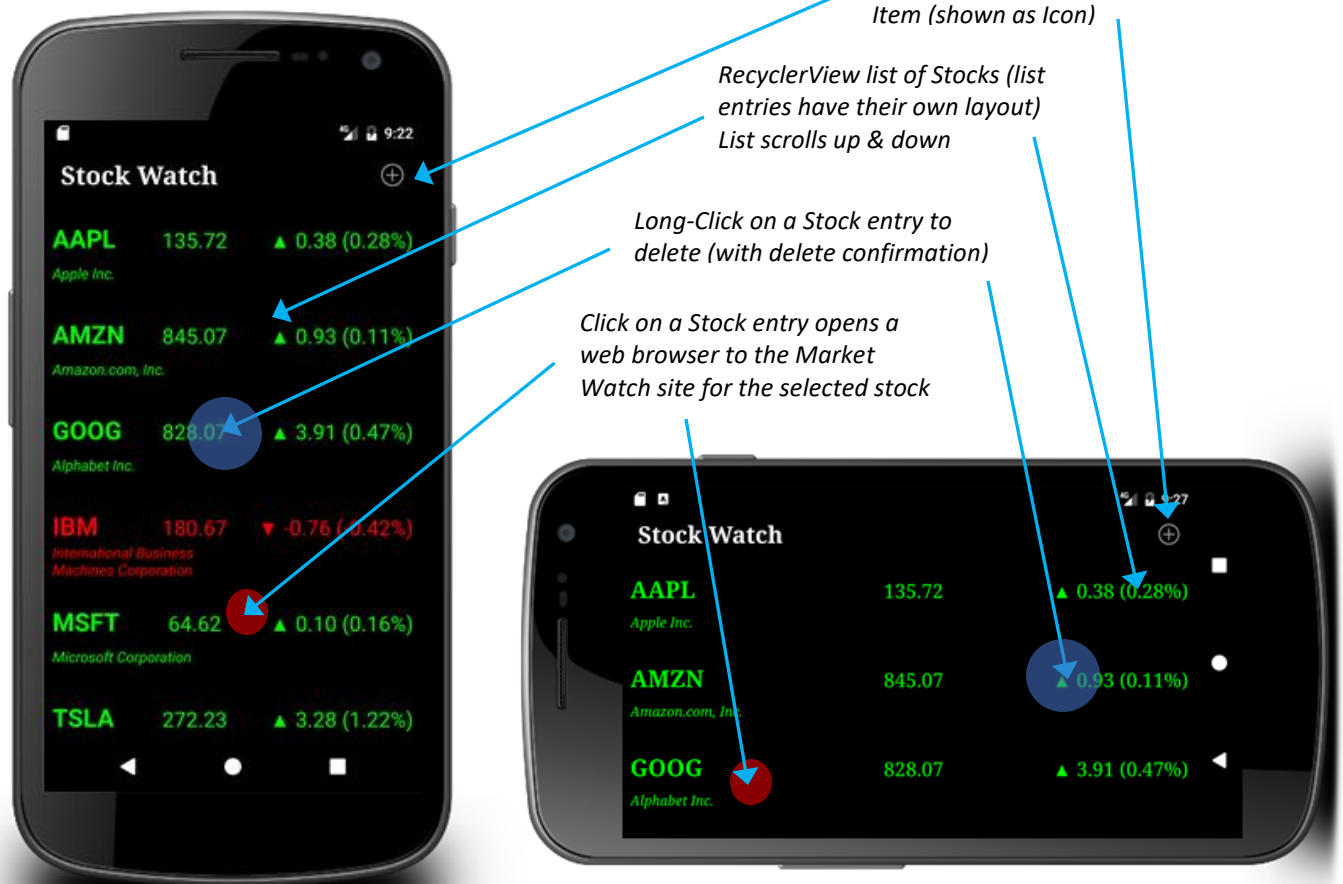


**CS 442: Mobile Applications Development****Assignment 4 – Stock Watch (300 pts)**

Uses: Internet, RecyclerView, Option-Menus, Multiple AsyncTasks, JSON Data, Swipe-Refresh, Dialogs, SQLite Database

App Highlights:

- This app allows the user to display a sorted list of selected stocks. List entries include the stock symbol (and company name), the current price, the daily price change amount and price percent change.
- There is no need to use a different layout for landscape orientation in this application.
- Selected stock symbols and the related names should be stored in the device's SQLite Database.
- A Stock class should be created to represent each individual stock in the application. Required data includes: Stock Symbol (String), Company Name (String), Price (double), Price Change (double), and Change Percentage (double).
- Clicking on a stock opens a browser displaying the *Market Watch* web page for that stock
- Swipe-Refresh (pull-down) refreshes stock data
- The application is made up of 1 activity, shown below:



Document Contents:

- A) Internet Data
- B) Application Behavior Diagrams:
- C) Application Behavior Flowcharts
- D) Database
- E) Development Plan

Application Behaviors:

- A) Add Stocks
- B) Delete Stocks
- C) Refresh Stock Data
- D) Open Browser with Stock Data

A) Internet Data:

Downloading data for a stock symbol requires 2 downloads – one download to acquire the full stock symbol and company name, and a second download to acquire the financial data for the stock. See the “Adding a Stock” flowchart later in this document for development details.

Download 1: Stock Symbol & Company Data

When selecting a stock, a user can enter a partial or complete stock symbol or company name. This information is used to download a list of stocks matching the provided name data. This downloaded list is presented to the user as a selectable list of stocks that match their criteria.

Download Source: <http://stocksearchapi.com>

NOTE: You MUST sign up with the <http://stocksearchapi.com> website to get an API KEY. Your API KEY must be supplied with all stock queries. Once you sign up (the only data they need is an email address) – log in to access your API Key (shown with Your Account).

Query Format: http://stocksearchapi.com/api/?api_key=your-api-key-here&search_text=user-entered-string

For example, if your API Key was “ABC123xyz” and the user-entered text was “CSO”, your full URL would be:

http://stocksearchapi.com/api/?api_key=ABC123xyz&search_text=CSO

Download Results Example:

Results are returned in JSON format, as a JSONArray containing the results data from the query. The data we are interested in is the “company_name” and “company_symbol”.

Your results will include either **ONE stock** (Example using search text “TSLA”):

(When only one stock matches the user’s criteria, no list-selection-dialog is needed)

```
[
  {
    "company_name": "Tesla Motors, Inc.",
    "company_symbol": "TSLA",
    "listing_exchange": "NASDAQ"
  }
]
```

Or **two or more Stocks** (Example using search text “CSO”)

(When multiple stocks matches the user’s criteria, a list-selection-dialog is needed)

```
[
  {
    "company_name": "Cornerstone OnDemand, Inc.",
    "company_symbol": "CSOD",
  }
]
```

```

        "listing_exchange": "NASDAQ"
    },
    {
        "company_name": "CSOP FTSE China A50 ETF",
        "company_symbol": "AFTY",
        "listing_exchange": "NYSE ARCA"
    },
    {
        "company_name": "CSOP MSCI China A International Hedged ETF",
        "company_symbol": "CNHX",
        "listing_exchange": "NYSE ARCA"
    },
    {
        "company_name": "CSOP China CSI 300 A-H Dynamic ETF",
        "company_symbol": "HAHA",
        "listing_exchange": "NYSE ARCA"
    }
]

```

Or **NO stock** (no match found)

Response Code: 404 Not Found

Download 2: Stock Financial Data

When you have the desired stock symbol (and company name), you use the *stock symbol* to download financial data for that stock.

Download Source: <http://finance.google.com> (No sign up or API Key is necessary)

Query Format: <http://finance.google.com/finance/info?client=ig&q=user-selected-stock-symbol>

For example, if the selected stock symbol was TSLA, your full URL would be:

<http://finance.google.com/finance/info?client=ig&q=TSLA>

Download Results:

Results are returned in JSON format, as a JSONArray containing the results data from the query. The data we are interested in is highlighted below.

Your results will include either **ONE stock** (Example using search text "TSLA"):

Note, the first 2 characters in the JSON results are "//". These must be removed before parsing as JSON data.

```

//[
{
    "id": "12607212",
    "t": "TSLA",
    "e": "NASDAQ",
    "l": "277.99",
    "l_fix": "277.99",
    "l_cur": "277.99",
    "s": "0",
    "lft": "3:50PM EST",

```

```

    "lt": "Feb 21, 3:50PM EST",
    "lt_dts": "2017-02-21T15:50:19Z",
    "c": "+5.76",
    "c_fix": "5.76",
    "cp": "2.12",
    "cp_fix": "2.12",
    "ccol": "chg",
    "pcls_fix": "272.23"
  }
]

```

Note, the first 2 characters in the JSON results are `"/"`. These must be removed before parsing as JSON data.

Or **NO stock** (no match found)

Response Code: 400 Bad Request

The data contained in the JSON financial data is as follows (those we need are highlighted):

- `id` Internal Stock Identifier
- `t` Ticker
- `e` Exchange
- `l` Last Trade Price (lowercase "L")
- `l_fix` (Same as last trade price)
- `l_cur` Last Trade Price With Currency (Same as Last Trade Price if stock trades in Dollars, otherwise alternate currency is shown.)
- `s` LastTradeSize (*always seems to be zero*)
- `lts` Last Trade TimeStamp
- `lt` Last Trade Time & Date Formatted
- `lt_dts` Last Trade Time & Date Formatted – Z time
- `c` Price Change Amount (always with +/-)
- `c_fix` Price Change Amount (No "+" when positive, "-" if negative)
- `cp` Price Change Percentage
- `cp_fix` (Always the same as Price Change Percentage)
- `ccol` "chr" or "chg"
- `pcls_fix` Previous Close Price

Always check to be sure the "Ticker" value (the stock symbol) matches the symbol you were trying to download.

The **Stock Symbol** and **Company Name**, and the **Last Trade Price**, **Price Change Amount** & **Price Change Percentage** make up the data for one stock. Your Stock class should contain these 5 data elements.

Using these 5 data elements, you can create a Stock object with all data reflecting the user's choice.

B) Application Behavior Diagrams:

1) App MainActivity

Each stock entry contains the Stock Symbol (AAPL), the company name (Apple Inc.), the Last Trade Price (135.72), the price change direction (▲ for positive Price Change Amount, ▼ for negative Price Change Amount), the Price Change Amount (0.38), and the Price Change Percentage (0.28%) in parentheses.

If the stock's Price Change Amount is a positive value, then entire entry should use a green font. If the Price Change Amount is a negative value, then entire entry should use a red font.

Clicking on a Stock entry opens a web browser to the Market Watch site for the selected stock



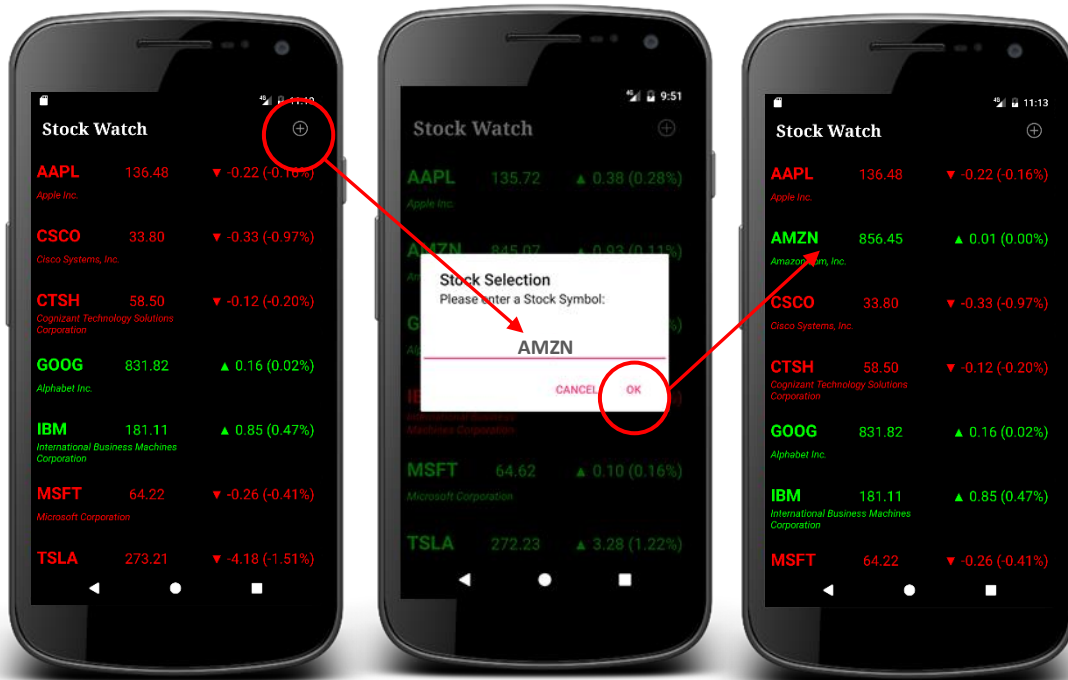
"Add Stock" action is an Options-Menu with a single item (shown as Icon)

A scrollbar should be present along the right-hand side

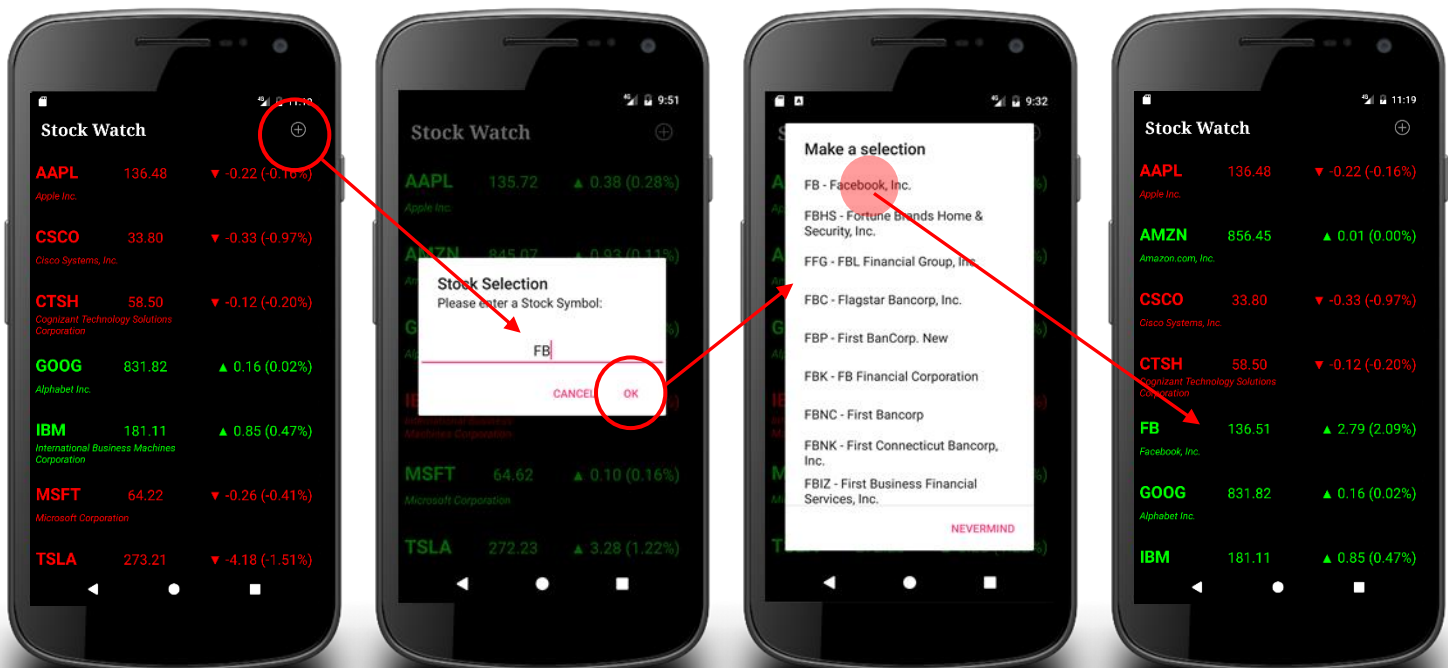
Long-Click on a Stock entry to delete (with delete confirmation)

RecyclerView list entries have their own layout

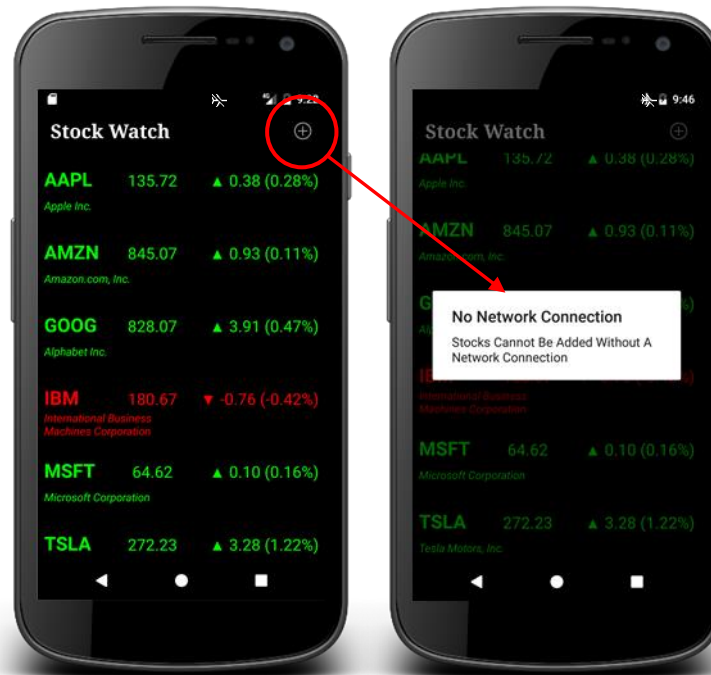
- 1) Adding a stock – only **one stock matched** the search string (*Stock Selection dialog should only allow capital letters*):



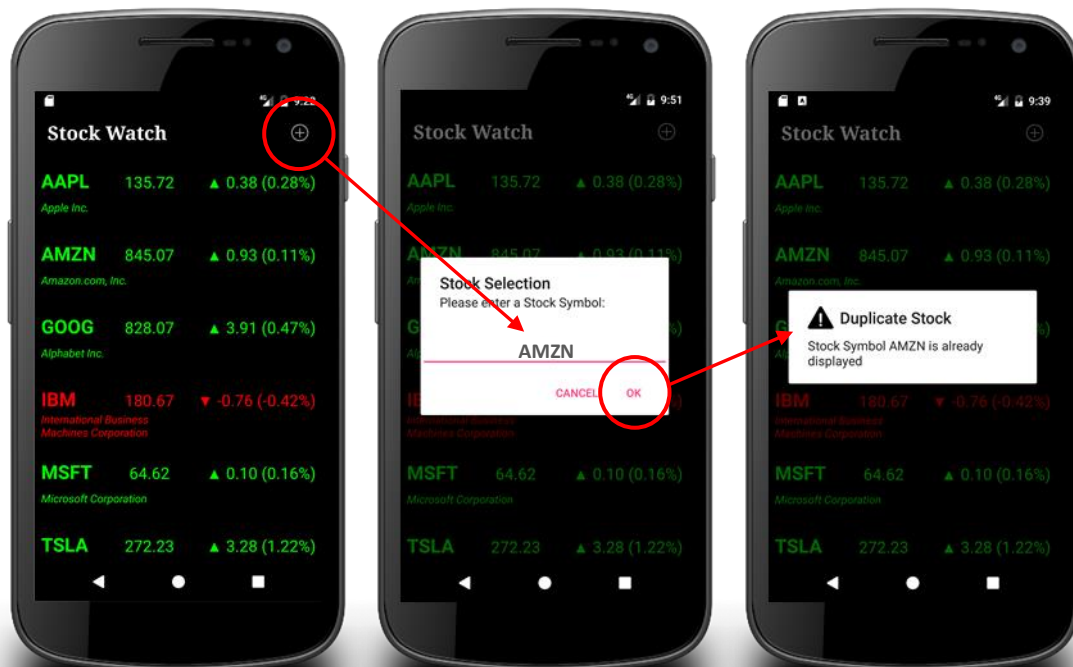
- 2) Adding a stock – **multiple stocks matched** the search string (*Stock selection dialog should display the stock symbol and company name*):



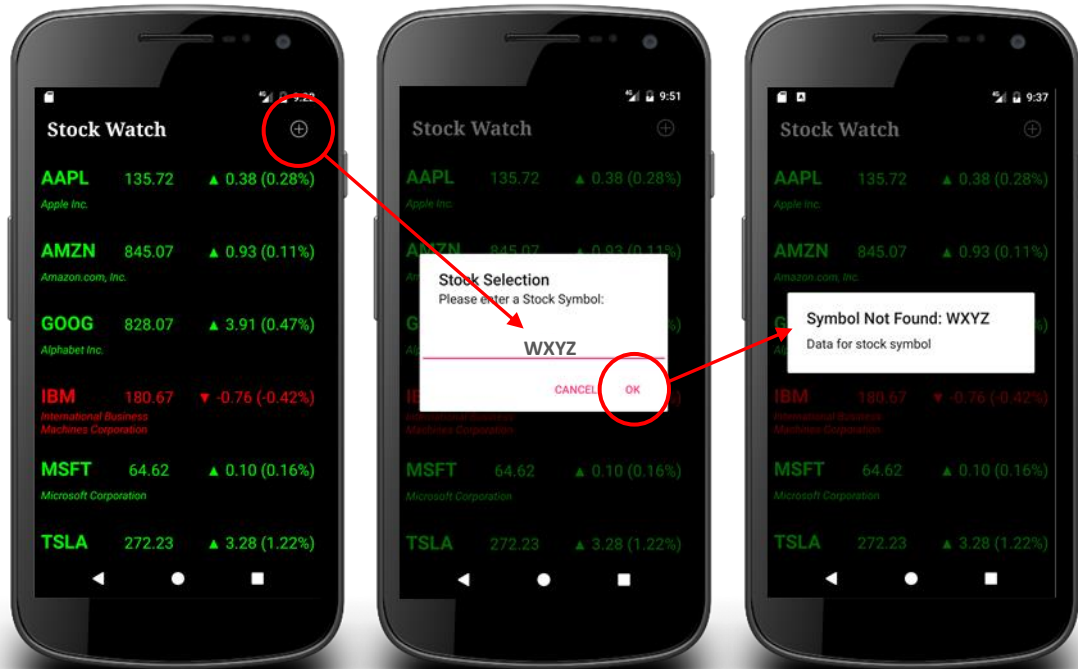
- 3) Adding a stock with no Network Connection (*No buttons on the error dialog*):



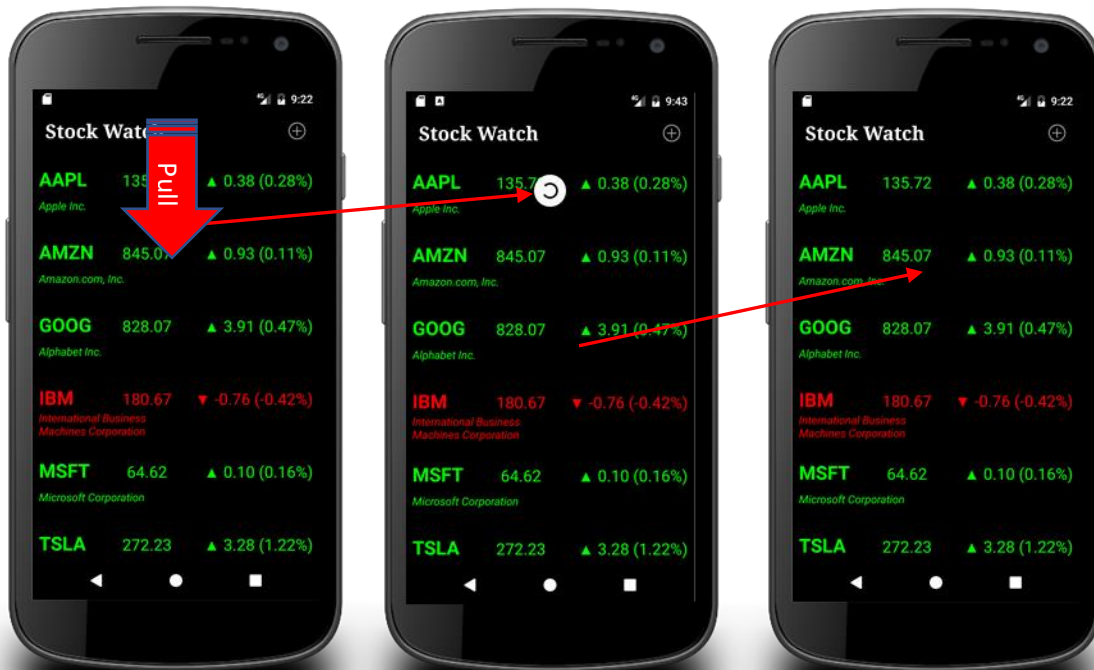
- 4) Adding a stock – specified stock is a duplicate (*Stock Selection dialog should only allow capital letters, No buttons on the dialog*):



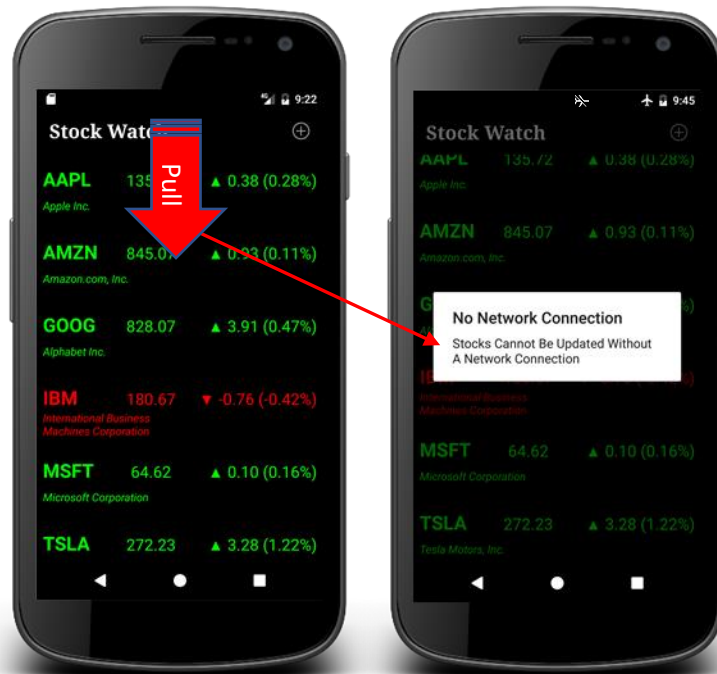
- 1) Adding a stock – specified stock is not found (*Stock Selection dialog should only allow capital letters, No buttons on the dialog*):



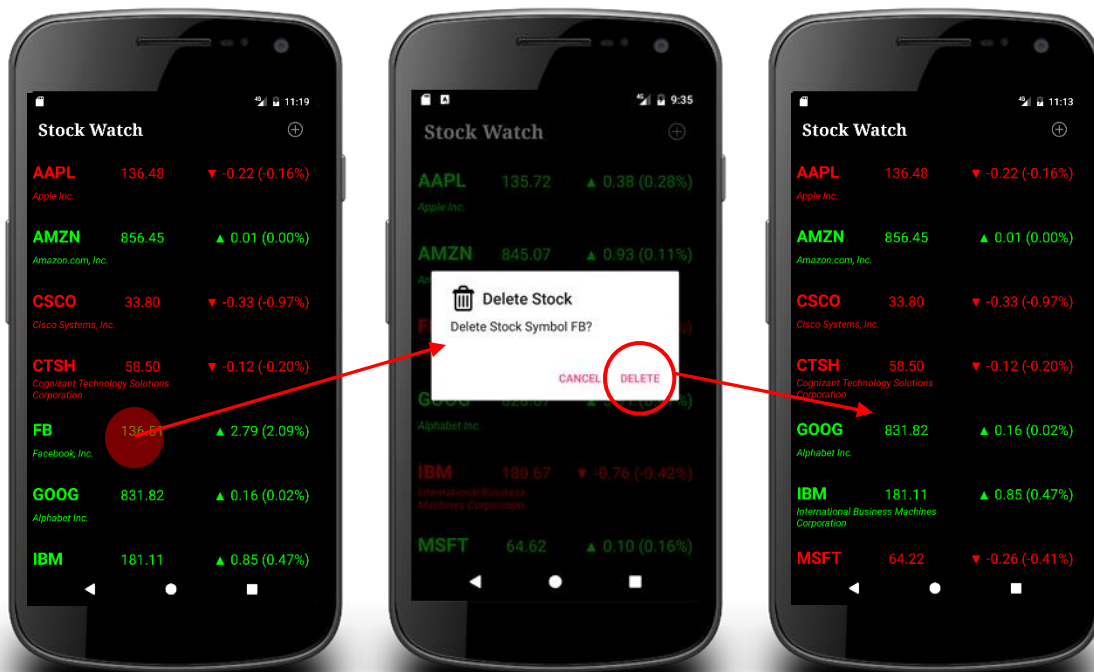
- 6) Swipe-Refresh (pull-down) reloads (re-downloads) current stock data:



7) Swipe-Refresh attempt with no network connection (*No buttons on the error dialog*):



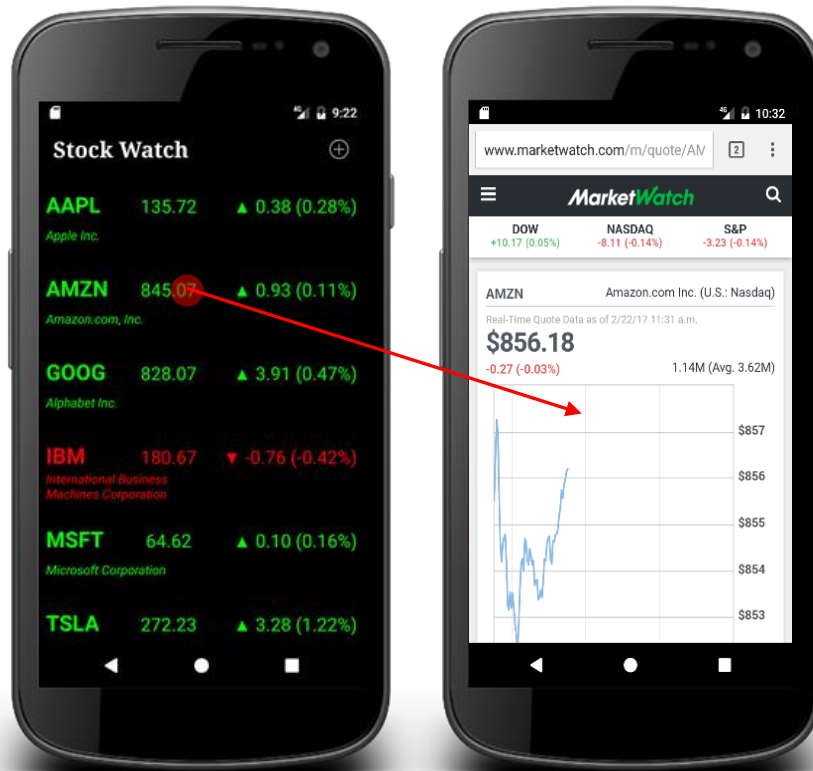
8) Long-Press on a stock to delete it:



9) Click on a stock to open the *MarketWatch.com* website entry for the selected stock:

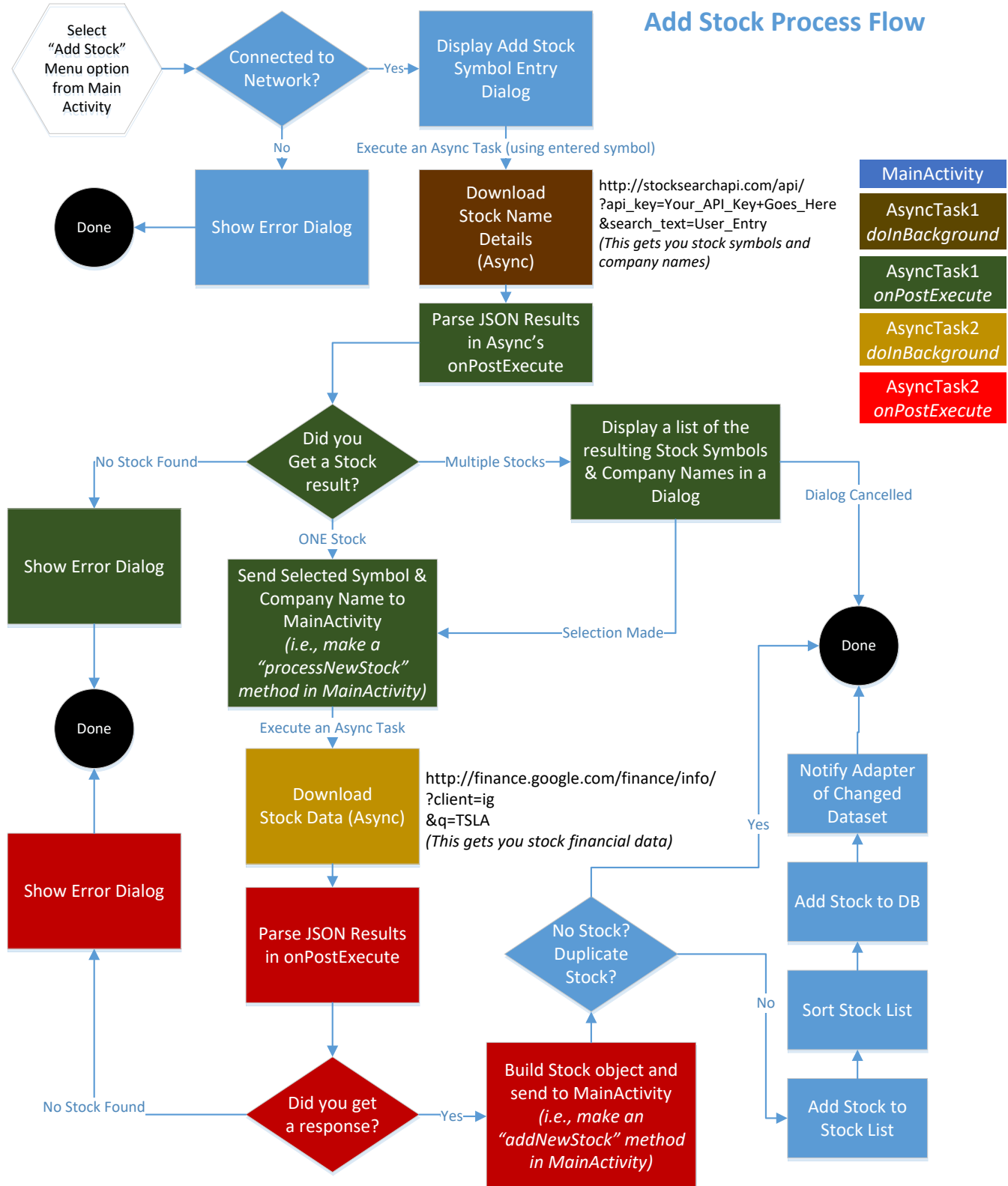
MarketWatch URL's are in the form: http://www.marketwatch.com/investing/stock/some_stock

For example: <http://www.marketwatch.com/investing/stock/TSLA>



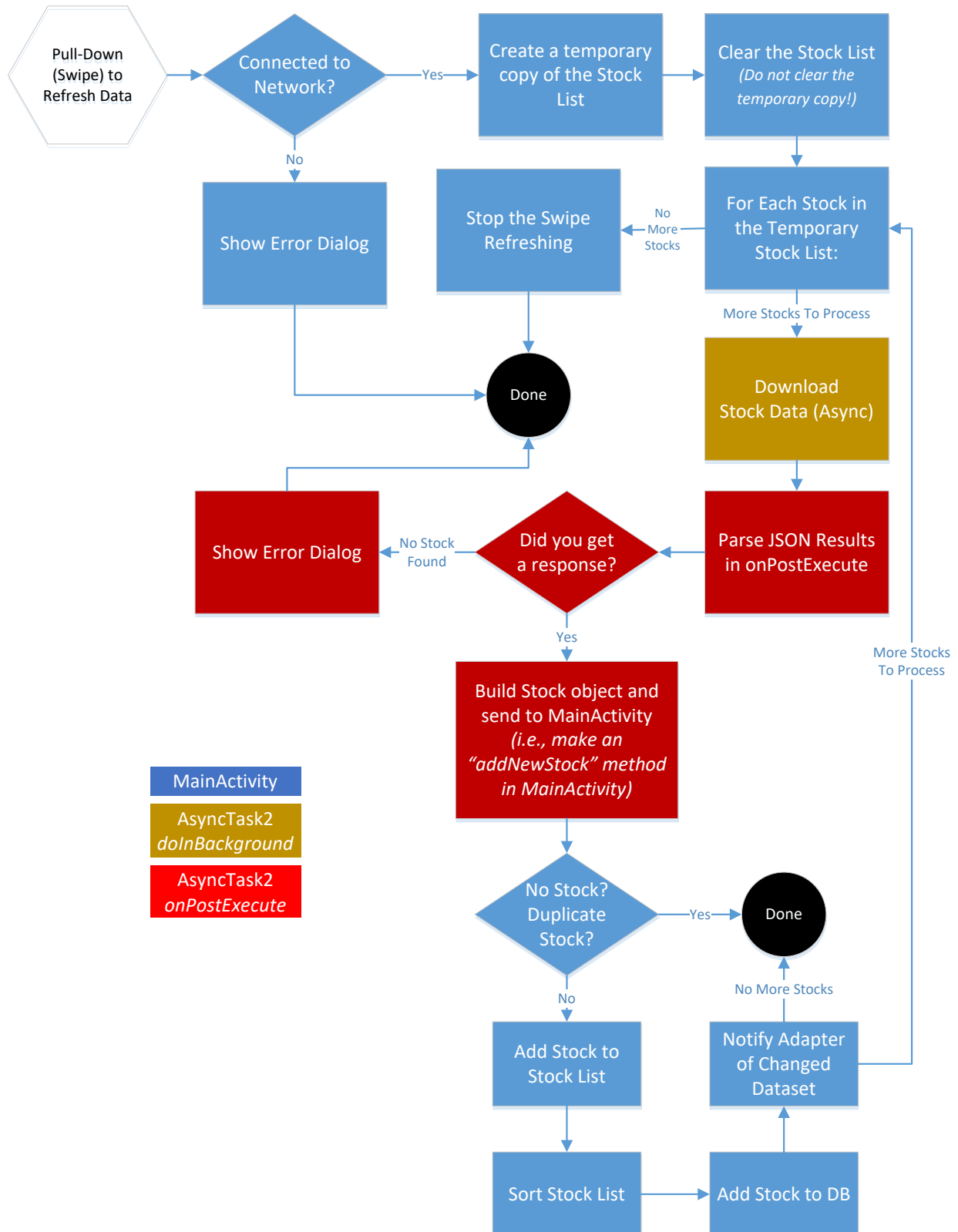
C) Application Behavior Flowcharts:

a) Adding a Stock



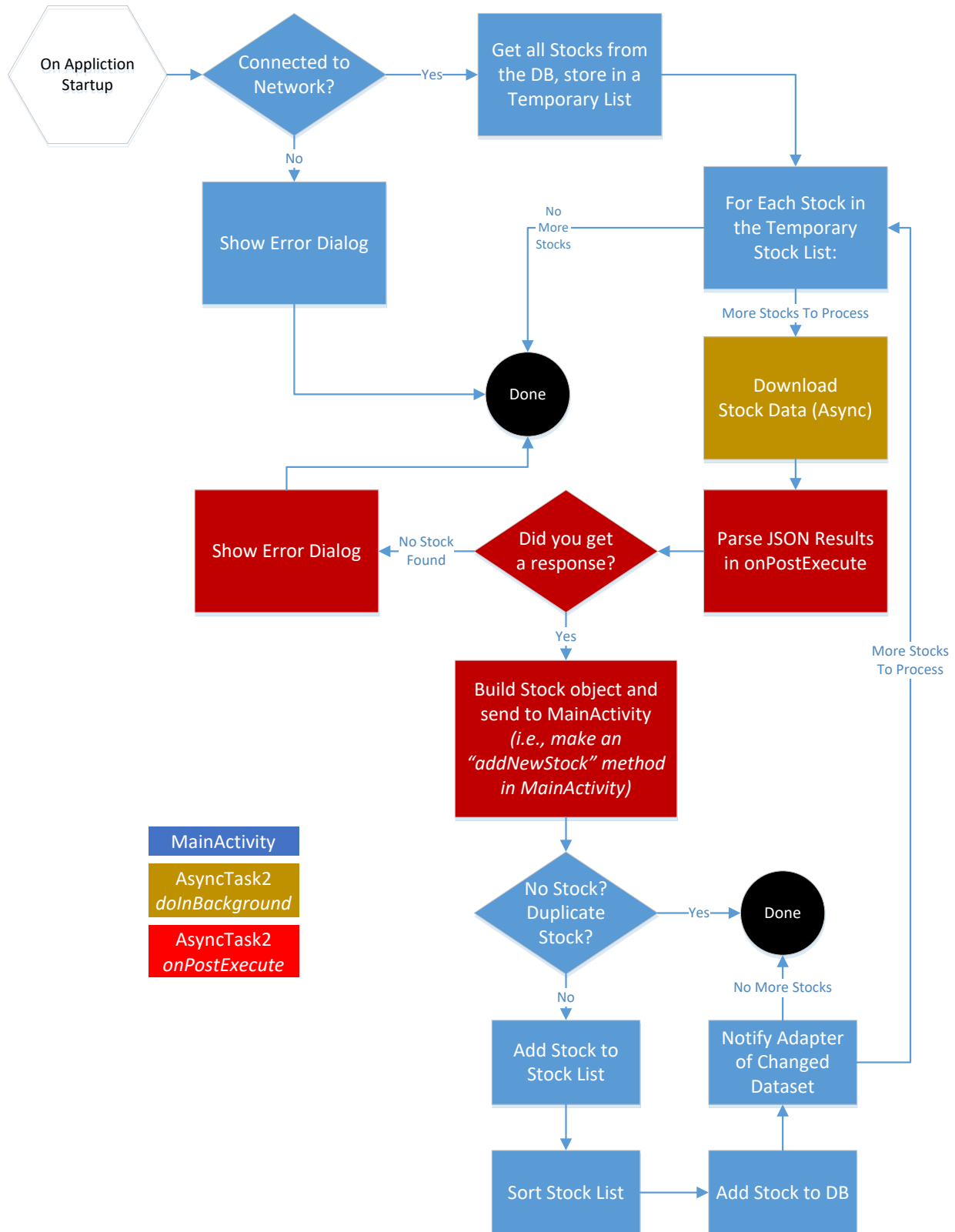
b) Swipe-Refresh (pull-down) List:

Swipe-Refresh Process Flow

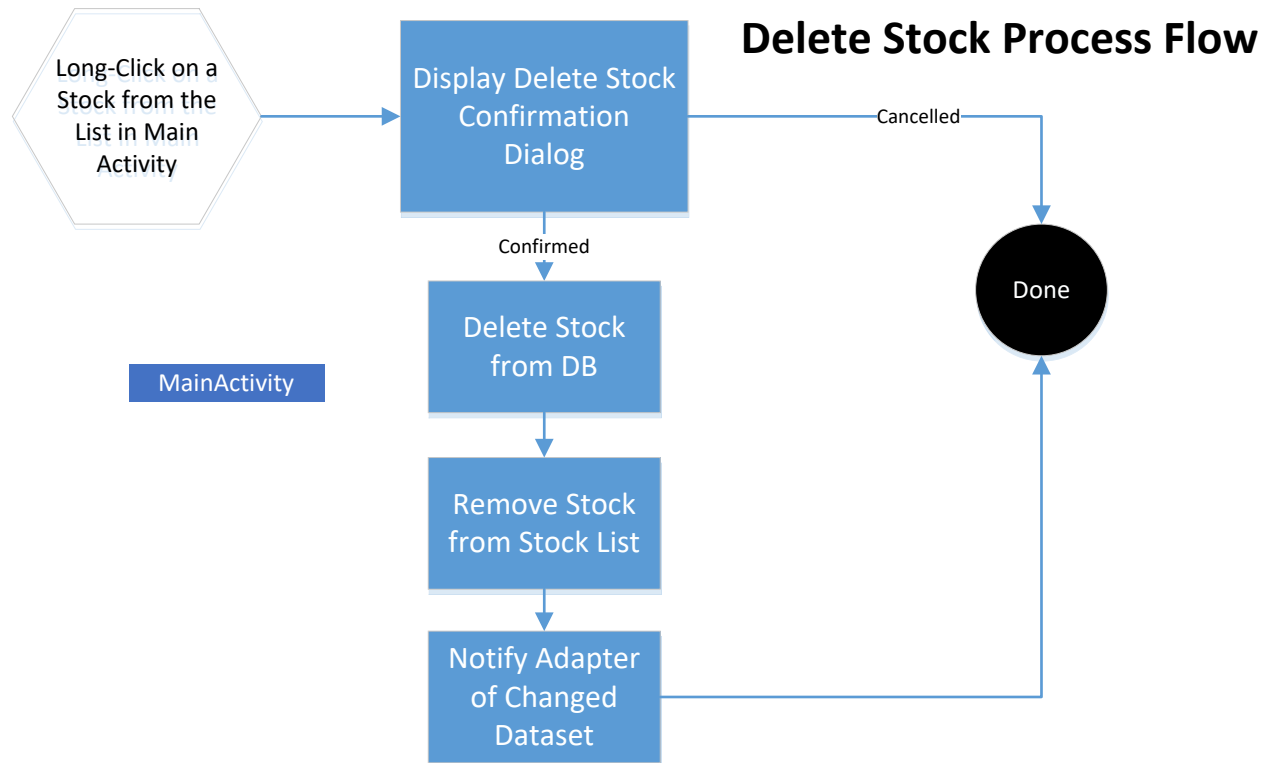


c) Sartup Process:

Startup Process Flow



d) Long-Press Delete Stock:





D) Database

Your application must store the Stock Symbol and Company Name in the android device's SQLite database. You will need a Database handler class as we have done in class (you have a posted example of a Database handler that you can use as a model).

StockWatchTable	
StockSymbol	CompanyName
AAPL	Apple, Inc
AMZN	Amazon.com, Inc.
...	...

```
private static final String DATABASE_NAME = "StockAppDB";
private static final String TABLE_NAME = "StockWatchTable";
private static final String SYMBOL = "StockSymbol";
private static final String COMPANY = "CompanyName";
```

- DB creation (done in onCreate):

```
CREATE TABLE TABLE_NAME (
    SYMBOL TEXT not null unique,
    COMPANY TEXT not null)
```

- DB Add (Sample method to add a stock to the DB):

```
public void addStock(Stock stock) {
    Log.d(TAG, "addStock: Adding " + stock.getSymbol());

    ContentValues values = new ContentValues();
    values.put(SYMBOL, stock.getSymbol());
    values.put(COMPANY, stock.getCompany());

    database.insert(TABLE_NAME, null, values);

    Log.d(TAG, "addStock: Add Complete");
}
```

- DB Delete (Sample method to delete a stock from the DB):

```
public void deleteStock(String symbol) {
    Log.d(TAG, "deleteStock: Deleting Stock " + symbol);

    int cnt = database.delete(
        TABLE_NAME, "SYMBOL = ?", new String[] { symbol });

    Log.d(TAG, "deleteStock: " + cnt);
}
```



- DB Load All (Sample method to get all stock-company entries from the DB):

```
public ArrayList<String[]> loadStocks() {

    Log.d(TAG, "loadStocks: Load all symbol-company entries from DB");

    ArrayList<String[]> stocks = new ArrayList<>();

    Cursor cursor = database.query(
        TABLE_NAME, // The table to query
        new String[]{ SYMBOL, COMPANY }, // The columns to return
        null, // The columns for the WHERE clause, null means "*"
        null, // The values for the WHERE clause, null means "*"
        null, // don't group the rows
        null, // don't filter by row groups
        null); // The sort order

    if (cursor != null) { // Only proceed if cursor is not null
        cursor.moveToFirst();

        for (int i = 0; i < cursor.getCount(); i++) {

            String symbol = cursor.getString(0); // 1st returned column
            String company = cursor.getString(1); // 2nd returned column

            stocks.add(new String[] {symbol, company});

            cursor.moveToNext();
        }
        cursor.close();
    }

    return stocks;
}
```

E) Development Plan

1) Create the base app:

- a. MainActivity with RecyclerView & SwipeRefreshLayout
- b. Stock Class
- c. RecyclerView Adapter
- d. RecyclerView ViewHolder
- e. Create fake "dummy" stocks to populate the list in the MainActivity onCreate.
- f. Add the onClick and onLongClick methods. The onLongClick should delete an entry. The onClick can open a Toast message for now.
- g. Add Stock options-menu opens dialog, on confirmation you can open a Toast message for now.
- h. SwipeRefresh callback method can open a Toast message for now.

2) Add the database elements:

- a. Create the database handler.
- b. Add database handler calls to MainActivity (load, add, delete)
- c. Code the onClick method to open the browser to the stock's Market Watch site.

3) Add the internet elements and final integration:

- a. Create the Stock Symbol - Company Name downloader/parser AsyncTask (stocksearchapi.com)
- b. Create the Stock Financial Data downloader/parser AsyncTask (finance.google.com)
- c. Add a method to MainActivity that allows the Stock Symbol - Company Name downloader/parser AsyncTask to send the newly downloaded Stock Symbol & Company Name data back to MainActivity. This method should create and execute the Stock Financial Data downloader/parser AsyncTask.
- d. Add a method to MainActivity that allows the Stock Financial Data downloader/parser AsyncTask to send the newly created Stock back to MainActivity.
- e. Implement the Add Stock feature (this uses the results of the above tasks)
- f. Implement the SwipeRefresh callback to re-download the Stock Financial Data for the loaded stocks
- g. Add alerts when startup, add, & refresh are attempted when no internet connection is available.

**Assignment Assistance**

The TAs for our course is available to assist you with your assignment if needed. Questions on assignment requirements and course concepts can be sent to the instructor.

Submissions & Grading

- 1) Submissions must consist of your zipped project folder (*please execute Build =>Clean Project before generating the zip file*).
- 2) Submissions should reflect the concepts and practices we cover in class, and the requirements specified in this document.
- 3) Late submissions will be penalized by 10% per week late. (i.e., from one second late to 1 week late: 10% penalty, from one week plus one second late to 2 weeks late: 20% penalty, etc.).
- 4) Grading will be based upon the presence and proper functionality of *all features and behaviors* described in this document.

NOTE

This assignment is worth 300 points. This means (for example) that if you get 89% on this assignment, your recorded score will be:

(89% * 300 points = 267 points)

Note that this also means that the 10% late submission penalty will be 10% * 300 points = 30 points.

If you do not understand anything in this handout, please ask.

Otherwise the assumption is that you understand the content.

Unsure? Ask!