

Assignment #5
Mobile Device Development
Stock Search Application
Due Date: 12/03/2020@11:59pm

In this assignment, you will develop an Android Mobile application, which will have the following functionality:

- Auto-complete edit box is provided to enter the stock name or symbol.
- The user selects a stock from the suggestions using 'AutoCompleteTextView'
- Stock info is retrieved using HTTP calls using the tiingo API (See Appendix)
- All HTTP API calls and JSON processing needs to happen in a background thread
- Every time the user has to wait to see the data, you must display a progress bar. The progress bar is a circular progress bar that just says "Fetching Data...".

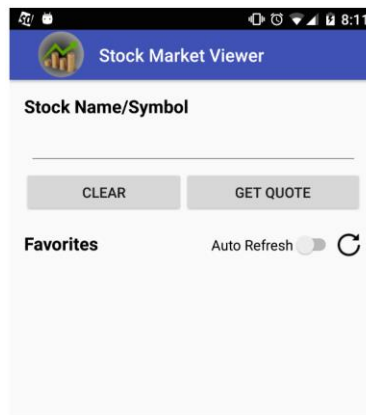


Figure 1. Search Form

Once the user has provided data and selected a result from the autocomplete list she would click on 'Get Quote', when validation must be done to check that the entered data is valid. Once the validation is successful, we would get the stock details using tiingo APIs, which would return the result in JSON format.

Implementation

Search Form

You must replicate the Search Form, as shown in Figure 1.

The interface consists of the following:

- An 'AutoCompleteTextView' for the user to enter company name or symbol.
- The user is provided with suggestions of keywords using the Tiingo Autocomplete API (See Appendix)
- Two buttons for interaction in the Search Form.
- A button 'CLEAR' to clear the 'AutoCompleteTextView' component.
- A button 'GET QUOTE' to get the quote, after validation.
- The Favorite RecyclerView showing the list of favorite stocks.
- The Favorite List starts with an empty favorite list.

The form has two buttons:

- a) GET QUOTE: Validations are first performed, when the button is clicked. If the validations are successful, then the stock details would be fetched in the Stock Details activity. However, if the validations are unsuccessful, appropriate messages would be displayed and no further requests would be made to the web service.
- b) CLEAR: This button would clear the 'AutoCompleteTextView' and also clear any validations error, if present.

AutoComplete

The user can enter the stock name or symbol in the text view to get the stock information using tiingo API. Based on the user input, the AutoComplete would display the all the matching companies and symbols (see Figure 2) by making a HTTP API request. The requests would be made only when the user enters a minimum of 3 characters to avoid unnecessary network calls. This needs to be implemented using AutoCompleteTextView.

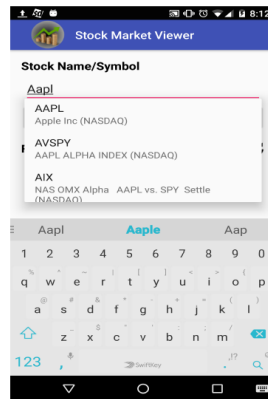


Figure 2: AutoComplete Suggestions

Validations

The following validations need to be implemented:

- Empty Entry: If the user does not enter anything in the AutoComplete component, you need to display an appropriate message to indicate the same.
- Invalid Selection: If the user enters an invalid entry which does not correspond to any valid stock symbol, display an appropriate message.

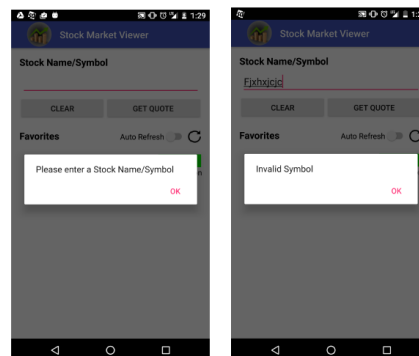


Figure 3: Validations

Once if the user input is successfully validated, the API calls should be made, otherwise appropriate messages should be displayed.

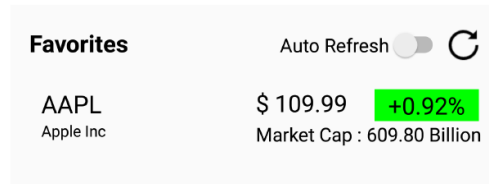
Get Quote Execution

Once the validation is successful, you should execute an HTTP request using tiingo API to retrieve. You should pass the company symbol as a parameter of the transaction when calling the API.

The tiingo API call would extract the stock details of the company symbol and returns the data in JSON format, launch the Stock details activity and display the results.

Favorite List

The Favorite stocks would be displayed in a list as per Figure 4. The data for the favorite list should be loaded from 'SharedPreferences'. For more about shared preferences please refer to the appendix.




Favorites		Auto Refresh <input type="checkbox"/> 
AAPL Apple Inc	\$ 109.99 Market Cap : 609.80 Billion	+0.92%

Figure 4: Favorite Stocks

Additionally, there needs to be a few important features:

- Automatic Refresh – A switch: when it is on it should refresh only the price and change percentage column every 10 seconds.
- Refresh button – Should refresh only the price and change column fields and not the rest of the table.
- Stock Details – The stock details activity should be loaded when the user clicks on any entry of the RecyclerView.

Stock Details

The stock detail activity should have 2 sections:

- Current Stock information
- Historical Data

Current stock information Historical Data is loaded by default. Furthermore, the stock details and stock historical data would have a list showing all the stock values. The list of the stock details would be implemented using a RecyclerView.

Submission

1. Push your project directory along with the source to remote bitbucket repository by the due date.
2. Invite and share your project repository the Grader (yan.chen01@sjsu.edu) and Instructor (ramin.moazeni@sjsu.edu).
3. Submit a Readme.pdf to Canvas including your name, repository access link, instructions to run your program (if any), snapshot of your running program
4. Your project directory will be graded according to the state your project directory was in at due time when fetched.

Appendix

API's description -- TiingoAPI calls

A comprehensive reference about this API is available at:

1. <https://api.tiingo.com/documentation/end-of-day>
2. <https://api.tiingo.com/documentation/iex>
3. <https://api.tiingo.com/documentation/utilities/search>

Company's Description

For Company's Description, use the following API.

<https://api.tiingo.com/tiingo/daily/<ticker>?token=<APIKeyTiingo>>

URL parameter in API Call:

- Ticker: Ticker symbol of the stock. E.g.: MSFT
- Token: The API access Token.

An example URL constructed from the parameters will look similar to this:

<https://api.tiingo.com/tiingo/daily/AAPL?token=12PrIvA32tEmYtEmpToKeN23>

Response Keys	Details
ticker	Ticker symbol.
name	Company's Name
description	Company's Description
startDate	Company's Start Date
exchangeCode	Company's Exchange Code

Response received for Company's Meta Data API call.

Company's Latest Price of the stock

For Company's Latest Price, use the following API.

<https://api.tiingo.com/iex/?tickers=<ticker>&token=<APIKeyTiingo>>

URL parameter in API Call:

- Ticker: Ticker symbol of the stock. E.g.: AAPL
- Token: The API access Token.

An example URL constructed from the parameters will look similar to:

<https://api.tiingo.com/iex?tickers=AAPL&token=12PrIvA32tEmYtEmpToKeN23>

Response Keys	Details
ticker	Ticker symbol.
timestamp	Timestamp of the data.
last	Company's latest price based on timestamp
prevClose	Company's previous closing based on timestamp
open	Company's open price based on timestamp
high	Company's high price based on timestamp
low	Company's low price based on timestamp
mid	Company's mid price based on timestamp
volume	Company's volume based on timestamp
bidSize	Company's bid size based on timestamp
bidPrice	Company's bid price based on timestamp
askSize	Company's ask size based on timestamp
askPrice	Company's ask price based on timestamp

Response received for Company's Latest Price of the Stock

Value of 'mid', 'bidPrice', 'bidSize', 'askPrice', 'askSize' will be null when market is closed. Value of mid can be null even when the market is open, if this happens then you should display '-' instead of null. Market Status must be open if the difference between currentTimestamp and 'timestamp' key is less than 60 seconds.

Company's Historical Data:

For Company's Historical data, use the following API.

<https://api.tiingo.com/tiingo/daily/<ticker>/prices?startDate=<startDate>&resampleFreq=<resampleFreq>&token=<APIKeyTiingo>>

URL parameter in API Call:

- Ticker: Ticker symbol of the stock. E.g.: AAPL
- startDate: the date from which we need data.
- resampleFreq: the interval between 2 data lists/ arrays.
- Token: The API access Token.

An example URL constructed from the parameters will look similar to this:

<https://api.tiingo.com/tiingo/daily/AAPL/prices?startDate=2019-09-10&resampleFreq=4min&token=12PrIvA32tEmYtEmpToKeN23>

Example Response:

Response Keys	Details
date	Date and time of the data.
open	Open Price at the specific date and time.
high	High Price at the specific date and time.
low	Low Price at the specific date and time.
close	Close Price at the specific date and time.
volume	Volume at the specific date and time.

Details regarding Company's Historical Data

Autocomplete:

For Autocomplete, use the following API.

<https://api.tiingo.com/tiingo/utilities/search?query=<query>&token=<APIKeyTiingo>>

URL parameter in API Call:

- Query: query means the combination of letters for which you want results for autocomplete. E.g.: AA
- Token: The API access Token.

An example URL constructed from the parameters will look similar like:

<https://api.tiingo.com/tiingo/utilities/search?query=AA&token=12PrIvA32tEmYtEmpToKeN23>

Example Response:

Response Keys	Details
Ticker	Ticker symbol.
Name	Company's Name

Response received for autocomplete API call.