Applied Information Technology And Computer Science Vol. 4 No. 1 (2023) 1558-1568

AITCS

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

Budget Planning for Student App

Nabil Izzat Mazlan[1], Norhamreeza Abdul Hamid[1]*,

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

*Corresponding Author Designation

DOI: https://doi.org/10.30880/aitcs.2023.04.01.093
Received 16 June 2022; Accepted 12 June 2023; Available online 30 June 2023

Abstract: Budgetify is a budget planning application for students. The purpose is to help students manage their financial better as one of common and crucial problem among students. The main features of the proposed application are to track daily spending and saving money. Moreover, the application is easy and simple to use as it develops using prototyping methodology for achieve that best user experience. The project was using Figma for design prototype, Visual Studio Code as code editor, and Flutter as framework. The application is expected to help students manage their financial better and help them focus more on studying than stressing on financial problems.

Keywords: Application, Budget planning, Students

1. Introduction

Money management is a thing that all people has been thought to do since they were kids. Even though people were thought to use money wisely and manage it well, the real challenge for them to manage their spending is when they are become a student because they need to manage their own spending. Consequently, people will learn how to self-control, responsibility, dedication, making decision, and have a great life management in daily life. Besides, good financial management does not only help to increase the saving, it leads to a happier life as people will have money for emergencies, education, stress reduction things, helping others, and most importantly minimizing financial risk [1].

Most of the student nowadays do not a have good budget plan. A classic way of financial planning by a student that is by manually writing in a book or mentally plan of their budget. By mentally plan, meaning that the student just briefly plans it in their mind without having a proper structure plan. These methods are not efficient and accurate for a student to manage their spending.

However, with these classic ways, student needs to make an extra effort to always keeps track on their daily spending such as needs always carry around their budget planning book, and if they forget to bring it, they need to memorize all their daily spending to record it. Moreover, the student easily forgets from time to time. Hence, it will be hard for a student to plan their budget proficiently. Furthermore, with no proper budget planning and tracking, student will continue spend their money carelessly since they do not know the prioritize to spend the money.

*Corresponding author:norhamreeza@uthm.edu.my

Thus, this proposed project, can help to overcome those problems. Firstly, the application will be simple for student to use it. Next, with the help of application student can easily plan their spending budget in which category they want to prioritize spending more. In addition, with accurate statistic shown in the application, student will have more will to fight their urge to spend more as they saw how much money they already spend. Consequently, with the proposed application, student will have more proper and efficient budget planning to help them have better financial management.

2. Related Work

2.1 Overview of student financial planning

Bad financial planning become crucial among university students. This is vital issue especially for student with loans and scholarship. It also affects students that come from underprivileged or low-income families [2]. Based on the study of financial status and pressure among students, statistics among 255 respondents stated that 23.5% did not create a saving account, 30.5% of them did not take note of their spending and income, and 33.7% do not know how to manage their financial properly [3]. To support this study, another study said that There is a disparity between student income and expenditure. The majority of respondents experienced financial difficulties as a result of low financial resources and poor budgeting [4].

2.2 Comparison with Existing Systems

All three current existing applications will be compared with the project application. The comparison will be based on feature, functionality, and user friendliness. The comparison will be shown in a Table 1 below.

Table 1: System's Comparison

Modul	Application			
	Monify	Money Cats	Spendee	Budgetify
Login	X	√	√	√
Analytics Data	√	X	√	√
Budget/Ledger Module	X	√	√	√
User friendliness	√	X	√	√
Recurrence Spending/savings	√	X	√	√

3. Methodology/Framework

The prototyping model is a systems development process in which a prototype is produced, tested, and then tweaked until an acceptable output is obtained from which the entire system or product may be developed [5]. The reason for choosing prototyping model as the methodology is because the project will be built entirely for a user and this model main purposes is to satisfy the user's need. The prototyping model will work close with the user as developers will create a prototype and give it to the client for review. Following the review, the client proposes changes to the prototype. The recommended changes are subsequently included into the prototype, which is ultimately delivered to the customer for review [6]

Prototyping model is divided into six phases. The first phase is Requirements which is to gather information and do an analysis on it. The second phase is Quick Design which to create a simple design of the project. Next, third phase is to create a prototype. The fourth phase is to get user evaluation based on the prototype and based on it come fifth phase which is refining the prototype, fourth and fifth phase will be repeated until user is satisfied. Lastly, is implement and maintain.

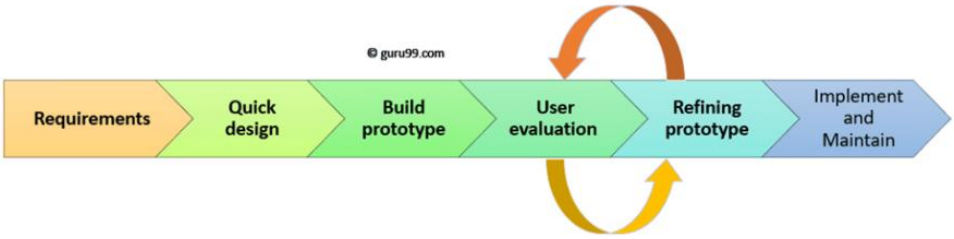


Figure 1: Prototyping Model [7]

3.1.1 Requirements Phase

This phase objective is to gather information on the project that going to be develop in terms of analysis of past alike project, and system users will be asked to determine what they anticipate from the system. Based on the analysis and requirements gathers, developers will get the basic idea of the project where problems background, scope, objective, and expected result is already decided.

3.1.2 Quick Design Phase

In this phase is where brief idea from requirement phase is brought to live. However, it is not a finished design. It provides only the important aspects such as input, output, and focuses on the characteristics that are visible to the user rather than detailed plan. The rapid design will aid in the development of the prototype later on.

3.1.3 Build Prototype Phase

Based on all information gathers on previous phase, an actual prototype will be design. The prototype mainly will be focus on user's interest likes user interface, applications functionality, applications design, and so on. The reason of focusing on user interest is to maintain the speed and cost of prototype development.

3.1.4 User Evaluation Phase

Once the prototype is done, here come user evaluation phase. In this phase, the user will test the prototype and evaluate it. This way, strength and weakness can be found from a user perspective. Missing functionality and errors can be easily detected as user evaluate if they did not satisfy in certain module. As if they are many unsatisfied users, the prototype will be brought to next phase which is refining prototype together with the users comment and evaluation.

3.1.5 Refining Prototype Phase

Refining prototype phase is work close with user evaluation phase as these two works in a loop. Process of refining the current prototype to meets with all the user's comments and suggested modifications will be done in this phase. The new refining prototype then will be presented again to the user to see if it meets their needs and expectation on the application. As if the new prototype is already satisfied the user and there is no new problem detected. The final complete version will be built based on the approved refined prototype. If not, its need to cycle through the refining prototype phase again.

3.1.6 Implement and Maintain Phase

Once the prototype's goal has been met, it is discarded, and the software is built using different process models. The prototype's primary goal is to thoroughly grasp the customer's requirements. Because all of the criteria are now understood, the developers create the software and deliver it to the client, with the assumption that the generated software fits all of the customer's expectations. Then, routine maintenance is performed on the system to save downtime and prevent large-scale breakdowns.

3.2 System Requirement Analysis

The process of defining the expectations of users for an application that is to be created or altered is known as requirements analysis. It includes all of the tasks that are carried out in order to determine the demands of users. Requirement analysis is referring to the process of analyzing, documenting, validating, and managing software or system requirements. Which is divided into multiple section likes functional requirement and non-functional requirement, software and hardware requirement, and user requirement. below show table 2 for functional requirement and Table 3 for non-functional requirements.

Table 2: Functional requirement

Modules	**Functional requirement**
Signup	• User needs to insert unique username that different than others. • After signup will directed to login page
Login	• User could login using an already sign up account • After login will directed to profile page
Home feed	• Will show pinned savings module, daily, weekly, and monthly spending, and recent transaction • Can access insert savings and insert spending module • Can access details of pinned savings module
Profile	• Option to logout at profile page • User could update personal details on profile
New Saving module	• User choice to create saving module or not • Apps allow user to create multiple module
Savings folder	• Apps show list of available saving module • Allow user to pin one module to be shown at homepage • Can access module full details • Apps allow user to delete module
Insert savings	• Apps only allow show numeric keypad to insert amount • Apps validate if the input is valid • Apps will calculate cumulative of total savings for each module
Insert spending	• Apps only allow show numeric keypad to insert amount • Apps validate if the input is valid • Apps will calculate cumulative of total spending for day, weeks, and months

Table 3: Non-Functional Requirements

Requirements	
Performance	• The application should be able to use anytime
Operational	• The application should be user friendly • The application should be use on android • The application should be easily maintained and updated
Security	• User should able to keeps their data with login feature
Usability	• Users more satisfies with the efficiency and effectiveness system
Maintainability	• The application should be in maintained condition to increase application performance

3.2.1 Use case Diagram

Use case is one of the important diagrams needed for application development. Use cases describe a system's functional needs from the perspective of the end user, resulting in a goal-focused sequence of activities that users and developers can easily follow. A full use case will have one primary or fundamental flow as well as several variant flows. The alternate flow, also known as an extended use case, describes both typical variants on the basic flow and uncommon instances. As for Budgetify application, there is going to be one actor which is budgeter the user, Figure 3.2 will show the full Use Case Diagram for the application.

Figure 2: Use Case for Budgetify

3.2.2 Class Diagram

Class diagrams are important to UML. They are built on object orientation concepts and may be deployed at various stages of a project. They appear as the domain model during the analysis, attempting to produce a representation of reality. Class diagrams allow you to use UML to design models with attributes, relationships, actions, and intersections. A class diagram depicts the relationships between classes through aggregations and linkages, as well as the transmission of attributes and behavior between classes. In Figure 3.3 below shown the relationship between the classes for Budgetify application.

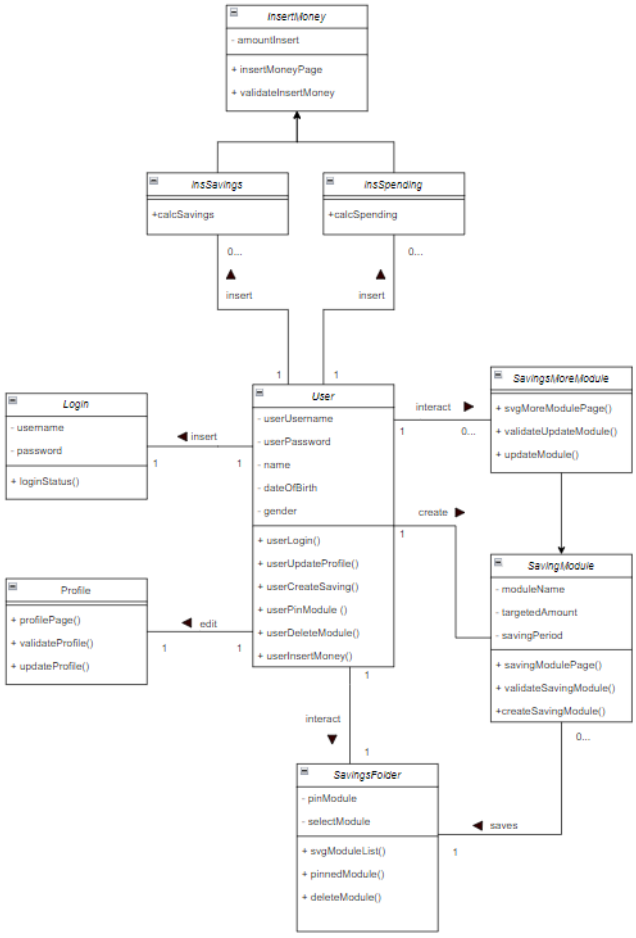


Figure 3: Class Diagram for Budgetify

3.3 Implementation phase

Through all the process and phase above, will come the implementation phase where all the plan and design will be fully developed, this is where front-end programming like interface will be link with back-end programming which is the database. After that, the user will be involved in using the Budgetify application where it is tested to see if it meets all the user requirements. This phase also includes testing, inspection, adjustment, correction, and certification of facilities and systems to guarantee project performance as required.

A few pages of interfaces are shown below where Figure 4 is interface of user login and the code segment is on Figure 7. Next, Figure 5 is an interface of insert spending module of user daily spending and the code segment is on Figure 8. Lastly, the last interface shown is an interface of user profile on Figure 6 and the code segment of it is on Figure 9.

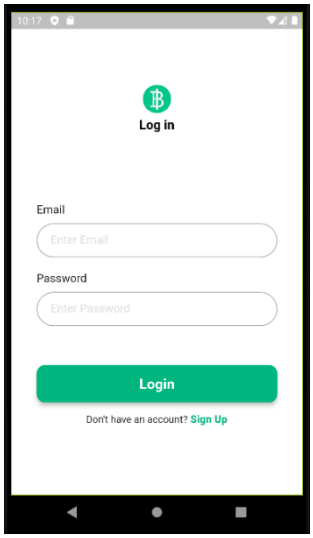


Figure 4: Login page

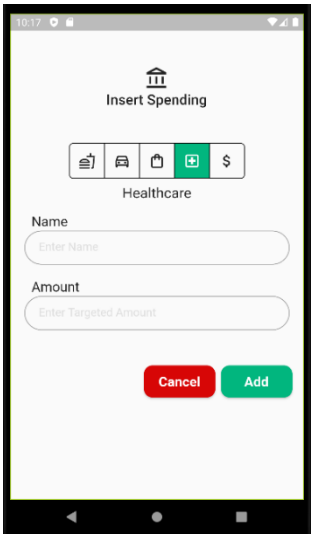


Figure 5: Insert spending page

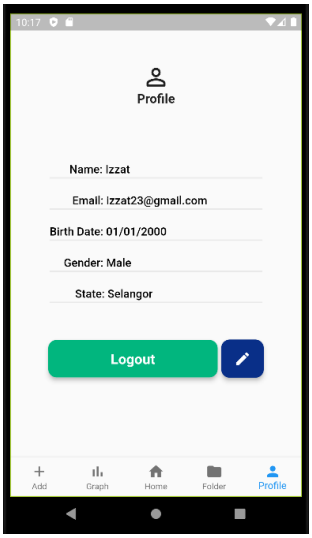
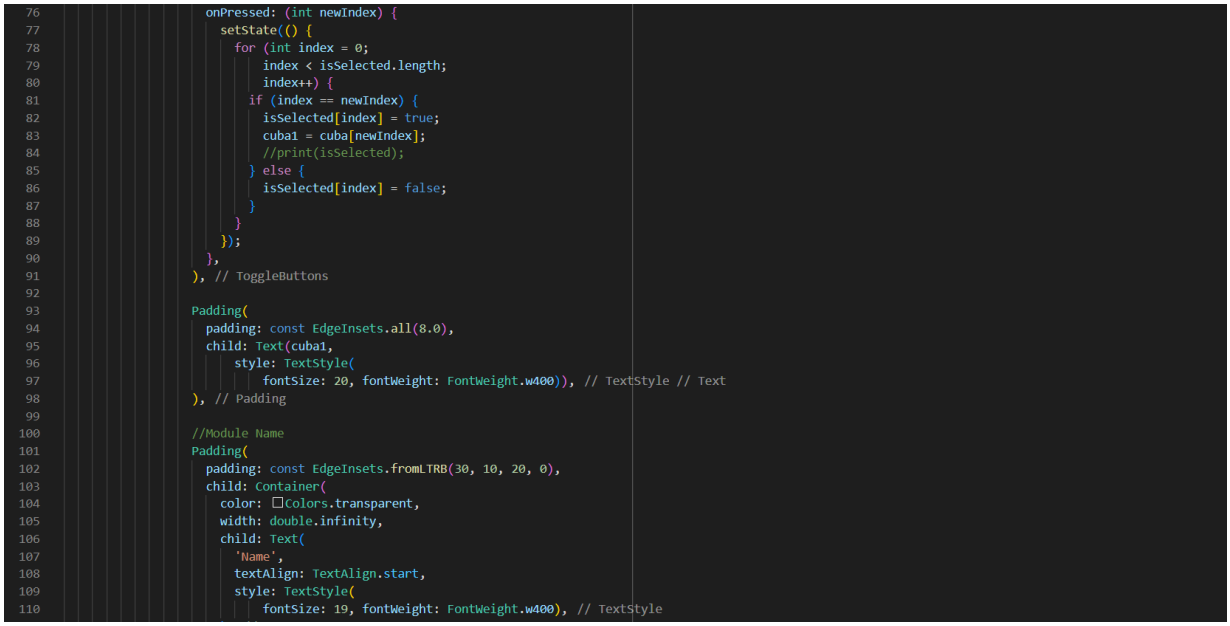


Figure 6: User profile page

```dart
@override
Widget build(BuildContext context) {
  final textLogin = Text(
    "Log in",
    style: TextStyle(
        color: Colors.black, fontWeight: FontWeight.bold, fontSize: 18), // TextStyle
  ); // Text
  //email field
  final emailField = TextFormField(
      autofocus: false,
      controller: emailController,
      keyboardType: TextInputType.emailAddress,
      validator: (value) {
        if (value!.isEmpty) {
          return ("Please Enter Your Email");
        }
        // reg expression for email validation
        if (!RegExp("^[a-zA-Z0-9+_.-]+@[a-zA-Z0-9.-]+.[a-z]")
            .hasMatch(value)) {
          return ("Please Enter a valid email");
        }
        return null;
      },
      onSaved: (value) {
        emailController.text = value!;
      },
      textInputAction: TextInputAction.next,
      decoration: InputDecoration(
        contentPadding: EdgeInsets.fromLTRB(20, 10, 20, 10),
        hintText: "Enter Email",
        hintStyle: TextStyle(color: Colors.grey[300]),
        border: OutlineInputBorder(
          borderRadius: BorderRadius.circular(58),
        ), // OutlineInputBorder
      )); // InputDecoration // TextFormField

```

Figure 7: Code segment for login page

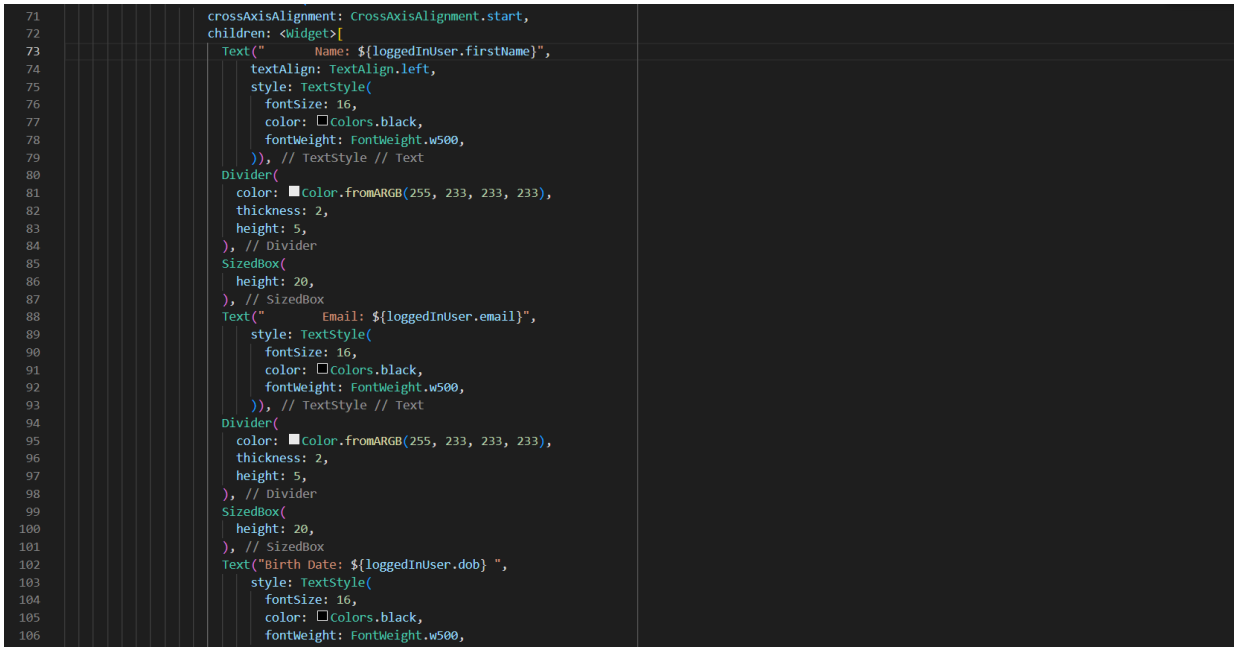


```
onPressed: (int newIndex) {
  setState(() {
    for (int index = 0;
        index < isSelected.length;
        index++) {
      if (index == newIndex) {
        isSelected[index] = true;
        cuba1 = cuba[newIndex];
        //print(isSelected);
      } else {
        isSelected[index] = false;
      }
    }
  });
},
), // ToggleButtons

Padding(
  padding: const EdgeInsets.all(8.0),
  child: Text(cuba1,
      style: TextStyle(
          fontSize: 20, fontWeight: FontWeight.w400)), // TextStyle // Text
), // Padding

//Module Name
Padding(
  padding: const EdgeInsets.fromLTRB(30, 10, 20, 0),
  child: Container(
    color: Colors.transparent,
    width: double.infinity,
    child: Text(
      'Name',
      textAlign: TextAlign.start,
      style: TextStyle(
          fontSize: 19, fontWeight: FontWeight.w400), // TextStyle
```

Figure 8: Code segment for insert spending page



```
crossAxisAlignment: CrossAxisAlignment.start,
children: <Widget>[
  Text("       Name: ${loggedInUser.firstName}",
      textAlign: TextAlign.left,
      style: TextStyle(
        fontSize: 16,
        color: Colors.black,
        fontWeight: FontWeight.w500,
      )), // TextStyle // Text
  Divider(
    color: Color.fromARGB(255, 233, 233, 233),
    thickness: 2,
    height: 5,
  ), // Divider
  SizedBox(
    height: 20,
  ), // SizedBox
  Text("        Email: ${loggedInUser.email}",
      style: TextStyle(
        fontSize: 16,
        color: Colors.black,
        fontWeight: FontWeight.w500,
      )), // TextStyle // Text
  Divider(
    color: Color.fromARGB(255, 233, 233, 233),
    thickness: 2,
    height: 5,
  ), // Divider
  SizedBox(
    height: 20,
  ), // SizedBox
  Text("Birth Date: ${loggedInUser.dob} ",
      style: TextStyle(
        fontSize: 16,
        color: Colors.black,
        fontWeight: FontWeight.w500,
```

Figure 9: Code segment for user profile

4. Results and Discussion

After user testing on the developed application, result and discussion is important to obtain feedback and find errors or flaws on the application. The user will test all part of the application to make sure every module is function as it should be. At this phase, testing could be for the apps to be tweak and adjust to meets the user requirement.

Functional testing is carried out to make sure that all module is function properly, functional testing can be done by using test case where it is a series of activities done on Budgetify application to assess whether or not it meets requirements and performs properly. Next, user acceptance testing is where feedback is obtained from users to see the satisfactions of the user towards the Budgetify

application. After user used the application, a set of questionnaires is distributed to see whether user satisfied and accept the application or not.

Table 4: Test Case

No	Description	Result
1	New user allows to register a new account	Success
2	User with an account can login	Success
3	All important modules shown on homepage	Success
4	User could update profile details and the details will display on their profile	Success
5	Saving module can be created	Success
6	A list of create saving module is shown on savings folder	Success
7	Savings module could be enlarged to see the details	Success
8	Insert savings module to insert and record saving	Success
9	Insert spending module to insert and record spending	Success

5. Conclusion

To conclude, the Budgetify application is developed to help students track and manage their spending and saving. The application managed to achieve all the project objective stated. Besides that, the main expected result and project significance to help student with their money management and gives benefits to them is also achieved in this project. However, this application is not on perfect form and still has more room for improvement later on. Despite the hindrance, this project has achieved their main objective and it was a success.

Acknowledgment

The authors would like to thank the Faculty of Computer Science and Information Technology, Universiti Tun Hussein Onn Malaysia

Appendix A

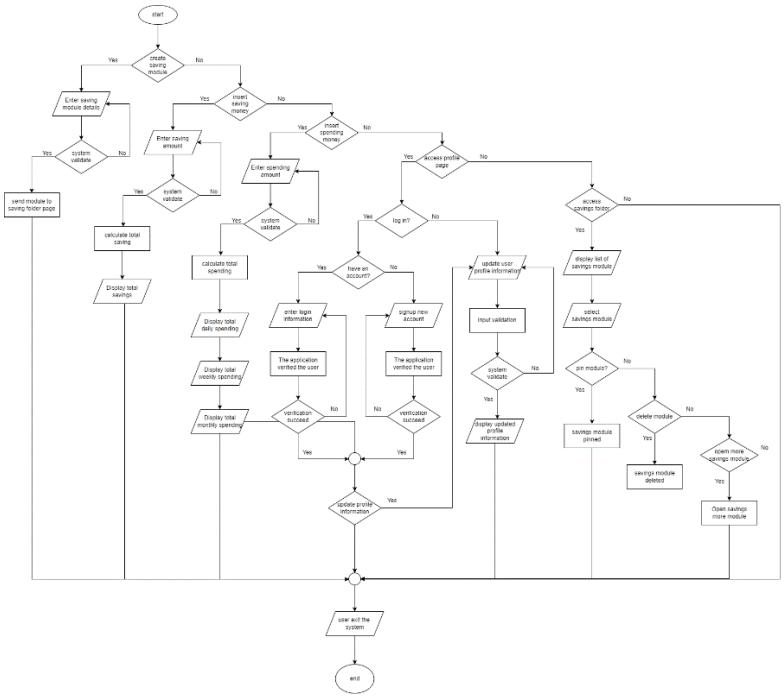


Figure A: User Flowchart for Budget Planning for Student

Appendix B

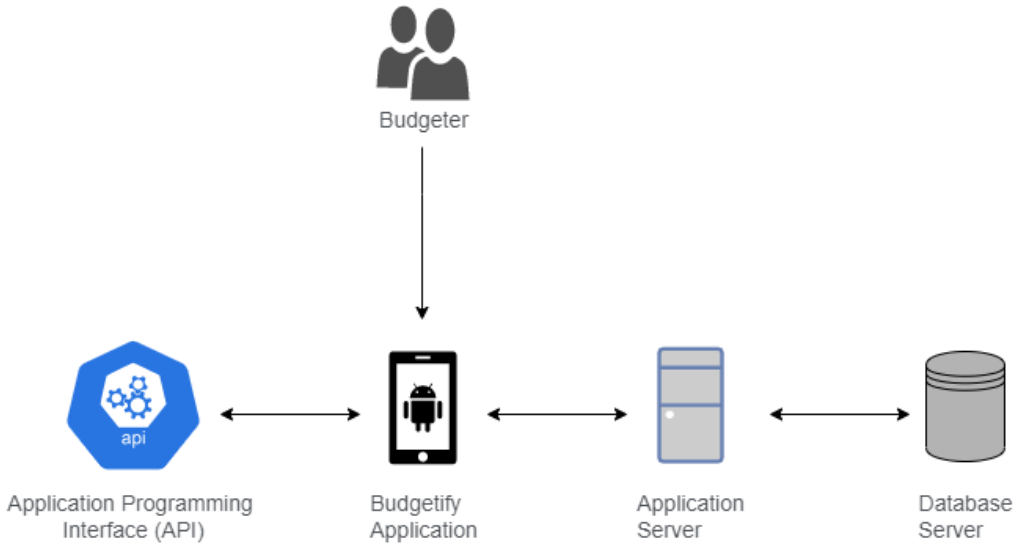


Figure B: System Architecture

References

[1] Agarwal, N. *How adopting a savings culture can minimise stress*. [online] Arabian Business. Available at: <https://www.arabianbusiness.com/opinion/comment/469801-how-adopting-savings-culture-can-minimise-stress> 2021

[2] NORAZLAN, N., YUSUF, S., & MOHAMED HAMOUD AL-MAJDHOUB, F. A. T. I. M. A. The financial problems and academic performance among public university students in Malaysia. The Asian Journal of Professional and Business Studies, 2(2). 2020

[3] Mod Asri, N., Abu Bakar, N., & Saad, S. Status kewangan dan tekanan dalam kalangan mahasiswa. *Jurnal Pengguna Makaysia*, 63-83. 2017

[4] Daud, N., Norwani, N. M., & Yusof, R. Students Financial Problems in Higher Education Institutions. International Journal of Academic Research in Business and Social Sciences, 8(10), 1158–1565. 2018

[5] Lewis, S. What is the Prototyping Model?. [online] SearchCIO. Available at: <https://searchcio.techtarget.com/definition/Prototyping-Model> 2021

[6] T, N. What is Prototyping Model? Phases, Types, Advantages & Disadvantages - Binary Terms. [online] Binary Terms. Available at: <https://binaryterms.com/prototyping-model.html> 2021

[7] Martin, M. Prototyping Model in Software Engineering: Methodology, Process, Approach. [online] Guru99. Available at: <https://www.guru99.com/software-engineering-prototyping-model.html> 2021