation preparation.

2. Software Architecture

Reference template:
(http://www.ecs.csun.edu/~rlingard/COMP684/Example2SoftArch.htm)

2.1 Introduction

2.1.1 Purpose

The purpose of software architecture document is to cover project aspect from architectural point of view. Diagrams included in this document are of general character and represent information in easy and understandable way.

2.2 Overview

This document uses 4+1 architecture view model consisting of:

2.2.1 Logical view.

This view represents the functionality that is provided for the end users of the system. It addresses developers and clients and is represented with Sequence Diagram.

2.2.2 Physical view.

This view represents the hardware and software components used within system. Components and connections between them are represented in Deployment Diagram of UML Deployment Diagram notation.

2.2.3 Process view.

Process view represents activity and communication between processes of the backend of the BI system. Communication is represented with Activity Diagram.

2.2.4 Development view.

Development view represents system from programmers point of view and is targeted for developers. It`s represented with packages and three different layers which include **Data Layer, Logic Layer and User Interface Layer.**

2.3 Stakeholders and concerns

In this project, stakeholder is anyone who takes participation in developing, testing and

deployment of the system, as well as supervisor and an examiner.

Main concerns for the developers is ease of maintainability of the system, as well as ability to modify and upgrade the system. Main concern for customers is ease of use and access of the system through different clients, which include platform independent desktop application, Web client and Android application.

2.3.1 Stakeholders

There are five major stakeholders of the system.

- Developers - people developing the BI system.
- Tester - person responsible for testing and quality.
- Supervisor - person who followed development of the product from the beginning, guiding developers team to meet requirements and providing with required information.
- Examiner - person who grades the product.
- User - final users of the product, who interact with the system.

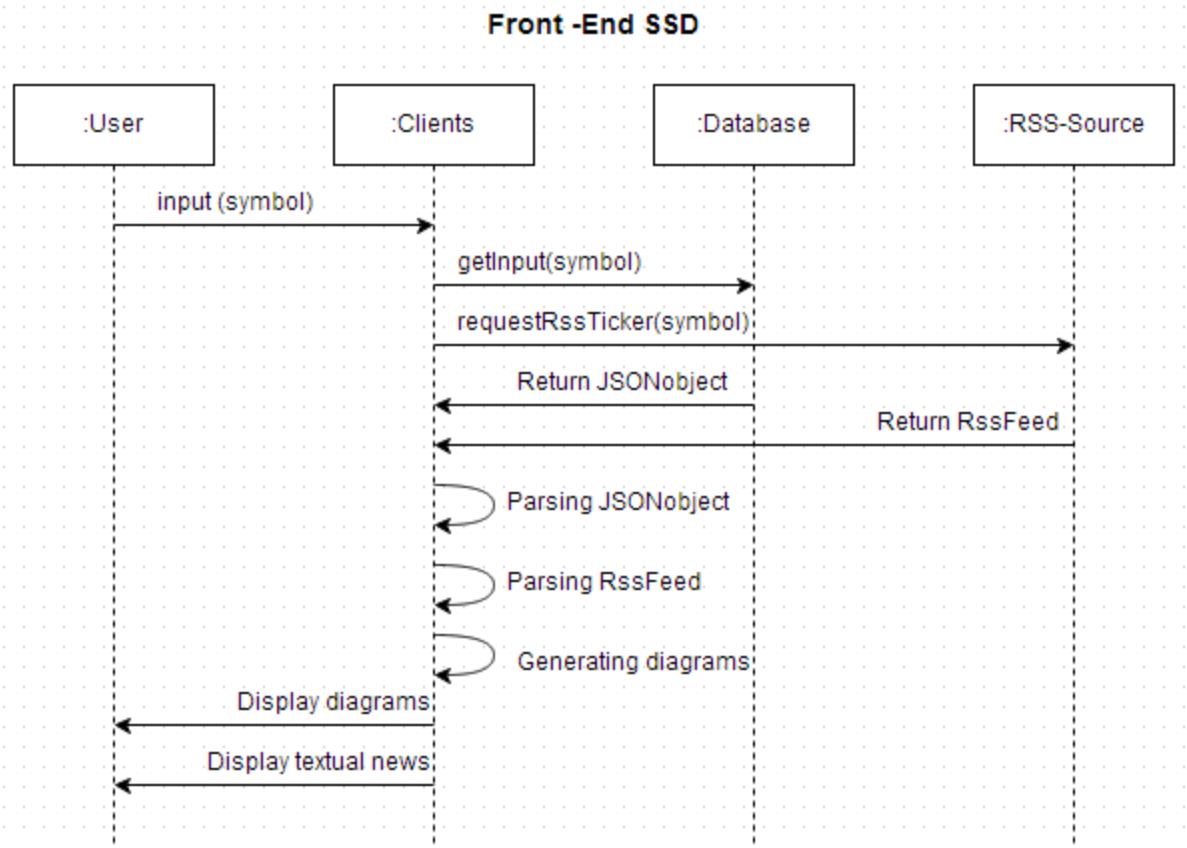
2.3.2 Quality attributes

The quality attributes of the system include:

- performance - the system should not take more than 2 second to display search results.
- maintainability - the system should be easy to maintain and upgrade.
- reliability - the system should be up and running constantly.
- multi-platform - the desktop application should be platform independent.
- user friendly - the system should be easy and intuitive to use.
- secure - the data on the database should be securely stored, and clients should use getters and setters to pattern-match username and password.

2.4 Logical View

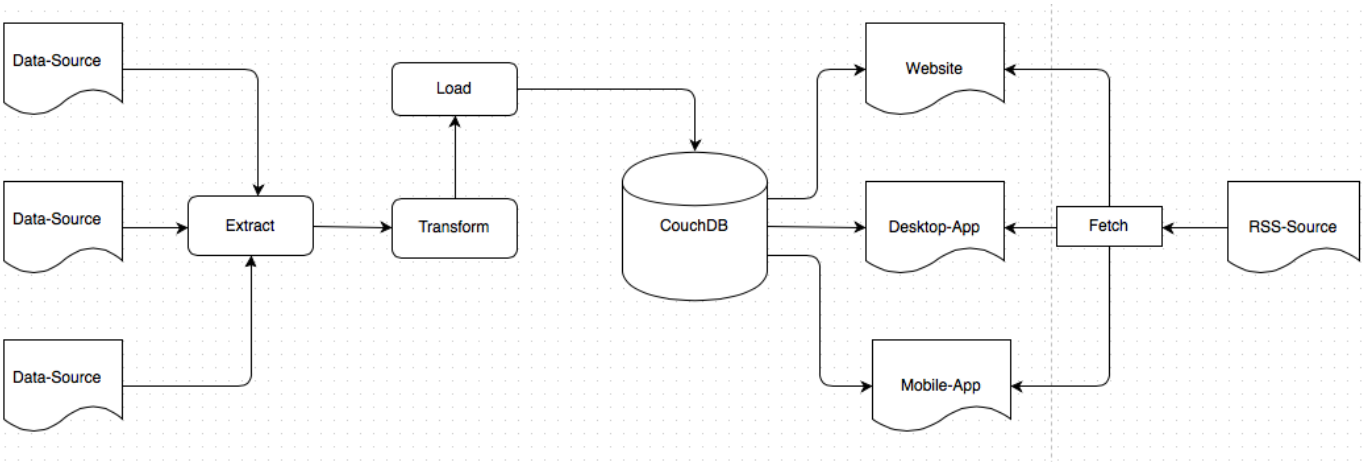
Diagram 2.1 - Sequence diagram



The logical view is concerned with functionality that system provides to the end user. The interaction between user and system is represented with sequence diagram. The interaction starts when user inputs the symbol to the client, here, client might be desktop application, mobile application or the website. After performing sequence of operations on the back end, client outputs information in forms of graphs and textual news.

2.5 Physical View

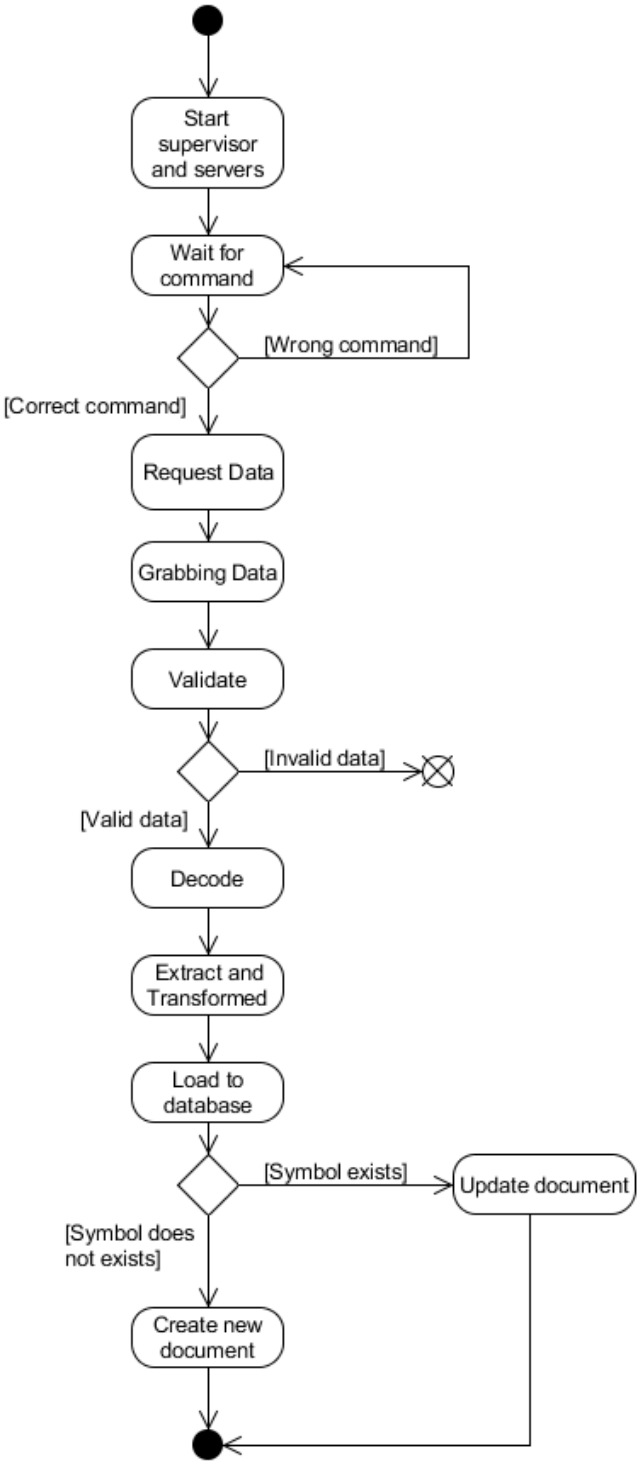
Diagram 2.2 - Physical View.



Physical view diagram represents the hardware and software components and interaction between components.
The process starts with extracting data from three different sources, in parallel with extracting textual data from RSS source. Once data is extracted from three sources it`s being transferred to JSON format and loaded to non-relational cloud database. Once data is stored clients (website, desktop application and android app) access database and retrieve data as a JSON objects and parse it to represent it in understandable for user manner. In parallel, RSS source is fetched and news are represented for the searched keyword.

2.6 Process View

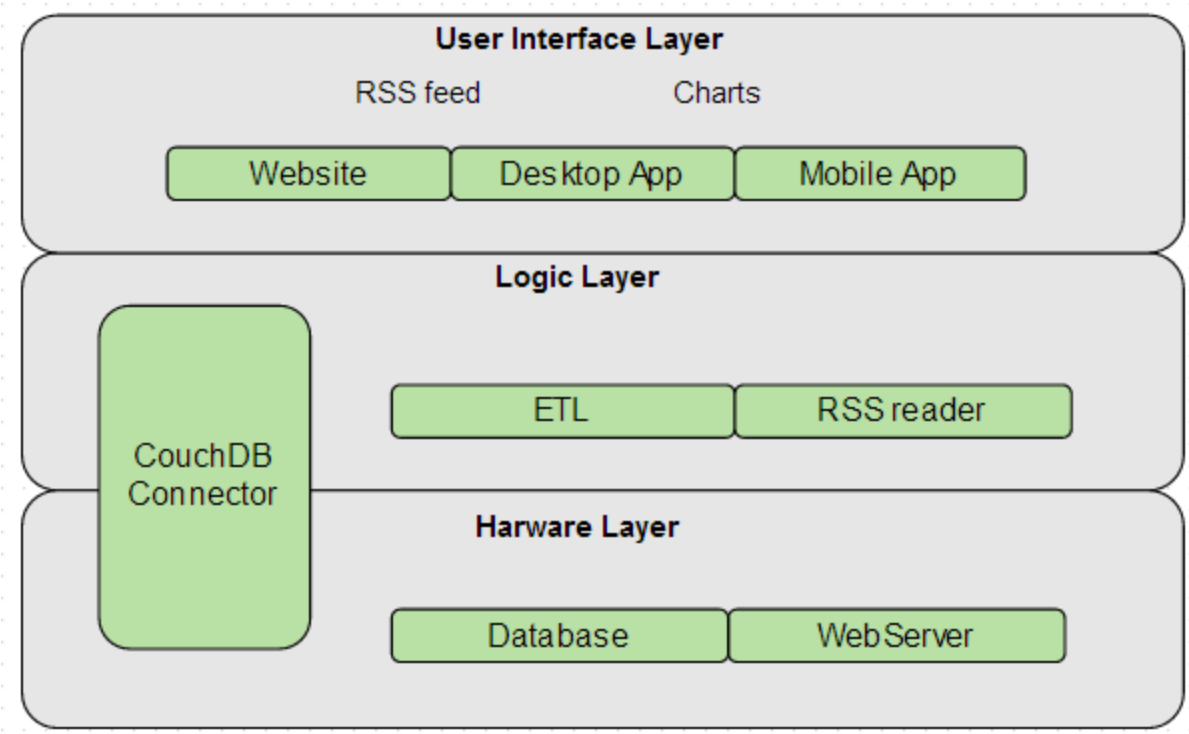
Diagram 2.3 - Process View.



The purpose of the activity diagram is to model the procedural flow of actions that are part of a back end. Since it models procedural flow of the system activity diagram represent states of the system and actions which trigger those states. It starts with starting the system and goes all the way down to loading and updating new information to database.

2.7 Development View

Diagram 2.4 - Development view



This diagram represents the layers of the system with belonging components. There are three layers of the system: Hardware layer, Logic layer and User Interface layer.

User interface layer consists of three different clients that provide charts, news feeds and all the necessary information. Clients include website, desktop application and mobile application. Logic layer contains ETL (Extract Transform Load) module which extracts and parses necessary information; RSS parser for news feed. Hardware layer consists of database storage and website server. Database Connector provides connection between data and logic layer.

3. Software Design Specification

Reference guideline for SDS/SDD(http://www.ceng.metu.edu.tr/_media/course/ceng490/sdd_template.pdf?rev)

3.1. System Overview

3.1.1 Backend

The system generates urls for the ticker from a list of stock symbols from three different markets, it makes a request to the ticker with the generated url and reads/parsers the JSON data by decoding the JSON object into a struct structure. Then taking out the specific information. Before sending the useful data into the database we structure them in a way that the update handler can put the data into existing files or creating new ones.

3.1.2 Frontend

Information about the specific stocks will be displayed on three programs (Candlestick, volume and price) after the user have selected the stock they want to view in the search bar. At the same time the search bar passes the searched variable to generate a url to retrieve the RSS news for that specific stock.

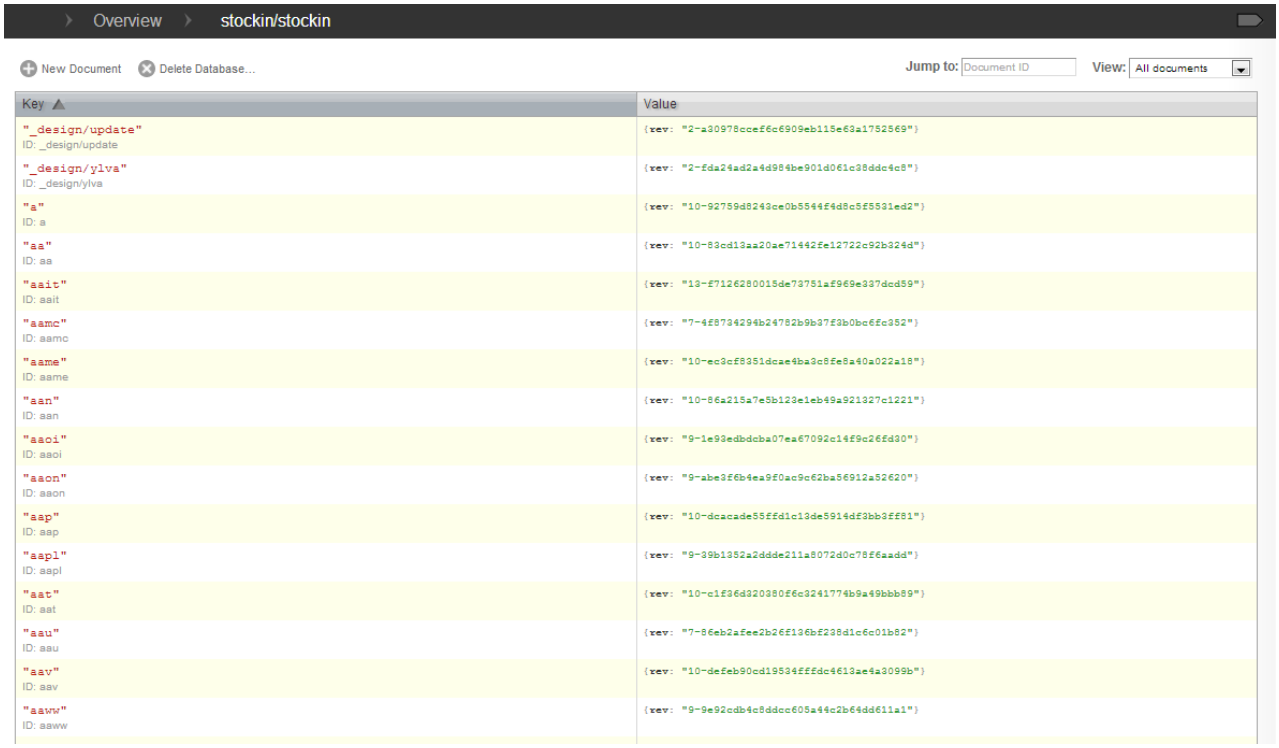
3.2 Data Design

3.2.1 Data description

The data that's retrieved from the sources (google,yahoo and markitondemand) is returned as JSON objects. We parse the data by taking out the specific type of information that we need (High, Low, Open, High etc) for the diagrams. Then we parse the data in the corrected order in order to store it in the database. In the database there is a design document written in JavaScript handling the info. If the data sent in belongs to a document (The id is the symbol) it will update the said document, otherwise it will create a new document using the symbol as ID.

3.2.2 Database structure

Main Database



Data Structure

Source


```
{
   "_id": "aapl",
   "_rev": "7-0a86594611d8822b9bf3b166372f53a3",
   "Stock": {
       "Data": [
           {
               "Time": "1386794540000",
               "Open": "566.89",
               "High": "570.97",
               "Low": "560.20",
               "Close": "561.08",
               "Volume": "10597400"
           },
           {
               "Time": "1386865723000",
               "Open": "562.14",
               "High": "565.34",
               "Low": "560.34",
               "Close": "563.35",
               "Volume": "37100000"
           },
```


Here is the structure of the database, we have one database holding all the stocks. Every document holds the data for one stock and the ID of the document is the symbol name of the

stock. This means every time we send in a stock with the same symbol, the data(Time, Open, High etc) will be stored inside a list of tuples after the latest stock data.

3.3 Component design

3.3.1 Backend:

Supervisor

- Start up server/monitoring the server
- Restart in case of crash

JSON parser

- Parsing JSON objects
- Extract JSON data
- Parse data into right format

CouchDB Updater

- Updating CouchDB documents

Other modules – various functions

- Volume parser - Parsers all different volume format to one format (e.g. 1,2M or 1,999.00 to 1200000 and 1999)
- Unixtimeconverter - Converting current date into unixtime
- Text converter - converting text file containing the necessary symbols into string of symbols

Libraries

- Mochiweb MochiJson2parser (https://github.com/mochi/mochiweb)

3.3.2 Frontend:

Webpage:

Server

- Based on the Sinatra framework
- Written in ruby and conforms to the Rack specification
- Uses the Couchrest gem as a client to the couchdb
- Contains three routes
    - /
    - /stock
    - /search

Design

- Default minimalistic frontpage
- Features a fast ajax autocomplete
- Stock view
- Three different charts focused data
- RSS feed from the yahoo finance api
- Custom HTML5 font
- Uses Highcharts for rendering pretty svg based charts
- Uses a main layout and renders templates inside

Libraries

- jQuery (http://jquery.com/download/)
- Highcharts (http://www.highcharts.com/)
- Google font api (https://developers.google.com/fonts/)
- Google rss api (https://developers.google.com/feed/)
- Sinatra(http://rubygems.org/gems/sinatra)
- Couchrest (https://github.com/jchris/couchrest)

3.3.3 Desktop:

Java:

- Establishing connection with database
- Parsing JSON objects
- Generating charts
- Parsing RSS feed
- Building RSS feed
- Searching for stock quote symbols in database and holding them as variables for RSS feed and charts

Libraries:

- JFreechart (http://www.jfree.org/jfreechart/)
- LightCouch (http://www.lightcouch.org/)
- Feed4j (http://www.sauronsoftware.it/projects/feed4j)
- Gson (http://code.google.com/p/google-gson/)

3.3.4 Android:

Java:

- Establishing connection with database
- Parsing JSON objects
- Building charts

- Parsing RSS feed
- Building RSS feed
- Searching for stock quote symbols in database and holding them as variables for RSS feed and charts

XML:

- creating layout of application

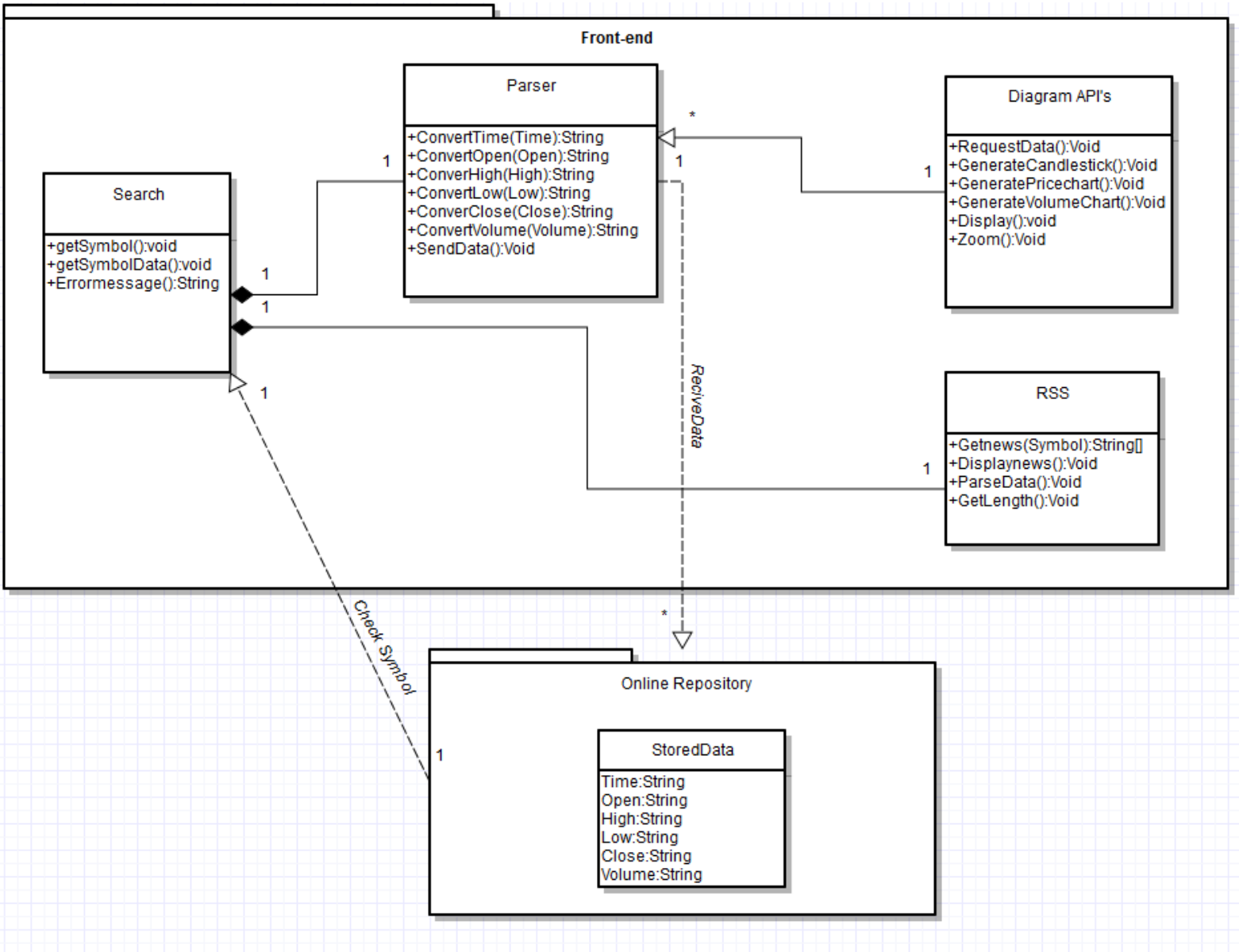
Libraries:

- android-support-v4.jar (built in)
- stock-chart.jar (http://stockchartview.org/)
- android-rss.jar (https://www.assembla.com/code/andoridtutorials/subversion/nodes/LunchList/libs/android-rss.jar?rev=76)

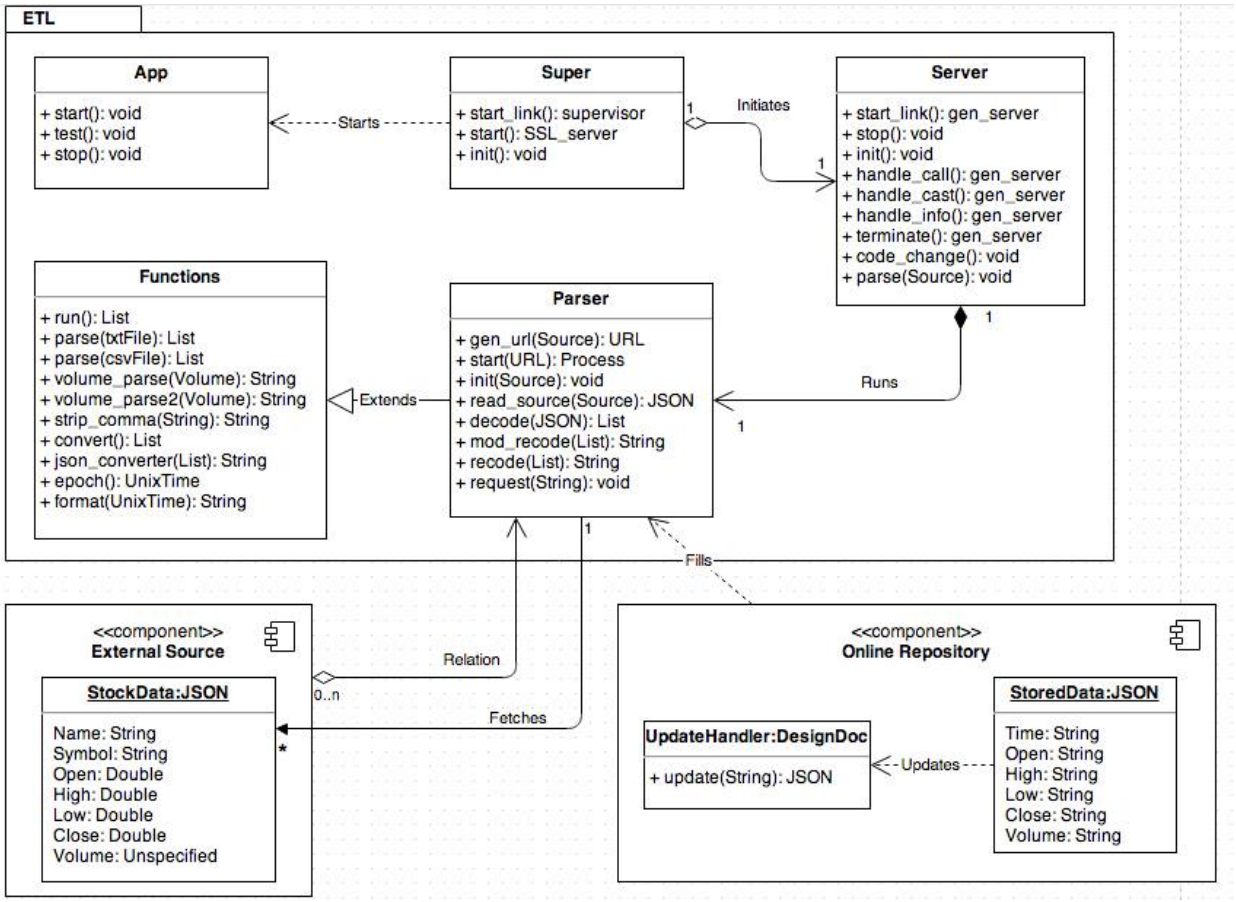
3.4 System design

3.4.1 Class Diagram with Components

Frontend

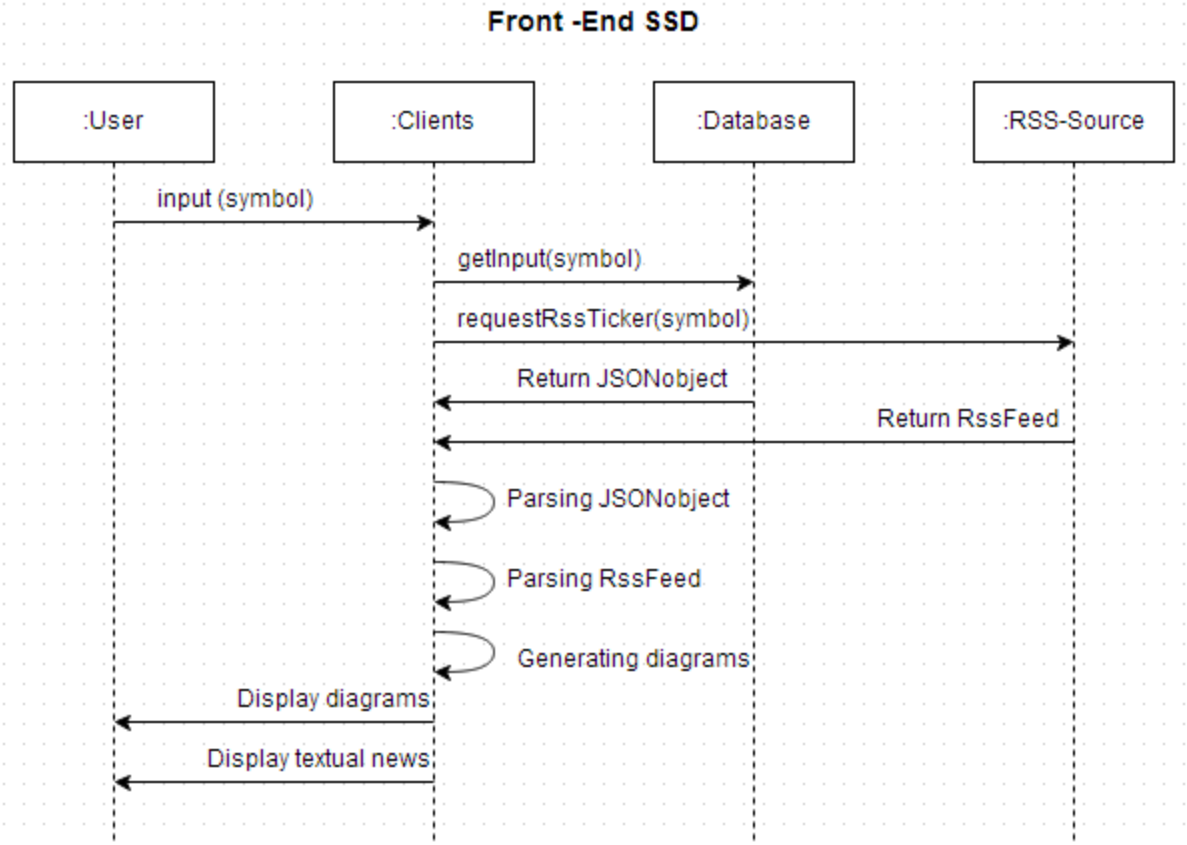


Backend

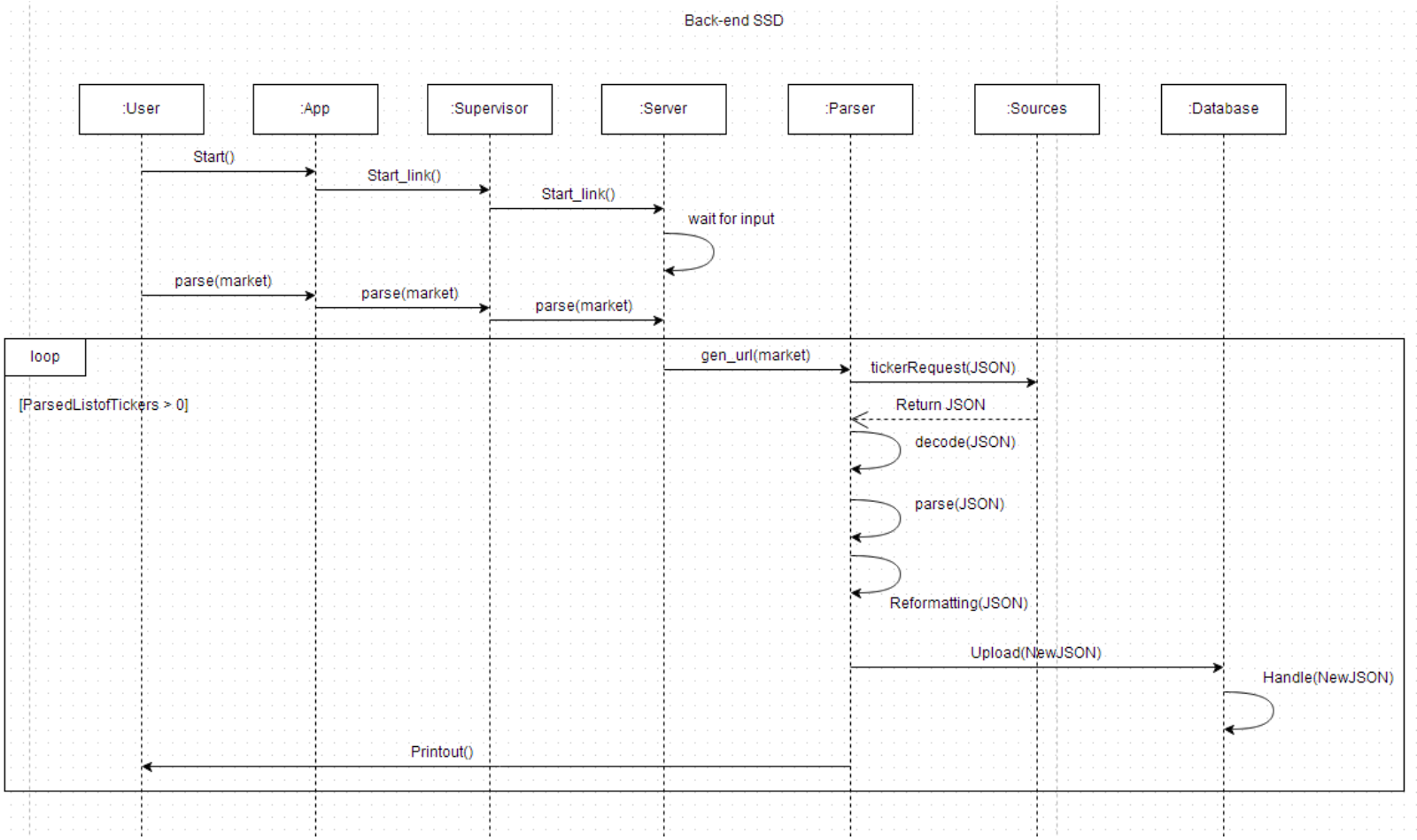


3.4.2 Sequence diagram

Frontend



Backend


Back-end SSD
:User
:App
:Supervisor
:Server
:Parser
:Sources
:Database
Start()
Start_link()
Start_link()
wait for input
parse(market)
parse(market)
parse(market)
loop
[ParsedListofTickers > 0]
gen_url(market)
tickerRequest(JSON)
Return JSON
decode(JSON)
parse(JSON)
Reformatting(JSON)
Upload(NewJSON)
Handle(NewJSON)
Printout()

4. System requirement Specification

Referece document(http://users.tricity.wsu.edu/~mckinnon/cpts322/cpts322-srs-v1.doc**)**

4.1 Functional requirements:

- The system is required to fetch stock data from 4 sources.
- Data is to be gathered on a frequent timely basis.
- Stock data is required to be processed through ETL for a common structure.
- Major ETL parts are to be coded in Erlang.
- The system needs to be available on 3 clients (Website, Desktop application, and Mobile application).
- Stock data is required to be displayed through a Candlestick diagram as well as two more graphical views.
- The user is able to view historical data for a particular stock.

4.2 Non-functional requirements:

- The user is able to search for stock symbols.
- A news feed is displayed for every stock upon search.

4.3 System Features

Backend

4.3.1 Extracting

- Acquiring stock symbols from files -
  Lists of stock symbols are gathered from parsing .txt-files as well as .csv-files, removing any unwanted information and keeping just symbols.
- Generating ticker URL -
  A new URL is needed for each stock symbol in order to request data, these are generated with the lists of symbols previously acquired.
- Requesting data from 3 stock data sources -
  The system uses three sources (Google finance, Yahoo finance, and Markitondemand) to request individual stock data from three stock markets (NASDAQ, NYSE, and AMEX).
- Retrieving JSON data -
  Stock data is gathered in JSON-format.
- Running on server -
  All back-end code runs on Erlang gen_servers with a proper supervision connection, to ensure sufficient error handling and ability to follow all processes.

4.3.2 Transforming

- Decoding -
  Data is decoded using Mochijson to allow changes to structure, data types, and formats.
- Parsing stock data -
  The acquired data is parsed to include exactly the information wanted.
- Formatting stock data -
  Data is modified to ensure same formatting of dates, volumes, prices etc from the different sources, also making sure names and data structures are the same for a well structured database.

4.3.3 Loading

- Uploading to database -
  All data is uploaded concurrently to a CouchDB database hosted online.
- Update manager -
  Updating, saving multiple information and creating new documents for each stock is done by an update handler written in JavaScript.

Frontend

4.4 Search

- Symbol search
  User has the opportunity to search for a stock symbol name. Website will return a message if the provided symbol is not found in the database.
- Autocompletion (Website)
  When searching for a symbol name our website will provide suggestions for a symbol name with link to the webpage.

4.5 Charts

- Retrieving data from database
  Once the symbol name is found in the database it will collect the information.
-Parsing data from database
  Once the information has been gathered the system will parse the data and select specific data and save it to different variables for further usage-
- Generating charts.
  Once the steps above has completed the system will use the provided data to display different charts.

4.6 Rss retrieve-

- Gets the news for the searched symbol name and displays them in the applications.

5. Test Plan & Test Report

5.1 Test Plan

5.1.1 Introduction

1. Purpose

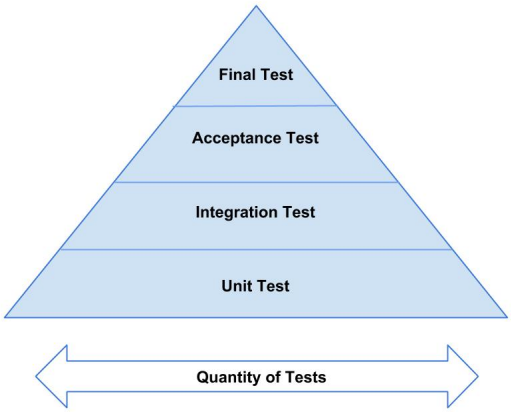
The purpose of this document is to describe and specify the testing method performed during this project in order to assure the quality and functionality of the applications.

2. Scope

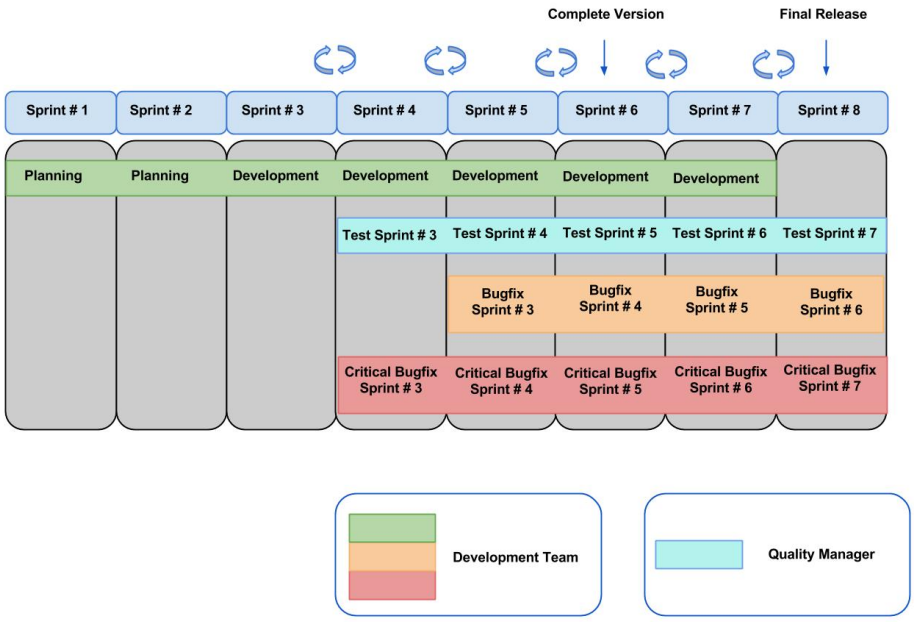
This document will present the approach and testing approach utilized for this specific project, and it is based on the Requirement Specification and Project Plan documents.

5.1.2 Process

The Scrum is the methodology in use for this project therefore the testing process will also contain an agile approach. The test cases will be based on the functional requirements. At the end of each sprint, a Unit Test will be performed, followed by Integration Test and Acceptance Test, in order to obtain a better visibility of the project progress.



The tests cases will be performed accordingly to the functional requirements developed through the sprints. The following graph describes how the test are performed during the sprints.



5.1.3 Design

1. Deliverables

- Test Cases
- Test Report
- Requirement Specification

2. Features to be tested

- The functionality of the applications as described in the User Cases
- The functionality of the specific requirements as described in the Requirements Specification.

5.1.4 Approach

1. Guidelines

- The test cases will be performed accordingly with the requirements for each sprint.
- The test cases will be written while coding and executed after implementation.
- The test cases will be performed at end of each sprint.
- The test cases must run one hundred percent before the following sprint.
- Each requirement must contain at least one or more functional tests.
- Acceptance testing will be performed on the system at the end of each interaction.
- To be considered complete, a User Story must have passed its acceptance test.

- The test cases, as well its outcomes and defects will be documented through an online management tool.

2. White-Box Testing(Unit Test)

The white-box testing will focus on the functionality of each internal systems component, since the the first part of the development will fccus on the back-end of the system, with a very poor user interface.

3. Black-Box Testing(Integration Test)

Once all the internal attributes of the system are tested, the white-box testing will be replaced by the black-box testing, in order to assure that all the requirements are properly functioning with the front-end.

4. Integrity Test

Manual Integrity Testing will be performed on the CouchDB database. It must be executed once the database is complete and it will focus on loading the right information to the database, updating and/or creating a new document if it doesn't exist in the system.

5. Tools

- eUnit Library – used for the Unit Tests.
- OnTime – used for defect tracking and documenting Test Cases and defects.

5.2 Test Report

5.2.1 Introduction

This test report describes the results of the tests performed during the development of the applications. It will make references to the Test Plan document.

5.2.2 Test Plan Reflection

The testing process was successfully completed as previous stated in the test plan guidelines. Most of the functional requirements were tested and completed using the eUnit tests. All test cases and defects were documented and tracked using the online management system.

5.2.3 Test Cases

TC1 - Requesting/Parsing data from 3 stock data sources

General	
ID:	TC1
Name:	Requesting data from 3 stock data sources
Description:	Extracting the specific data to be uploaded to the database

Test Case			
Objective:	To test if the system is fetching the specific data from the sources		
Work Product:	US46: Grabbing data		
Method:	Automated		
Risk:	High		
Type:	Functional		
Priority:	Important		
Pre-Conditions:	Markets are open in order to get live data. Fetch NASDAQ, NYSE, AMEX data.		
Steps:		Input	Expected results
	1.	parse(google)	JSON data printout
	2.	parse(yahoo)	JSON data printout
	3.	parse(markitondemand	JSON data printout
	4.		
	Validation		
Post-Conditions:	Retrieving JSON data		

Notes	
Notes:	Sometimes the yahoo source gives error due to limitation of requests.

Result	
Run:	Verdict:
22-11-13	Pass
06-12-13	Pass

TC2 - Database loading data

General			
ID:	TC2		
Name:	Database loading data		
Description:	All data must be uploaded concurrenlty to the database.		
Test Case			
Objective:	To test if the CouchDB database is loading the data in timely basis		
Work Product:	US84: Backend functions and loading to database		
Method:	Automated		
Risk:	Medium		
Type:	Functional		
Priority:	Important		
Pre-Conditions:	The database must upload the content, and if a document doesn't exist, a new document must be created		
Steps:		Input	Expected results
	1.	parse(market)	Update/New Document
	2.		
	3.		
	4.		
	Validation		
Post-Conditions:	Database is being loaded as excpected		
Notes			
Notes:	Test case updated according to functions changes.		

Result	
Run:	Verdict:
22-11-13	Failed See DE3
06-12-13	Pass

TC3 - Desktop Search function

General	
ID:	TC3
Name:	Desktop Search function
Description:	The system must search for a specific stock symbol inside database

Test Case			
Objective:	To test if the system is searching for stocks symbol name		
Work Product:	US75: Search function		
Method:	Automated		
Risk:	Medium		
Type:	Functional		
Priority:	Important		
Pre-Conditions:	Connection to local DB/online DB.		
Steps:		Input	Expected results
	1.	search(symbol)	JSON Data
	2.		
	3.		
	4.		
	Validation		
Post-Conditions:	The system searching and returning corret JSON Data from the database.		

Notes	
Notes:	To test if the system is searching for stocks symbol name.

Result	
Run:	Verdict:
22-11-13	Pass
06-12-13	Pass

TC4 - Website Search function

General	
ID:	TC4
Name:	Website Search function
Description:	The system must search for a specific stock symbol inside database

Test Case				
Objective:	To test if the system is searching for stocks symbol name			
Work Product:	US67: Search Function			
Method:	Automated			
Risk:	Medium			
Type:	Functional			
Priority:	Important			
Pre-Conditions:	Connection to local DB/online DB.			
Steps:		Input	Expected results	
	1.	search(symbol)	JSON Data	
	2.			
	3.			
	4.			
	Validation			
Post-Conditions:	The system searching and returning corret JSON Data from the database.			

Notes	
Notes:	Test case updated according to sources changes

Result	
Run:	Verdict:
08-11-13	Failed See DE1
22-11-13	Pass
06-12-13	Pass

TC5 - Android Search function

General	
ID:	TC5
Name:	Android Search function
Description:	The system must search for a specific stock symbol inside database

Test Case				
Objective:	To test if the system is searching for stocks symbol name			
Work Product:	US71: Search Function			
Method:	Automated			
Risk:	Medium			
Type:	Functional			
Priority:	Important			
Pre-Conditions:	Connection to local DB/online DB.			
Steps:		Input	Expected results	
	1.	search(symbol)	JSON Data	
	2.			
	3.			
	4.			
	Validation			
Post-Conditions:	The system searching and returning corret JSON Data from the database.			

Notes	
Notes:	Test case updated according to sources changes

Result	
Run:	Verdict:
22-11-13	Failed See DE5
06-12-13	Pass

TC6 - Desktop Charts

General	
ID:	TC6
Name:	Desktop Charts
Description:	The system collects info from a specific stock and display it in charts

Test Case			
Objective:	To test if the system is collecting and displaying the specific information		
Work Product:	US41: Adding API diagrams		
Method:	Automated		
Risk:	High		
Type:	Functional		
Priority:	Important		
Pre-Conditions:	Able to search/parse Data.		
Steps:		Input	Expected results
	1.	search(symbol)	Display results
	2.		
	3.		
	4.		
	Validation		
Post-Conditions:	Charts showing the data for serach symbol.		

Notes	
Notes:	

Result	
Run:	Verdict:
22-11-13	Failed See DE6
06-12-13	Pass

TC7 - Website Charts

General	
ID:	TC7
Name:	Website Charts
Description:	The system collects info from a specific stock and display it in charts

Test Case			
Objective:	To test if the system is collecting and displaying the specific information		
Work Product:	US66: Organize API diagrams		
Method:	Automated		
Risk:	High		
Type:	Functional		
Priority:	Important		
Pre-Conditions:	Able to search/parse Data.		
Steps:		Input	Expected results
	1.	search(symbol)	Display results
	2.		
	3.		
	4.		
	Validation		
Post-Conditions:	Charts showing the data for serach symbol.		

Notes	
Notes:	

Result	
Run:	Verdict:
22-11-13	Failed See DE7
06-12-13	Pass

TC8 - Android Charts

General	
ID:	TC8
Name:	Android Charts
Description:	The system collects info from a specific stock and display it in charts

Test Case			
Objective:	The system collects info from a specific stock and display it in charts		
Work Product:	US70: Diagram API		
Method:	Automated		
Risk:	High		
Type:	Functional		
Priority:	Important		
Pre-Conditions:	Able to search/parse Data.		
Steps:		Input	Expected results
	1.	search(symbol)	Display results
	2.		
	3.		
	4.		
	Validation		
Post-Conditions:	Charts showing the data for serach symbol.		

Notes	
Notes:	

Result	
Run:	Verdict:
22-11-13	Failed See DE8
06-12-13	Pass

TC9 - Desktop RSS retriver

General	
ID:	TC9
Name:	Desktop RSS retriver
Description:	It gets the news for specific stock and displays it in the application

Test Case			
Objective:	To test if the RSS are grabbing and displaying the right news		
Work Product:	US94: RSS receiver		
Method:	Automated		
Risk:	High		
Type:	Functional		
Priority:	Important		
Pre-Conditions:	Able to search.		
Steps:		Input	Expected results
	1.	search(symbol)	RSS for the symbol
	2.		
	3.		
	4.		
	Validation		
Post-Conditions:	Displayed RSS for the searched symbol.		

Notes	
Notes:	

Result	
Run:	Verdict:
08-11-13	Failed See DE2
22-11-13	Failed See DE4
06-12-13	Pass

TC10 - Website RSS retriver

General	
ID:	TC10
Name:	Website RSS retriver
Description:	It gets the news for specific stock and displays it in the application

Test Case			
Objective:	To test if the RSS are grabbing and displaying the right news		
Work Product:	US45: RSS reader for textual news		
Method:	Automated		
Risk:	High		
Type:	Functional		
Priority:	Important		
Pre-Conditions:	Able to search.		
Steps:		Input	Expected results
	1.	search(symbol)	RSS for the symbol
	2.		
	3.		
	4.		
	Validation		
Post-Conditions:	Displayed RSS for the searched symbol.		

Notes	
Notes:	

Result	
Run:	Verdict:
24-10-13	Pass
08-11-13	Pass
22-11-13	Pass
06-12-13	Pass

TC11 - Android RSS retriver

General	
ID:	TC11
Name:	Android RSS retriver
Description:	It gets the news for specific stock and displays it in the application

Test Case			
Objective:	To test if the RSS are grabbing and displaying the right news		
Work Product:	US105: Putting everything together		
Method:	Automated		
Risk:	High		
Type:	Functional		
Priority:	Important		
Pre-Conditions:	Able to search.		
Steps:		Input	Expected results
	1.	search(symbol)	RSS for the symbol
	2.		
	3.		
	4.		
	Validation		
Post-Conditions:	Displayed RSS for the searched symbol.		

Notes	
Notes:	

Result	
Run:	Verdict:
06-12-13	Pass

5.2.4 Defect Tracking

#	Name	Requirement	Severity	Priority	State	Submitted by	Owner
DE1	Search not connected to the database	US67: Search function	3 - Minor Problem	2 - High Attention	Fixed	Lídia N	Ylva F
DE2	Not displaying RSS info from the specific symbol	US44: RSS reader	3 - Minor Problem	3 - Medium Attention	Fixed	Lídia N	Odzaya B
DE3	Not updating docs in database	US84: Backend functions and loading DB	2 - Major Problem	1 - Resolve Immediately	Fixed	Lídia N	Ale L/ Carl N
DE4	Not displaying RSS info from the specific symbol	US45: RSS reader	3 - Minor Problem	3 - Medium Attention	Fixed	Lídia N	Lídia N
DE5	Search not connected to to the database	US71: Search function	3 - Minor Problem	2 - High Attention	Fixed	Lídia N	George M
DE6	Return the JSON data is not parsed	US90: Parsing data from DB	3 - Minor Problem	2 - High Attention	Fixed	Lídia N	Daniel H
DE7	Return the JSON data is not parsed	US98: Parse data from DB to API	3 - Minor Problem	2 - High Attention	Fixed	Lídia N	Ylva F
DE8	Parse error	US70: Diagram API	3 - Minor Problem	2 - High Attention	Fixed	Lídia N	George M

5.2.5 Detailed Defect Report

Defects

State:	Fixed	Release:	ETL
Priority:	Resolve Immediately	Severity:	Major Problem
Submitted By:	Lídia N	Creation Date:	22-11-13
Resolution:	Remake a new database handler, using symbol name as documents in DB.		
User Story:	US84: Backend functions and loading DB		

Schedule

Schedule State:	Closed

1 Search not connected to the database

General

ID:	DE1
Name:	Search not connected to the database
Description:	The search is currently working with a textfile.
Owner:	Ylva F

Defects

State:	Fixed	Release:	Website
Priority:	High Attention	Severity:	Minor Problem
Submitted By:	Lídia N	Creation Date:	08-11-13
Resolution:	Search should be connected to the local/online database and not to textfile.		
User Story:	US67: Search function		

Schedule

Schedule State:	Closed

2 Not displaying RSS info from the specific symbol

General

ID:	DE2
Name:	Not displaying RSS info from the specific symbol
Description:	RSS is currently only receiving the RSS from a static link.
Owner:	Odzaya B

Defects

State:	Fixed	Environment:	Desktop
Priority:	Medium Attention	Severity:	Minor Problem
Submitted By:	Lídia N	Creation Date:	08-11-13

Resolution:	Implement a function to grab the searched symbol and retrieve its info from RSS.
User Story:	US44: RSS reader

Schedule

Schedule State:	Closed

3 Not updating docs in database

General

ID:	DE3
Name:	Not updating docs in database
Description:	Could not update an exiting document on database, only create new ones.
Owner:	Ale L/ Carl N

Defects

State:	Fixed	Environment:	Desktop
Priority:	High Attention	Severity:	Minor Problem
Submitted By:	Lídia N	Creation Date:	22-11-13
Resolution:	Fix the parsing for the returned JASON data from database.		
User Story:	US90: Parsing data from DB		

Schedule

Schedule State:	Closed

4 Not displaying RSS info from the specific symbol

General

ID:	DE4
Name:	Not displaying RSS info from the specific symbol
Description:	RSS is currently only receiving the RSS from a static link.
Owner:	Lídia N

Defects

State:	Fixed	Environment:	Desktop
Priority:	Medium Attention	Severity:	Minor Problem
Submitted By:	Lídia N	Creation Date:	22-11-13
Resolution:	Implement a function to grab the searched symbol and retrieve its info from RSS.		
User Story:	US44: RSS reader		

Schedule

Schedule State:	Closed

5 Search not conncted to the database

General

ID:	DE5
Name:	Search not connected to the database
Description:	Search is working for strings stored inside of the application.
Owner:	Lídia N

Defects

State: Fixed Environment: Desktop

Priority: High Attention Severity: Minor Problem

Submitted By: Lídia N Creation Date: 22-11-13

Resolution: Connect it to the local or online database.

User Story: US71: Search function

Schedule

Schedule State: Closed

6 Return the JSON data is not parsed

General

ID: DE6

Name: Return the JSON data is not parsed

Description: Charts could not display the correct data because the return data was not parsed.

Owner: Daniel H

the

State:	Fixed	Environment:	Desktop
Priority:	High Attention	Severity:	Minor Problem
Submitted By:	Lídia N	Creation Date:	22-11-13
Resolution:	Conect to the local/online database and create parser for the data.		
User Story:	US70: Diagram API		

Schedule

Schedule State:	Closed

7 Return the JSON data is not parsed

General

ID:	DE7
Name:	Return the JSON data is not parsed
Description:	Charts could not display the correct data because the return data was not parsed.
Owner:	Ylva F

Defects

State:	Fixed	Environment:	Website
Priority:	High Attention	Severity:	Minor Problem
Submitted By:	Lídia N	Creation Date:	22-11-13
Resolution:	Fix the parsing for the returned JASON data from database.		
User Story:	US98: Parse data from DB to API		

Schedule

Schedule State:	Closed

8 Parse error

General

ID:	DE8
Name:	Parse error
Description:	Search is not searching inside the database. The return data is the return data us not being parsed.
Owner:	George M

6. UI Design Report

6.1 Deciding the GUI

In this project we were assigned to make three different clients, a desktop program, a website and a mobile application. Considering they were all going to display the same thing we realized that it would be suitable if all the clients followed the same design as much as possible. By using the same design we hope that the user will get the impression that our clients are easy to manage regardless of which one he uses. Another purpose would be to build a trademark for the future.

The website was designed simple and clean for easy understanding that will facilitate the usage of the client. By putting the search bar centered and presenting images of the markets that we offer, the user simply searches directly for the desired stock. When the search is made the stock is presented in three different graphs (Candlestick, volume and price)
and an rss-feed that are all located on the same page for easy access.

The same design filosofi is also the used for both the desktop and the android application, where we aim to limit everything on two pages. First page containing the search and the markets we offer. The second page containing the diagrams showing the different graphs and the textual news(RSS feed) about the selected stock.

6.2 GUI Technical specification

The GUI for the website is written in:

- Javascript
- Ruby
- HTML/CSS
- jQuery
- Sinatra

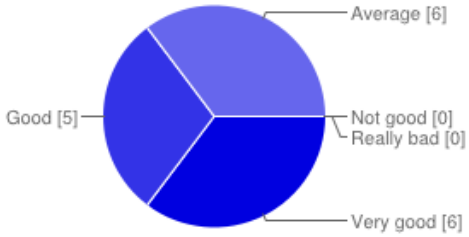
The languages that we use works together to display the website and lets the website be dynamic in terms of information handling.

6.3 Survey

We conducted survey between our classmates, friends and family members to receive their feedback about experience of using different clients. By doing so, we can visually represent what an average user can expect from our system.

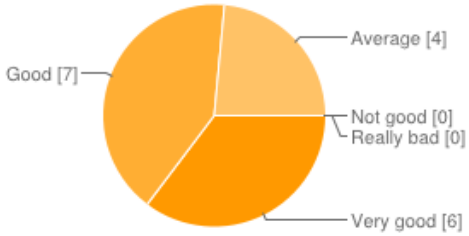
6.3.1 Website

First impression about the website



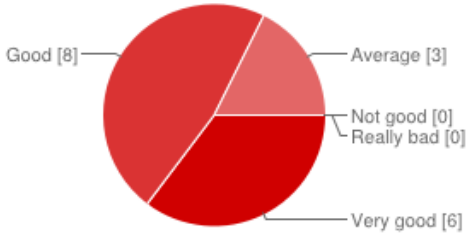
Very good	6	35%
Good	5	29%
Average	6	35%
Not good	0	0%
Really bad	0	0%

How do you like the overall design



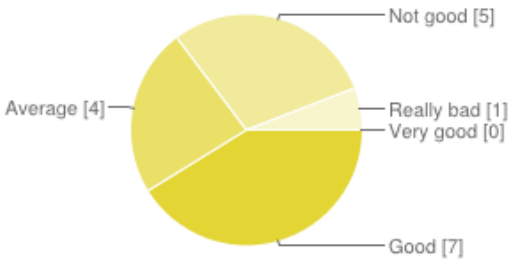
Very good	6	35%
Good	7	41%
Average	4	24%
Not good	0	0%
Really bad	0	0%

How do you like the charts



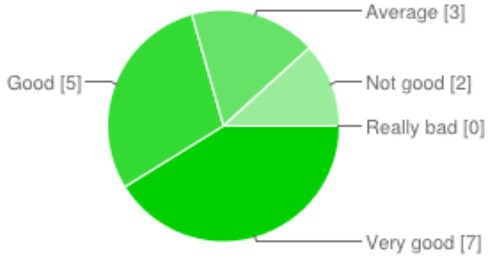
Very good	6	35%
Good	8	47%
Average	3	18%
Not good	0	0%
Really bad	0	0%

How do you like the RSS feed



Very good	**0**	0%
Good	**7**	41%
Average	**4**	24%
Not good	**5**	29%
Really bad	**1**	6%

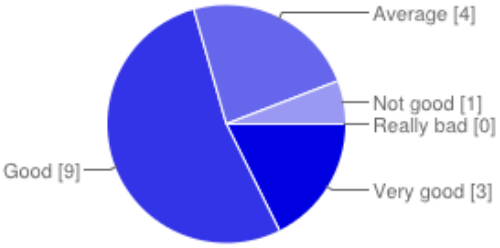
How do you like the performance



Very good	**7**	41%
Good	**5**	29%
Average	**3**	18%
Not good	**2**	12%
Really bad	**0**	0%

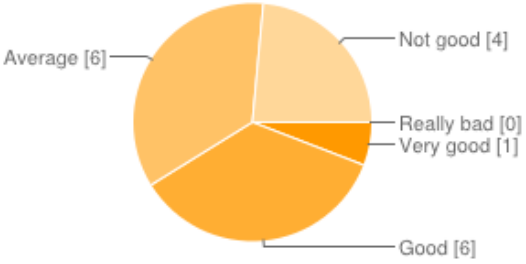
6.3.2 Desktop

First impression about the desktop application



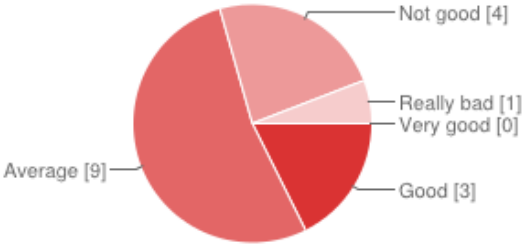
Very good	**3**	18%
Good	**9**	53%
Average	**4**	24%
Not good	**1**	6%
Really bad	**0**	0%

How do you like the overall design



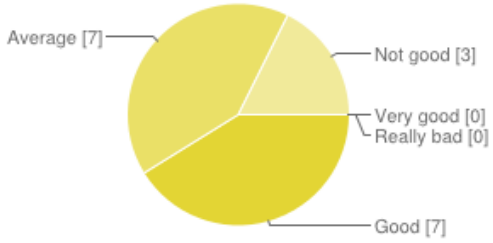
Very good	**1**	6%
Good	**6**	35%
Average	**6**	35%
Not good	**4**	24%
Really bad	**0**	0%

How do you like the charts



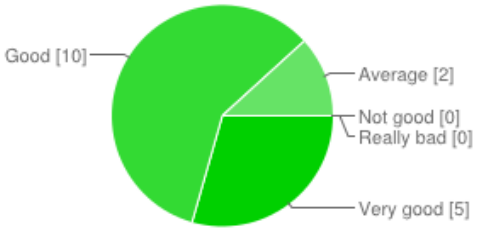
Very good	**0**	0%
Good	**3**	18%
Average	**9**	53%
Not good	**4**	24%
Really bad	**1**	6%

How do you like the RSS feed



Very good	**0**	0%
Good	**7**	41%
Average	**7**	41%
Not good	**3**	18%
Really bad	**0**	0%

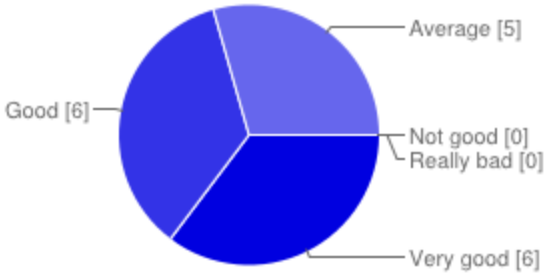
How do you like the performance



Very good	**5**	29%
Good	**10**	59%
Average	**2**	12%
Not good	**0**	0%
Really bad	**0**	0%

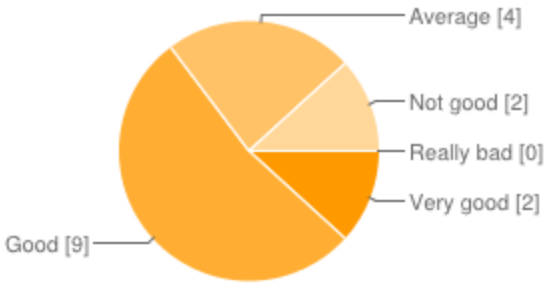
3.6.3 Android

First impression about the android



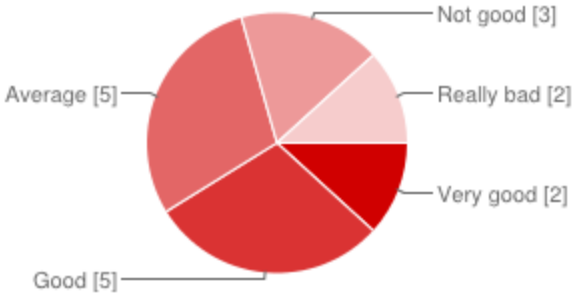
Very good	6	35 %
Good	6	35 %
Average	5	29 %
Not good	0	0 %
Really bad	0	0 %

How do you like the overall design



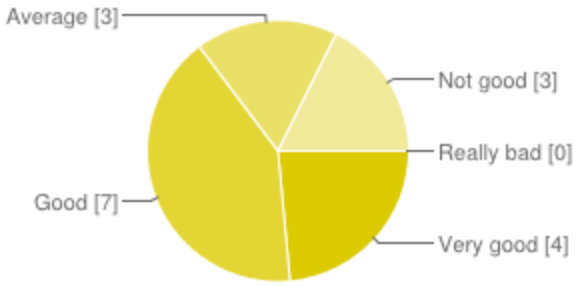
Very good	2	12 %
Good	9	53 %
Average	4	24 %
Not good	2	12 %
Really bad	0	0 %

How do you like the charts



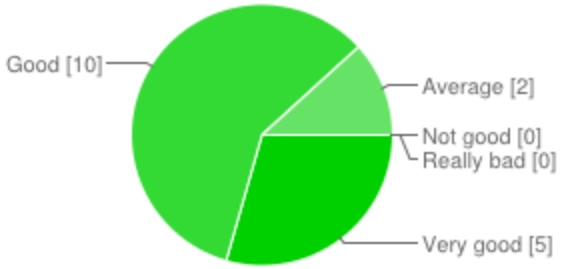
Very good	2	12 %
Good	5	29 %
Average	5	29 %
Not good	3	18 %
Really bad	2	12 %

How do you like the RSS feed



Very good	**4**	24 %
Good	**7**	41 %
Average	**3**	18 %
Not good	**3**	18 %
Really bad	**0**	0 %

How do you like the performance

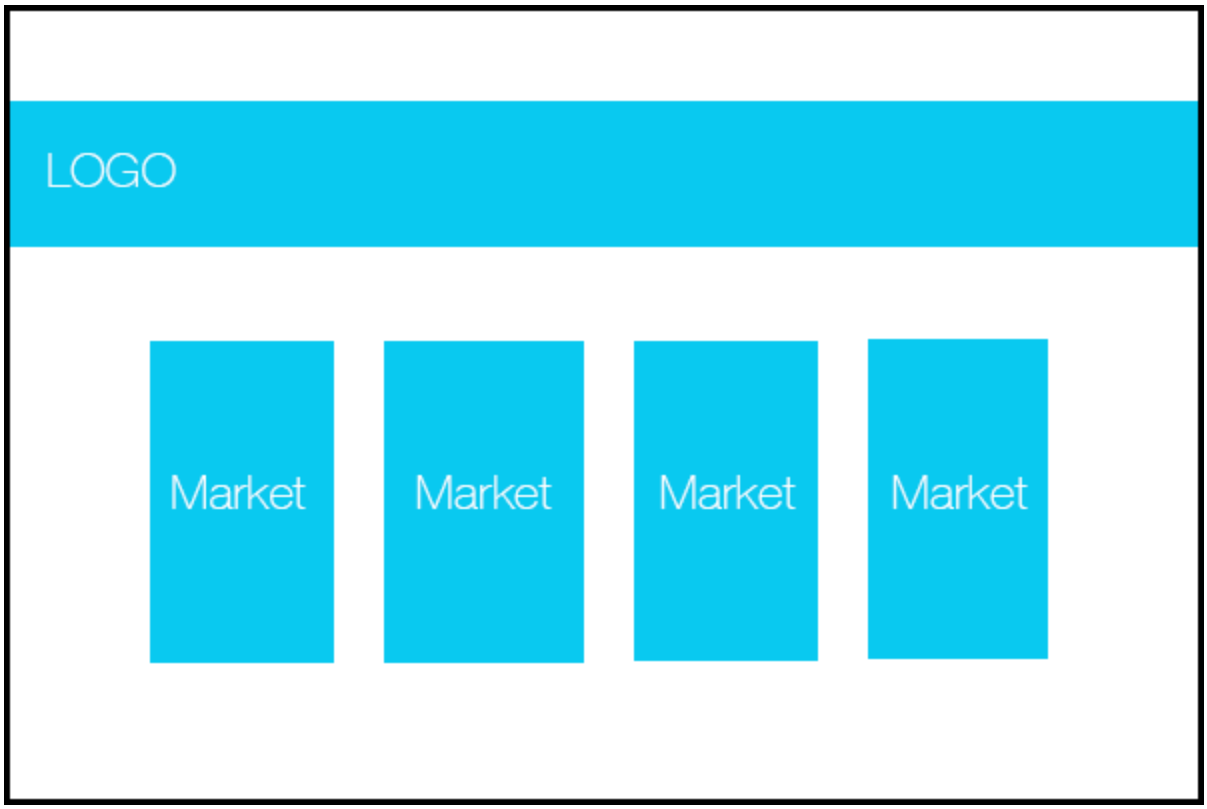


Very good	**5**	29 %
Good	**10**	59 %
Average	**2**	12 %
Not good	**0**	0 %
Really bad	**0**	0 %

6.4 Mockup & Changes

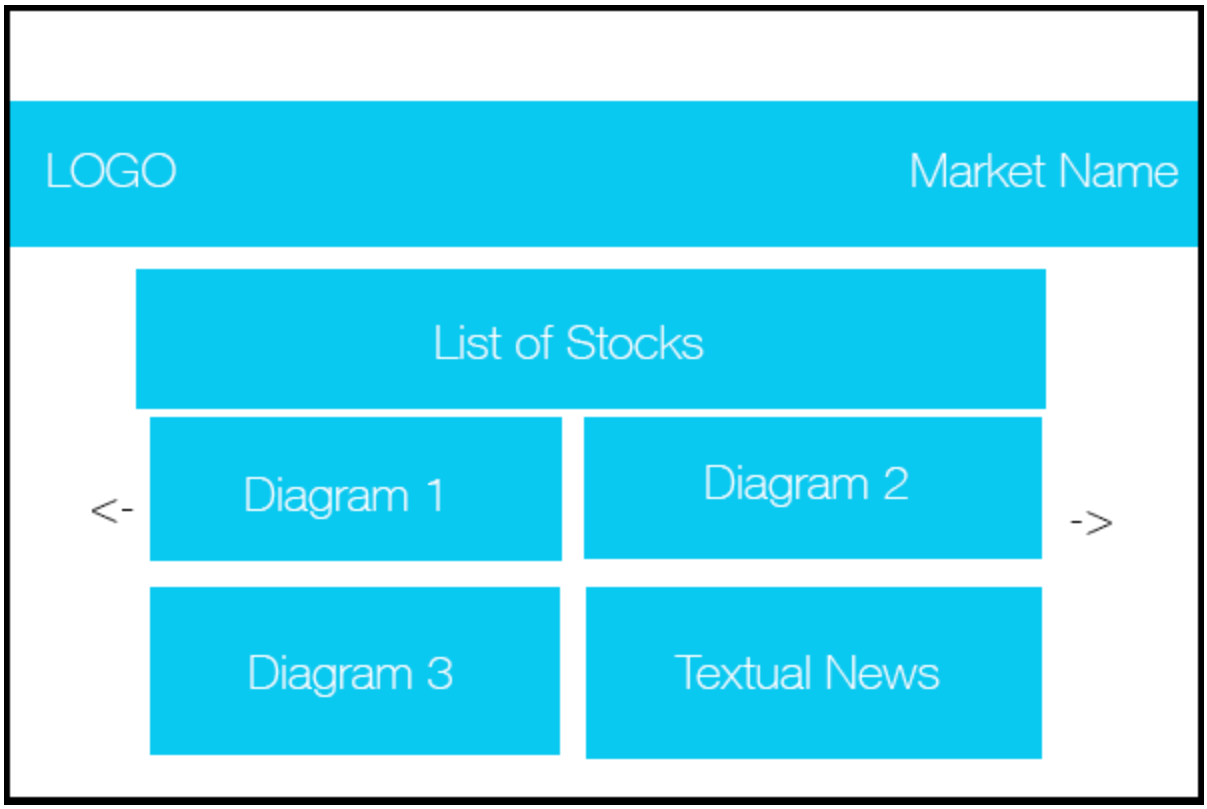
6.4.1 Website

Front Page v1



This was the first design we ever did, in the beginning there were things that we didn't really know due to just starting the project. We knew this but we decided to at least put out an overall design. Here we decided it was best using two pages to make it convenient. As you can see on the first page, there is the big stripe with a logo of the project and 4 different stock markets. We still wanted a minimalistic and interactable picture representation of each stock market. Ones clicking on one of the stock market pictures it will lead you to the next page.

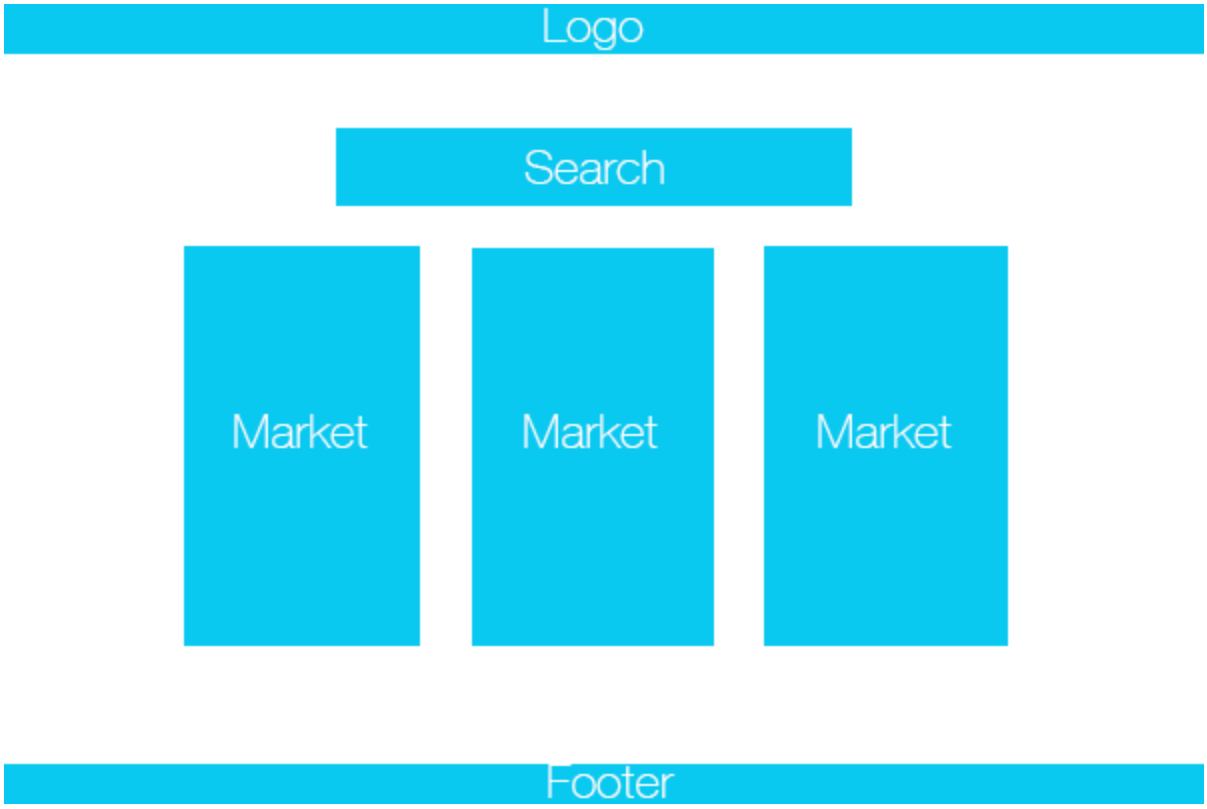
Market Page v1



Ones the user have clicked on one of the markets it will take you here to the market page, to the right of the top banner you can see the market name and underneath you will be shown with a list containing all the stocks that is clickable for the user. When user selects one of the stocks in the stock list, the diagrams and news will be shown for that stock inside the stock market. The user can also switch between markets by clicking on one of the arrows at the end of the page.

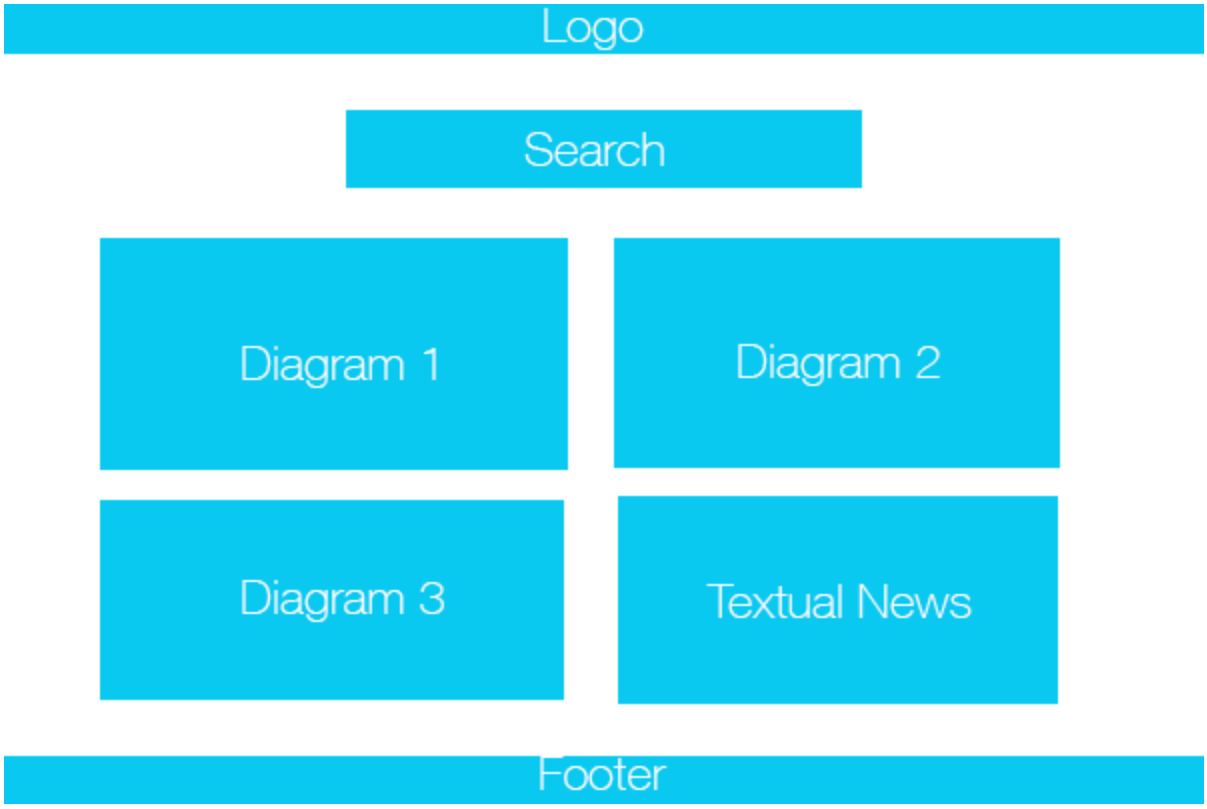
While this sounded good at the start of the project, later on we realised this was not a possibility when you have over 6000 stocks in the database. It would be too time consuming to scroll through the list and find one single stock. This would not help the user at all and thats why on the mockup v2 which also is our final version, we changed the list to a search bar instead. With this change, there was also no more need for individual pages for each market. Because now the user can directly search through all the markets and find the stock they are looking for.

Frontpage v2



This is the v2 of the website which also is the final design for what we wanted the website to look like. We kept the markets but only as a show to display what markets we have. With the addition of a search bar at top, the user can browse through all the different stock markets at once and therefore eliminate the need of click on individual stock markets.

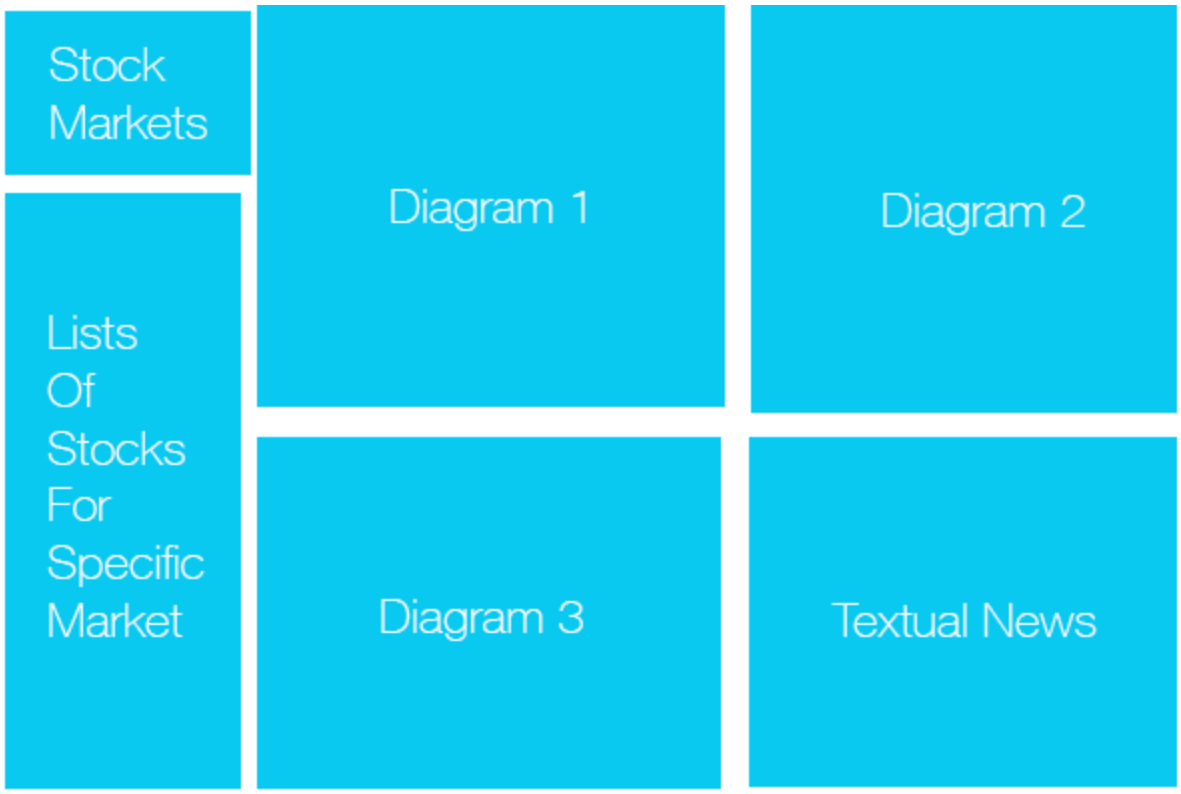
Resultpage v2



After finding the stock that the user looked for, it brings the user to the resultpage which has a similar look to the frontpage, the search bar at top is still here. This eliminates any extra steps that the user must do to search for another stock. The user can simply type a new stock and the page will be refreshed and all the diagrams with the textual news for that specific stock.

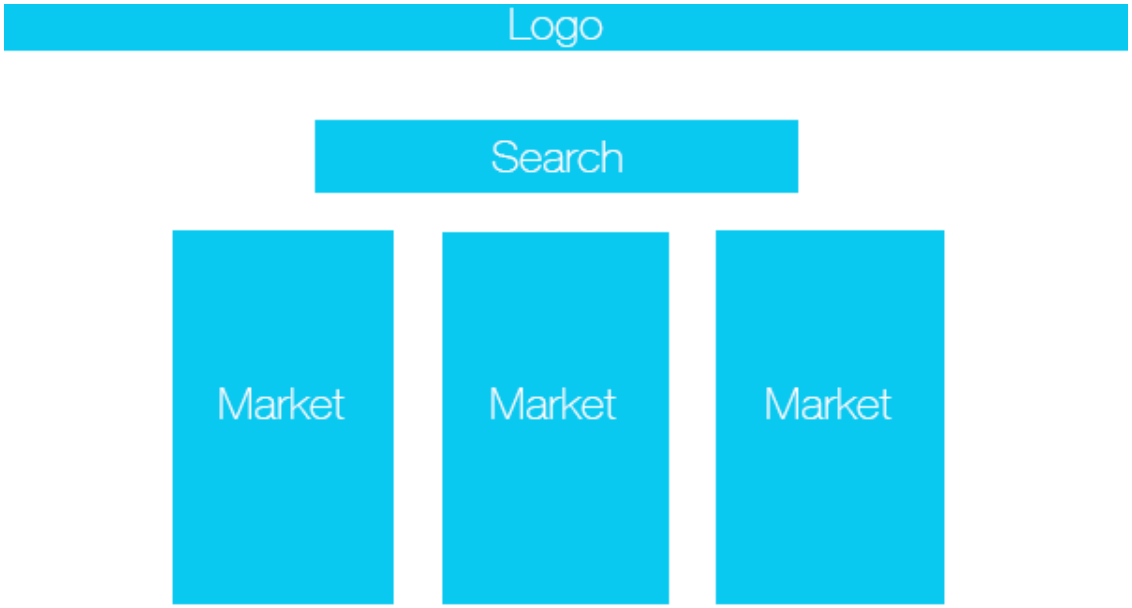
6.4.2 Desktop

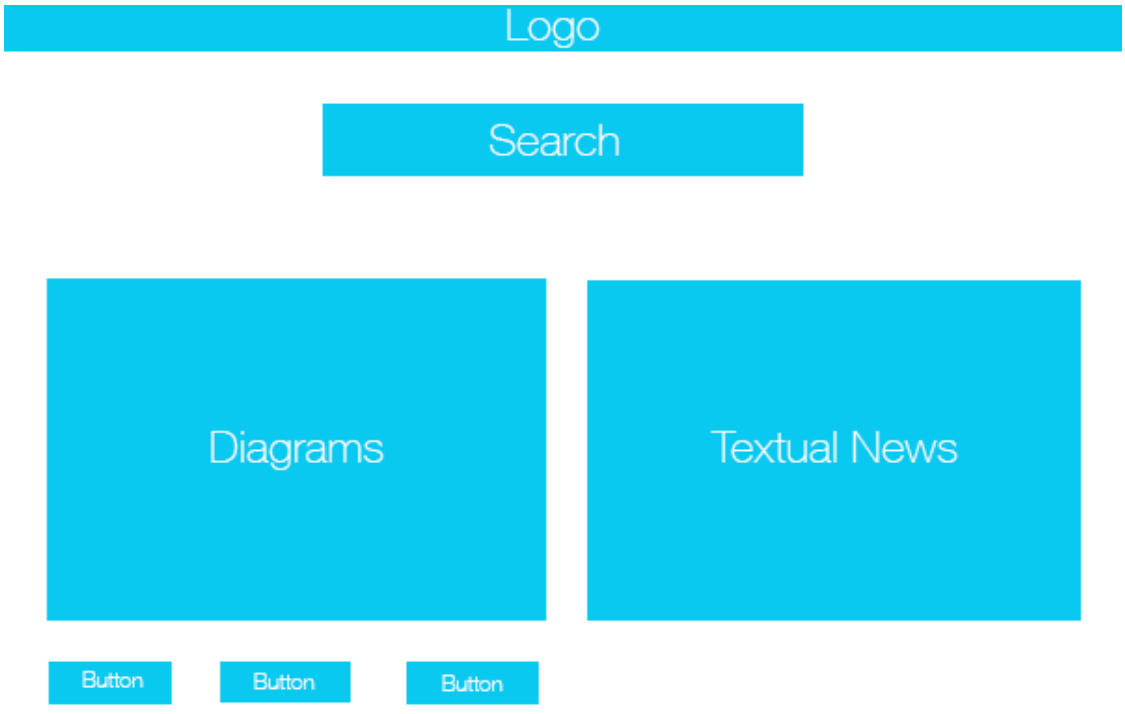
Desktop V1



Same as website, we did the design very early on in the project before we had a 100% grip of the whole project. A lot of the things were what we wanted to have. You can see we were going for a clickable stock markets and the lists of stocks for that specific market. But this was later changed due to the amount of stock stored in the database(same as the first website mockup). The diagrams are shown next to each other similiar to the website as well. But here we wanted to have all the functionalities on one page.

Desktop V2



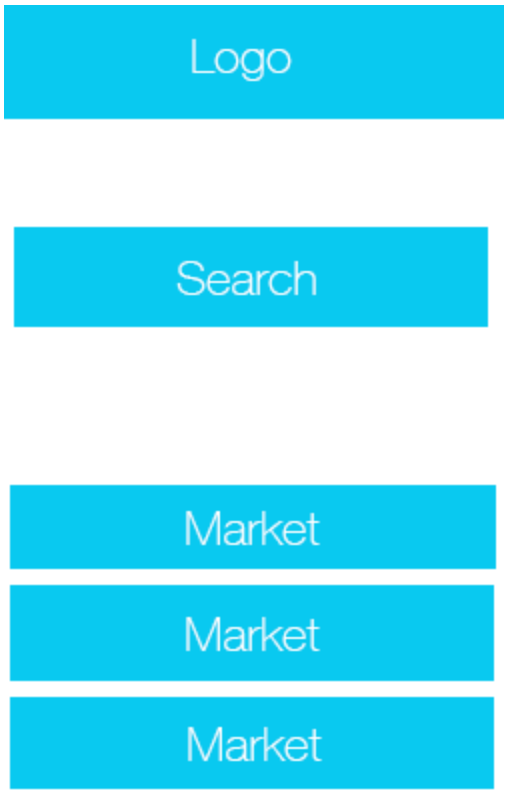


As explained above, we later had to change the whole layout of the desktop. This time we wanted to give the user an as close experience with the new website as possible. The search

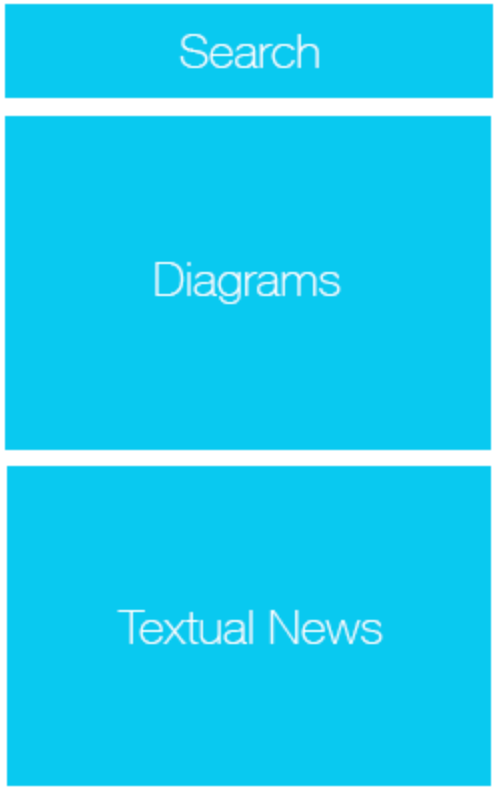
bar was added similar to the website, once user reaches the result page he/she can then search for new diagrams just like the website. But during the project the java group encountered a problem with putting all the diagrams into one frame. We tried to get it working but couldn't, in the end it was not doable for the group and we had to re- design and move along in order to finish everything in time. This made it so that the the diagrams were instead stored into different frames and by clicking on one of the buttons at the bottom of the screen the user could switch between diagrams.

6.4.3 Android

Front



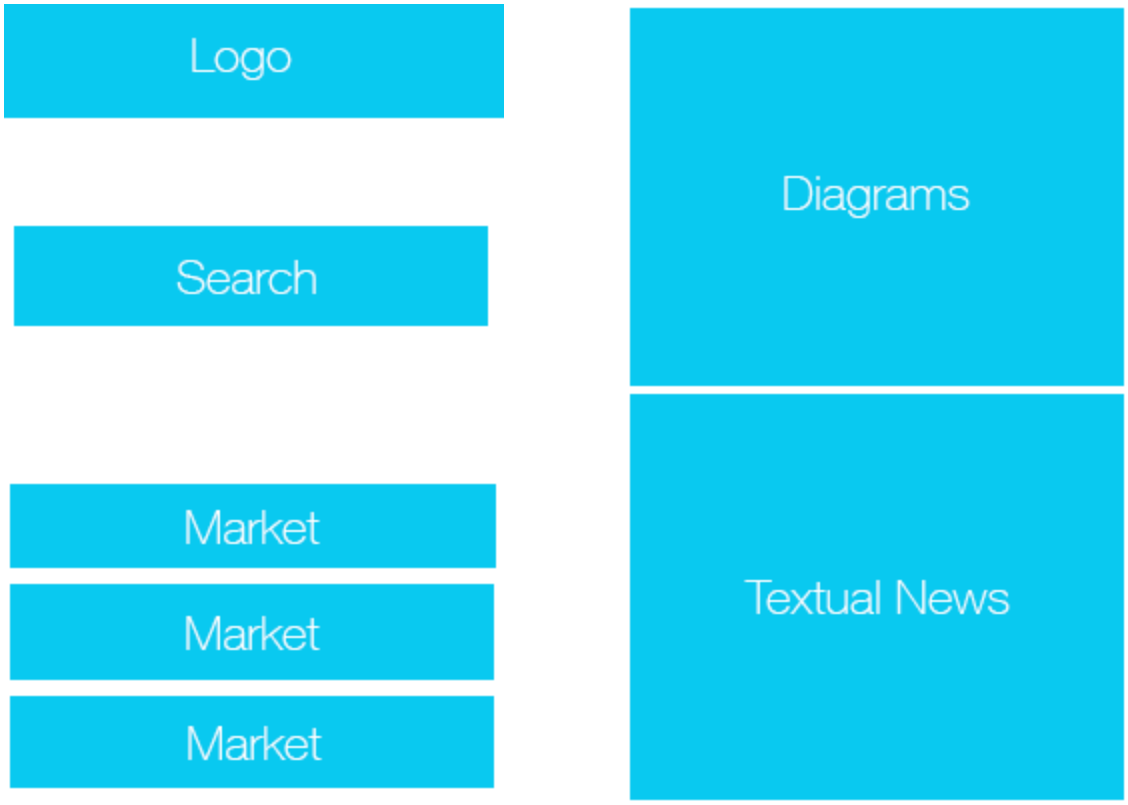
Result



One of the first things we thought of we designing the android UI is that we knew there wasn't much we could fit in due to the various sizes of different phones. But we decided to do it in two pages, similar to the website and desktop in order to keep a similar feel across all over devices. We knew that we wanted to go with a search function instead of lists of stocks. This made it much more user friendly for the application. On the first page once starting up the application, you are met with a logo and the the search bar. Once you search for a stock it will take you to the second page where you are met with a search bar at the top, the diagrams which you can

swipe between to view all three view and at the bottom the textual view.

Front **Result**

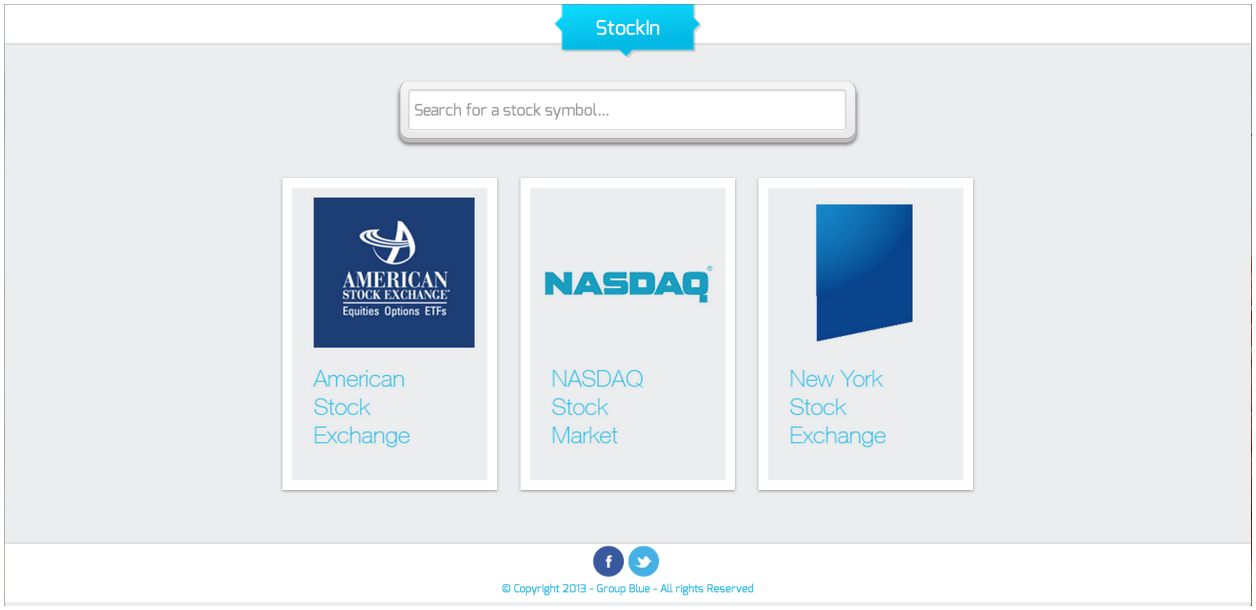


Not much have changed here, but once we started with android. We quickly realised that it was not possible to have both search and the diagram with textual news on the same page, it would take up too hard for smaller resolution to view everything. With this in mind we removed the search from the second page and solved the problem by using navigation drawer.

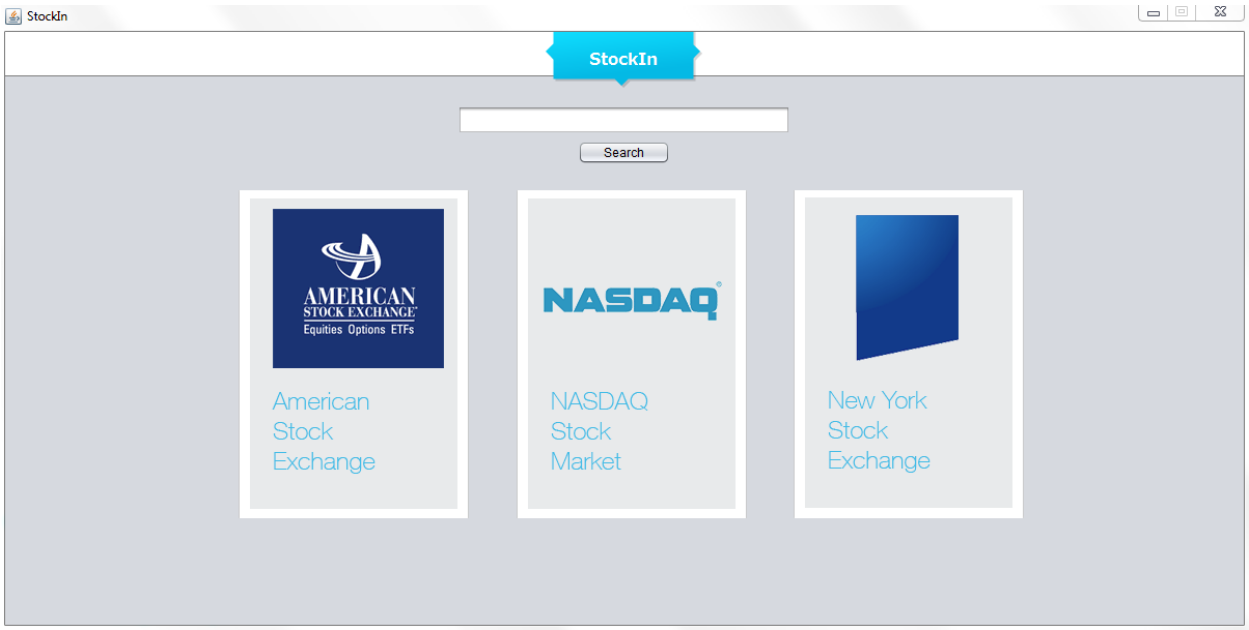
Navigation drawer(More info: http://developer.android.com/design/patterns/navigation-drawer.html) has been very popular lately with many big applications. You simply swipe from left to right and it brings out a menu, in our case search, charts and news(Fullscreen rss feed). Then by clicking search it will bring the user back to the first page. This will come naturally to the average android user since it's used in many applications nowadays.

6.5 Result

Website

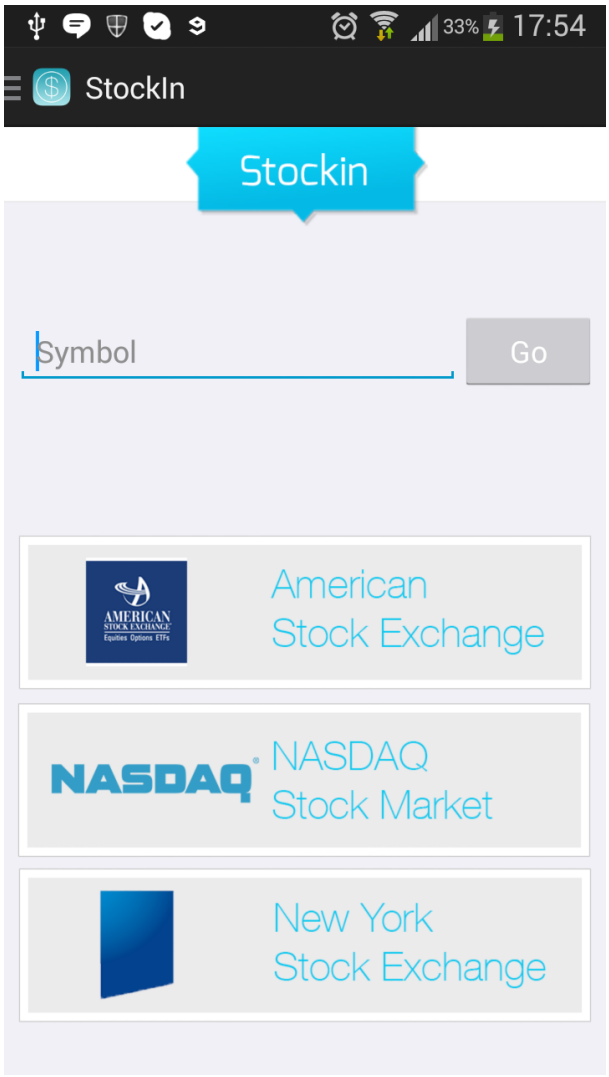


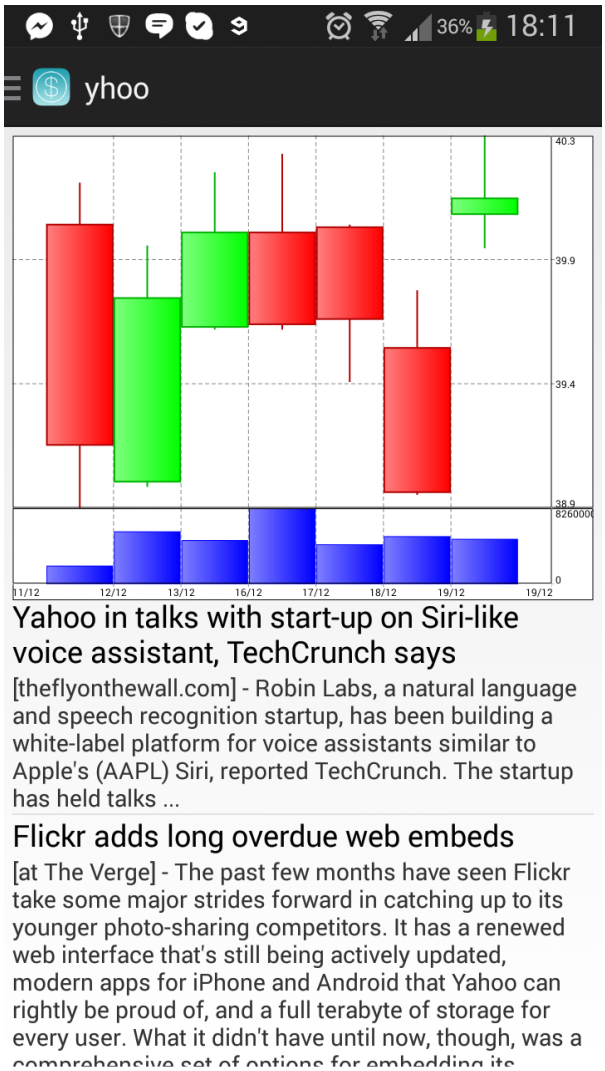
Java


StockIn
StockIn
Search
AMERICAN
STOCK EXCHANGE
Equities Options ETFs
American
Stock
Exchange
NASDAQ
NASDAQ
Stock
Market
New York
Stock
Exchange


StockIn
StockIn
yhoo
Search
Charts
Candlestick
Price
42.0
41.5
41.0
40.5
40.0
39.5
39.0
38.5
38.0
12-Dec
13-Dec
14-Dec
15-Dec
16-Dec
17-Dec
18-Dec
19-Dec
20-Dec
News
Google Fined $1.2 Million by Spain's Privacy Watchdog
EU has "serious doubts" about Italy's proposed Web tax
Will Obama's legacy be stained by technology scandals?
[video] What Does 2014 Hold for M&A and Technology?
YouTube for Roku Renders Chromecast Obsolete
Yahoo in talks with start-up on Siri-like voice assistant, TechCrunch says
Flickr adds long overdue web embeds
Flickr adds long overdue web embeds
Yahoo to Facebook Get What They Wanted From NSA Review
Yahoo/Siri App Is Actually A Real Prototype Built By Robin Labs And Commissi
Flickr's redesign continues with a new style for embedding pics on other website
Video of Yahoo's answer to Siri and Google Now is not a Yahoo product
Yahoo may be trying to build a Siri and Google Now killer
Fantasy Football: Yahoo's Time To Catch The Ball Of Profit
Brindese el Regalo del Tiempo con Texas.gov/Conductor
Candlestick
Volumechart
Pricechart

Android


33% 17:54
StockIn
Stockin
Symbol
Go
American
Stock Exchange
NASDAQ
NASDAQ
Stock Market
New York
Stock Exchange


36% 18:11
yhoo
40.3
39.9
39.4
38.9
11/12
12/12
13/12
16/12
17/12
18/12
19/12
19/12
Yahoo in talks with start-up on Siri-like voice assistant, TechCrunch says
[theflyonthewall.com] - Robin Labs, a natural language and speech recognition startup, has been building a white-label platform for voice assistants similar to Apple's (AAPL) Siri, reported TechCrunch. The startup has held talks ...
Flickr adds long overdue web embeds
[at The Verge] - The past few months have seen Flickr take some major strides forward in catching up to its younger photo-sharing competitors. It has a renewed web interface that's still being actively updated, modern apps for iPhone and Android that Yahoo can rightly be proud of, and a full terabyte of storage for every user. What it didn't have until now, though, was a

7. Technology Report

7.1 Introduction

The purpose of this document is to provide a detailed overview of the tools, libraries and programming languages used by the group during the development of the project. Also motivating why we choose to use the database and tools that we did.

7.2 Development Tools

- Group members used either Sublime Text 2 or Eclipse as their developing environment.
- Erlang shell was used in the backend of our system.
- Windows, OS-X and Linux was used under the development of this project.
- For the project management we used Scrum.
- In this project we used a non-relational database (CouchDB)
- Android phone for testing the android application.

Sublime Text 2 is a simple text editor used by both backend and webpage. The reason we chose it is because its a very simple and clean text editor with syntax highlighting. The reason why we didn't use emacs for the project was because during the beginning of the term when we had erlang lessons with Robert Virding (One of the co-founders of erlang). He told us to stick to our basic text editors if we never used emacs before.

Android & Desktop both used Eclipse because they were working with Java as programming language, the IDE was used in the first term and something the members are very familiar with.

The reason why we choose a non-functional database (CouchDB) was because first off we wanted to try out the non-functional database and learn it's way of operating with data. Because during the first term project we used relational database SQL. It was also because CouchDB was written in erlang which is correspondent to this terms course and a major part of what the project has to be done in.

7.3 Hosting

7.3.1 Cloudant - CouchDB

Cloudant (https://cloudant.com/) is distributed database as a service(DBaaS), good in handling JSON data. It's used by many big companies such as, Samsung, IBM, DHL, Microsoft, Pearson etc. We tried many other online databases to host our data. This was by far the fastests we have found. Other options were Iris couch (http://www.iriscouch.com/). Compared to Iris couch

Cloudant was offering a much faster connection. By trying the our applications, a search with iris couch would take from 5-6 seconds while cloudant offered less than 1 second search.

7.3.2 Heroku - Website

Heroku (https://www.heroku.com/) is a very fast and easy to update hosting service. It was first developed to host ruby in it's core and this fits our website perfectly because our core is made with ruby. With only a few git command the live website updates in mere seconds, this made it very convenient while developing the website itself.

7.4 Libraries

7.4.1 JFreechart

JFreechart provided us with the necessary charts that fulfilled the requirements of the project. Jfreechart is used in our Java desktop application which uses the Candlestick, volume and bar chart to provide the user with information based on their choice of stock.
Link to library: http://www.jfree.org/jfreechart/

7.4.2 Highcharts

We used a Javascript based library on the website with a friendly interface that also provides the necessary charts. It's compatible with JSON which is holding all the information for a stock that can be implemented in the graphs later on.
Link to library: http://www.highcharts.com/

7.4.3 Mochiweb/Mochijson2

We are using mochijson2 of the mochiweb library in the backend, for decoding the JSON object returned to us from the sources. By decoding it we are easily work with the JSON data and extract the data needed.

Link to library: https://github.com/mochi/mochiweb

7.4.4 LightCouch

In order to create a connection between the desktop application and the database we used a library called LightCouch which has multiple defined functions that helps us to make a connection with CouchDB. LightCouch can also request different documents from the database and return the data.

Link to library: http://www.lightcouch.org/

7.4.5 Gson

Gson is a library that we used for converting Java objects into JSON and can also be used to convert a JSON string to a java object. This basically gave us the correct structure in order for our charts to work.

Link to library: http://code.google.com/p/google-gson/

7.4.6 Feed4j

This library helped us to request news for the searched stock by parsing XML files and returning the specific data that was selected.

Link to library: http://www.sauronsoftware.it/projects/feed4j/

7.4.7 jQuery

JQuery is a javascript library that can add a lot of different functions and animations to your website for a better looking interface.

Link to library: http://jquery.com/

7.4.8 Google rss

The library is helping us to fetch news from different sources by parsing the xml files.The library is coded in Javascript which gives us a lot of flexibility.

Link to library: https://developers.google.com/feed/

7.4.9 Sinatra

Sinatra is a framework written in Ruby that gave us the opportunity to quickly set up a web-application.

Link to library: http://rubygems.org/gems/sinatra

7.4.10 Couchrest

Couchrest was used to establish a connection with our online database and gave us the possibility to request specific data.

Link to library: https://github.com/jchris/couchrest

7.4.11 stock-chart

stock-chart is a open source project that provides us with different charts that we can use for our android device.

Link to library: http://stockchartview.org/

7.4.12 android-rss

The library is used for grabbing news for our stocks.

Link to library:
https://www.assembla.com/code/andoridtutorials/subversion/nodes/LunchList/libs/android-rss.jar?rev=76)

7.5 Documentation Tools

- OnTime (http://www.ontimenow.com/) was used to set up individual goals and managing the time.
- Google drive helped us to work jointly on the documentations.
- Dropbox to store and manage our backups in case of failures.
- Visual Paradigm & Draw.io was used for drawing UML diagrams.

7.6 Programming Languages

- HTML/CSS
- Javascript
- Java
- Erlang
- Xml
- Ruby

7.7 Definitions, Acronyms and Abbreviations

JSON – JavaScript object notation is a way to transfer data in a compact way across the network connections.

RSS – Allows you to easily stay informed by retrieving the latest news from different websites. RSS stands for “Really Simple Syndication”.

NoSql – Non relational gives us more control over the database but requires a lot of configuration and is time consuming. NoSql is best fitted for databases like CouchDB and MongoDB.

Backend – Basically an application that interacts directly with the front end of the system. Back-end biggest priority is to make hidden calculations and send data to the interface.

Frontend – Grabbing data from the Back-end and displaying the data in a user friendly way. Can be a website, Desktop or a mobile application.