CS 4720 - F18 - Final Project Proposal

Device Name: Nexus 7 Platform: Android

Name: Guangda Zhu Computing ID: gz6xw

Name: Siteng Zhang Computing ID: sz4dd

App Name: Stock Watcher

Project Description:

Our app will provide a platform for users to find stocks they like and save information about stocks. Users can register their emails to get updates on the stocks based on their preferences. They will be able to save stock they want to continue to observe, and all the stocks they have saved up should be similar to the bucket list created for our android mini-app with each stock being an item in the list.

What we propose to do is create an app that will do the following:

- The system shall allow users to create account with their email address;
- The system shall allow users to enter stock name to pull information and display it to the user.
- The system shall allow users to save preferred stock using SQLite
- The system shall allow users to config whether they would like to receive daily email about stock price change they saved.
- The system shall allow users to set up user configuration, such as time period between email or alert when a stock price drops too much.
- The system shall have a page displaying all the stocked saved by the user.

We plan to incorporate the following features:

20pts: data storage using SQLite, saving saved stock information.

10pts: consume presumed web services by pulling stock info from the internet.

10pts: Shared Preferences using key-value pair, such as user configuration.

5pts: email to alert user of big stock price change.

One of the two listed below:

15pts: firebase for user login information

15pts: saving stock price/change to a file and send it to user if they allow that in sharedpreferences.

Wireframe Explanation:

Our wireframe showed what we expected our app to do. The launch screen should be simple, containing what our app does and give a chance for the user to enter their email. Of course, the app does not have any login, therefore the user are free to not enter anything in the email section.

We want a central screen to take care of our favorite list of stocks, and connect between all other different functionalities. Therefore, we included the second screen as the central screen facilitating between all functionalities. In the main screen, user to connect to the settings page, which handles font size and email frequency changed. User can also type in the stock name of their choice, then click on detailed info to see the details of that stock. As the detailed info page show up, they can choose to go back if it did not suit their need, or click on “add to favorite” to add this stock to their list of favorite stocks. We also wanted the ability to generate a summary email to send to users if they wish to, which is why we included an email functionality in a separate screen.

PlatForm Justification:

One of the main reason we chose android is for the more precise sqlite control. We intended to use sqlite to store all the saved stocks from the user, therefore, a precise control is needed to avoid unintended consequences. IOS have an abstraction layer called CoreData which does not allow very precise controls, and we were worried that we will not be able to handle data as well using it.

Secondly, only one person in the group has an ios device, and MacinCloud has proven to be very slow and somewhat costly to use. This is also a factor in our group choosing to do Android instead.

Lastly, we think that with our type of free app that doesn’t require the newest version of android or high-power processor, it makes more sense to open it to more potential users, and android have much more user compared to Ios.

Major Feature/Screen:

1. The most important feature we have is the api call, by using an outside api with the help of retrofit, we are able to pull of live stock prices and changes when user enters the stock name.
2. We store some user information in sharedpreferences. User can enter their email address, which will be saved in sharedpreferences and used when user want a summary of the list of favorite stocks, as it will put the email entered as the receiver of such summary.
3. We have a list of favorite stocks, saved using sqlite data storage. User can freely add and remove stocks from this list, and all the changes will be stored in the local database.
4. We have a settings page which allows user to adjust font size of the app using sharedpreferences. The main adjustment is for the list of favorite stocks and the detailed info page. This allows user to adjust the app according to their need and have accessibility benefits.

Optional Features:

1. Data Storage using SQLite: 20pts.
    a. Click on continue in the first screen
    b. In the main screen, type FB in to the stock name input.
    c. Click on detailed info
    d. Click on “add to favorite”
    e. The main list should have the FB stock added.
    f. Quit and reopen the app.
    g. The main list should still have the newly added stock.
2. Data Storage using File read write & Email: 15+5 = 20pts.
    a. Enter your email address in the first screen, and click “save email”.
    b. Click continue to the main screen.
    c. Click on “email summary” on the bottom.
    d. If needed, Google may ask to setup an account.
    e. Then a email draft screen should pop up, with receiver being the email entered. (EMAIL)
    f. It should have an attachment, which is written about the summary of the list of favorite stocks. (FILE WRITE)
3. Data Storage using SharedPreference: 10pts:
    a. The above section also detailed a usage of sharedprefernece, automatically bringing the saved email address as the receiver of the email.
    b. Also, go to the main page as normal
    c. Click on setting button in the top right
    d. Click on “general”
    e. Click on “font size”
    f. Choose a font size of small or Huge for clarity.

    g. Go back to the main page, the list should have a different font size. This is achieved using sharedpreference.
4. API call done with retrofit: 10pts:
    a. Go to the main page as normal
    b. In the only input field, type in “AAPL”
    c. Click on detailed info button
    d. New page has information about the apple stock pulled from an api online.
5. This four add up to 60 pts.

Testing Methodology:

We tested our app in the emulator and using nexus 7 to make sure that everything works right. We tried typing in wrong things on purpose to make sure bad input doesn’t break the app. We also tried deleting all stock from the list, to test if it can handle having nothing in the sqlite database. We have also switched orientation of the emulator/tablet itself to make sure the app still works in landscape.

Usage:

We intended this app to be used by mostly casual users that does not care about fancy graphs or graphics. We wish to provide a simple experience for the user, making them only a few click away from looking at stock information for all the stock they want.

It is really easy to use and doesn’t require any login information. However, if the email field in the first page is not saved in the sharedpreference, the email summary functionality won’t work as well, as the recipient of the email field is auto-filled with the email information from sharedpreference.

Lesson Learned:

The biggest lesson I learned is probably that it is really hard to have a good looking UI. In ios, doing storyboard is simply drag and drop most of the time. Although android layout have similar functionalities, it is a lot more strict on position and constraints. Therefore, making a good looking UI is rather difficult.

Also, I learned how useful sharedpreference is. I really enjoyed using it due to how easy, quick and responsive it is to use. Doing font size using it is a breeze and everything just clicked.

I expected data storage to be a lot more difficult than it actually is. Using part of the code from Core Skills app, I only have to tweak a few things for storage to work. It is actually rather simple and also works really well.

Overall, I think the hardest part of the development process is probably to make a good UI. Also, Google please make it easier to hide keyboard, I have no idea why I can’t do something like keyboard.hide() to make keyboard auto-hide after clicking out of focus.