GitHub Username: xionger

CryptoCoin

Description

Investing cryptocurrencies could make you a millionaire within one year or even several months. You can get both real time and historical market changes of cryptocurrencies from this app. The app also picks news worldwide of selected cryptocurrencies for your references. It is easy to use and customize.

Intended User

CryptoCoin is intended for users who are interested in cryptocurrency market and/or investment.

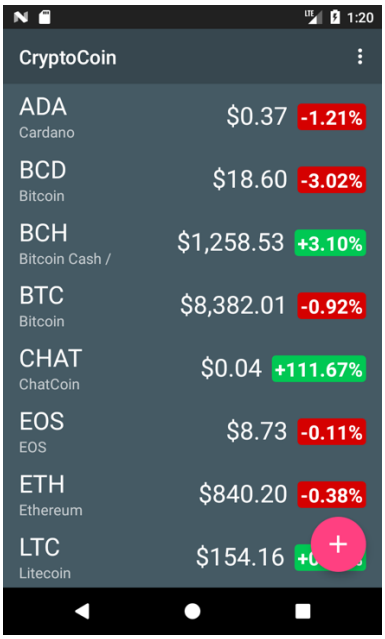
Features

- Monitor current cryptocurrency market
- Check the historical data of selected cryptocurrencies
- Retrieve the news of interested cryptocurrencies
- Add targeted cryptocurrencies into the list
- Customize settlement currency and intervals of historical data

User Interface Mocks

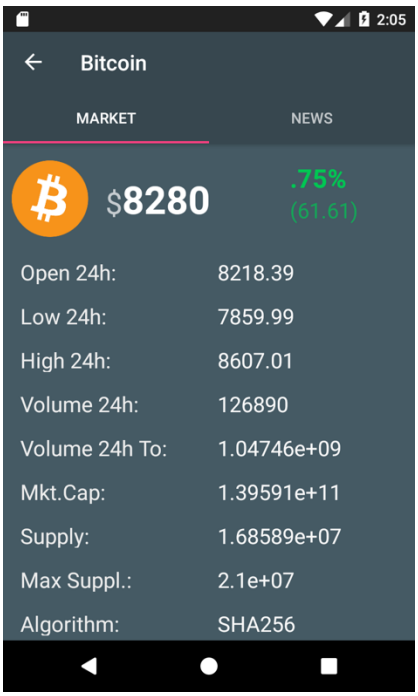
These can be created by hand (take a photo of your drawings and insert them in this flow), or using a program like Google Drawings, www.ninjamock.com, Paper by 53, Photoshop or Balsamiq.

Screen 1

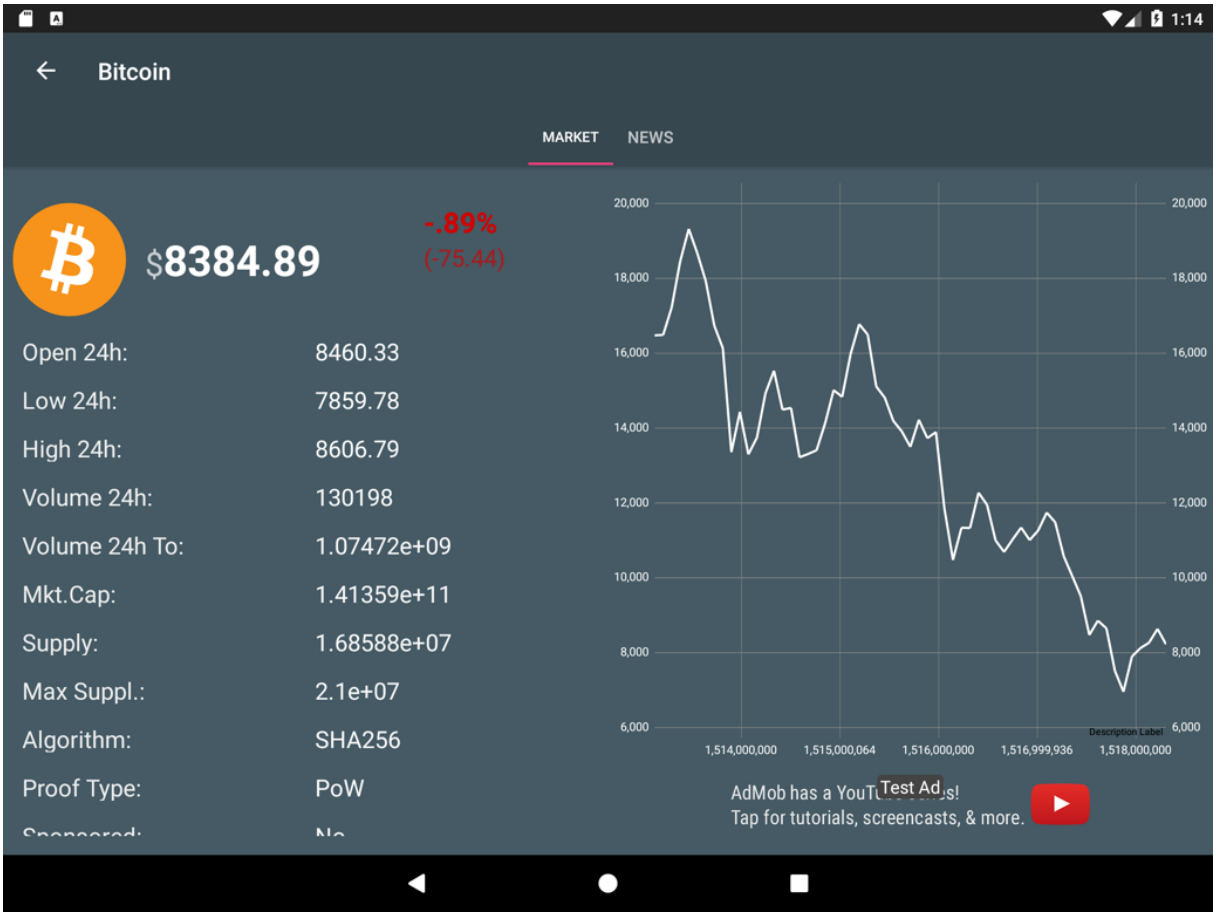


Screen 2

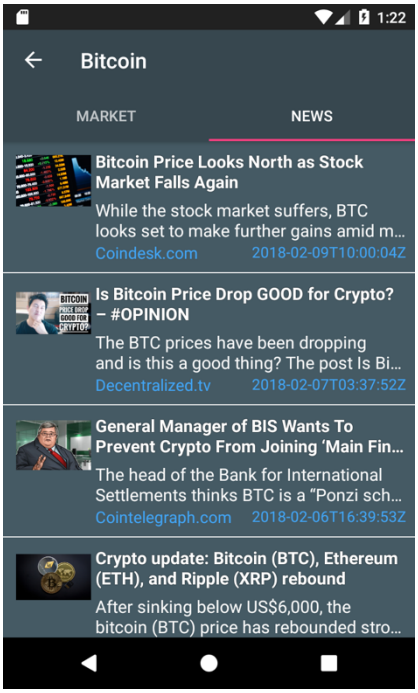
Screen 3



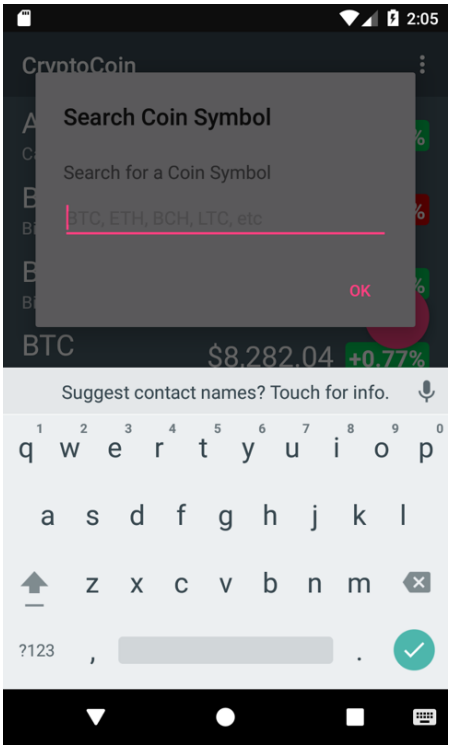
Screen 4



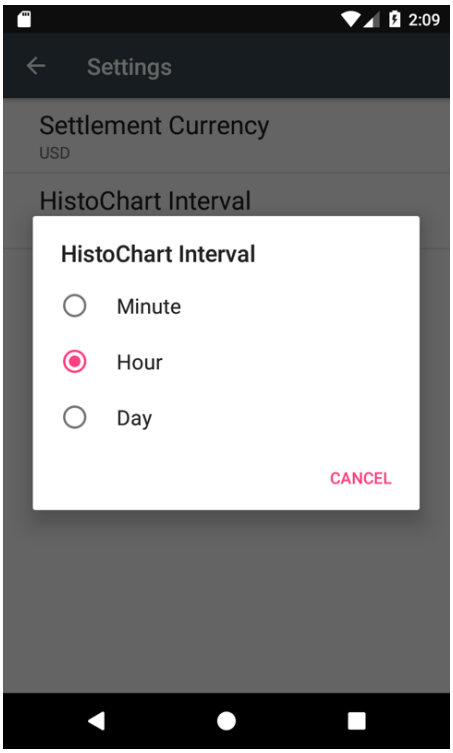
Screen 5



Screen 6



Screen 7



Key Considerations

How will your app handle data persistence?

The data will be fetched from both local (a Json file at assets) and online (CryptoCompare API and News API) and stored in SQLite database. The App will handle data persistence via content provider.

Describe any edge or corner cases in the UX.

- Invalid or missing data: set as “0” or “N/A” in accordance with specific conditions
- Missing logo image: using default app logo
- Nonsense data: If a user sets “BTC” as settlement currency, the price of Bitcoin will be constant (1.0) and the historical price chart will be a horizontal line. The app hides the chart in this case.

Describe any libraries you’ll be using and share your reasoning for including them.

- Glide: handle the loading and caching of images;
- Butterknife: bind views;
- Timber: log messages and help to debug;
- Okhttp3: make network calls and help to fetch online data;
- MPAndroidChart: visualize historical data;
- Material-dialogs: help to generate nice dialog UI.

Describe how you will implement Google Play Services or other external services.

- GcmTaskService (play-services-gcm): handle data sync tasks upon user request; it will also schedule data update automatically within a specific period.
- Mobile Ads (play-services-ads): show ads inside the app.

Next Steps: Required Tasks

This is the section where you can take the main features of your app (declared above) and break them down into tangible technical tasks that you can complete one at a time until you have a finished app.

Task 1: Project Setup

- Configure libraries
- Sign up and configure APIs
- Design database and content provider
- Design models

Task 2: Implement UI for Each Activity and Fragment

- Build UI for CoinsActivity
- Build UI for CoinDetailFragment
- Build UI for NewsFragment
- Implement PagerAdapter to integrate CoinDetailFragment and NewsFragment

Task 3: Implement Sync Data Tasks

- Implement IntentService to fetch basic information of cryptocurrencies
- Implement IntenService to fetch detail information of selected cryptocurrency
- Schedule data sync automatically
- Limit the frequency of API calls

Task 4: Implement "Add" Functions

- Build UI for adding user selected cryptocurrency
- Fetch data for the added cryptocurrency

Task 5: Implement Settings

- Design and build UI for settings
- Implement shared preferences

Task 6: Implement Widget

- Implement widget
- Build UI for widget
- Implement automatically data sync for widget

Task 7: Polish UI

- Modify layout for both phone and tablet
- Visualize historical data of cryptocurrency
- Design and implement the theme
- Polish layout, color, font, etc
- Using material design features if needed

Task 8: Testing

- Write basic testing cases using Junit and Expresso
- Fix bugs if needed

Submission Instructions

- After you've completed all the sections, download this document as a PDF [File → Download as PDF]
    - Make sure the PDF is named "**Capstone_Stage1.pdf**"
- Submit the PDF as a zip or in a GitHub project repo using the project submission portal

If using GitHub:

- Create a new GitHub repo for the capstone. Name it "**Capstone Project**"
- Add this document to your repo. Make sure it's named "**Capstone_Stage1.pdf**"