Personal Budget and Portfolio Mobile Application

Natnapat Keeratirakornpisut
Department of Computer Engineering, Walailak University
Nakhonsithammarat, Thailand
Natnapat.ke@gmail.com

***Abstract*—People nowadays are interested in budgeting and investing in markets, particularly stocks as well as cryptocurrencies. It is possible to do so using a smartphone, but there is a drawback in that users must move between applications to view their portfolio.Therefore I create a mobile application that allows users to manage their budget expenses and investment profits. Users can track their expense and manage their budget. Users may see an overview of their investment, particularly the profit made by each asset, as well as any details that can be used to make investment decisions. Despite the fact that the major tasks have been completed, the development process is still ongoing. Furthermore, the application must be improved in order to satisfy users and be more applicable**

Keywords—Mobile application, Budget plan, Expense accounting, cryptocurrency, investment

I. INTRODUCTION

Finance and investing are gaining popularity among the younger generation. Investments such as funds and stocks have become more popular as a result of advances in computer technology and the internet. In addition to its role in altering the financial system, blockchain technology fosters trust. This includes the use of blockchain in a variety of sectors. This expands investment choices for existing market participants as well as new generations looking to get into the market. The majority of investments may be done by smart phone. Users must switch between applications to track their investment portfolio in different types of markets. As a result, users can not obtain an overview of their investment that causes uncomfortable situations.

People are interested in cryptocurrencies for a variety of reasons. One of the most important reasons is that market capitalization and volume have increased dramatically in recent years. As of May 6, 2018, the entire market value of cryptocurrencies exceeded 450 billion USD. A large capitalization is accompanied with a large profile margin. More and more people are drawn to this sector in the expectation of profiting from the trend.

Portfolio management is a critical skill for efficiently managing investments. It aids in determining how much to invest in whatever asset or debt. It is essentially a collection of instruments such as stocks, mutual funds, bond, and so on. Portfolio management, in other terms, is a collection of assets that helps to diversify investment risk and distribute money to the appropriate location.

The main objectives of the development is to create a mobile application that allows users to manage their own money, including not only investment but also expense accounting to make users, particularly younger generations interested in cryptocurrencies, stocks and money management, be able to track all of their assets and expenses. I also provide a design template and a concept that may be further improved.

A. *Paper Organization*

This mobile application is divided into two sections. The first is expense accounting, and the second is investment portfolio. In section II, I present the application functionality and in section III, I present the method and software development tools.Section IV is about API's data structure. Section V is a result and conclusion of the paper.

II. APPLICATION FUNCTIONALITY

A. *User Definition*

Because this application focus on young generation, the user in this article is someone between the ages of 18 and 30 who is interested in investing and money management

B. *Specification*

This mobile application was created with Flutter, a cross-platform mobile application development framework created by Google. So that it could run cross-platform.

The application can only used to do budgeting when it is offline. When the user is online, the user may access asset features.

C. *Budget Plan*

The user must first create a budget plan before adding any expense statements. It is possible to do on the screen shown in Figure 1(a). The user must specify the plan's saving goal, income, budget allocation and deadline. Following the addition of a budget plan, monthly budget plans are shown on the screen seen in Figure 1(b). The screen show how much money the user spends each month.

(a) Create Budget Screen *(b) Monthly Budget Plan Screen*

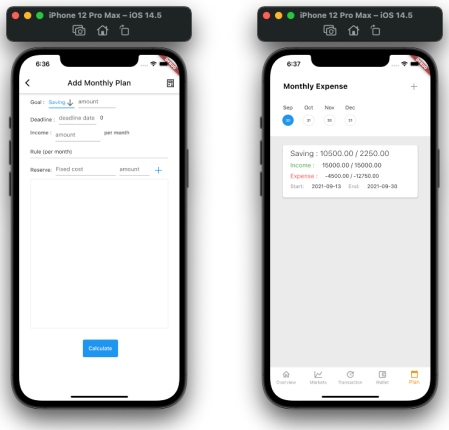
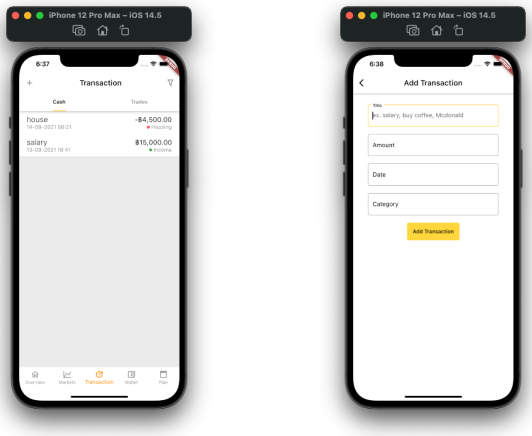


Figure 1. Screenshots of the budget plan screens

D. *Budgeting*

This mobile application aims at making users to be able to track their expense and manage their budget.

The screen shown in Figure 2(a) provides features such as a list of expense and income statements that the user may add by tapping the plus icon in the upper left corner of the screen and filling in the blank seen in Figure 2(b)

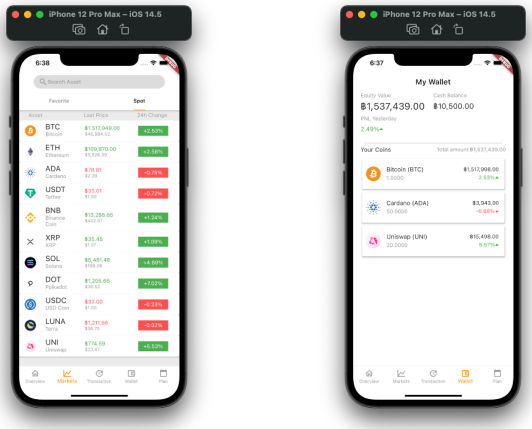


(a) Expense Statements Screen *(b) Adding Statement Screen*

Figure 2. Screenshots of the expense statements screens

E. *Investing*

The screen shown in Figure 3(a) provides a list of all supported assets as well as basic information such as name and price change in 24 hours. The screen as Figure 3(b) shows the details of the user's assets, cash balance, percentage change and net worth.



(a) All Assets Screen *(b) Wallet Screen*

Figure 3. Screenshots of the assets screens

Identify applicable funding agency here. If none, delete this text box.

III. TOOLS, DESIGNING AND DEVELOPMENT

Before you begin to format your paper, first write and save the content as a separate text file. Complete all content and organizational editing before formatting. Please note sections A-D below for more information on proofreading, spelling and grammar.

A. *Tools*

- Flutter is Google's UI toolkit for building beautiful, natively compiled applications for mobile, web, desktop, and embedded devices from a single code base. All of the UIs of this application are created by using Flutter.
- Lambda is a compute service that lets user run code without provisioning or managing servers. It is used to create APIs for retrieving asset information. And the programming language is python.
- The sqlite package, which is based on SQlite, was used to create this application as an internal database.

C. *Designing*

This application is consist of four parts: mobile application, internal database, external API, and API. The mobile application retrieves data from both the created API (using Lambda) and the external API (Coingecko's API) which provides realtime cryptocurrency price and details.

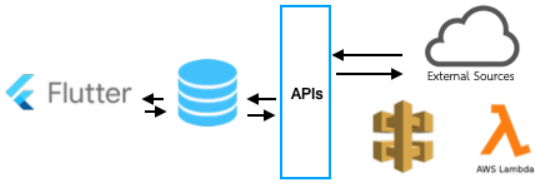


Figure 4. Overview of System

- Mobile application : every user interactions are on this part of system. Mobile application retrieve data from API to show on mobile screens.
- All user data is saved in an internal database (In-App database). As a result, the user can utilize the program while it is not connected to the internet (only budgeting features).
- Nowadays, numerous companies give information on assets, particularly cryptocurrencies and stock markets, through APIs. There are both free and paid options. However, in order to get information from these APIs, the user must be online in order to access the API-related functionality (assets features).
- A serverless microservice, or AWS Lambda in this case, is a service that does not need developers to configure any servers. We may use it to host our backend functions as micro services. In this case, I use it as an API endpoint to deliver a list of supported assets.

D. Database Design

Database that used to create this mobile application is a relational database (SQLite). As a result, I created database table as shown below.

TABLE I USER'S ASSETS

Attribute	Type	PK	FK	Not Null	Unique	Description
Id	integer	/		/	/	User's asset ID
symbol	Text			/		Abbreviation of assets
hodl	Text			/		Amount of asset
Favorite	Integer			/		Is this asset is set to favorite

TABLE 2 STATEMENT

Attribute	Type	PK	FK	Not Null	Unique	Description
Id	integer	/		/	/	Id of statement
Title	Text			/		Title of the statement
Amount	Real			/		How much does it cost ?
Timestamp	Text			/		Timestamp
monthlyPlanID	Integer		/	/		ID of budget plan in each month

TABLE 3 BUDGET PLAN

Attribute	Type	PK	FK	Not Null	Unique	Description
planID	integer	/		/	/	Id of budget plan
goalType	Text			/		Budget plan goal type
Amount	Real			/		Amount
startDate	Text			/		When this plan start ?
endDate	Text			/		When this plan end ?

TABLE 4 MONTHLY BUDGET PLAN

Attribute	Type	PK	FK	Not Null	Unique	Description
MonthlyPlanID	integer	/		/	/	Id of monthly budgets plan
planID	Integer		/	/		Id of budget plan
Saving	Real			/		Amount of saving
startDate	Text			/		When it start ?
endDate	Text			/		When it end ?
Income	Real			/		Amount of income monthly
actualIncome	Real			/		Actual income
Expense	Real			/		Amount of expense monthly
actualExpense	Real			/		Actual expense

TABLE 5 FIXED COST MONTHLY

Attribute	Type	PK	FK	Not Null	Unique	Description
Rid	Integer	/		/	/	Id of fixed cost
planID	Integer		/	/		Id of budget plan
reservedType	Text			/		Type of fixed cost
reservedAmount	Real			/		Amount
actualAmount	Real			/		Actual amount
Checked	Integer			/		Check if fixed cost completed
monthlyPlanID	Integer		/	/		Id of monthly budget plan

There are five tables in the database that are related to one another as illustrated in Figure 5.

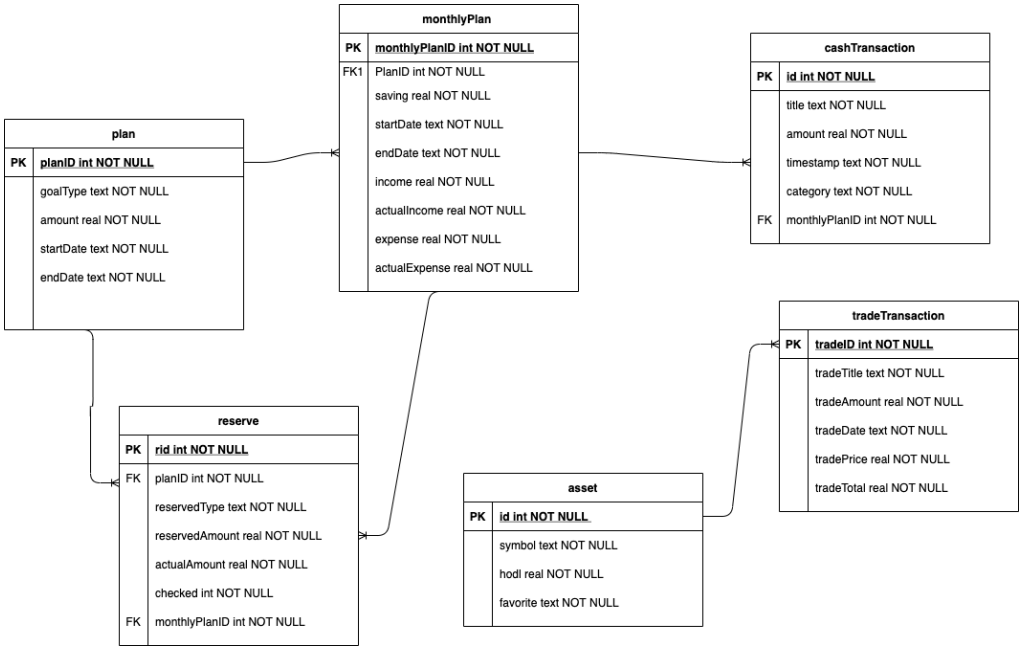


Figure 5. ER diagram

IV. API'S DATA STRUCTURE

As previously said, this application retrieves data from an external API (CoinGecko api). The structure of the data returned from the API is determined by the api endpoint that was requested.

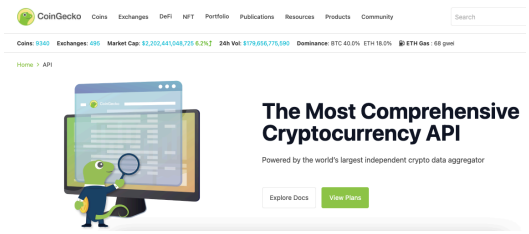


Figure 6. Coin Gecko Api website


```
{
  "id": "bitcoin",
  "symbol": "btc",
  "name": "Bitcoin",
  "image": "https://assets.coingecko.com/coins/images/1/large/bitcoin.png?
  1547033579",
  "current_price": 46816,
  "market_cap": 880644677219,
  "market_cap_rank": 1,
  "fully_diluted_valuation": 982871504257,
  "total_volume": 58143048545,
  "high_24h": 46968,
  "low_24h": 44380,
  "price_change_24h": 2317.7,
  "price_change_percentage_24h": 5.20857,
  "market_cap_change_24h": 41074726588,
  "market_cap_change_percentage_24h": 4.89235,
  "circulating_supply": 18815825,
  "total_supply": 21000000,
  "max_supply": 21000000,
  "ath": 64805,
  "ath_change_percentage": -27.67931,
  "ath_date": "2021-04-14T11:54:46.763Z",
  "atl": 67.81,
  "atl_change_percentage": 69016.52017,
  "atl_date": "2013-07-06T00:00:00.000Z",
  "roi": null,
  "last_updated": "2021-09-14T15:56:15.371Z"
}
```


Figure 7. Structure of cryptocurrency data from CoinGecko API

A. *Data from external API*

The API data covers all of the usage for the cryptocurrency portfolio mobile application. There are several endpoints given. However two endpoints are enough to provide data for the application.

Here is links from the CoinGecko API (https://api.coingecko.com/api/v3/coins/markets?vs_currency=usd&ids=bitcoin&order=market_cap_desc&per_page=100&page=1&sparkline=false) which when the api structure is accessed using browsers, the data structure will appear as shown as Figure 7. This API endpoint provides data about fundamental details of cryptocurrency : symbol, name, image, current price, market capital, market capital ranking, total volume, high price in 24hr, low price in 24hr, price change in 24hr, price change percentage in 24hr, market capital change in 24hr, market capital percentage change in 24hr, total supply, max supply, all-time-high price and date. The link and Figure 7 show data only one cryptocurrency ,bitcoin in this case, but we can change the link to concatenate the link "ids=". This is where the API server determine which cryptocurrencies data are requested.

This link () is another external API endpoint that the application access As shown in Figure 8. It provides OHLC data. These data are used to create candlestick graph.


```
[
    1631561400000,
    1471419.93,
    1473542.12,
    1471419.93,
    1472293.8
],
```

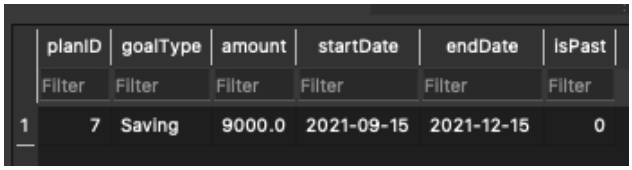

Figure 8. One of the OHLC dataset from API

B. *Data from micorservice API*

The assets supported by this application are limited, therefore, I create a list of supported assets. If supported assets list is up-to-date. User does not have to update the application because the list is provides online by API

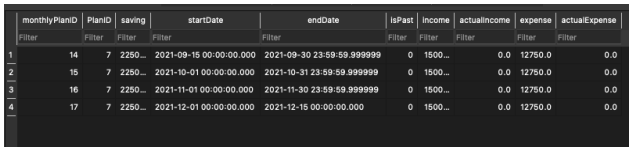
V. RESULT

The user must first create a budget plan before adding any statements. On the adding budget screen, the user must fill in every blank and all information must meet standards such as income not exceeding expenses. Following the creation of a budget plan, the application will divide user's budget plan into monthly budget plan. Therefore, the user can view monthly budget plans that are subsets of the budget plan. In the budget plan, the user may track and budget each month. However, the user is unable to view any previous budget plans. Each of the monthly plans and expenses fixed cost provided by the user is saved in a database as shown as Figure 9, 10 , and 11



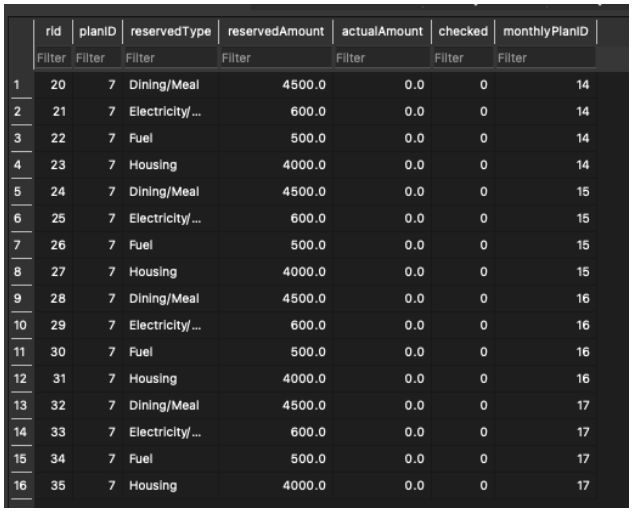
	planID	goalType	amount	startDate	endDate	isPast
	Filter	Filter	Filter	Filter	Filter	Filter
1	7	Saving	9000.0	2021-09-15	2021-12-15	0

Figure 11. Monthly budget plans created upon the submission of a budget plan



	monthlyPlanID	planID	saving	startDate	endDate	isPast	income	actualIncome	expense	actualExpense
	Filter	Filter	Filter	Filter	Filter	Filter	Filter	Filter	Filter	Filter
1	14	7	2250...	2021-09-15 00:00:00.000	2021-09-30 23:59:59.999999	0	1500...	0.0	12750.0	0.0
2	15	7	2250...	2021-10-01 00:00:00.000	2021-10-31 23:59:59.999999	0	1500...	0.0	12750.0	0.0
3	16	7	2250...	2021-11-01 00:00:00.000	2021-11-30 23:59:59.999999	0	1500...	0.0	12750.0	0.0
4	17	7	2250...	2021-12-01 00:00:00.000	2021-12-15 00:00:00.000	0	1500...	0.0	12750.0	0.0

Figure 10. Monthly budget plans created upon the submission of a budget plan



	rid	planID	reservedType	reservedAmount	actualAmount	checked	monthlyPlanID
	Filter	Filter	Filter	Filter	Filter	Filter	Filter
1	20	7	Dining/Meal	4500.0	0.0	0	14
2	21	7	Electricity/...	600.0	0.0	0	14
3	22	7	Fuel	500.0	0.0	0	14
4	23	7	Housing	4000.0	0.0	0	14
5	24	7	Dining/Meal	4500.0	0.0	0	15
6	25	7	Electricity/...	600.0	0.0	0	15
7	26	7	Fuel	500.0	0.0	0	15
8	27	7	Housing	4000.0	0.0	0	15
9	28	7	Dining/Meal	4500.0	0.0	0	16
10	29	7	Electricity/...	600.0	0.0	0	16
11	30	7	Fuel	500.0	0.0	0	16
12	31	7	Housing	4000.0	0.0	0	16
13	32	7	Dining/Meal	4500.0	0.0	0	17
14	33	7	Electricity/...	600.0	0.0	0	17
15	34	7	Fuel	500.0	0.0	0	17
16	35	7	Housing	4000.0	0.0	0	17

Figure 11. Every fixed cost of monthly budget plan stored in database

The user may enter any statements, and the monthly budget plan will be updated. The user can also delete the statements, and the monthly budget plan will be restored.

The user can add asset statements that will be shown on screen as Figure 12(a) as well as Figure 3(b) if they are still available in portfolio. Moreover the user can view graphical data of each asset.

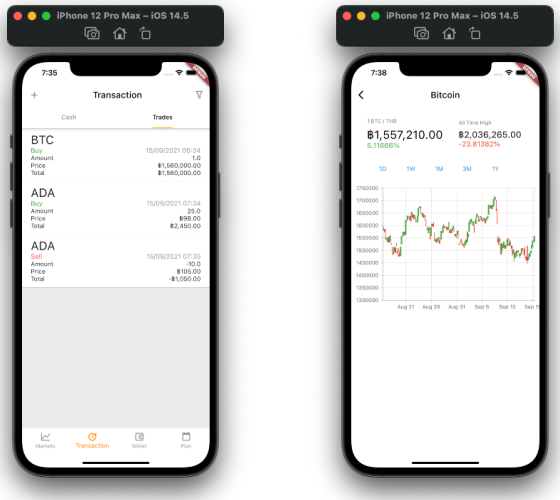


Figure 12(a) screenshot of list of asset statements

Figure 12(b) screenshot of graphical data of an asset

ACKNOWLEDGMENT

I would like to appreciate my adviser who have helped me to put ideas, well above the level of simplicity and into something concrete.

REFERENCES

1. D. E. Yurochkin, A. A. Horoshiy and S. A. Karpukhin, "Development of an Application for Expense Accounting," 2021 IEEE Conference of Russian Young Researchers in Electrical and Electronic Engineering (ElConRus), 2021, pp. 753-757, doi: 10.1109/ElConRus51938.2021.9396201.
2. T. Giridher, A. Bulchandani, R. Kim, P. Naik, A. Wasilewska and J. L. Wong, "Social mobile applications," 2010 IEEE Long Island Systems, Applications and Technology Conference, 2010, pp. 1-6, doi: 10.1109/LISAT.2010.5478281.
3. R. Parlika and P. W. Atmaja, "Realtime monitoring of Bitcoin prices on several Cryptocurrency markets using Web API, Telegram Bot, MySQL Database, and PHP-Cronjob," 2020 6th Information Technology International Seminar (ITIS), 2020, pp. 253-257, doi: 10.1109/ITIS50118.2020.9321109.
4. Q. Xie et al., "Chatbot Application on Cryptocurrency," 2019 IEEE Conference on Computational Intelligence for Financial Engineering & Economics (CIFEr), 2019, pp. 1-8, doi: 10.1109/CIFEr.2019.8759121.