Android S&P 500 Stock Quote Application
Project Design

Completed by: Mathew Yamasaki

Document version PD-1.0

Completion date: 12/21/12

Document Change History

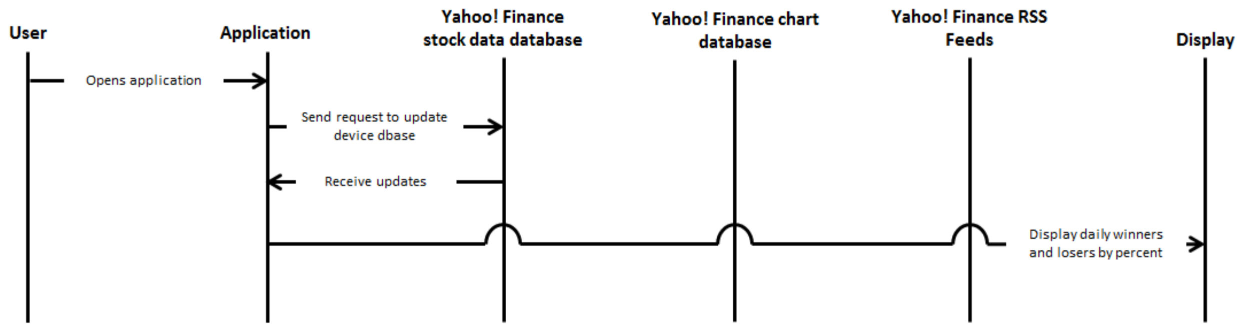
Version Number	Date	Contributor	Description
PD-D-0.1	11/18/12	Mathew Yamasaki	Document creation
PD-D-0.2	11/25/12	Mathew Yamasaki	• Add event trace scenario for when a user enters an invalid ticker symbol or company name. • Correct `class SQLiteDatabase()` variable `patio` to `peRatio`. • Add `dailyHigh` to `class SQLiteDatabase()` and change the order of the variables to reflect database column order. • Remove `dailyLowPrice` and replace with `dailyLow`
PD-D-0.3	12/20/12	Mathew Yamasaki	• Remove "Team C" logos
PD-D-0.4	12/20/12	Mathew Yamasaki	• Change section designators from Roman numerals to numbers • Add additional event trace diagrams scenarios to Section 1 • Remove all instances of "or company name"
PD-1.0	12/20/12	Mathew Yamasaki	• Update Section 2, Class Design

Table of Contents

1. Event-Trace Diagrams

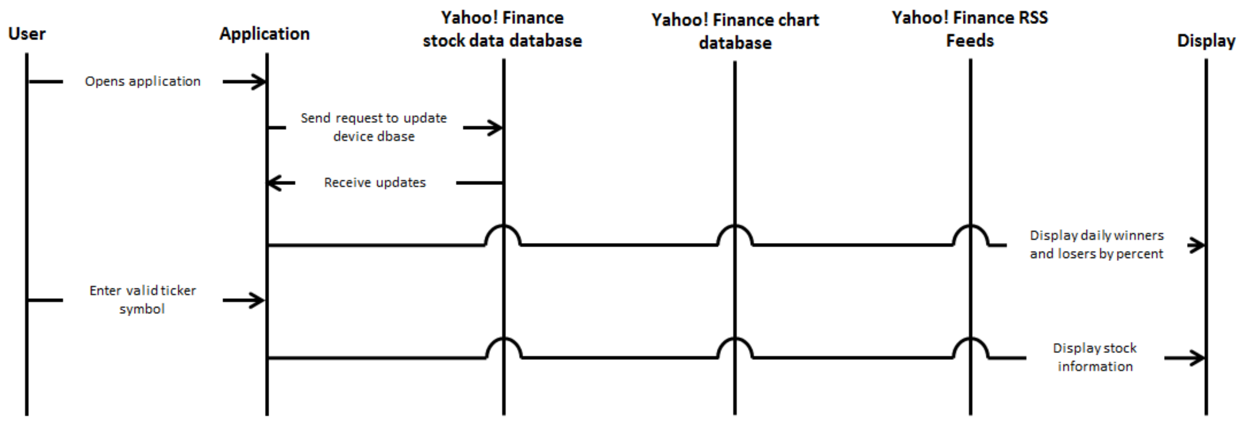
1.1 Scenario 1: Partial Application Execution

In Scenario 1, the user opens the application which then sends an update request to the Yahoo! Finance data server. The requested data is sent from the server and updates that application database. The user then enters a stock ticker symbol and the predefined data are output to the display.


User
Application
Yahoo! Finance stock data database
Yahoo! Finance chart database
Yahoo! Finance RSS Feeds
Display
Opens application
Send request to update device dbase
Receive updates
Display daily winners and losers by percent

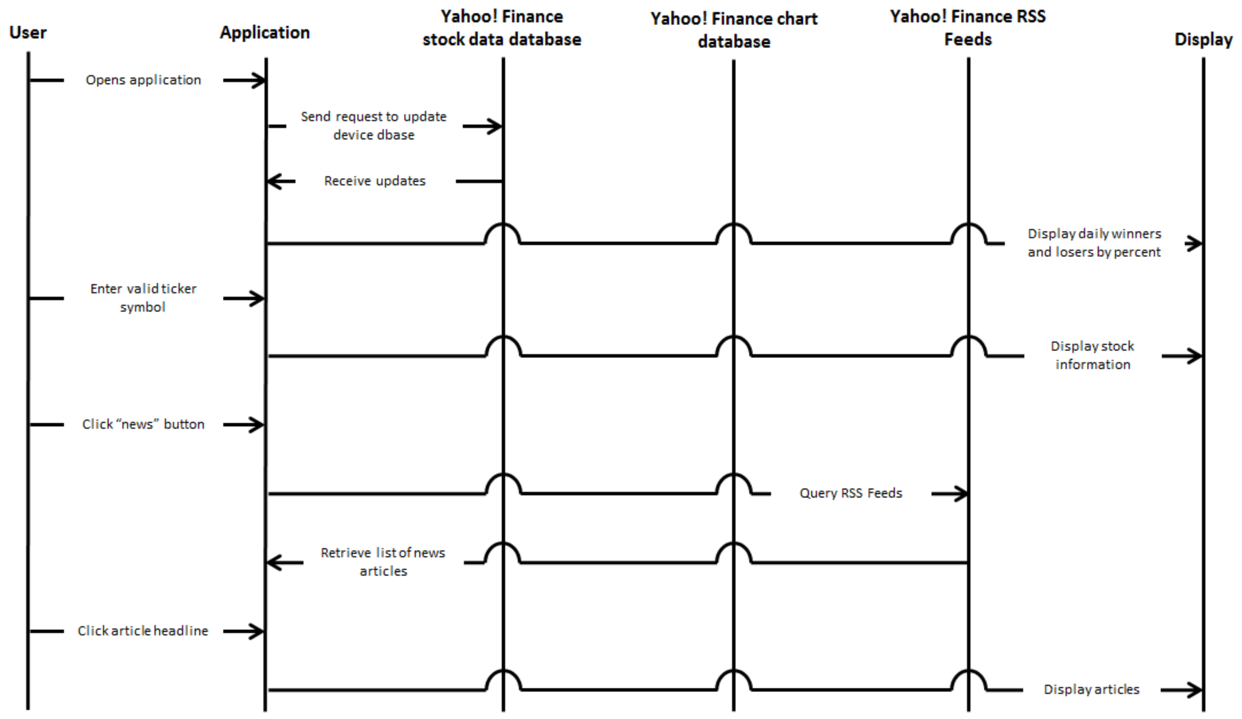
1.2 Scenario 2: Retrieving Stock Information

In Scenario 2, the user opens the application which then sends an update request to the Yahoo! Finance data server. The requested data is sent from the server and updates the application database. The user then enters a valid stock ticker symbol. The application displays the stock's information in the second Activity.


User
Application
Yahoo! Finance stock data database
Yahoo! Finance chart database
Yahoo! Finance RSS Feeds
Display
Opens application
Send request to update device dbase
Receive updates
Display daily winners and losers by percent
Enter valid ticker symbol
Display stock information

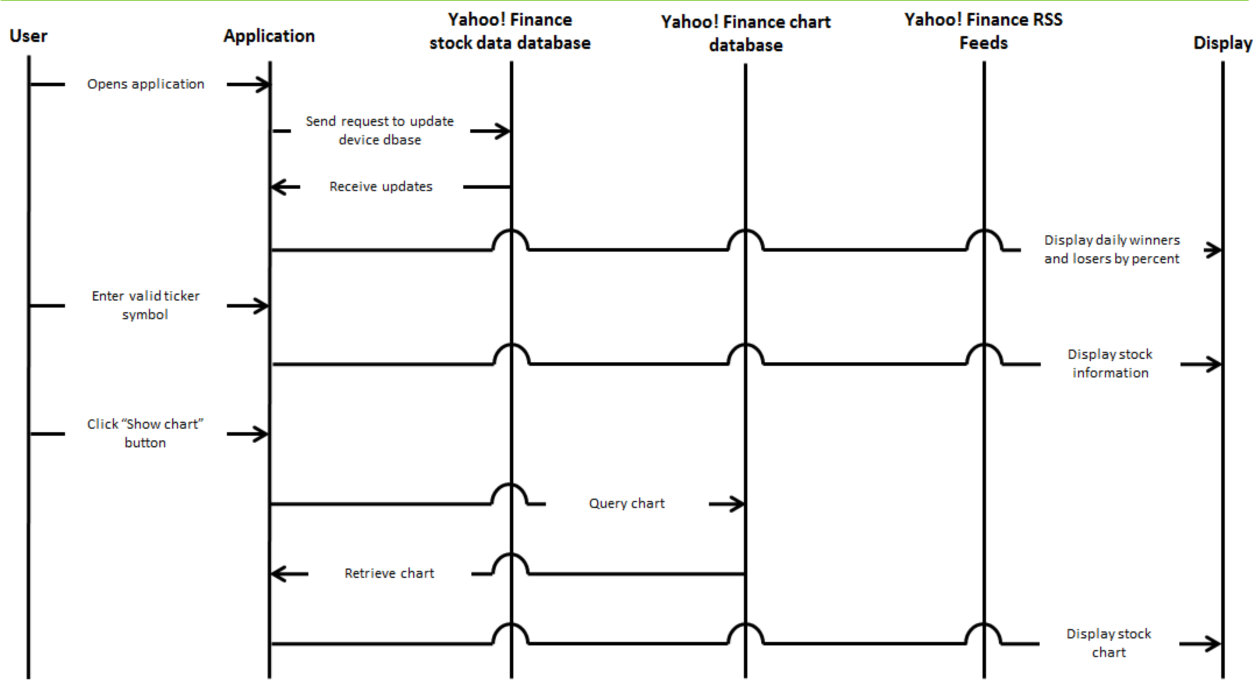
1.3 Scenario 3: Retrieving Stock News

In Scenario 4, the user opens the application which then sends an update request to the Yahoo! Finance data server. The requested data is sent from the server and updates the application database. The user then enters a valid stock ticker symbol. The application displays the stock's information in the second Activity. The user then clicks the "Show news" button which sends a query to Yahoo! Finance RSS Feeds. A list of news articles containing references to the stock's ticker symbol is returned. The user selects a desired article headline which is then displayed in the Internet browser.


User
Application
Yahoo! Finance stock data database
Yahoo! Finance chart database
Yahoo! Finance RSS Feeds
Display
Opens application
Send request to update device dbase
Receive updates
Display daily winners and losers by percent
Enter valid ticker symbol
Display stock information
Click "news" button
Query RSS Feeds
Retrieve list of news articles
Click article headline
Display articles

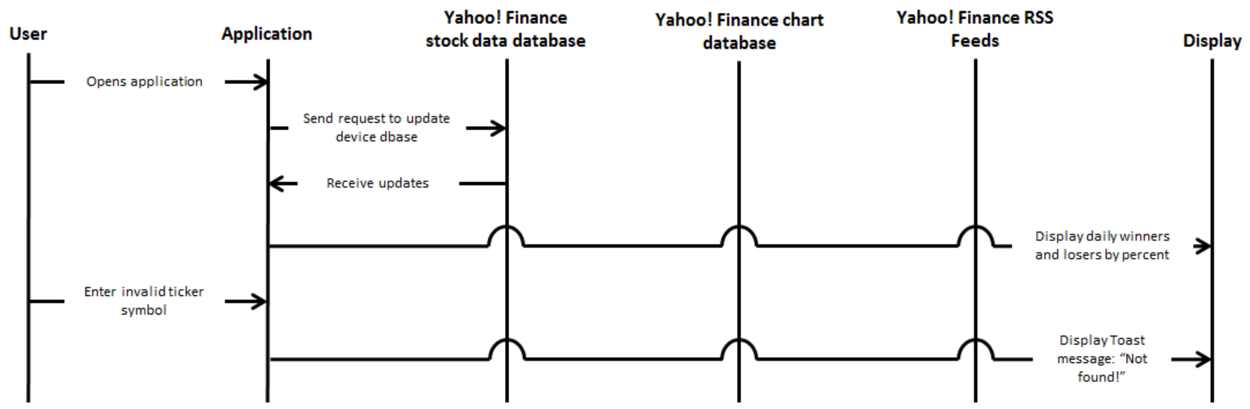
1.4 Scenario 4: Retrieving a Stock Chart

In Scenario 4, the user opens the application which then sends an update request to the Yahoo! Finance data server. The requested data is sent from the server and updates the application database. The user then enters a valid stock ticker symbol. The application displays the stock's information in the second Activity. The user then clicks the "Show chart" button which sends a query to Yahoo! Finance. The chart image is retuned and displayed to the user.


User
Application
Yahoo! Finance stock data database
Yahoo! Finance chart database
Yahoo! Finance RSS Feeds
Display
Opens application
Send request to update device dbase
Receive updates
Display daily winners and losers by percent
Enter valid ticker symbol
Display stock information
Click "Show chart" button
Query chart
Retrieve chart
Display stock chart

1.5 Scenario 5: User Enters an Invalid Ticker Symbol

In Scenario 4, the user opens the application which then sends an update request to the Yahoo! Finance data server. The requested data is sent from the server and updates the application database. The user then enters an invalid stock ticker symbol. A Toast message, "Not found!" is displayed. The user can elect to enter another ticker symbol.


User
Application
Yahoo! Finance stock data database
Yahoo! Finance chart database
Yahoo! Finance RSS Feeds
Display
Opens application
Send request to update device dbase
Receive updates
Display daily winners and losers by percent
Enter invalid ticker symbol
Display Toast message: "Not found!"

2. Class Design

```
class MainActivity extends Activity {

}

class chartActivity extends Activity {
     Get and display chart image
}

class newsActivity extends Activity {
     Retrieve and display news articles pertaining to stock
     ticker symbol
}

class Stock {
     private int _id;
     private String _ticker;
     private double _closing_price;
     private double _daily_dollar_change;
     private String _daily_percent_change;
     private double _daily_high;
     private double _daily_low;
     private double _52_week_high;
     private double _52_week_low;
     private double _pe_ratio;
     private int _volume;
     private double _50_day_moving_average;
     private String _name;

     All getter and setter methods
}

class YahooFinanceDataHelper {
     Methods to query Yahoo! Finance for
}

void getInput(String tickerSymbol or companyName) {
     Scan SQLiteDatabase;
     if (tickerSymbol or companyName is true) {
          Call displayData();
     }
     else {
          Display "not found" message
     }
}

class Display {
     void displayData(double currentPrice, double
          dollarChange, double percentChange, double
          week52high, double week52low, double dayLowPrice,
          double peRatio, int dailyVolume, double ma50day)
          {

     create TextView object;
     set text size;
     displayData() using TextView onject;
     send TextView to display activity;
     }
}

class SQLiteDatabase() {

     String stockID;
     String ticker;
     String companyName;
     String closingPrice;
     String dailyDollarChange;
     String dailyPercentChange;
     String dailyHigh;
     String dailyLow;
     String week52High;
     String week52Low;
     String peRatio;
     String volume;
     double ma50day;

     public void onCreate database(SQLiteDatabase db) {
          Create database table with indicated column names
     }

     void addStock(Stock stock) {
          Use getter methods of the Stock class to insert
          values into the db
     }

     public getStock(int id) {
          Return a Stock by db ID number
     }
} // end class
```

3. Unresolved Risks and Mitigation

The application is mostly dependent upon the functioning of external resources which are beyond the risk mitigation scope of this project

The system may be slow in responding and/or updating the application database. This could be due to diminished cellular network connectivity caused by service provider technical issues or device location in relation to cellular towers. This could also be attributed to technical issues with Yahoo! Finance. The application's preexisting design feature of utilizing an Android SQLite database provides some risk mitigation by allowing the user to use the application with the latest updates. It may be possible for the user to allow the application to automatically update the database periodically throughout the day.