Technical University of Denmark

62416 Advanced Mobile Application Development Fall (21)

CryptoApp

Gruppe 13
2th December 2021

s176492
Mohamad
Ashmar
Github: m0-ar

s176485
Mohamad Javad
Zamani
Github: JVDZMN

s141479
Anthony
Haidari
Github: Anthai87

Click: Github Repository

Contents

Introduction **2**

1 User guide **2**

Sprint 1 **4**

2 Analysis **4**

2.1 Stakeholder descriptions (Requirements) 4

Stakeholder descriptions (Requirements) 4

2.2 Epics and User Stories 5

Epics 5

2.2.1 Epic **Investment on Crypto** 5

2.2.2 Epic **Overview** 5

2.2.3 Epic Transactions 5

Product overview 6

Sprint 2 - Beta **7**

3 Design **7**

3.1 Use Cases 7

3.2 Class Diagram 7

3.3 Sequence Diagram 8

4 Prepare Sprint Backlog **9**

Sprint 3 - Beta **11**

5 Implementation **11**

5.1 Architecture 11

y11

5.2 Hilt-Dagger 11

y11

5.3 DataBinding 12

y612**Test** **13**

6.1 Junit Test 13

Junit Test 13

Sprint 3 - Release 1.0 **14**

7 Conclusion **14**

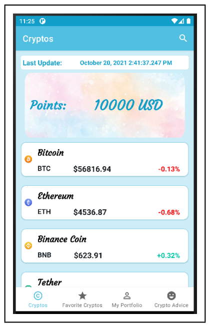
Introduction

In this document we will summarize the process of making Cryptocurrency app, so the following sections will show an overview of the product, its stakholders, its key features, and the requirements underpining the application.

1 User guide

The Crypto Currency Application is an Android App that can be used as a tracker of crypto exchange rates. But the main purpose with this app is to help users making investments in the cryptocurrency market with fictional money. In the following section a descriptive guidance is given to the users that will utilize the application.

The first screen in the app, after the first launch is pictured below. The user will start with 10,000 fictional rewarded money to buy and sell cryptocurrency. As it is showed the users points is in the header and below that you will find a scrollable list of different cryptocurrencies with their respective icon, name, symbol, recent value in dollars as well as negative and positive percentage change in the last 24 hours.



Last Update: October 20, 2021 2:41:37.247 PM

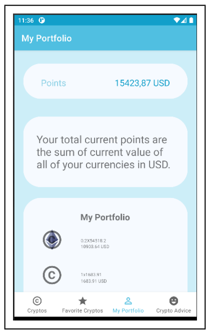
Points: 10000 USD

Bitcoin			
BTC	$56816.94	-0.13%
Ethereum			
ETH	$4536.87	-0.68%
Binance Coin			
BNB	$623.91	+0.32%
Tether			

Figure 2: Screen I

The purpose with the screen 2, as it is shown below, is to list the users portfolio. The portfolio is also a scrollable list of owned currencies withe their icon, volume, recent rates and total value in dollars. The screen also shows the users collected points in the header and it informs the math behind the calculated points. The button has click action, hence once it is tapped on the app will navigate the user to 'Transactions' screen.

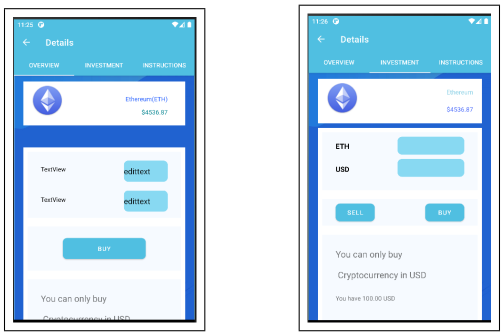
The 'Transactions' screen will then show a list of the user's sold and bought currencies which were stored in a database table by clicking the buy and sell



Your total current points are the sum of current value of all of your currencies in USD.

Figure 3: Screen II

buttons. This leads us to screen 4 and screen 5, which are imaged below:



You can only buy

Cryptocurrency in USD

You can only buy

Cryptocurrency in USD

You have 100.00 USD

Figure 4: From left: Screen IV, Screen V

As it is seen screen 4 shows a header containing the icon, name, symbol, recent rate of the cryptocurrency whose row was tapped on, on screen 1. Furthermore, it is shown how much the user owns valued in dollars. From this screen the user is able to buy or sell this currency. Note! the cryptocurrencies can only be sold if the user owns this. There is also shown a graph on the screen indicating fluctuations in the value of cryptocurrency over an interval. From Screen 5 the user is only able to buy the wanted cryptos.

Sprint 1

2 Analysis

2.1 Stakeholder descriptions (Requirements)

Table 1: Stakeholder summary

Name	*Description*	*Product responsibilities*	*Project responsibilities*
User(s)	*Users interested in learn how to invest in cryptocurrencies*	*Fetching the latest rates of cryptocurrencies and provide to the user. Share relevant informations every time the user opens the application.*	*If some users provide inputs through surveys, and feedback on release of the application.*
Project team	*Student Developers working on the project.*	*Make the product design flexible and adaptable for future needs. Make the product design scalable to serve more users in future. Make the product user interface intuitive for quick product adoption*	*Deliver the software as required and communicated by Product owner. Keep the product owner and others (stakeholders) informed on project progress, as needed.* .
Product owner(s)	*Shorcut.io and Professor Ian*	*Provide vision, goals and identify requirements for the project. Help the developers prioritize product features to align with goals.*	*Participate in important meetings and provide feedback and guidance.*

2.2 Epics and User Stories

2.2.1 Epic Investment on Crypto

- As a user I want to learn how to invest in cryptocurrency market with fake money.
- As a user I want to start with USD10,000 fake money for investing in cryptocurrency market.
- As a user I am able to track the latest rates of all cryptocurrencies.
- As a user I want to buy cryptocurrencies which I find profitable with my fake money.
- As a user I want to sell cryptocurrencies, which I think are worthless or devalued in future.
- As a user I want to see a list of all cryptocurrencies you can invest in.
- As a user I want to see the prices of all the listed cryptocurrencies.
- As a user I want to see how much the value of a given crypto has changed in the last 24 hours.
- As a user I am able to look at my points based on the net worth of my portfolio in dollars, at any given time.

2.2.2 Epic Overview

- As a user I want a list of cryptos and what has changed in their value in the last 24 hours.
- As a user I want to see how much money I have in my account.
- As a user I am able to search for a given cryptocurrency.

2.2.3 Epic Transactions

- As a user I want to see a list of my transactions, hence sold and bought cryptocyrrencies in the past.

Product overview

Need	Feature according to User Stories	Priority
Investment and Selling	Buy a certain amount of cryptos	5
	Sell a certain amount of cryptos	5
Overview	View a list of cryptos	4
	View networh of my portfolio	4
	Search for a given crypto	4
Transactions	View a list of transactions	3

Table 2: Features estimation

Sprint 2 - Beta

3 Design

3.1 Use Cases

In the following section we describe use cases in order to record the requirements. As it is showned in the picture below, there is one user/actor. This is the actor with our application at hand who has the desire to learn investing within the cyptocurrency market.

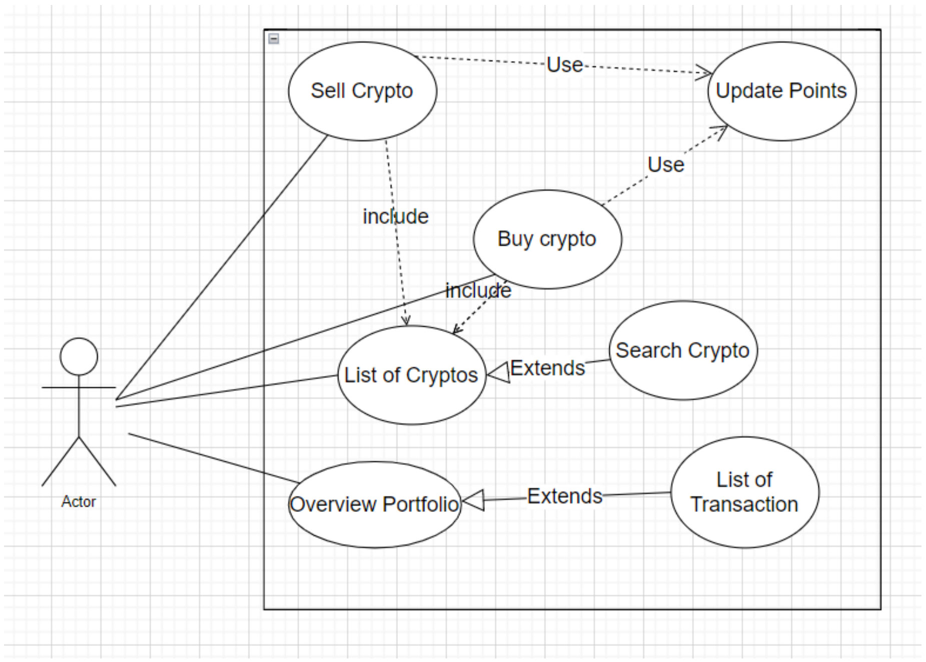


Figure 5: Use Case Diagram

3.2 Class Diagram

There are 6 Different classes in the app, the main class is crypto class, which made by converting json respone from coinbase API to kotlin Data class with json2kotlin extension. In the app both Invested crypto class and Assest class have a list of Crypto. Since it is imposibble to store complex data in room database, we use type converter to convert list of cryptos to string and vice versa. portfolio class store a list of invested cryptos and userpoints and Transactions class contains the history of user transactions in form of a list of transaction class.

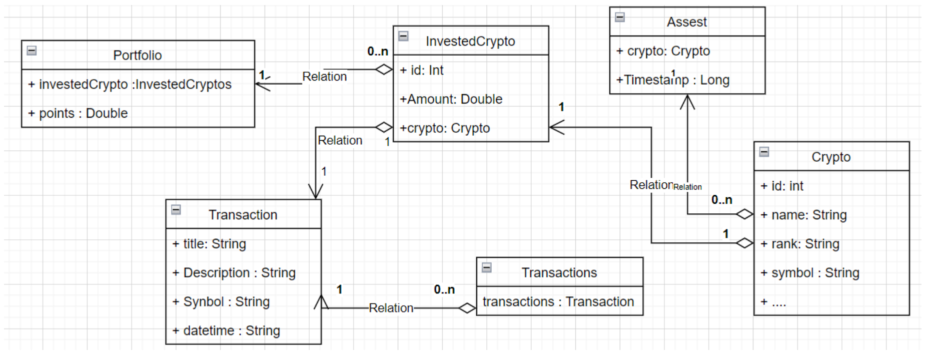


Figure 6: classDiagram

3.3 Sequence Diagram

Below is showned a image of UML Sequence Diagram to illustrate how the different parts of the system within our application interact with each other to carry out functionalities, and the order in which the interactions occur when a particular use case is executed.

Brief description of how the cryptos are displayed on first fragment is with the sequence diagram.
First fragment, which name is cryptosFragmant, request the all data of MainViewModel. MainViewModel requests all data from repository and repository which have one instance of both local and remote data sources. Here we send request first to Local database due to not wasting time, if there are some data in local database, so local database send data as response to repository's request, but if there are no data in local, so repository send a request via retrofit library to coincap.io API. These steps are reversed, until data is displayed on screen.

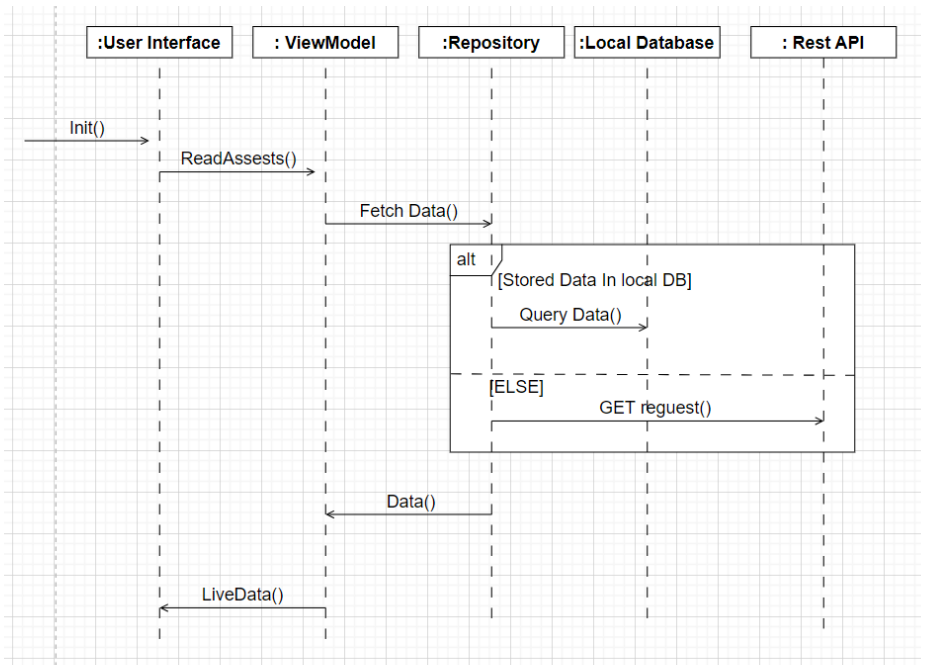


Figure 7: Sequence Diagram

4 Prepare Sprint Backlog

User Story - Create a service

As a investor I want to invest in a chosen cryptocurrency.

Use case - Invest in a cryptocurrency

Primary actor: User
Secondary actor: None
Description: A user can invest in a chosen crypto.
Precondition: The user should has to be rewarded with 10,000 fake USD to buy cryptocurrency.
Basic flow

1. User taps on a chosen cryptocurrency from the scrollable list of all cryptocurrencies.
2. The application navigates the user to a screen where he can buy the chosen crypto.
3. User enters USD values he wants to invest in.

4. Resultant value of the users portfolio will be calculated using the recent rate.

5. The bought currency will be added in the scrollable Transactions List.

   User repeats all the steps until he has a portfolio of chosen cryptocurrencies.

Alternate flow

From setp 1 of basic flow:

1. User finds another crypto he can afford investing in from the list.

Postcondition: Portfolio contains fully with chosen cryptocurrencies.

Sprint 3 - Beta

5 Implementation

5.1 Architecture

Since we had to get our data from either coincap API (using retrofit library) or the stored data in the local database(using room database library), which is fetched and stored before, we needed an architecture, which can separate our data sources. Our chosen architecture is the recommended architecture at android developer web site, which has separated every component like f.x UI and logic from each other. using this architecture clearly abstracts the logic of the actions that can be performed in the app.

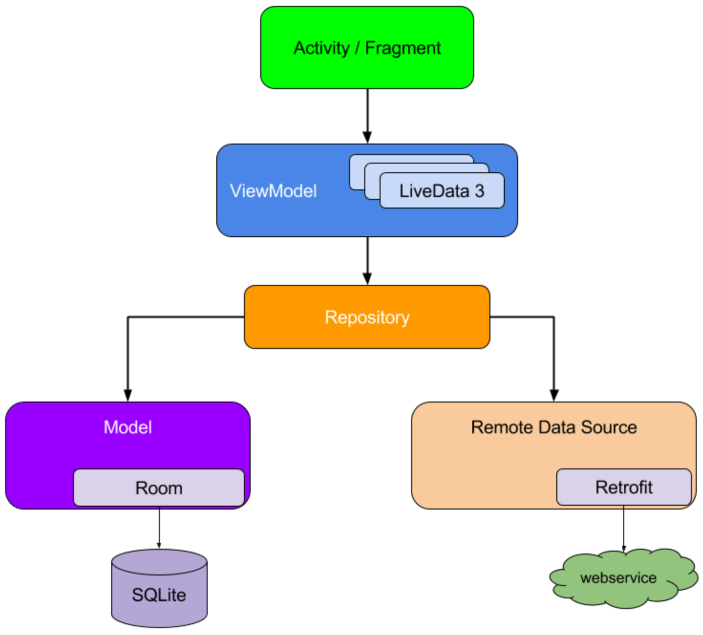


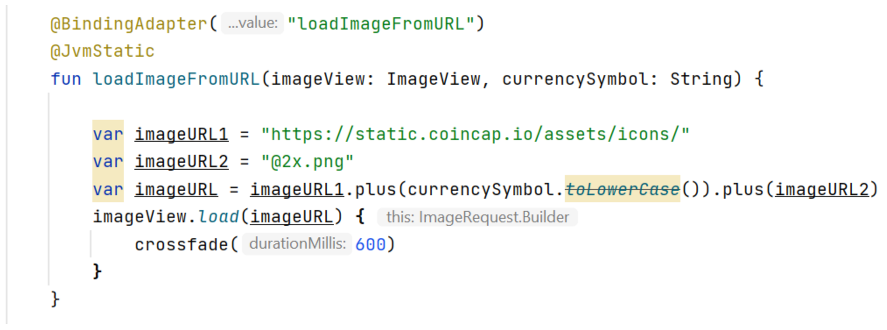
Figure 8: Android App Architecture

5.2 Hilt-Dagger

In this hierarchy, each upper component depends on its own lower component. we have implemented this dependency Injection with hilt-Dagger. Hilt-Dagger is 3rd-part injection library for Android that helped us to reduce the boilerplate of doing dependency injection. This requires to much work constructing every class and its dependency by hand. Hilt supported

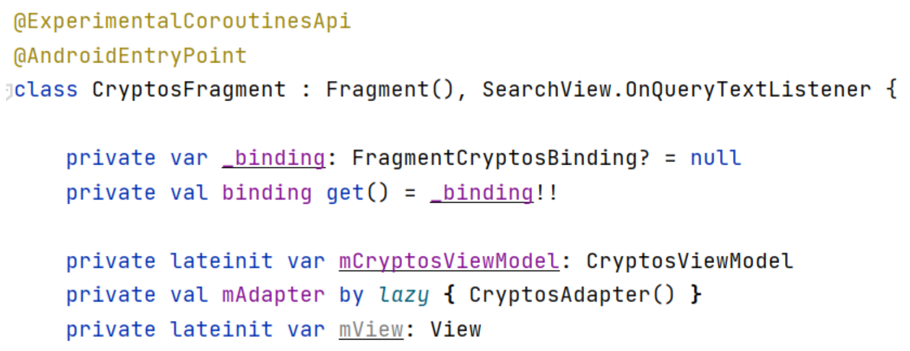
5.3 DataBinding

in this project we use databinding library to decrease model - View connection coding. To connect model with UI, we used to write a lot of code, but with Databinding library we can do that with smaill part of codes. we have even bonded a fx viewmodel function with UI buttns. here are some part of databinding codes that we have used in our app.



```
@BindingAdapter( ...value: "loadImageFromURL")
@JvmStatic
fun loadImageFromURL(imageView: ImageView, currencySymbol: String) {

    var imageURL1 = "https://static.coincap.io/assets/icons/"
    var imageURL2 = "@2x.png"
    var imageURL = imageURL1.plus(currencySymbol.~~toLowerCase~~()).plus(imageURL2)
    imageView.load(imageURL) {  this: ImageRequest.Builder
        crossfade( durationMillis: 600)
    }
}
```

Figure 9: An adapter for displaying image on ImageView by using coil library



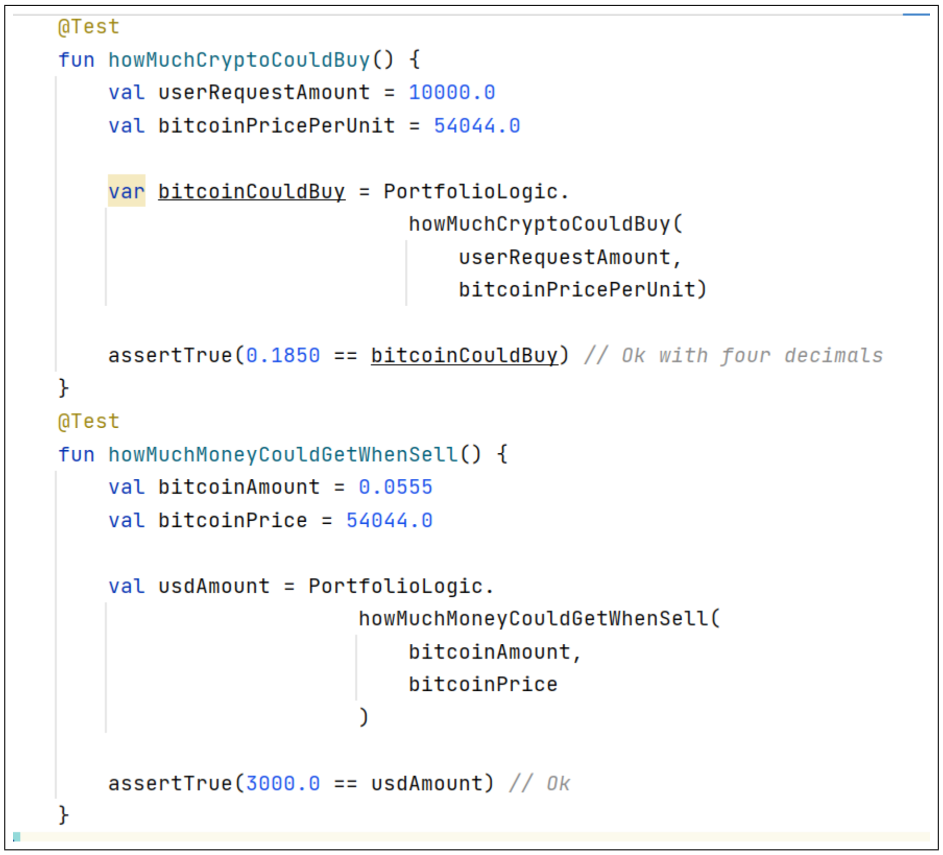
```
@ExperimentalCoroutinesApi
@AndroidEntryPoint
class CryptosFragment : Fragment(), SearchView.OnQueryTextListener {

    private var _binding: FragmentCryptosBinding? = null
    private val binding get() = _binding!!

    private lateinit var mCryptosViewModel: CryptosViewModel
    private val mAdapter by lazy { CryptosAdapter() }
    private lateinit var mView: View
```

Figure 10: using databinding in fragmemnt

6 Test

6.1 Junit Test

In howMuchCryptoCouldBuy() we insure that its return the right amount with four decimals number. For howMuchmoneyCouldGetWhenSell() it will insure to get the rignt value by using ceil() in PortfolioLogic class, which ceil() will round up usdAmount example if usdAmount equal to 2999.95 so it will be 3000.0.



```
@Test
fun howMuchCryptoCouldBuy() {
    val userRequestAmount = 10000.0
    val bitcoinPricePerUnit = 54044.0

    var bitcoinCouldBuy = PortfolioLogic.
                        howMuchCryptoCouldBuy(
                            userRequestAmount,
                            bitcoinPricePerUnit)

    assertTrue(0.1850 == bitcoinCouldBuy) // Ok with four decimals
}
@Test
fun howMuchMoneyCouldGetWhenSell() {
    val bitcoinAmount = 0.0555
    val bitcoinPrice = 54044.0

    val usdAmount = PortfolioLogic.
                        howMuchMoneyCouldGetWhenSell(
                            bitcoinAmount,
                            bitcoinPrice
                        )

    assertTrue(3000.0 == usdAmount) // Ok
}
```

Figure 11: Junit test over logic part

Sprint 4 - Release 1.0

7 Conclusion

The purpose with this project in the course 62416 Advanced Mobile Application Development at Technical University of Denmark was to develop an Android Mobile Application using Kotlin language and the latest technologies and different 3rd part libraries. We have chosen to develop an application where a user can learn how to navigate within the cryptocurrency market and make profitable investments by buying or selling cypto coins.

We have managed to meet the key requirements set by the product owner. Due to lack of time management and pressure from other courses during the project's progress we decided to ignore some features in order to hand in a finished product within the deadline. At the moment we have few bugs which has to be fixed. For instance before we had to hand in the project we got some problems in our applications. We could'nt figure this out. But we promise to fix it before the oral examination dated at 9th december. Other than that the application has flexible and intuitive user interface which makes the navigation through screen quite feasible. We have managed to fetch data from coincap.io RESTful API and display them within a scrollable recyclerview which makes the user able to track the cryptocurrencies with the latest rates of currencies. From the list the user can click/tap on a chosen crypt0 and begin to invest with the rewarded fake money amounted to 10,000 USD. At any given tie m, the user's portfolio net worth and points will be calculated based on the investments he/she had made.

References

[1] Document: this document is being inspired by
`https://github.com/LinkedInLearning/Software-Design-Requirements-Release-2825344`

[2] Most of the application has been developed by helping of Stevdza-San videos.
`https://www.youtube.com/channel/UCYLAirIEMMXtWOECuZAtjqQ`