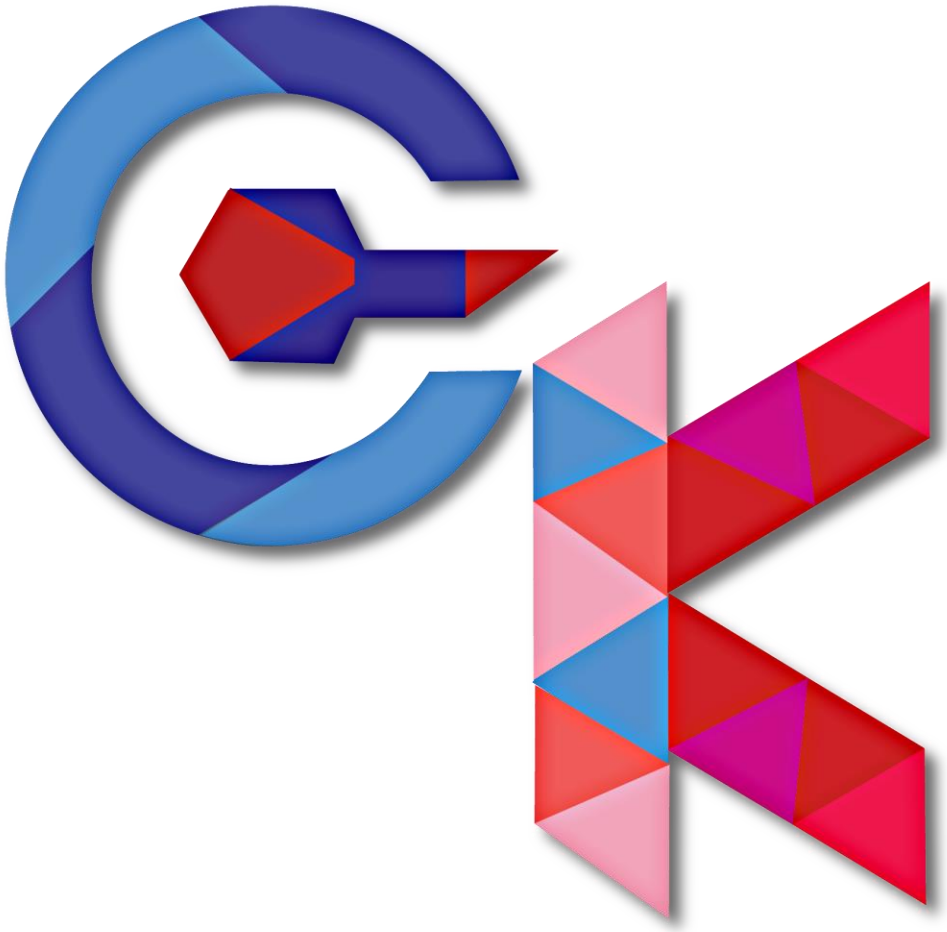


CryptoKit

CSI 3130 and CSI 3308 Final Project

By Sean Blonien

Dr. Tomas Cerny

April 26, 2018

Vision Statement

CryptoKit will be a free, easy to use, user-friendly Android application that shows up-to-date crypto asset (crypto currency) information, such as the current price of Bitcoin. This experience will be as intuitive and simplistic as possible such that it shows current prices and doesn't overwhelm the user with unnecessary, superfluous information present in many other apps of this genre.

Project Outline

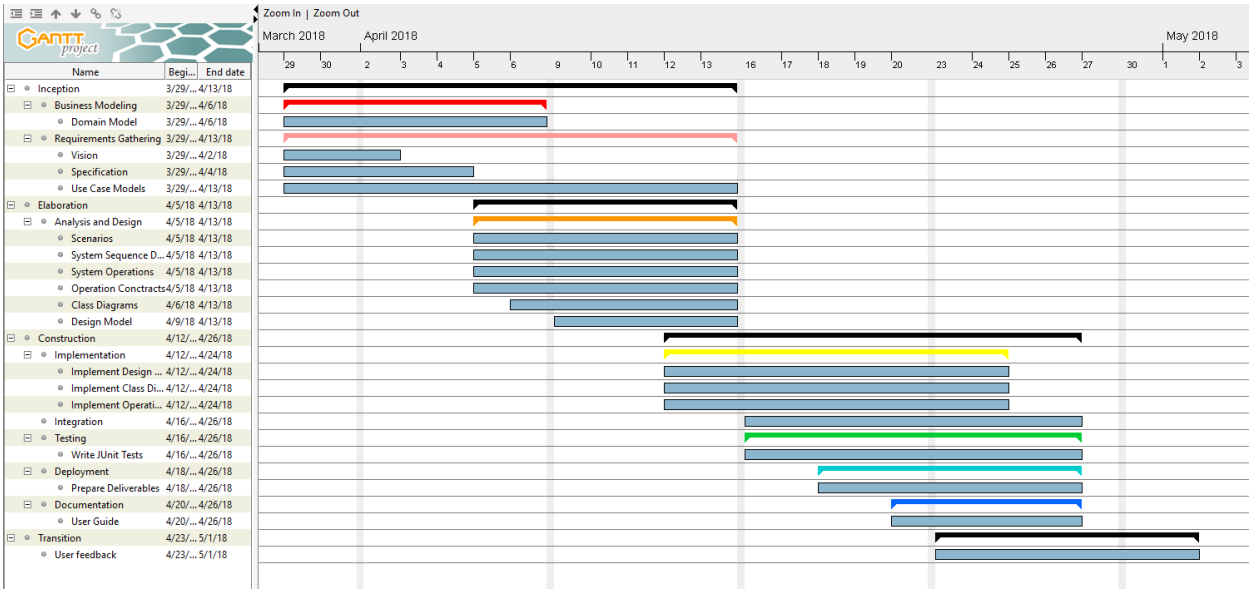
The scope of this project will be to design and develop an Android application in Java that has the following general features: a fully-functional registration, login, and account management system; and the ability to easily view crypto assets; the ability to learn more and interact with the crypto assets.

Responsibilities

Since I am in a solo group, my responsibilities include everything involved when making the Java based Android application, which includes project manager, designer, analyst, developer, and tester among other miscellaneous roles and responsibilities required for this project.

Gantt Diagram

Here is a snapshot of the Gantt diagram used for scheduling tasks for this project. See the "CryptoKit Gantt Schedule.pdf" to see the diagram and schedule in full.



Deliverables

- Link to GitHub Repository
- Android APK file that is runnable on an Android device or emulator (not provided)
- Gradle-based Java source files
- Documentation (e.g. everything in this document especially UML diagrams)
- Explanation on implementation such as use of GRASPS, design patterns, and testing

Functional Requirements

- The user can access the app through the Google Play Store.
- The user will be able to register with the CryptoKit system.
- The user will need to confirm their registration through their email.
- The user needs to be able to recover their password if lost or forgotten.
- The user can logout.
- Users can see up-to-date crypto asset prices.
- The user can see a crypto asset's name and logo to be able to easily identify the asset at a glance.
- The user can select a specific asset to show more information.
- This supplementary information should show a crypto asset's 24 hour volume, its market cap, its available token supply, and its maximum token supply.
- This information should contain as brief description of the asset as well that describes the technology, history, and purpose of the specific crypto asset.
- The user can filter through the crypto according to their filter criteria.
- The user can search the crypto assets for a specific crypto asset by name.
- The user can favorite crypto assets that appear at the top of the app.

Non-Functional Requirements

- Perform well on all compatible Android devices, with quick response time.
- Portable such that anyone with an Android phone will be able to use the service.
- Be secure with the user's data as to not expose any personal information.
- As available and reliable as possible so that it can be accessed by all users whenever they need it.
- Functionally intuitive and easy to use.
- Maintainable with good documentation and organization.
- Easily extendable so that other features can be implemented.

Business Rules

- The user needs to register to login.
- To login, the user must enter in a confirmed email account with password.

- The user must have internet connection before launching the app.

Actor Roles

- Unregistered user
- Registered user
- Firebase account management service
- Crypto asset price service
- Crypto asset description service

Use Cases

Use Case: Register for CryptoKit
Primary Actor: User
Scope: Registration subsystem
Level: User goal
Brief: The user can register (i.e. make an account) in order to login to the app.
Stakeholders: User
Precondition: the user has access to a valid email address
Postcondition: the user will have an account registered with CryptoKit
Triggers: the user presses the "Not a registered? Register here." option on the login screen to be brought to the Registration page that has a "Register" button to be pressed once the user enters in valid account information.
Main success scenario:

1. User wants to register to CryptoKit
2. User opens the CryptoKit app and is shown starting options
3. The user navigates to the "Not a registered? Register here." option below the "Forgot your password?" option and clicks it.
4. The user is brought to the registration page.
5. The user enters in a valid email address and password to register with.
6. The user confirms their email address.
7. The user now has an account registered with CryptoKit.

Alternative Paths:

2.1 The user launches the app and nothing displays.
    2.1.1 The user will have to relaunch the app if something went wrong while loading.
    2.1.2 The user will see some sort of notification about why the app cannot load correctly.
    2.1.3 The user can contact the support team by email at contact@seanblonien.com
3.1 The user might first try to put his information in the login page.
    3.1.1 The user will be notified that he cannot login because his account does not exist.
    3.1.2 The user will be notified how to register.
3.2 The user accidently clicks another options/button.
    3.2.1 A notification will appear displaying what the invalid operation was, such as an invalid login as no email or password was provided.

4.1 The registration page is confused for the login page.
4.1.1 If the user already has an account, a message will display to tell the user to login normally instead of registering under an account that already exists.
5.1 The user enters in an invalid email.
5.1.1 User will be prompted to enter a valid email with a valid domain, such that the email is in the general format of <name>@<domain>.<top level domain>
5.2 The user enters an invalid password.
5.2.1 The user will be shown the password requirements (minimum number of characters, required special characters, invalid characters, etc.).
6.1 The user is not sent a confirmation email.
6.1.1 The user can try to register again under a different email.
6.1.2 The user can still try to login even if his account was not confirmed.
6.1.3 The user can contact support at contact@seanblonien.com

Use Case: Login to CryptoKit
Primary Actor: User
Scope: Login subsystem
Level: User goal
Brief: The user logs into the app.
Stakeholders: User
Precondition: the user has an account already registered with CryptoKit
Postcondition: the user is logged in and brought to the main activity in CryptoKit
Triggers: the user presses the "Login." option on the login screen
Main success scenario:
1. User wants to login to Crypto Kit
2. User launches the app and is shown the login screen.
3. The user enters his email and password.
4. The user presses login and is shown the main activity.

Alternative Paths:
2.1 The user launches the app and nothing displays.
2.1.1 The user will have to relaunch the app if something went wrong while loading.
2.1.2 The user will see some sort of notification about why the app cannot load correctly.
2.1.3 The user can contact the support team by email at contact@seanblonien.com
3.1 The user enters in an invalid email.
3.1.1 User will be prompted that the email entered was not registered with CryptoKit.
3.1.2 The user will be told to register an account before logging in.
3.2 The user enters an invalid password.
3.2.1 The user will be shown the password requirements (minimum number of characters, required special characters, invalid characters, etc.).
3.2.2 The user will be shown that the wrong password was entered for that registered account

3.2.3 The user will be prompted to reset their password if they forgot it or are unable to login.

3.2.4 An email will be sent to that users email account with instructions on how to reset their password.

4.1 The login is unsuccessful even though the proper info was entered.

4.1.1 The user will have to ensure he is connected to the internet before logging in.

4.1.2 The app must be restarted in order to try logging in again.

4.2 The user logs in but does not see the main activity displayed.

4.1.1 The user will have to relaunch the app if something went wrong.

4.1.2 The user will have to reload that page because it is likely the API service was down.

Use Case: View Details of a Crypto Asset
Primary Actor: User
Scope: Price Checking subsystem
Level: User goal
Brief: The user sees more information about a specific crypto asset.
Stakeholders: User
Precondition: the user has an account and is logged in to the app and is on the price checker activity page
Postcondition: the user can view more financial and descriptive information about the asset
Triggers: the user selects or clicks on a specific asset in the price checker view
Main success scenario:

1. User wants to see more information on a crypto asset.
2. The user selects, taps, or clicks on an asset that is displayed.
3. The information is loaded and displayed to the user expanding the current view of that asset.

Alternative Paths:

2.1 The app crashes when attempting to view more information.

2.1.1 The user must reload the app and return to the main activity to try again.

2.1.2 If the issue persists, the user can submit the bug to contact@seanblonien.com

3.1 The user taps on an asset, but no information is displayed.

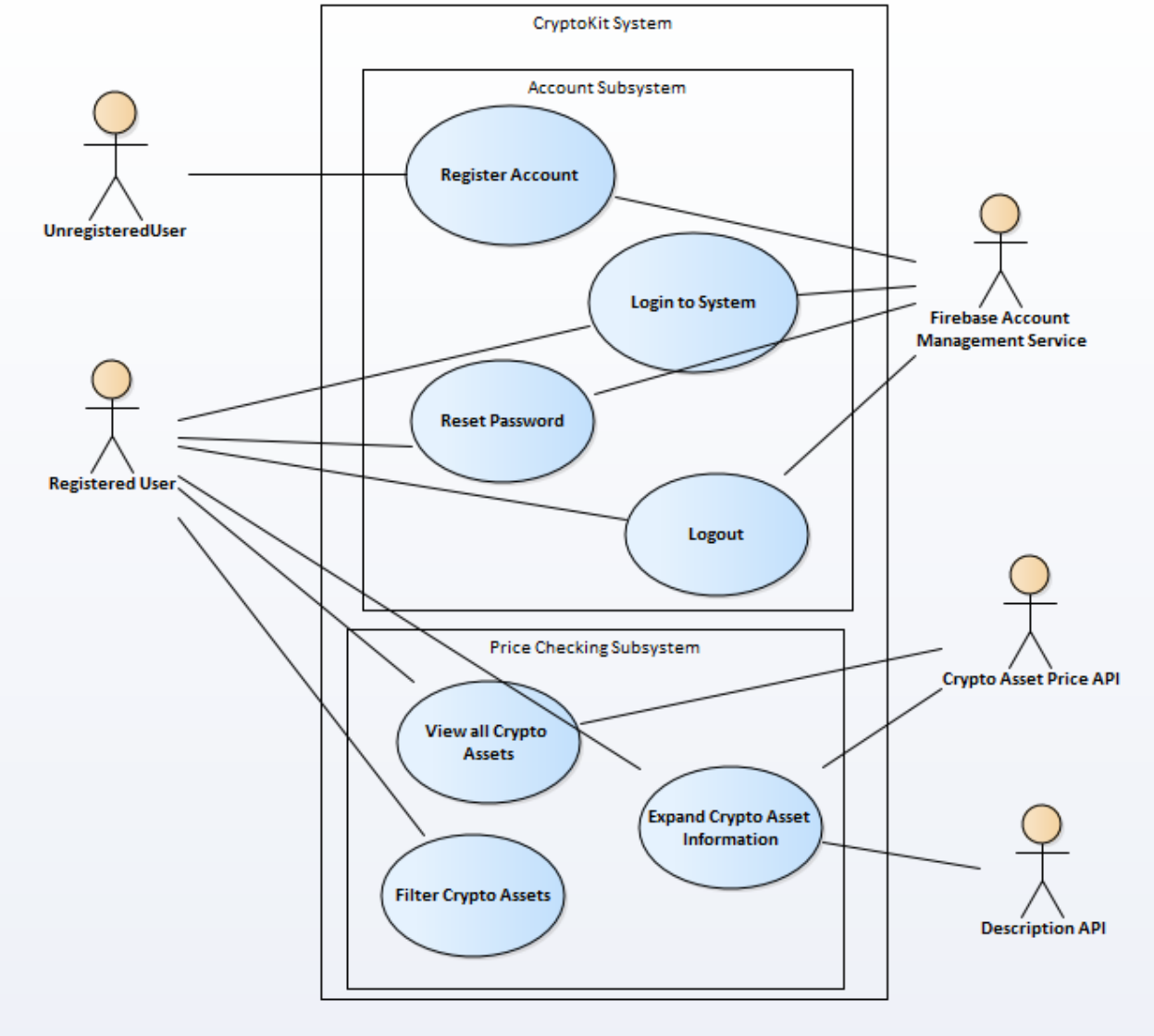
3.1.1 It is likely the API or connection went wrong, so the user must try tapping the asset again to refresh the data.

3.2 Some, but not all, of the information is displayed.

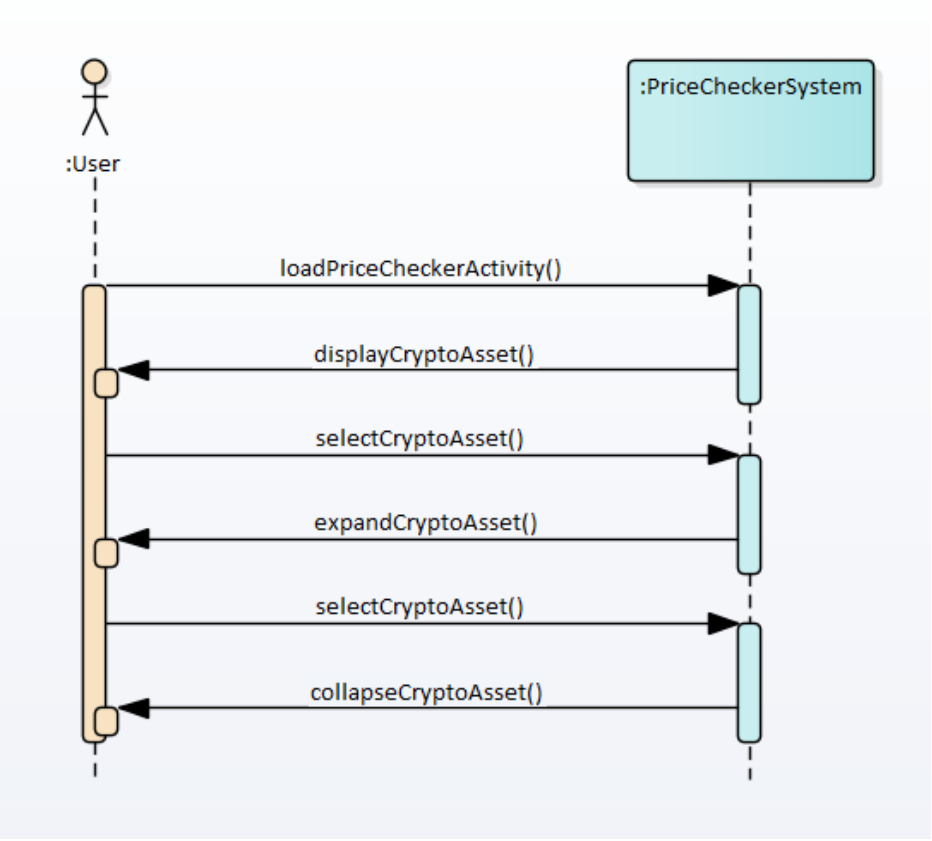
3.2.1 If a description is not provided, this means the Wikipedia API interface was unable to find an article about that particular asset.

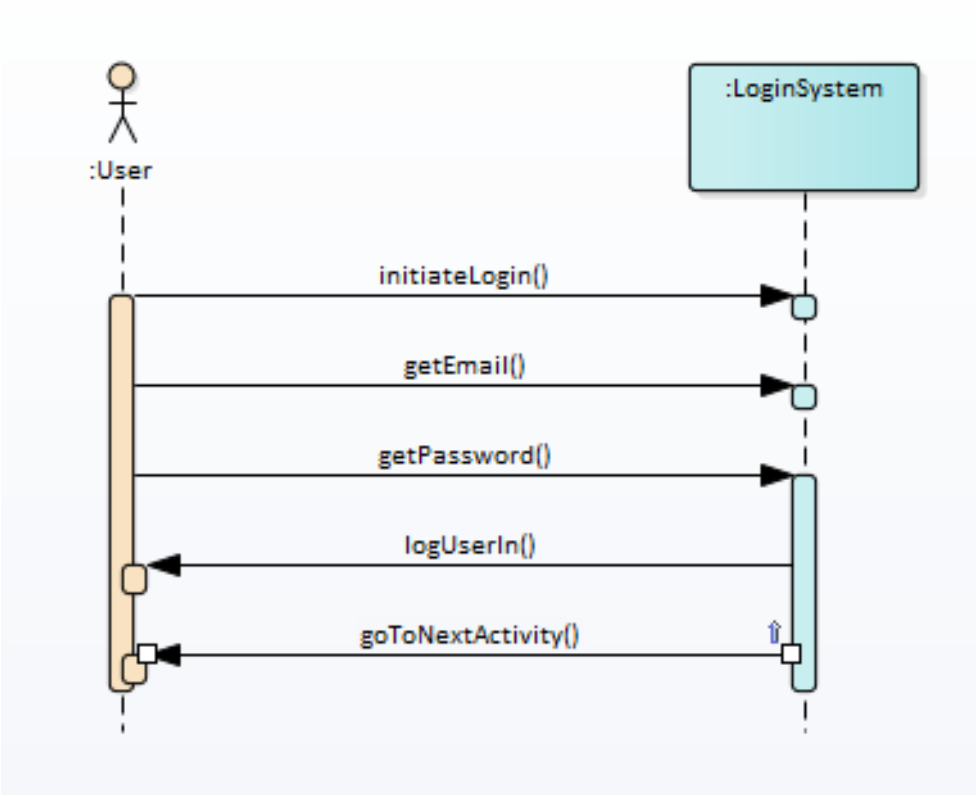
3.2.2 The user will be shown a default description indicating there is no description.

Use Case Diagram

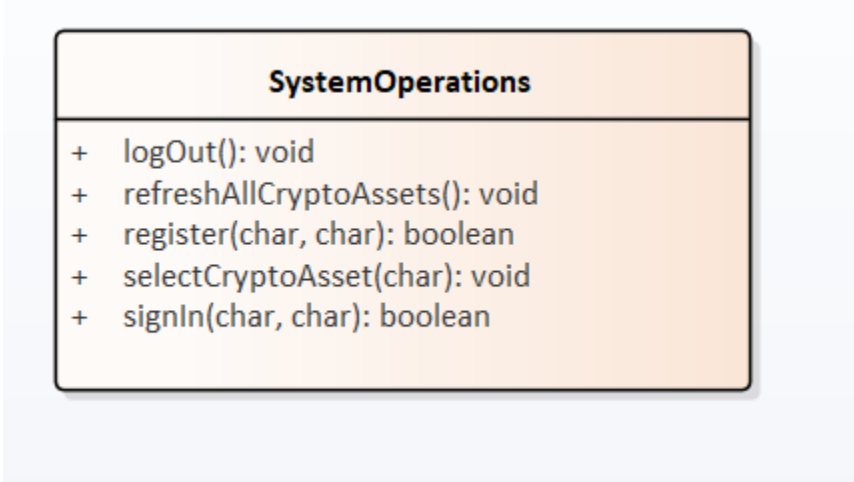


System Sequence Diagrams

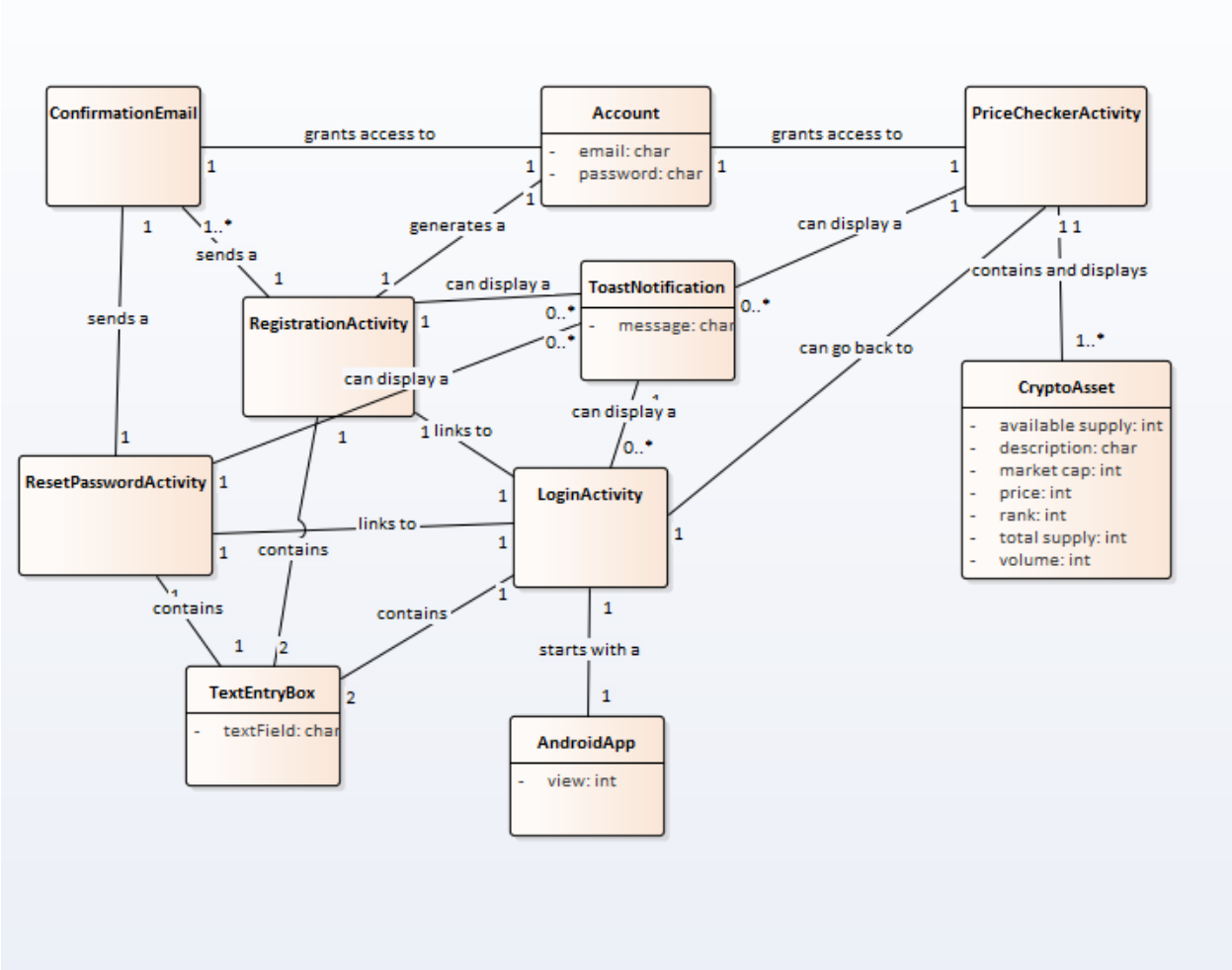




System Operations



Domain Model



Operation Contracts

Operation: signIn(email, password)
Cross Reference: Use Case: Login to CryptoKit
Pre-conditions: the user has already registered for an account
Post-conditions:

- the user will log in with the system
- the display will then switch to the PriceChecker activity

Operation: register(email, password)
Cross Reference: Use Case: Register for CryptoKit
Pre-conditions: the user has a valid email account
Post-conditions:

- the user will be registered with the CryptoKit system
- the user now has an account that can be used to logIn() to view the crypto assets

Operation: logout()
Pre-conditions: the user is already logged into an existing account
Post-conditions:

- the user will log out of the system and his account
- the app will switch back to the log in screen, requiring a logIn() again

Operation: selectCryptoAsset(CryptoAsset)
Pre-conditions: the user is on the PriceChecker activity and the assets have loaded
Post-conditions:

- the user will see more information about the selected crypto asset
- the asset card will expand and show a brief description of the asset
- the view is modified due to the asset information expanding

Operation: refreshAllCryptoAssets()
Pre-conditions: the user is on the PriceChecker activity and the assets have loaded, and the user swipes down to invoke this operation
Post-conditions:

- the user will see a loading icon to indicate the new data is being loaded
- the user will see the prices and information change accordingly

Justification of GRASPS

Creator: ViewHolder creates the crypto assets in the recycle view adapter class in order to free up memory in the recycle view.
Information Expert: the activity classes play the role of information expert because they create and contain all the objects and UI within themselves, and some of them even have their own private classes in order to control threaded functionality.
Low Coupling: the crypto asset class only interfaces with the AssetViewHolder, and that class only interacts with the recycle view holder which means these classes have very low coupling.
Polymorphism: all the main activities, onClickListeners, adapters, Views, buttons, and so on, all use polymorphism and the nature of overriding functions to implement polymorphic behavior. There are many different types of views for example, but the app uses the polymorphic behavior to display views differently.

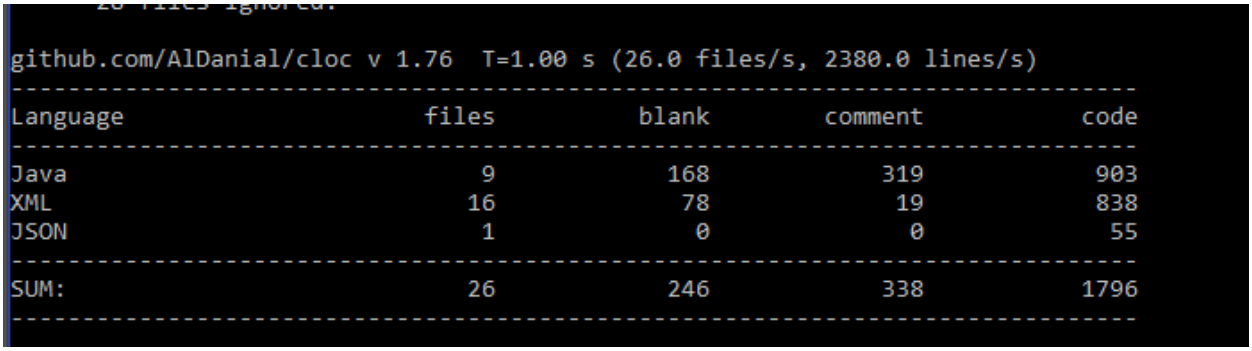
Design Patterns

Adapter: the RVAdapter interfaces the crypto asset class that contains all the asset data with the recycle viewer that displays the information. This adaptation allows the data to be displayed by only overriding an implementing just a few functions.
Iterator: the iterator is used when querying and loading information on the crypto assets when the image URL is added to each of the objects .
Composite: the main activities such as PriceChecker compose all the Views and objects related to graphical display which allows for easier control of the display and view.
Command: updateThisAsset and GetJSON are perfect examples of the command design pattern because they both encapsulate a request into an objection which allows for multithreading of this asynchronous task. These classes are invoked without the parent class knowing what it does, because both use the execute() method to start execution.
Proxy: the RVAdapter acts as the meadiator between the display and the actual crypto assets themselves. This class controls the behavior of these seemingly unrelated objects that enables changes in object interaction independent of the objects themselves and without them knowing about eachother.

Hours Spent on Project

Approximately 60 hours.
All code and XML files was made by and owned by me.

Logical Lines of Code



```
github.com/AlDanial/cloc v 1.76  T=1.00 s (26.0 files/s, 2380.0 lines/s)
-------------------------------------------------------------------------------
Language                     files          blank        comment           code
-------------------------------------------------------------------------------
Java                             9            168            319            903
XML                             16             78             19            838
JSON                             1              0              0             55
-------------------------------------------------------------------------------
SUM:                            26            246            338           1796
-------------------------------------------------------------------------------
```

Gradle was used to add dependencies and libraries instead of Maven. Since they are functionally the exact same software, it should not be an issue because anything you can do Maven you can do in Gradle.

Link to Repository with Git tracking

https://github.com/sblonien/CryptoKit