Applied Information Technology And Computer Science Vol. 3 No. 2 (2022) 442-459
© Universiti Tun Hussein Onn Malaysia Publisher's Office

AITCS

Homepage: http://publisher.uthm.edu.my/periodicals/index.php/aitcs
e-ISSN :2773-5141

MyMoney: Money Management and Tracking Application

Ngoh Guan Ji, Rozanawati Darman*

Faculty of Computer Science and Information Technology,
Universiti Tun Hussein Onn Malaysia, Parit Raja, Batu Pahat, 86400, MALAYSIA

DOI: https://doi.org/10.30880/aitcs.2022.03.02.029
Received 14 June 2022; Accepted 26 September 2022; Available online 30 November 2022

Abstract: MyMoney is an application developed to allow the public to manage money and track income or expenses easily. MyMoney is developed to help the public manage money without financial literacy and track daily income or expenses with simple steps. In this application, users can record income and expenses, create a shopping list, add a planned payment, design a retirement financial plan and do analysis for monthly income and expenses. The project is developed based on an iterative development methodology. Implementation of the application will be done by using Android Studio software and Firebase as the database. After implementation, there are a testing phase will be done for collecting feedback and do improvement in the next version to ensure the latest version of the application can be satisfied by the public. On the whole, this application can help the users to manage money and record income or expenses.

Keywords: Money management, financial management, expense tracking

1. Introduction

Financial management is useful for the public in the personal life which can help to create a comfortable life with an assurance of a secured future and freedom to spend money for happiness. The importance of financial planning and management is reflected in all areas of personal and business life. All individuals no matter what financial capacity is must learn and study financial management and adapt it to improve quality of life [1].

According to the EPF annual report in 2015, many Malaysians have insufficient retirement savings in the pension funds as the age reaches 55, compelling Malaysians to prolong the working years [2]. It is also known that 50% of retirees exhaust their EPF savings within 5 years. Many Malaysians rely on the little nest egg accumulated through mandatory contributions to the EPF, which may not necessarily be enough for the life after retirement [3].

The current money management and tracking application can only be used to record and track their expenses or income. Therefore, the application developed in this project has a function which is called a retirement financial plan. It can help users to design a financial plan depending on the lifestyle or expected lifestyle. Users should insert monthly expenses, active or passive income, retirement age and

**Corresponding author: zana@uthm.edu.my*

choose the lifestyle pattern, then it will help users to calculate how much monthly income is needed. This can help users to know how much monthly income is needed to afford the expected lifestyle.

According to the accessing of some current money management and tracking application, the process of recording expenses or income of the most current application, users cannot record their expenses or income in detail. It only can record the amount, type and date of the expenses or income. Therefore, it is hard for the user to recall or check for the previous expenses or income. The application developed in this project can let users insert the details of the expenses or income like location, remark, payment method and image.

The search or filter feature is also bad in the current application because it can only check the expenses or income on a specific date and view the chart of the monthly expenses or income. It is hard to search for expenses or income in type or duration. In the developed application in this project, it can let users search for each or many expenses in a particular duration, type, location, date. It also can view the daily, monthly expenses in a chart and expenses chart of a particular type.

The objectives of this application are to:

1. Design money management and tracking applications using an object-oriented approach.
2. Develop money management and tracking application using Android technology.
3. Test the developed system.

The money management and tracking application developed in this project will be focused on helping people to manage money discipline. The language of this application will be English. The individuals involved in this case study are users and the administrator. There are 12 modules in the application: user login, user forget password, user register, analysis income or expenses, record income or expenses, income or expenses summary, retirement financial plan, shopping list, planned payment, contact us, user profile administrator login and administrator panel.

2. Literature Review

2.1 Financial Planning

Financial planning is a process that is always done by individuals or organizations which can help to achieve a target or objective by estimating the capital necessary. In financial planning, there is a financial policy that has been formulated which is related to profit or value of assets, investments and money management. The process of financial planning helps individuals or organizations to formalize the targets, procedures, activities, policies and budget. There are six steps in the financial planning process.

2.2 Study on the existing applications

In this study, there are three existing applications are selected to do a comparison with the proposed application in the project. The applications are Money Manager, 1Money and Wallet. There will be a comparison table to show the similarity and differences of the existing applications are studied and the proposed application.

Money Manager is money management and tracking application owned by Realbyte Inc [4]. First of all, it allows photo saving when recording new expenses. Users can save the receipt or memory together with the expenses recorded. The application also has an asset graphs feature that will help users to review the asset trend in a chart. It will show a graph of the monthly expenses and income. There is a feature which is called a reinforced filter. It can help users to review the transactions with more filtering options. In this feature, it will show the income and expenses of the account that is selected by the users only. the application allows the use of advanced budget features. Users can set a monthly budget and budget for every type of expense.

1Money is a money management and tracking application that supports Android users only [5]. Users can add a new transaction easily and faster. Users need to select payment method and type of expenses, then enter the amount and click the tick button to save. One of the features of this application is expenses at a glance. The application will help users analyze all the expenses of a day that has been logged by users and show it in an informative chart. There is a budget feature that allows users to set their budgets every month for each of the types of expenses. The application can track savings and debits which can help users to keep an eye on the balances of the account.

Wallet is money management and tracking application that can help users to do finance management personally [6]. The application can automatically update bank details if the user syncs the bank account with the application. The application will help users to categorize the payment method and review the expense or income with the type. It also will list out all the amounts of each type of expense and income. There is a planned payment feature that can let users add the payment they need to pay monthly or on a specific date in the future. It will remind the users by a phone notification when the payment date is closed. In the application, users are allowed to set a budget for spending. Wallet also can let users share the selected account with others who can be trusted and cooperate on a budget.

Table 1: Comparison between the existing applications and the proposed application

Categories	System / Feature	Money Manager	1 Money	Wallet	Proposed application
Platform support	Android	✓	✓	✓	✓
	IOS	✓	✕	✓	✕
	Windows	✕	✕	✓	✕
User Interface Design (UI)	Dark/Light Mode	✓	✓	✓	✕
User Experience Design (UX)	Banner advertisements and pop-ups	✓	✓	✕	✕
	Able to change the icon of recommended expense categories	✕	✕	✓	✕
	Additional charge for a specific feature	✕	✓	✕	✕
	Able login with social network	✕	✓	✓	✕
Functions	Expenses and income tracking	✓	✓	✓	✓
	Planned payment	✕	✕	✓	✓
	Budget	✓	✓	✓	✓
	Shopping lists	✕	✕	✓	✓
	Reinforced filter	✓	✕	✓	✓

	Export report	✓	✓	✓	✓
	Retirement financial plan	X	X	X	✓
	Bank account sync	X	X	✓	X
	Expenses and income analysis	✓	✓	✓	✓
	Photo saved in process of logging new expenses	✓	X	X	✓

*✓ = Present

*X = Not present

3. Methodology

The methodology that is applied in the project is iterative development. The reason for using iterative development as the methodology is money management and the tracking application proposed in the project is developed in an object-oriented approach. According to the testing has done by Bosman in 1996, object-oriented software could be developed under any software development process model [7]. In the testing, Booch has suggested that the use of an object-oriented paradigm does not change any basic testing practices [8]. Unit testing, integration testing and system testing all have useful roles in object-oriented software testing. Additionally, regression testing also plays an important role whenever classes and objects are added or modified. For iterative development, it has to do many times of testing and adjustment then only come out with the latest version of the application.

Figure 1.0 shows the process of iterative development. First of all, it needs to do the planning and analysis phase. Then, it will break the overall project into a series of versions that are developed sequentially. The users can get a preliminary version of the application for testing the function and seeing the user interface. After that, users should give feedback for the preliminary versions which can help for analysis requirements and do improvement in the next version. This step will be repeating for four times in this project which ensure the latest version of the application is accepted by the users.

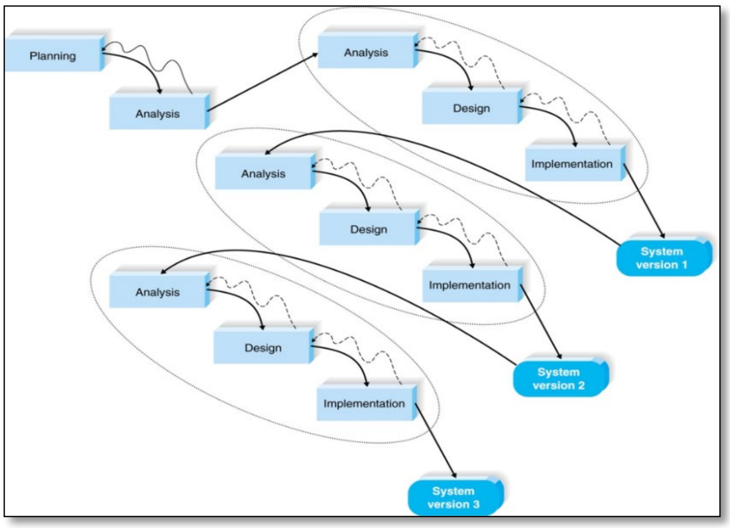


Figure 1.0 Process of iterative development

In this project, the first task in the planning phase is to discover the objective and problem statement of the project. After that, development tools use to develop the application should be decided which use

Visual Studio Code to do coding, Android Studio mobile simulator to display the user interface, Flutter as the framework of the project and coding language is Dart. Moreover, the function of the application also is decided in the planning phase. Then, a proposal will come out to list all information and planning about this project.

In the analysis phase, some tools or knowledge that will be used for the development should be studied. In the project, there will be three existing applications get analysis the features and development concept of the existing application. Then, the literature review is done for the knowledge of financial planning and application programming interface (API). In this phase, a case diagram, sequence diagram and activity diagram should be prepared.

The design phase is a phase to determine design strategy. Architecture, interface, database and features in the application should be designed. The design of the database is done and sketched in an entity-relationship diagram. The design that has been done in this phase will decide the implementation of the application in the next phase. In iterative development, the design phase will do adjustments according to the feedback from the user testing.

During the implementation phase, all modules and functions of the application are implemented. There are three steps in this phase which are system construction, installation and support plan. For the system construction, the application is built and tested to make sure it performs as designed. Then, the application should be allowed installed into the device. After that, a support plan will be prepared, including a post-implementation review and a systematic way of identifying changes needed for the system.

After the application is installed into the device, I will do testing by 15 users and give feedback. After that, feedbacks are collected will be analyzed for adjustment in the next version of the application. Total will be 4 times testing to be done to ensure the application will be satisfied by users.

4. Analysis and Design

4.1 System Analysis

4.1.1 Use Case Diagram

The use case diagram shown in Figure 2.0 consists of 15 main use cases, which are register, login, logout reset password, add new income/expense, search income/expense, create shopping list, edit shopping list, delete shopping list, create a planned payment, edit planned payment, delete a planned payment, design lifestyle financial plan and edit user account information. These use cases are the functional requirements of the proposed application. The actors in this application are the user and administrator.

The use case diagram shown in Figure 2.0 consists of 15 main use cases, which are register, login, logout reset password, add new income/expense, search income/expense, create shopping list, edit shopping list, delete shopping list, create a planned payment, edit planned payment, delete a planned payment, design lifestyle financial plan and edit user account information. These use cases are the functional requirements of the proposed application. The actors in this application are the user and administrator.

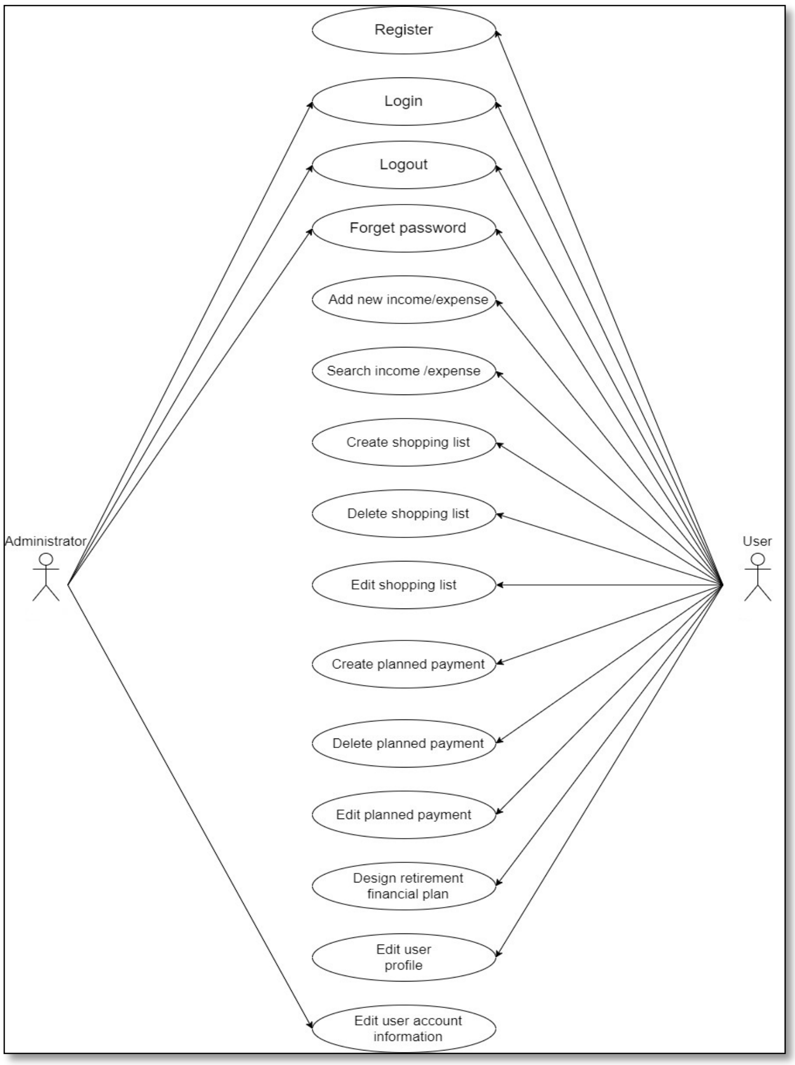


Figure 2.0: The user case diagram of the application

4.1.2 User Activity Diagram

Figure 3.0 illustrated the user activity diagram of the application. The overall process will start with the registration page. A new user will require to register a new account before login into the system. A user with an existing account can directly login into the application, which will be redirected to the homepage. However, a summary of income or expense will be set as the homepage.

In the summary of income or expense page, users allow to add, edit, delete and search income or expense. There is a hamburger button on the top left of the page which is the menu button. In the menu, there are functions such as contact us, planned payment, shopping list, user profile, income or expense analysis, lifestyle financial planning and logout.

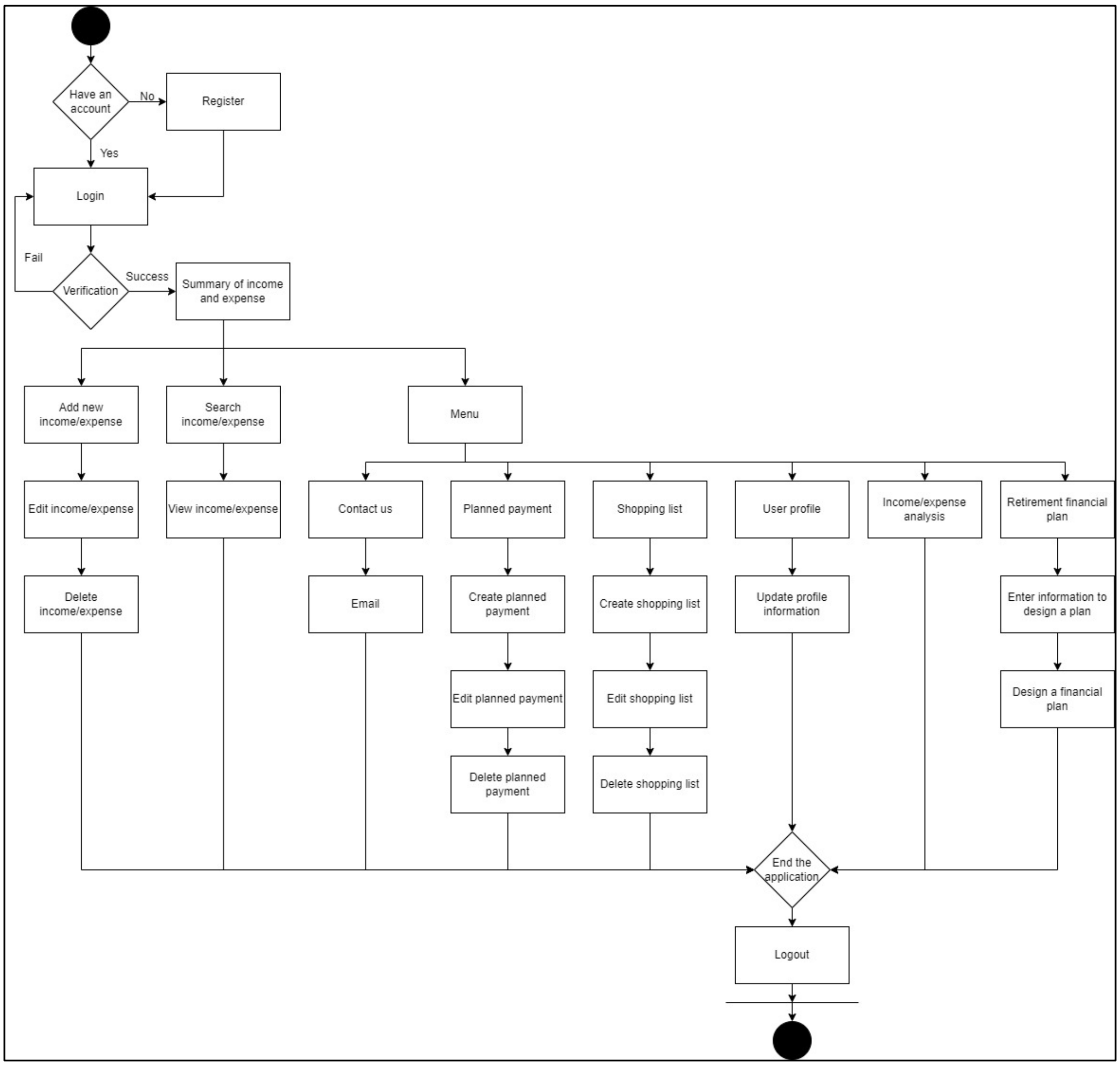


Figure 3.0: The user activity diagram of the application

5.1 Database configuration

Firebase Firestore Databse and Storage are picked as the database of the proposed application. It must be set up for the mobile and web application for connecting to the database. The dependencies need to be declared for the database in the mobile application as shown in Figure 5.1. On the other hand, initialization for the database connection is shown in Figure 5.2.

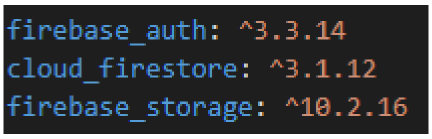
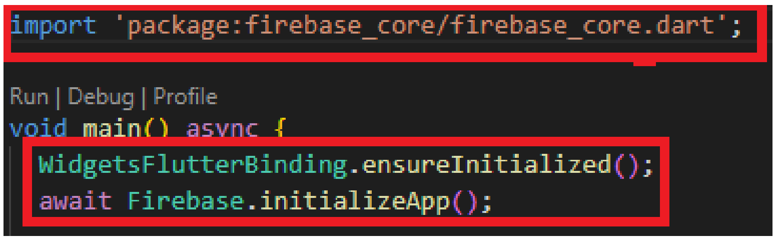

```
firebase_auth: ^3.3.14
cloud_firestore: ^3.1.12
firebase_storage: ^10.2.16
```

Figure 5.1: Dependency declared for Firebase connection of web and mobile application



```
import 'package:firebase_core/firebase_core.dart';

Run | Debug | Profile
void main() async {
  WidgetsFlutterBinding.ensureInitialized();
  await Firebase.initializeApp();
```

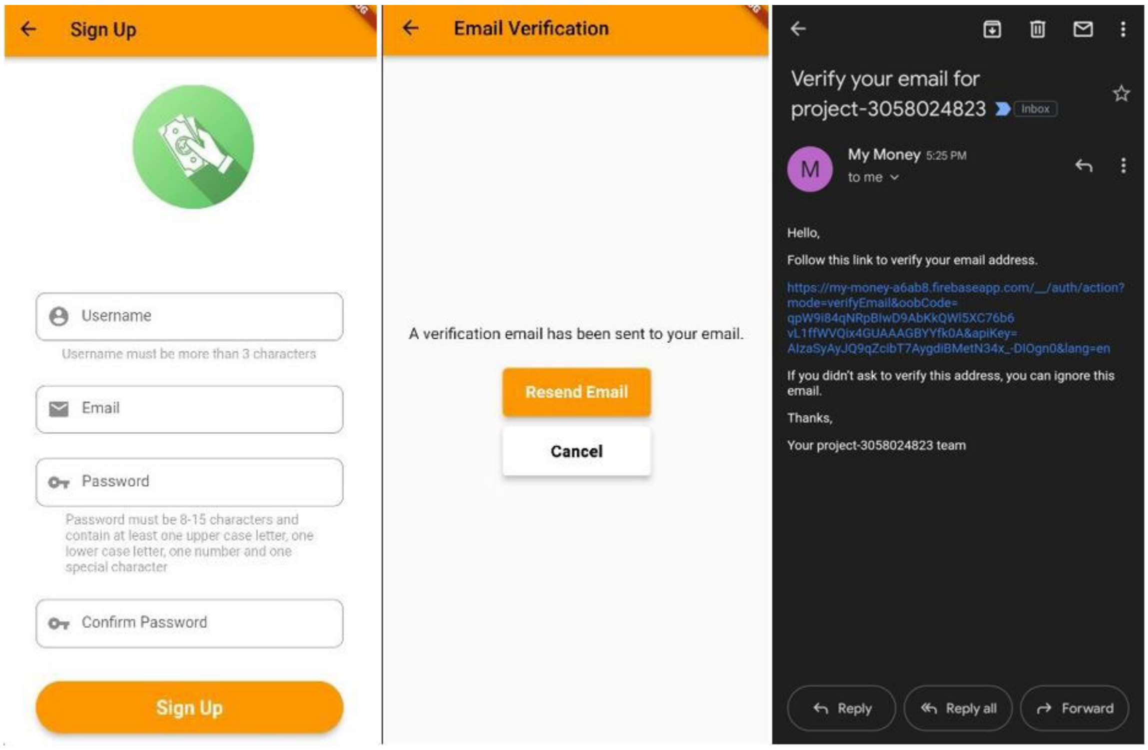
Figure 5.2: Inilization for Firebase connection of web and mobile application

5.2.1 User management module

The user management module typically allows users to manage their information that consists of register, login, logout, password reset, user profile, update medical history and update consultation details.

5.2.1.1 User management module

This activity will register an account for a new user, and there will be validation for the input field. Then, the system will check the availability of the username and email in Firebase Database whether it has been registered and start the registration process if the data is not available. The registration process can be shown in Figure 5.3.



Username must be more than 3 characters

Password must be 8-15 characters and contain at least one upper case letter, one lower case letter, one number and one special character

A verification email has been sent to your email.

Verify your email for project-3058024823

My Money 5:25 PM
to me

Hello,

Follow this link to verify your email address.

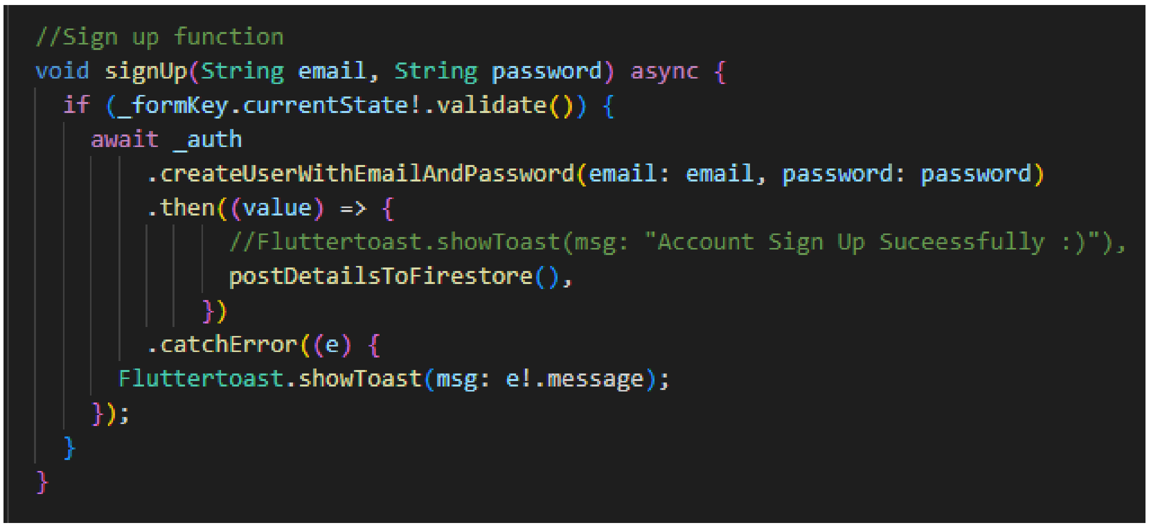
https://my-money-a6ab8.firebaseapp.com/__/auth/action?mode=verifyEmail&oobCode=qpW9I84qNRpBIwD9AbKkQWI5XC76b6vL1ffWVQix4GUAAAGBYYfk0A&apiKey=AIzaSyAyJQ9qZclbT7AygdIBMetN34x_-DIOgn0&lang=en

If you didn't ask to verify this address, you can ignore this email.

Thanks,

Your project-3058024823 team

Figure 5.3: Register process for user



```
//Sign up function
void signUp(String email, String password) async {
  if (_formKey.currentState!.validate()) {
    await _auth
        .createUserWithEmailAndPassword(email: email, password: password)
        .then((value) => {
              //Fluttertoast.showToast(msg: "Account Sign Up Suceessfully :)"),
              postDetailsToFirestore(),
            })
        .catchError((e) {
      Fluttertoast.showToast(msg: e!.message);
    });
  }
}
```

Figure 5.4: Code segment which processes the user registration

5.3.1.2 Login Function

Figure 5.5 shows the login interface for the application. The user can select the account type and login into the application by using their email and password. The application will check the status of email verification and role in database. If it is verified the application will redirect the user to homepage, else who is unverified it will redirect the user to email verification page. In Figure 5.9 shows the code segment for the user login.

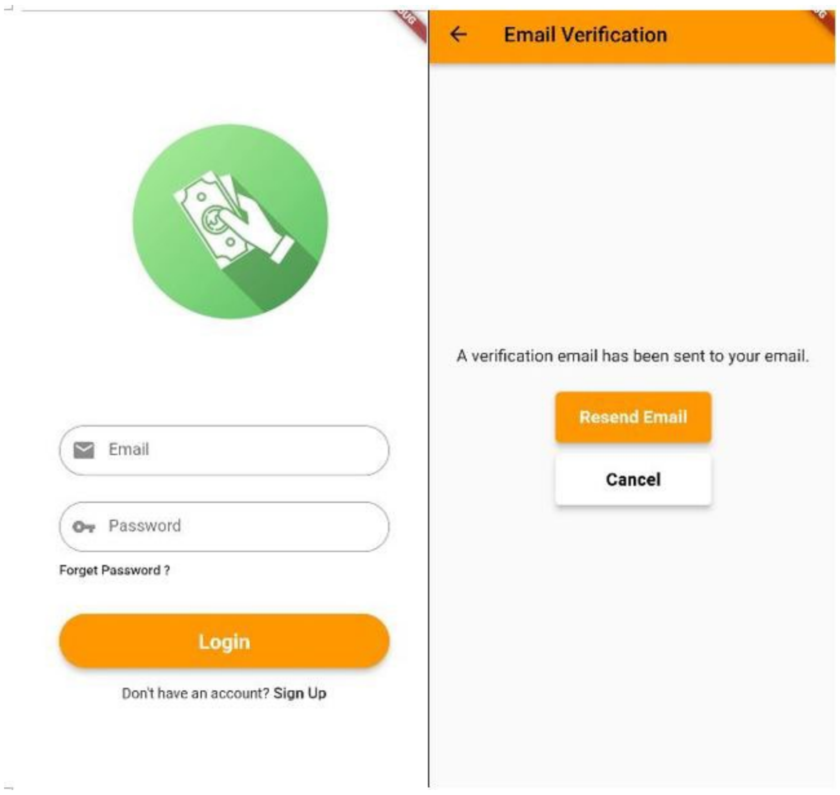
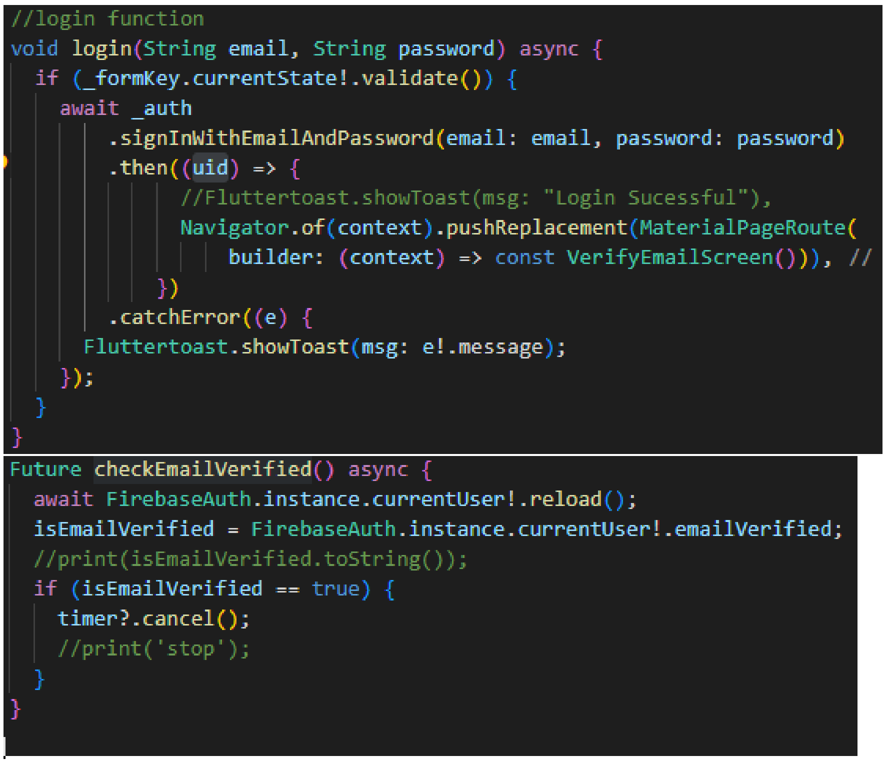

Email
Password
Forget Password ?
Login
Don't have an account? Sign Up
Email Verification
A verification email has been sent to your email.
Resend Email
Cancel

Figure 5.5: Login process for user

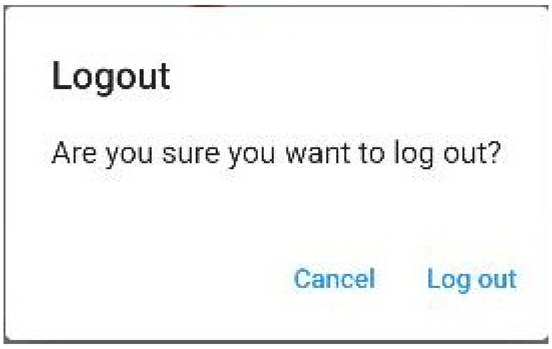


```
//login function
void login(String email, String password) async {
  if (_formKey.currentState!.validate()) {
    await _auth
        .signInWithEmailAndPassword(email: email, password: password)
        .then((uid) => {
              //Fluttertoast.showToast(msg: "Login Sucessful"),
              Navigator.of(context).pushReplacement(MaterialPageRoute(
                  builder: (context) => const VerifyEmailScreen())), //
            })
        .catchError((e) {
      Fluttertoast.showToast(msg: e!.message);
    });
  }
}
Future checkEmailVerified() async {
  await FirebaseAuth.instance.currentUser!.reload();
  isEmailVerified = FirebaseAuth.instance.currentUser!.emailVerified;
  //print(isEmailVerified.toString());
  if (isEmailVerified == true) {
    timer?.cancel();
    //print('stop');
  }
}
```

Figure 5.9: Code segment for the user login process

5.3.1.3 Logout Function

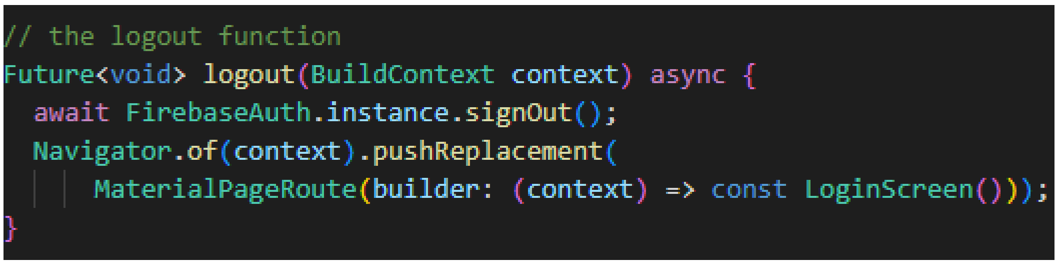
There will be an alert dialogue box to confirm the user's action before the logout process, as shown in Figure 5.6, and the code segment for the logout process is shown in Figure 5.7.



Logout

Are you sure you want to log out?

Figure 5.6: Logout confirmation



```
// the logout function
Future<void> logout(BuildContext context) async {
  await FirebaseAuth.instance.signOut();
  Navigator.of(context).pushReplacement(
      MaterialPageRoute(builder: (context) => const LoginScreen()));
}
```

Figure 5.7: Code segment for logout

5.3.2 Expenses and Income module

In this function user can view today's expenses and income have been recorded into the database. The user also can filter or search the expenses and income by day as shown in Figure 5.8. The code segment of display the expenses and income show in Figure 5.9 and Figure 5.10. The filter or search function code segment show in Figure 5.11.

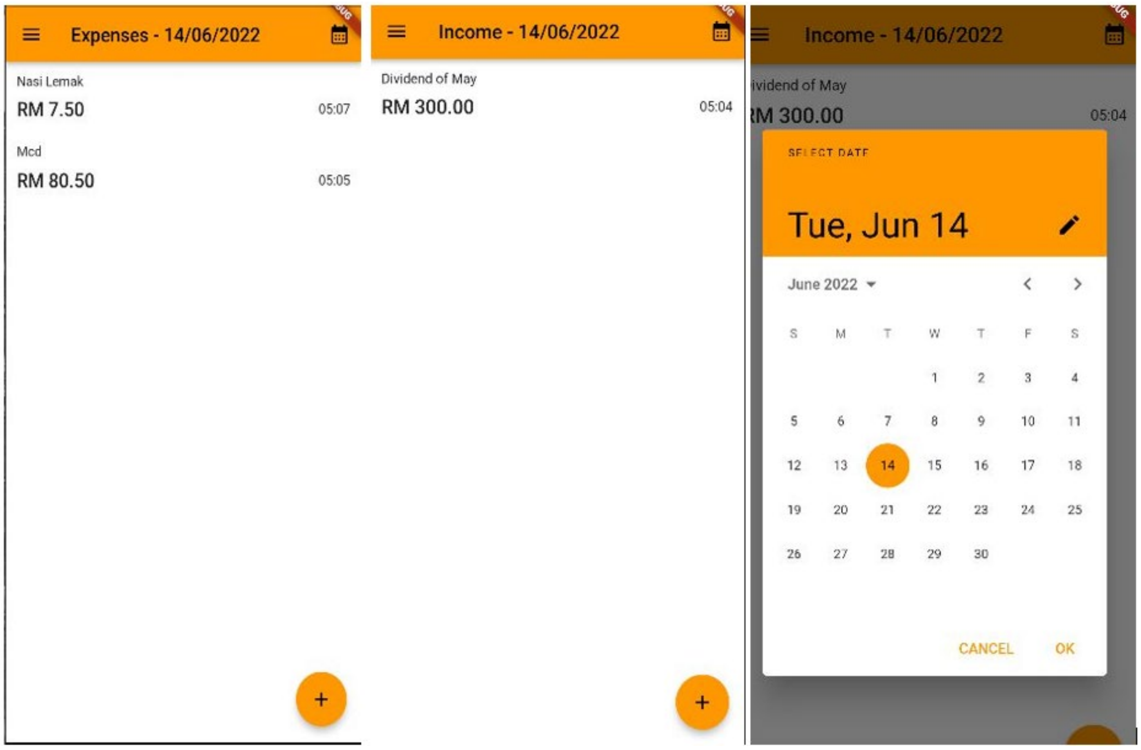
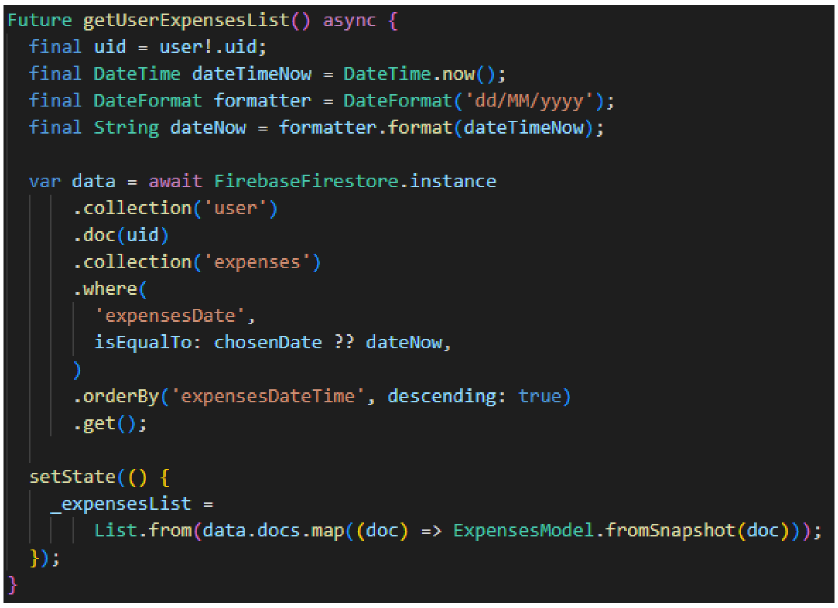


Figure 5.8: Display expenses and income and filter function

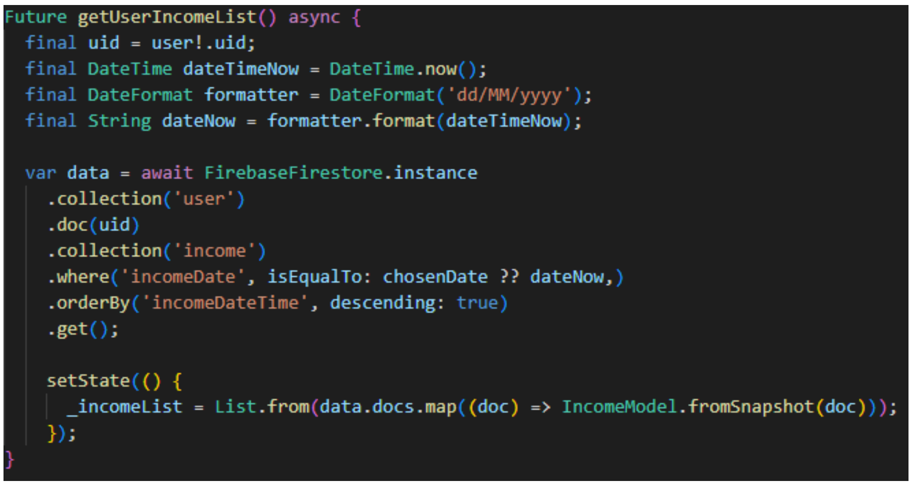


```
Future getUserExpensesList() async {
  final uid = user!.uid;
  final DateTime dateTimeNow = DateTime.now();
  final DateFormat formatter = DateFormat('dd/MM/yyyy');
  final String dateNow = formatter.format(dateTimeNow);

  var data = await FirebaseFirestore.instance
      .collection('user')
      .doc(uid)
      .collection('expenses')
      .where(
        'expensesDate',
        isEqualTo: chosenDate ?? dateNow,
      )
      .orderBy('expensesDateTime', descending: true)
      .get();

  setState(() {
    _expensesList =
        List.from(data.docs.map((doc) => ExpensesModel.fromSnapshot(doc)));
  });
}
```

Figure 5.9: Code segment to display expenses

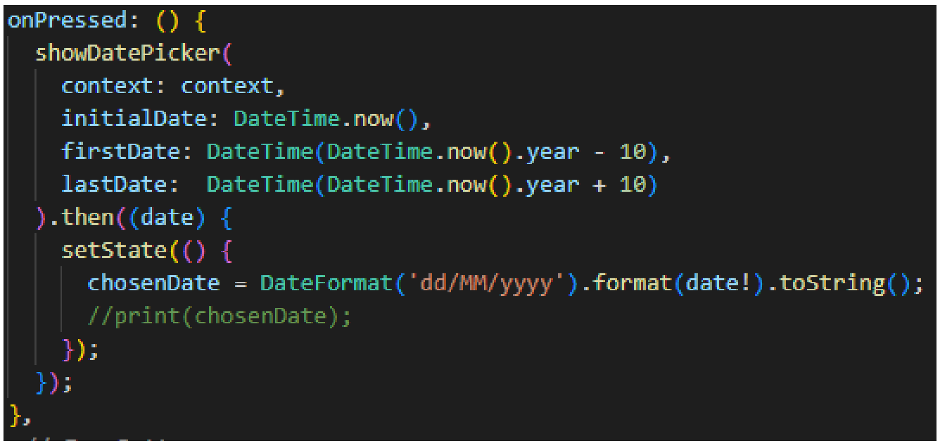


```
Future getUserIncomeList() async {
  final uid = user!.uid;
  final DateTime dateTimeNow = DateTime.now();
  final DateFormat formatter = DateFormat('dd/MM/yyyy');
  final String dateNow = formatter.format(dateTimeNow);

  var data = await FirebaseFirestore.instance
    .collection('user')
    .doc(uid)
    .collection('income')
    .where('incomeDate', isEqualTo: chosenDate ?? dateNow,)
    .orderBy('incomeDateTime', descending: true)
    .get();

    setState(() {
      _incomeList = List.from(data.docs.map((doc) => IncomeModel.fromSnapshot(doc)));
    });
}
```

Figure 5.10: Code segment to display income



```
onPressed: () {
  showDatePicker(
    context: context,
    initialDate: DateTime.now(),
    firstDate: DateTime(DateTime.now().year - 10),
    lastDate:  DateTime(DateTime.now().year + 10)
  ).then((date) {
    setState(() {
      chosenDate = DateFormat('dd/MM/yyyy').format(date!).toString();
      //print(chosenDate);
    });
  });
},
```

Figure 5.11: Code segment for the filter or search function

5.3.2.2 Edit and Delete expenses and income function

This function let user can edit and delete the expenses and income already record in the database. Figure 5.20 shows the interface of expenses detail, edit expenses and delete expenses. On the other hand, Figure 5.20 shows the interface of income detail, edit income and delete income. Figure 5.21 shows the code segment of edit expenses and income. Figure 2.22 shows the code segment to delete expenses and income.

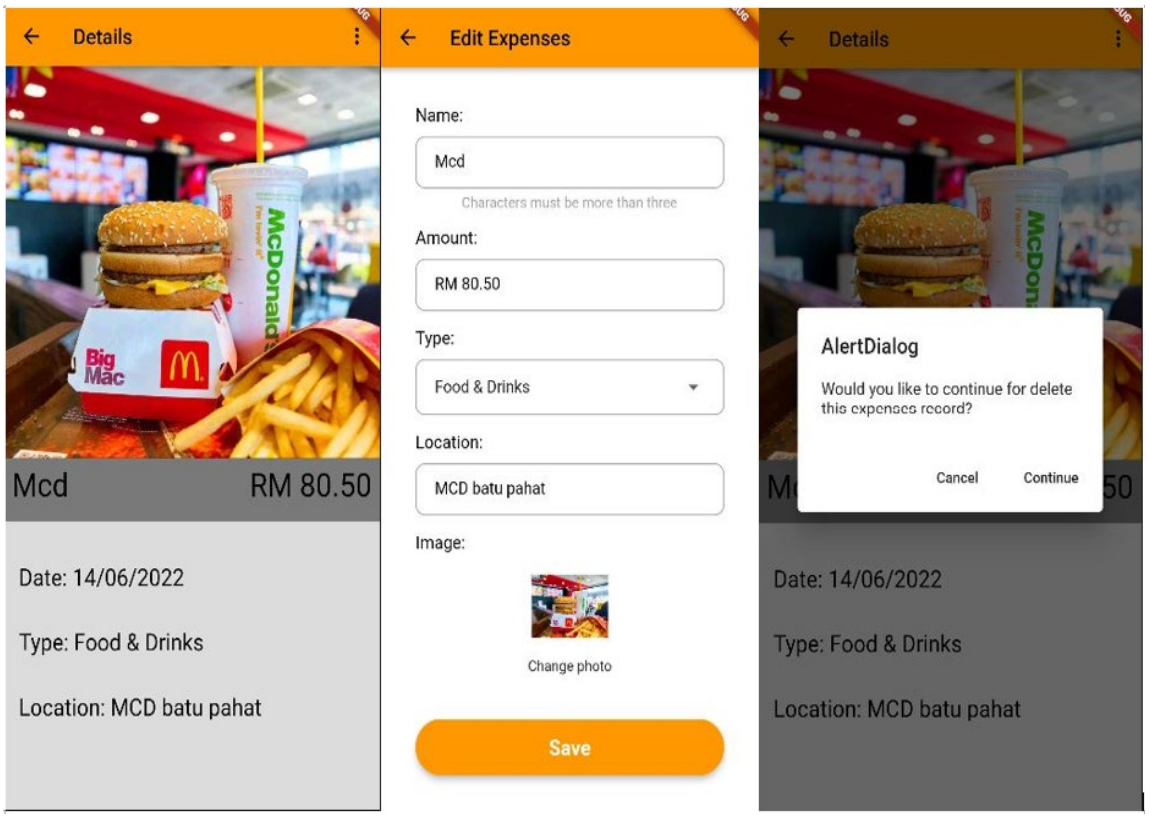


Figure 5.20: Interface detail of expenses, edit expenses and delete expenses

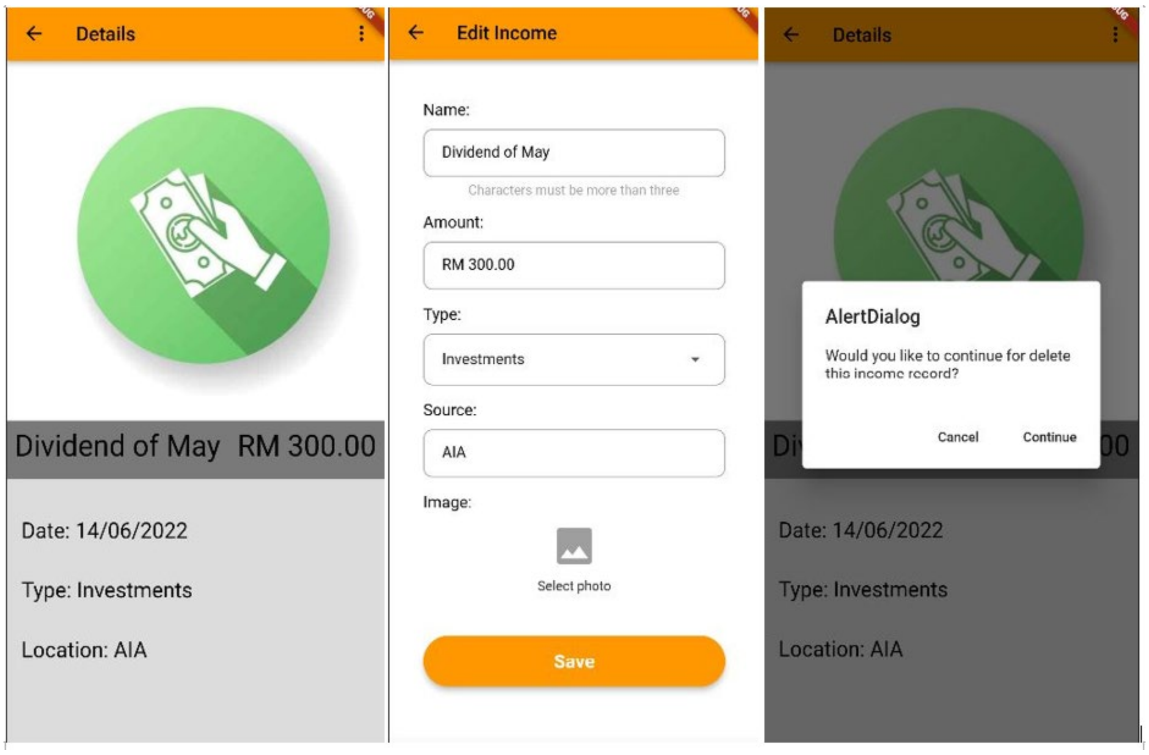


Figure 5.21: Interface detail of expenses, edit expenses and delete expenses

5.3.3 Analysis module

5.3.3.1 Cashflow analysis function

Figure 5.22 shows the cashflow analysis interface which show the total of income and expenses of the month and a graph of the cashflow of the month. Code segment of the cashflow analysis is shown in Figure 5.23.

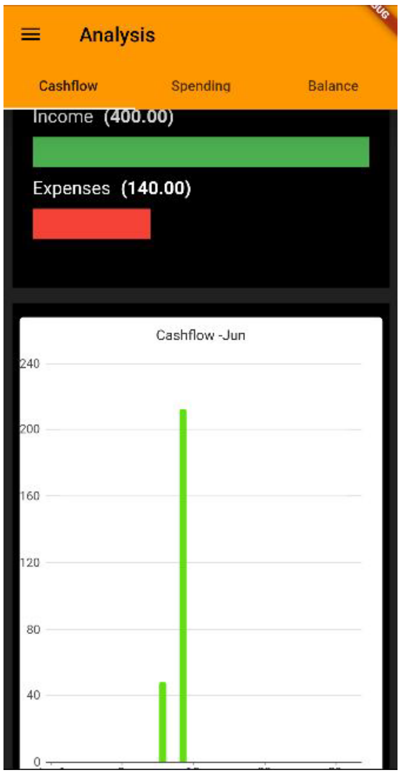
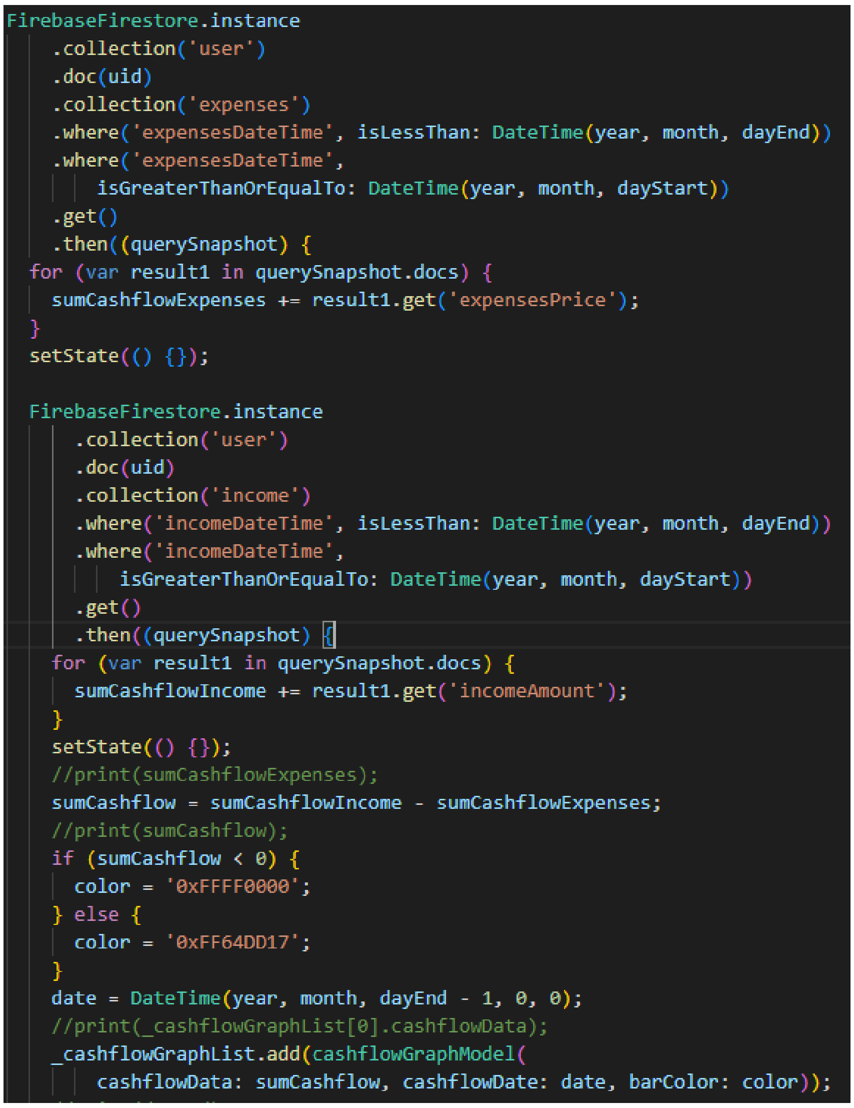


Figure 5.22: Interface of cash flow analysis



```
FirebaseFirestore.instance
    .collection('user')
    .doc(uid)
    .collection('expenses')
    .where('expensesDateTime', isLessThan: DateTime(year, month, dayEnd))
    .where('expensesDateTime',
        isGreaterThanOrEqualTo: DateTime(year, month, dayStart))
    .get()
    .then((querySnapshot) {
  for (var result1 in querySnapshot.docs) {
    sumCashflowExpenses += result1.get('expensesPrice');
  }
  setState(() {});

  FirebaseFirestore.instance
      .collection('user')
      .doc(uid)
      .collection('income')
      .where('incomeDateTime', isLessThan: DateTime(year, month, dayEnd))
      .where('incomeDateTime',
          isGreaterThanOrEqualTo: DateTime(year, month, dayStart))
      .get()
      .then((querySnapshot) {
    for (var result1 in querySnapshot.docs) {
      sumCashflowIncome += result1.get('incomeAmount');
    }
    setState(() {});
    //print(sumCashflowExpenses);
    sumCashflow = sumCashflowIncome - sumCashflowExpenses;
    //print(sumCashflow);
    if (sumCashflow < 0) {
      color = '0xFFFF0000';
    } else {
      color = '0xFF64DD17';
    }
    date = DateTime(year, month, dayEnd - 1, 0, 0);
    //print(_cashflowGraphList[0].cashflowData);
    _cashflowGraphList.add(cashflowGraphModel(
        cashflowData: sumCashflow, cashflowDate: date, barColor: color));
```

Figure 5.23: Code segment of analysis cashflow

5.3.3.2 Expenses and income analysis function

Figure 5.24 shows the expenses and income analysis which analysis the expenses and income of the month in a pie chart. Code segment of expenses analysis shown in Figure 5.25 and code segment of income analysis shown in Figure 5.26.

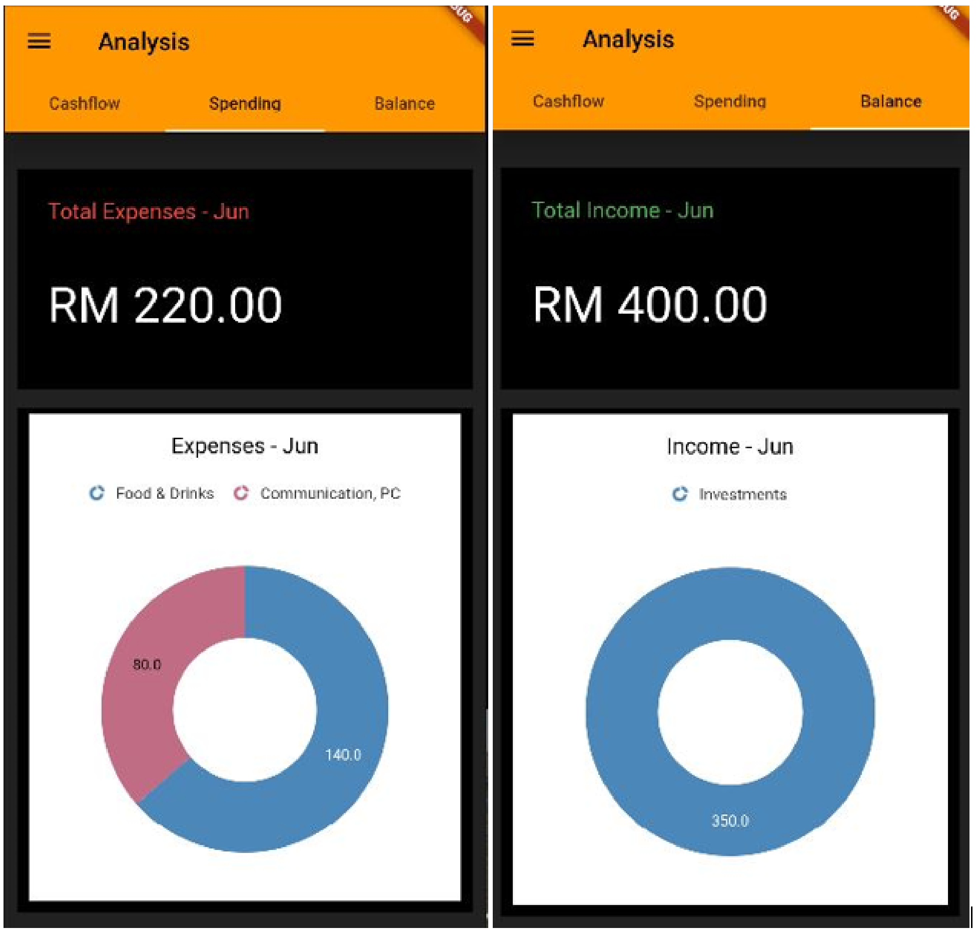
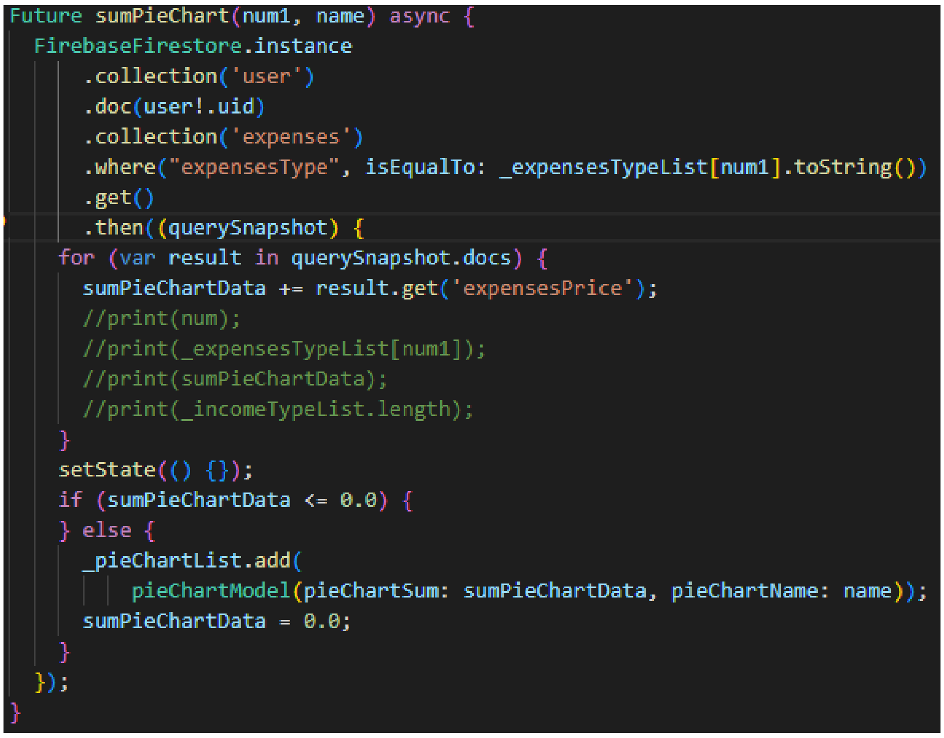
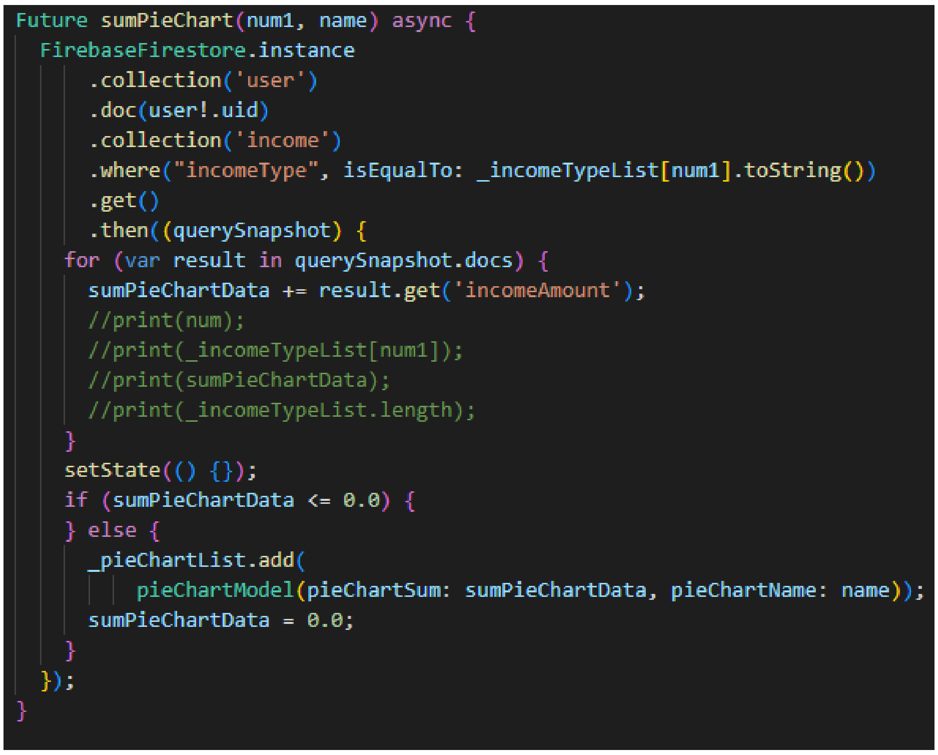


Figure 5.24: Interface of expenses and income analysis



```
Future sumPieChart(num1, name) async {
  FirebaseFirestore.instance
      .collection('user')
      .doc(user!.uid)
      .collection('expenses')
      .where("expensesType", isEqualTo: _expensesTypeList[num1].toString())
      .get()
      .then((querySnapshot) {
    for (var result in querySnapshot.docs) {
      sumPieChartData += result.get('expensesPrice');
      //print(num);
      //print(_expensesTypeList[num1]);
      //print(sumPieChartData);
      //print(_incomeTypeList.length);
    }
    setState(() {});
    if (sumPieChartData <= 0.0) {
    } else {
      _pieChartList.add(
          pieChartModel(pieChartSum: sumPieChartData, pieChartName: name));
      sumPieChartData = 0.0;
    }
  });
}
```

Figure 5.25: Code segment of expenses analysis



```
Future sumPieChart(num1, name) async {
  FirebaseFirestore.instance
      .collection('user')
      .doc(user!.uid)
      .collection('income')
      .where("incomeType", isEqualTo: _incomeTypeList[num1].toString())
      .get()
      .then((querySnapshot) {
    for (var result in querySnapshot.docs) {
      sumPieChartData += result.get('incomeAmount');
      //print(num);
      //print(_incomeTypeList[num1]);
      //print(sumPieChartData);
      //print(_incomeTypeList.length);
    }
    setState(() {});
    if (sumPieChartData <= 0.0) {
    } else {
      _pieChartList.add(
          pieChartModel(pieChartSum: sumPieChartData, pieChartName: name));
      sumPieChartData = 0.0;
    }
  });
}
```

Figure 5.26: Code segment of income analysis

6.1 Limitation

Although the majority function of the proposed application has been successfully developed and accomplished the objectives, this application has some limitations. The proposed application need more easy to use such as adding some characters reader which can scan and read the information from bill or receipt. However, in this development cannot make it which add a character reader.

Moreover, the application can add more type of filter data function. For the example, can let user to search a date range by pick the start date and end date. Moreover, it can make a sync function with bank or e-wallet to get the expenses and income.

6.2 Future work

In the future, the filter function should be improved to make the user can read the record in more pattern. This will let people know a date range of cashflow. Furthermore, the character reader API into the application which make the app more ease to use.

Acknowledgement

The authors would like to thank the Faculty of Computer Science and Information Technology, Universiti Tun Hussein Onn Malaysia for its support in every aspect.

References

[1] Bimal Bhatt. (2011). Financial Management Importance. Retrieved on 21 Dec 2021, from http://www.blognbuzz.com/financial-management-importance.html.

[2] Employees Provident Fund. (2016). 2015 Annual report of the employee's provident fund in Malaysia. Retrieved on 21 Dec 2021, from http://www.kwsp.gov.my/portal/en/aboutPdf_Folio:258 epf/annual-report/laporan-tahunan -2015.

[3] Kamal Halili Hassan, Fariza Ahmad, Rahim, Rohani Abdul Rahim, Tengku Noor Azira Tengku Zainuddin & Rooshida Merican A R Merican. "A review of Malaysia's private sector services retirement and pension system". The GAI International Academic Conferences. Turkey, Istanbul: Journal of GAI Business & Economics, pp 190-196, June 2015.

[4] (2021). Realbyte Inc. (Version 4.5.22) [Money Manager]. Retrieved from Google Play Store. https://play.google.com/store/apps/details?id=com.realbyteapps.moneymanagerfree.

[5] (2021). PixelRush (Version 2.5.1) [1Money]. Retrieved from Google Play Store. https://play.google.com/store/apps/details?id=org.pixelrush.moneyiq.

[6] (2021). BudgetBankers.com (Version 8.2.231) [Wallet]. Retrieved from Google Play Store. https://play.google.com/store/apps/details?id=com.droid4you.application.wallet.

[7] Oscar Bosman. Testing and iterative development adapting the Booch method. In Proceeding, 13th International Conference and Exposition on Testing Computer Software. Silver Spring, Md.: USPDI, June 1996.

[8] Grady Booch, Object oriented design with applications. Redwood City, Calif.: Bejamin/Cummings, 1991.