



الجامعة العربية الأمريكية
ARAB AMERICAN UNIVERSITY

Arab American University

Faculty of Engineering and Information Technology

Computer Systems Engineering

SENIOR PROJECT II

Smart Money Bank with A Mobile Application

By:

Abdallah Mohammed Lafi Odeh.....201610931..... CSE
Hamad Numan Abdullah Mohsen.....201712137..... CSE
Mohanad Fayyad Sobhi Atrash.....201712240..... CSE

Supervisor: Dr. Sami Awad

Feb.2022

Students Statement

We, the undersigned students, certify and confirm that the work submitted in this project report is entirely our own and has not been copied from any other source. Any material that has been used from other sources has been properly cited and acknowledged in the report.

We are fully aware that any copying or improper citation of references/sources used in this report will be considered plagiarism, which is a clear violation of the Code of Ethics of the Arab American University.

Abdallah Odeh

(201610931)

Hamad Mohsen

(201712137)

Mohand Atrash

(201712240)

Supervised by: *Dr. Sami Awad*

**Computer Systems Engineering Dept.
Submitted in partial fulfillment of the requirements of B.Sc. Degree in
Computer Systems Engineering**

Feb.2022

Supervisor Certification

This to certify that the work presented in this senior year project manuscript was carried out under my supervision, which is entitled:

“SDP Title Goes Here”

First Name Last Name (ID)

First Name Last Name (ID)

First Name Last Name (ID)

First Name Last Name (ID)

I hereby that the aforementioned students have successfully finished their senior year project and by submitting this report they have fulfilled in partial the requirements of B.Sc. Degree in ____ Engineering.

I also, hereby that I have **read, reviewed, and corrected the technical content** of this report and I believe that it is adequate in scope, quality and content and it is in alignment with the ABET requirements and the department guidelines.

Dr. Sami Awad

ACKNOWLEDGMENT

We would like to express our special thanks and sincere gratitude to our first supporter, our teacher, Dr. Sami Awad who gave us the best support to do this wonderful project. we are also grateful to our parents, friends, and everyone who helped us and supported us, thank you from our hearts.

ABSTRACT

This project solves the problem of counting and recognizing the banknotes in a money bank in different currency types mainly ILS, JOD, and USD. Also, this project can display the result of converting the savings in a money bank to different currency types via a mobile application. Where every time a user enters a banknote into the money bank the system recognizes this banknote, and the mobile application is notified of the change in total savings, so it can convert the savings to another currency. The money bank and the mobile application work with each other to achieve the best user experience in using the **"Smart Money Bank with a Mobile Application"** system.

TABLE OF CONTENTS

LIST OF FIGURES	v
LIST OF TABLES.....	vi
LIST OF ABBREVIATIONS.....	vii
chapter 1: INTRODUCTION	1
1 . 1 P r o b l e m S t a t e m e n t a n d P u r p o s e	1
1.2 Project and Design Objectives	1
1.3 Intended Outcomes and Deliverables.....	1
1.4 Summary of Report Structure	1
chapter 2: BACKGROUND	3
2.1 Overview	3
2.2 Related Work	3
2.2.1 Bill Acceptor	3
2.2.1.1 Introduction	3
2.2.1.2 Methodology.....	4
2.2.1.3 Advantages.....	4
2.2.1.4 Disadvantages	4
2.2.2 Digital Piggy Bank.....	5
2.2.2.1 Introduction	5
2.2.2.2 Methodology.....	5
2.2.2.3 Advantages.....	5
2.2.2.4 Disadvantages	5
2.2.3 Arduino Currency Counter	6
2.2.3.1 Introduction	6
2.2.3.2 Methodology.....	6
2.2.3.3 Advantages.....	6
2.2.3.4 Disadvantages	6
2.3 Analysis of Related Works.....	7
chapter 3: METHODS AND MATERIALS.....	8
3 . 1 S y s t e m D e s i g n a n d C o m p o n e n t s	8

3.1.1 System architecture	8
3.1.1.1 Big Picture View	8
3.1.1.2 Zooming on the Money Bank.....	9
3.1.1.3 Zooming on the Mobile Application	10
3.1.2 user interface design.....	11
3.1.2.1 Application User Interface	11
3.1.2.2 Hardware User Interface.....	12
3.1.3 Hardware components	13
3.1.3.1 Needs Analysis	13
3.1.3.2 Two Ways to Implement the Money Bank	14
3.1.3.3 Components Description	15
3.1.4 database design	21
3.1.4.1 Production Database.....	21
3.1.4.2 Prototype Database	21
3.2 Design Specifications, Standards and Constraints	22
3.2.1 Specifications	22
3.2.1.1 Mobile Application Specifications:.....	22
3.2.1.2 Money Bank Specifications:	22
3.2.2 Standards	22
3.2.3 Constraints	22
3.3 Design Alternatives	23
3.3.1 ALTERNATIVE HARDWARE Design	23
3.3.2 Description of alternative components	26
3.3.2.1 Raspberry Pi Camera:.....	26
3.3.2.2 LCD 2004A 20X4 with I2C.....	27
3.3.2.3 Sensor HC-SR501.....	27
3.3.2.4 Servo Motor	27
3.4 System Analysis and Optimization	28
3.4.1 System Requirements	28
3.4.1.1 Mobile Application Requirements	28
3.4.1.2 Money Bank Requirements.....	28

3.4.1.3 Cloud server Requirements	29
3.4.1.4 API Requirements	30
3.4.2 User Requirements	30
3.4.3 Use Case Diagrams.....	31
3.4.3.1 Use case diagram for user as actor.....	31
3.4.3.2 Use case diagram for money bank as actor	32
3.4.3.3 Use case diagram for cloud server as actor	33
3.4.3.4 Use case diagram for mobile application as actor	33
3.4.3.5 Use case diagram for API as actor.....	34
3.4.4 Use Case Diagrams Description	34
3.4.4.1 The interactions that the user could do in the system	34
3.4.4.2 The interactions that the money bank could do in the system:	36
3.4.4.3 The interactions that the cloud server could do in the system:	37
3.4.4.4 The interactions that the application could do in the system:	38
3.4.4.5 The interactions that the API could do in the system:.....	39
3.4.5 Sequence Diagram	40
3.4.5.1 The possible sequential interactions between the user, the money bank, and the cloud server.	40
3.4.5.2 The possible sequential interactions between the user, the mobile application, the cloud server, and the API.....	41
3.5 Software Design	42
3.5.1 Software Design of Mobile application.....	42
3.5.1.1 Mobile Application User Interface Flowchart.....	42
3.5.1.2 Fetching the real-time values of currencies from the API	42
3.5.1.3 Change Base Currency	43
3.5.1.4 Currency Converter.....	44
3.5.2 Software Design of Hardware components.....	44
3.5.2.1 Materials of Designing the Software for the Hardware Components	44
3.5.2.2 Software Architecture Diagram of Hardware Components.....	46
3.5.2.3 Software Modules of Hardware Components	47
chapter 4: RESULTS AND DISCUSSIONS	51
4.1 Results.....	51

4.2 Discussions	51
chapter 5: Project Management	52
5.1 Tasks, Schedule, and Milestones	52
5.1.1 Tasks:	52
5.1.2 Schedule and Milestones:	53
5.2 Resources and Cost Management	54
5.2.1 Resources:	54
5.2.2 Cost:	54
5.3 Lessons Learned	55
chapter 6: Impact of the engineering solution	56
6.1 Economical, Societal and Global	56
6.2 Environmental and Ethical	56
6.3 Other Issues	56
chapter 7: CONCLUSIONS and RECOMMENDATIONS.....	57
7.1 Summary of Achievements of the Project Objectives	57
7.2 New Skills and Experiences Learnt.....	57
7.3 Recommendations for Future Work	57
REFERENCES	Error! Bookmark not defined.

LIST OF FIGURES

FIG. 1. BELL ACCEPTOR.....	3
FIG. 2. STRUCTER OF THE BILL ACCEPTOR.....	4
FIG. 3. DIGITAL PIGGY BANK.....	5
FIG. 4. ARDUINO CURRENCY COUNTER.....	6
FIG. 5. BIG PICTURE VIEW.....	8
FIG. 6. ZOOMING ON THE MONEY BANK.....	9
FIG. 7. ZOOMING ON THE MOBILE APPLICATION	10
FIG. 8. APPLICATION USER INTERFACE	11
FIG. 9. HARDWARE USER INTERFACE.....	12
FIG. 10. NEEDS ANALYSIS	13
FIG. 11. RASPBERRY PI.....	16
FIG. 12. SKYWIRE ADAPTER.....	17
FIG. 13. SUN FOUNDER	18
FIG. 14. ARDAC ELITE	18
FIG. 15. DOXIE GO SE.....	19
FIG. 16. PRODUCTION DATABASE.....	21
FIG. 17. PROTOTYPE DATABASE	21
FIG. 18. SYSTEM DESIGN WITHE SEPARATING APPROACH.....	23
FIG. 19. SYSTEM DESIGN WITH MERGING APPROACH	24
FIG. 20. NEW BIG PICTURE VIEW	25
FIG. 21. ALTERNATIVE HARDWARE DESIGN	26
FIG. 22. RASPBERRY PI CAMERA	26
FIG. 23. LCD 2004A 20X4 WITH I2C	27
FIG. 24. SENSOR HC-SR501.....	27
FIG. 25. SERVO MOTOR	27
FIG. 26. USE CASE DIAGRAM FOR USER AS ACTOR	31
FIG. 27. USE CASE DIAGRAM FOR MONEY BANK AS ACTOR.....	32
FIG. 28. USE CASE DIAGRAM FOR CLOUD SERVER AS ACTOR	33
FIG. 29. USE CASE DIAGRAM FOR MOBILE APPLICATION AS ACTOR	33
FIG. 30. USE CASE DIAGRAM FOR API AS ACTOR	34
FIG. 31. SEQUENCE DIAGRAM1	40
FIG. 32. SEQUENCE DIAGRAM2	41
FIG. 33. USER INTERFACE FLOWCHART	42
FIG. 34. SOFTWARE ARCHITECTURE DIAGRAM OF HARDWARE COMPONENTS	46
FIG. 35. SCHEDULE AND MILESTONES1	53
FIG. 36. SCHEDULE AND MILESTONES2	53
FIG. 37. SCHEDULE AND MILESTONES3	53

LIST OF TABLES

TABLE 1. ANALYSIS OF RELATED WORKS	7
TABLE 2. SEPARATING VS MERGING	14
TABLE 3. MAIN COMPONENTS	15
TABLE 4. EQUIPMENT AND CABLES	20
TABLE 5. ALTERNATIVE COMPONENTS	25
TABLE 6. ACCEPTABLE BANKNOTES	29
TABLE 7. USE CASE: LOG IN.....	34
TABLE 8. USE CASE: CREATE ACCOUNT.....	34
TABLE 9. USE CASE: SEE IN DETAIL THE TOTAL AMOUNT OF MONEY ON THE DISPLAY SCREEN.	35
TABLE 10. USE CASE: SEE IN DETAIL THE TOTAL AMOUNT OF MONEY ON THE MOBILE APPLICATION.	35
TABLE 11. USE CASE: SEE THE REAL-TIME VALUE OF TOTAL MONEY ON THE MOBILE APPLICATION.	35
TABLE 12. USE CASE: ENTER PAPER MONEY	35
TABLE 13. USE CASE: TURN ON RESET MODE.....	36
TABLE 14. USE CASE: RECEIVE PAPER MONEY FROM USER	36
TABLE 15. RECOGNIZE PAPER MONEY.....	36
TABLE 16. USE CASE: STORE THE MONEY	36
TABLE 17. USE CASE: DISPLAY TOTAL_ILS AND TOTAL_JOD ON THE DISPLAY SCREEN	37
TABLE 18. USE CASE: SEND DATA TO CLOUD SERVER.....	37
TABLE 19. USE CASE: RECEIVE DATA FROM THE MONEY BANK	37
TABLE 20. USE CASE: STORE THE TOTAL AMOUNT OF MONEY	37
TABLE 21. USE CASE: RECEIVE USER DATA FROM THE MOBILE APPLICATION	38
TABLE 22. USE CASE: STORE THE USERS' DATA.....	38
TABLE 23. USE CASE: RECEIVE LOGIN DATA FROM THE APPLICATION	38
TABLE 24. USE CASE: COMPARE TO VALIDATE LOGGING IN	38
TABLE 25. USE CASE: CREATE AN ACCOUNT AND SEND THE USER DATA TO THE CLOUD SERVER.....	38
TABLE 26. USE CASE: LOGIN	39
TABLE 27. USE CASE: DISPLAY TOTAL_ILS AND TOTAL_JOD.....	39
TABLE 28. USE CASE: DISPLAY THE REAL_TIME VALUE OF TOTAL AMOUNT.....	39
TABLE 29. USE CASE: RECEIVE HTTP REQUEST FROM THE APPLICATION	39
TABLE 30. USE CASE: SEND DATA TO THE APPLICATION.....	40

LIST OF ABBREVIATIONS

API.....	Application Programming Interface
ILS.....	Israeli Shekel
JOD.....	Jordanian Dinar
USD.....	United States Dollar
HTTP.....	Hypertext Transfer Protocol
LCD.....	Liquid Crystal Display
US.....	United States
App.....	Application
USB.....	Universal Serial Bus
RAM.....	Random Access Memory
HDMI.....	High-Definition Multimedia Interface
CPU.....	Central Processing Unit
GPU.....	Graphics Processing Unit
GB.....	Giga Byte
SOC.....	System On Chip
SIM.....	Subscriber Identity Module
DC.....	Direct Current
UART.....	Universal Asynchronous Receiver-Transmitter
cm.....	Centimeter
MP.....	Mega Pixels
ID.....	Identity
CSI.....	Camera Serial Interface
MIPI.....	Mobile Industry Processor Interface
UK.....	United Kingdom
DC.....	Direct Current
TTL.....	Transistor-Transistor Logic
PIR.....	Passive Infrared

CHAPTER 1: INTRODUCTION

1.1 Problem Statement and Purpose

There are a lot of people who want to save money. Nowadays, we know that there are a lot of money banks products in the markets with different shapes and qualities. But, if we look at the money banks that can count the money inside them, we will find that there is a scarcity in the markets. And if we find something, it will recognize coins not banknotes. So, our project is found to solve this problem. In addition, we are trying to connect the money bank to the mobile like everything else in this era.

1.2 Project and Design Objectives

Our project doesn't focus on the money bank's outer shape or its ability to be unbreakable. What we are trying to do is to make it more interesting by highlighting other interesting things like the connectivity between the money bank and the mobile application, and the ability to recognize different types and values of paper money. Also, our project doesn't focus on dealing with coins, but it focuses on a more difficult challenge which is dealing, counting, and presenting paper money.

1.3 Intended Outcomes and Deliverables

- A money bank that can recognize and count the banknotes.
- A mobile application that can display the savings of the money bank and the result of conversion the savings into another currency type.

1.4 Summary of Report Structure

The structure of the report is dividing the project system into five subsystems, which are the user, the mobile application, the money bank, the cloud server, and the API. These subsystems are connected with each other to perform the functionality of the project. That is being structured under the criteria of ABET standards 2022.

CHAPTER 2: BACKGROUND

2.1 Overview

Nowadays, when you are browsing the internet for a 'Money Bank' you note that most of the results are focused on the outer shape and the unbreak ability. If we think a little bit out of the box and ask a question of what makes the money bank smarter or more interesting. What about a money bank that can know the total amount of banknotes inside it? Or you, if you are far away from it, can know the amount of money inside it from your phone by a mobile application. For what degree of intelligence, the Money bank can reach? Can it recognize the banknotes? or even the type of banknotes 'USD, ILS, JOD, ...etc.'? What about the ability to display the correct real-time amount of money in any currency you want on your mobile?

2.2 Related Work

In this section, we are going to introduce three of the most important current technologies and existing systems that are related to our project, which are Bill Acceptor, Digital Piggy Bank, and Arduino Currency Counter. Then, we are going to analyze and compare them with each other.

2.2.1 BILL ACCEPTOR

2.2.1.1 Introduction

This project aims to deal with paper money. The main part of this project is the Bill Acceptor. It consists of three parts: the input side, the output side, and an image processing unit internally. It receives, recognizes, and then forwards the entered paper money to the output side by electronic wheels. It recognizes a few types of currencies and their values. It can be programmed by connecting external devices to have specific functions [1].



Fig. 1. Bill Acceptor

2.2.1.2 Methodology

The user enters a paper money into the Bill Acceptor device through its input side. Inside the Bill Acceptor there is an image processing unit that recognizes the entered paper money. Bill Acceptor is connected to an Arduino Chip and an LCD display unit. It is based on image processing that detects paper money by a sensor and compares it with the reserved ones. Through comparison, it could accept the entered cash or reject it if it's not defined in the system.

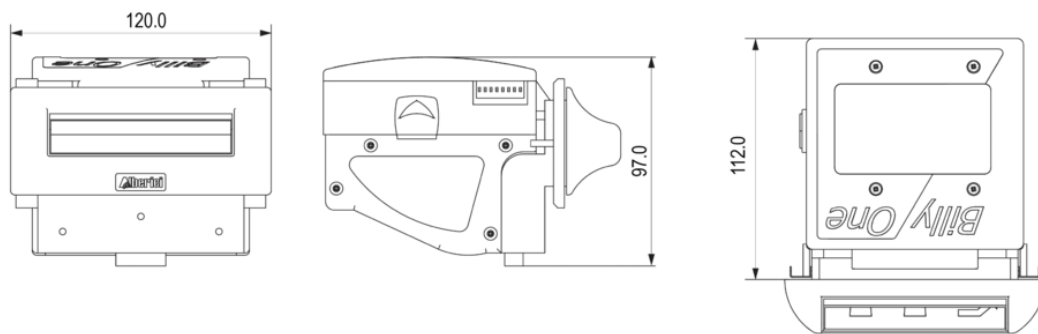


Fig. 2. Structer of the Bill Acceptor

2.2.1.3 Advantages

- The ability to recognize paper money.
- Reject unknown or undefined money.
- It could accept crimped or dirty paper money.

2.2.1.4 Disadvantages

- Performance is low.
- Does not connect to a mobile application.
- Doesn't recognize many currencies.

2.2.2 DIGITAL PIGGY BANK

2.2.2.1 Introduction

This is a good example of an electrical piggy bank. The main advantage of this product is its ability to count the coins and display them on an LCD screen. It accepts all US coins. Also, the manufacturer focused on the outer shape, quality, and the unbreak ability [2].



Fig. 3. Digital Piggy Bank

2.2.2.2 Methodology

The user enters a coin then the piggy bank recognizes the entered coin and displays the total amount on an LCD screen where the total amount equals the old total plus the value of the entered coin. If the user wants to withdraw coins, he must withdraw them manually and then report the withdrawal amount of coins. That's for record savings correctly. Also, there is an option to reset the counter and LCD screen, if the user wants to withdraw all the coins.

2.2.2.3 Advantages

- The ability to recognize US coins.
- Contains an LCD screen to display the total amount of coins.
- Contains functions that guarantee record savings correctly.

2.2.2.4 Disadvantages

- The inability to recognize paper money.
- The inability to recognize coins that's not US coins.

2.2.3 ARDUINO CURRENCY COUNTER

2.2.3.1 Introduction

This project consists of three main parts: an Arduino UNO Chip, LCD Display Unit, and TCS230 Color Sensor. The color sensor recognizes paper money and determines their value. The LCD display contains the initial value entered by the user [3].

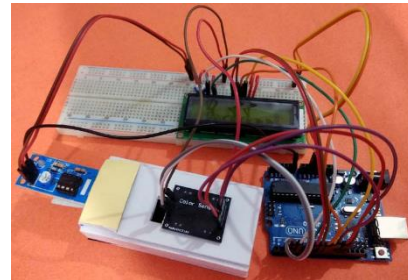


Fig. 4. Arduino Currency Counter

2.2.3.2 Methodology

This project works a little bit differently. The user defines the initial value. Then, he slides the paper money below the color sensor and it recognizes the paper money using infrared radiation, and then the Arduino chip performs a subtraction operation on the initial value from the value of paper money. Then, it sends the value of the paper money to the display unit.

2.2.3.3 Advantages

- The ability to recognize paper money by their colors and determines their values.
- The ease of implementing this project where it is based on infrared radiation to recognize the banknotes.

2.2.3.4 Disadvantages

- Inaccuracy in determining the value of paper currency.
- Errors occur in some readings because the color sensor might be affected by external lights. Therefore, the color sensor might incorrectly determine the paper money value.
- Unreliability. That is because the color sensor cannot validate the paper money correctly.

2.3 Analysis of Related Works

























	Bill Acceptor	Piggy Bank	Currency Counter	Our Project
Can it recognize paper money?				
connected to the internet?				
Contain a display screen?				
The system includes a mobile application				
Calculate the total money?				
Based on Image processing technology?				

Table 1. Analysis of Related Works

CHAPTER 3: METHODS AND MATERIALS

3.1 System Design and Components

3.1.1 SYSTEM ARCHITECTURE

3.1.1.1 Big Picture View

The following figure illustrates the big picture view of our system. The numbered elements are the actors in our system which are either subsystem or user. The endpoint of the arrows refers to the interactions that the element could do and with who. For example, the user can interact with the application and the application can interact with the user. while the cloud server cannot interact with the money bank even though the money bank can interact with the cloud server.

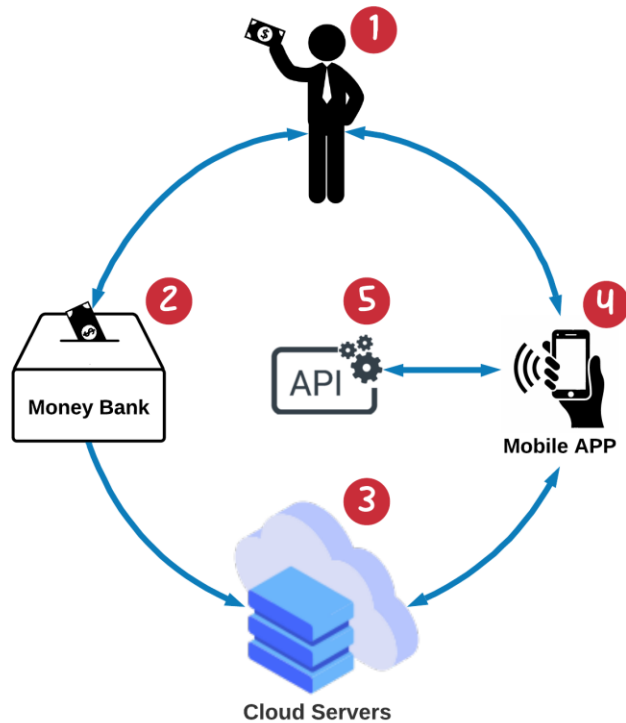


Fig. 5. Big Picture View

3.1.1.2 Zooming on the Money Bank

The following figure illustrates the structure of the internal components of the money bank and how they interact with each other and with the outside world. The money bank interacts with the user and with the cloud server outside its world. It receives from the user the banknotes, displays to him the total money inside it, and sends data to the cloud server. There are five main internal units in the money bank. The input and recognition unit, the storage unit, the process unit, the display unit, and the sending unit. The input and recognition unit receives the banknotes from the user to recognize them and send them to the storage unit. Also, it interacts with the process unit to recognize the entered paper money and then accept it or reject it. The process unit calculates the total amount of banknotes inside the money bank. Also, it interacts with the display unit and the sending unit. The process unit sends the total amount of the banknotes to the display unit to display them, and to the sending unit to send them to the cloud server.

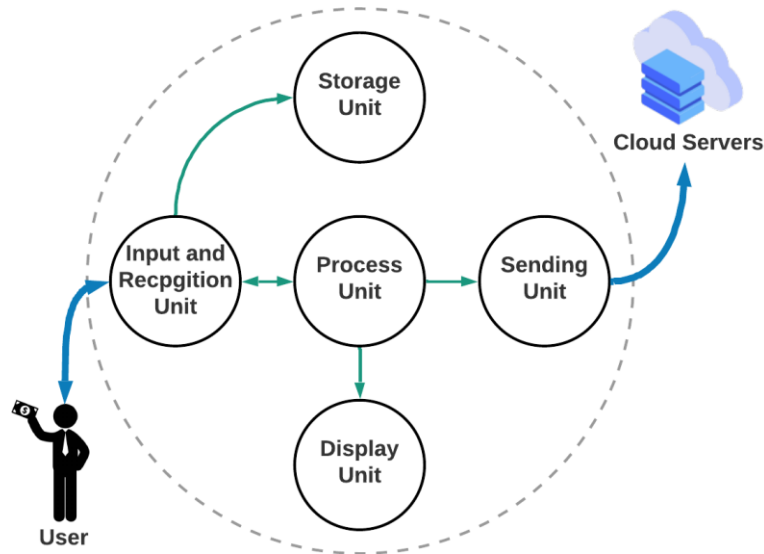


Fig. 6. Zooming on the Money Bank

3.1.1.3 Zooming on the Mobile Application

Any mobile application consists of two main layers: the frontend and the backend. The following figure illustrates our application's frontend and backend and how their components interact with each other and with the outside world.

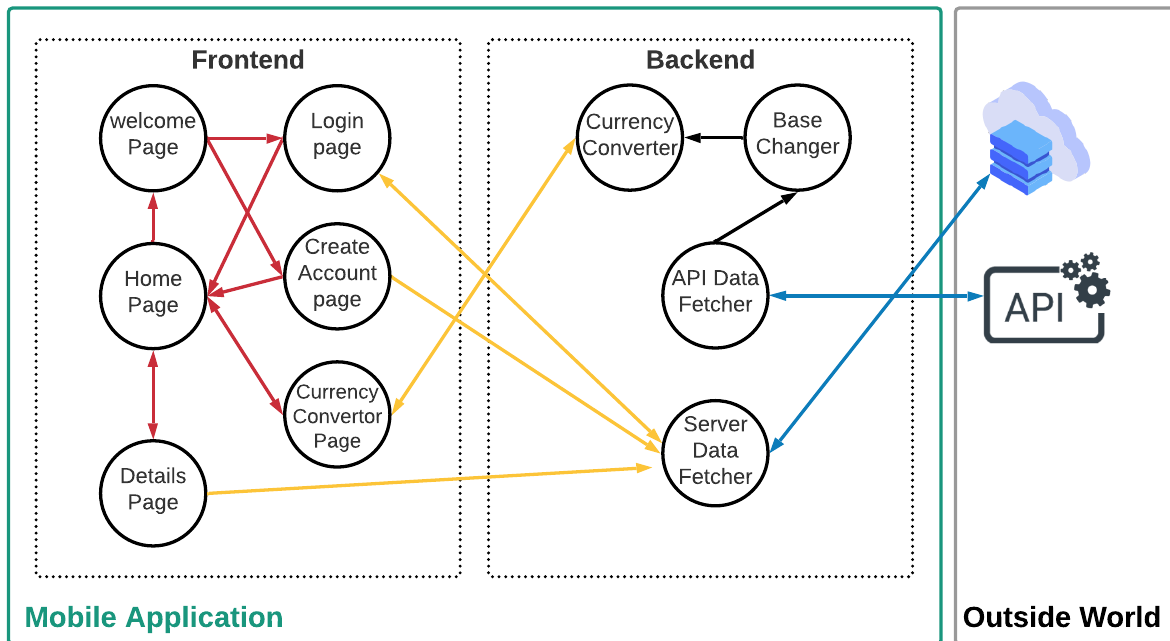


Fig. 7. Zooming on the Mobile Application

The green rectangle is the mobile application and the gray rectangle is the outside world. The circles in the frontend rectangle represent the application's pages and the circles in the backend rectangle represent the application's code functions. The red arrows and their endpoints represent the transition between the application's pages. In another word, represents for each page to which pages they can move. For example, the details page can move to the home page and vice versa. While the welcome page cannot move to the home page directly even though the home page can move to the welcome page directly. To move the welcome page to the home page, first it must move to the login page or the create account page then move to the home page. The yellow arrows and their endpoints represent the interactions between the frontend and backend components. For example, when the user successfully created an account in the create account page what happens is the entered information is sent to the backend and the backend is sent this information to the cloud server by a function called Server Data Fetcher. The two blue arrows and their endpoints represent the interactions between the backend components and the outside world.

3.1.2 USER INTERFACE DESIGN

3.1.2.1 Application User Interface

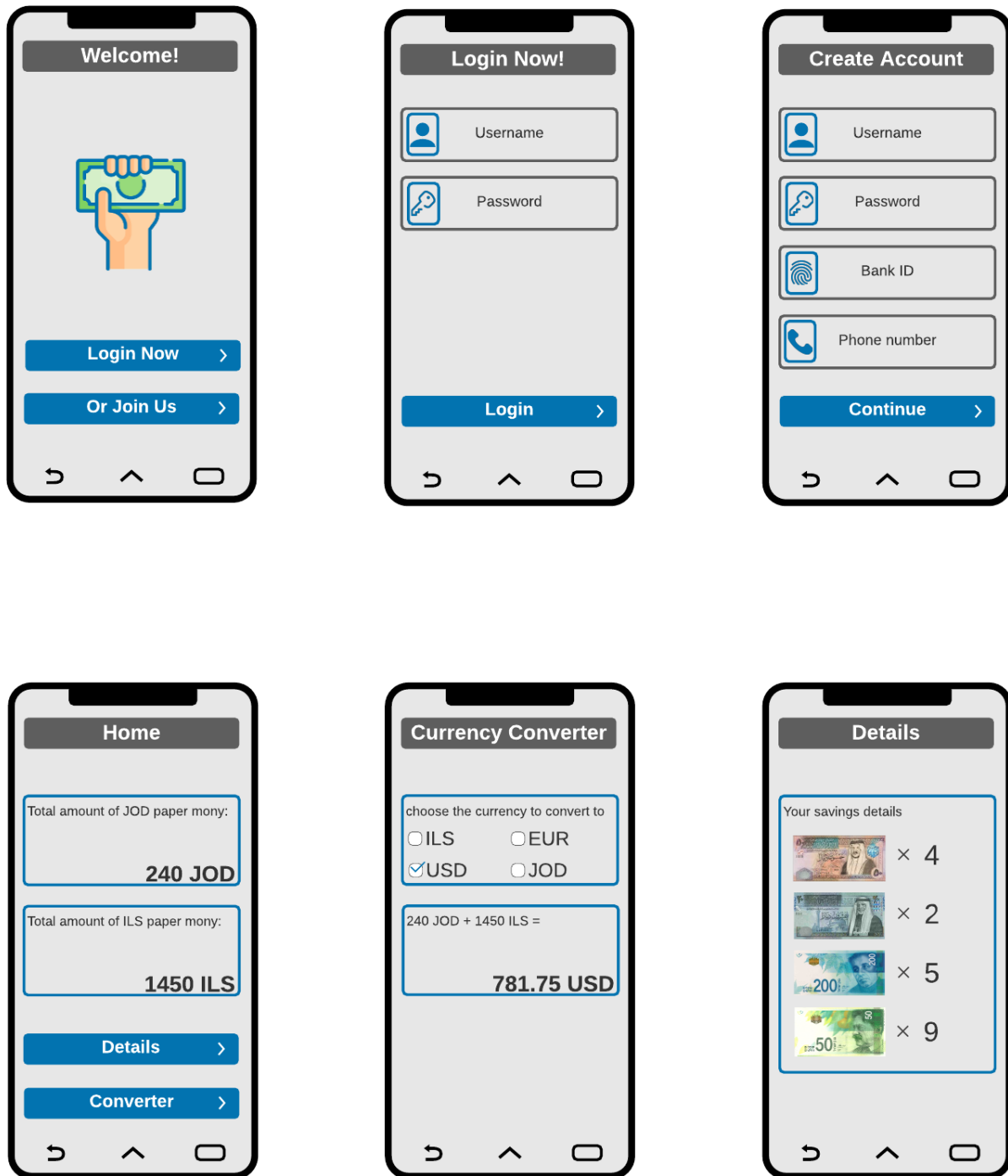


Fig. 8. Application User Interface

3.1.2.2 Hardware User Interface



Fig. 9. Hardware User Interface

3.1.3 HARDWARE COMPONENTS

In this section, we are going to talk about the hardware components that we need in our system. We start by highlighting our system's needs. Then, we talk about two various methods to implement our system. Finally, we analyze the components that our system needs.

3.1.3.1 Needs Analysis

Our system's hardware that we must care about is the money bank and its components. Now, if we look again at the money bank structure figure, we find that we need four main hardware components: A device that can receive, and recognize 'or scan' banknotes, a microcontroller to perform the processing operations that the system needs which maybe include recognize banknotes, a cellular modem to send the data to the cloud server, and a display device to display the money bank's information and status.

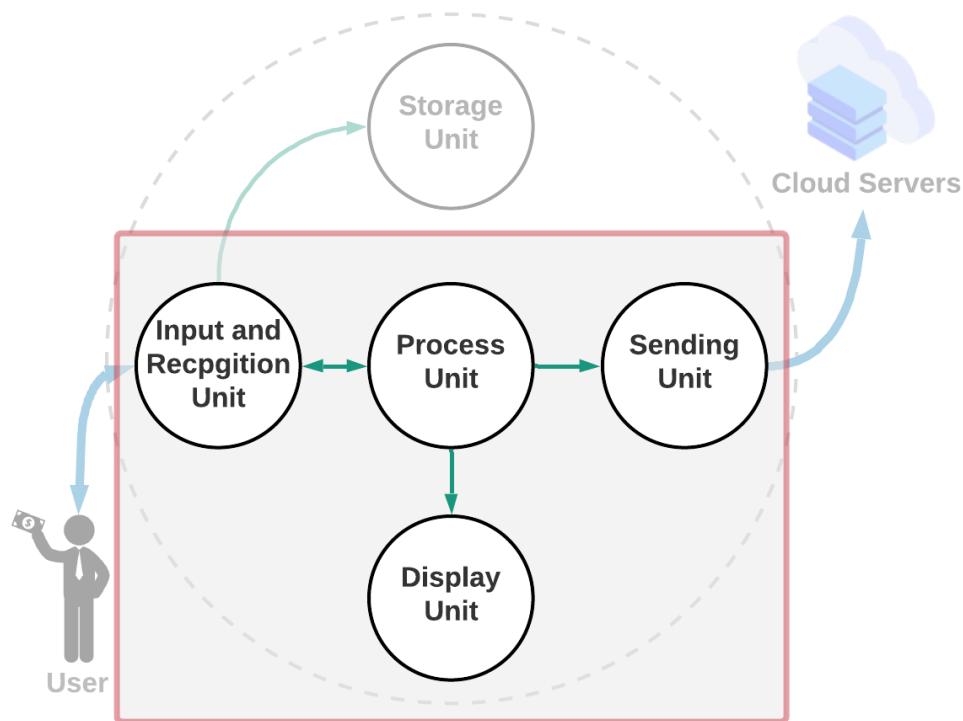


Fig. 10. Needs Analysis

3.1.3.2 Two Ways to Implement the Money Bank

We said in the previous paragraph that the input unit can be a recognition unit by itself or can be only a scanner and we said that the microcontroller maybe could be used to recognize banknotes. That's because we find that we can implement the money bank in two methods. The first method is by **separating** the recognition unit from the processing unit. In this case, the device that receives the banknotes recognizes them by itself and then reports the status of recognition to the microcontroller. The second method is by **merging** the recognition unit and the processing unit. In this case, the device that receives the banknotes will only scan them and then send the image to the processing unit to recognize it there. The following table shows the benefits and the disadvantages of the two implementation methods.

	Separating the recognition unit from the processing unit	Merging the recognition unit and the processing unit
We must program a recognition method based on image recognition technology	No	Yes
There is a limitation on the banknote's recognition methodology.	Yes	No
The system could have the ability to return invalid or undefined banknotes	Yes	Hard to make it

Table 2. Separating vs Merging

The hardware components differ in each method. The following table shows the four main components that we need in each method.






Components	Separation	Merging
Raspberry Pi Chip 	✓	✓
Skywire Adapter Cellular Modem 	✓	✓
Sun Founder LCD Display 	✓	✓
Ardac Elite Bill Acceptor 	✓	✗
Doxie Go SE Portable Scanner 	✗	✓

Table 3. Main Components

3.1.3.3 Components Description

In the previous table, the first three components are compatible with each other and designed to work with each other. In this section, we will describe them individually and then describe the last two components which are not compatible with the first three components. Therefore, we will mention the challenges that we face if we decide to implement the money bank in the Separation way or in the Merging way. Finally, we will mention some equipment and cables that we may use.

3.1.3.3.1 Raspberry Pi

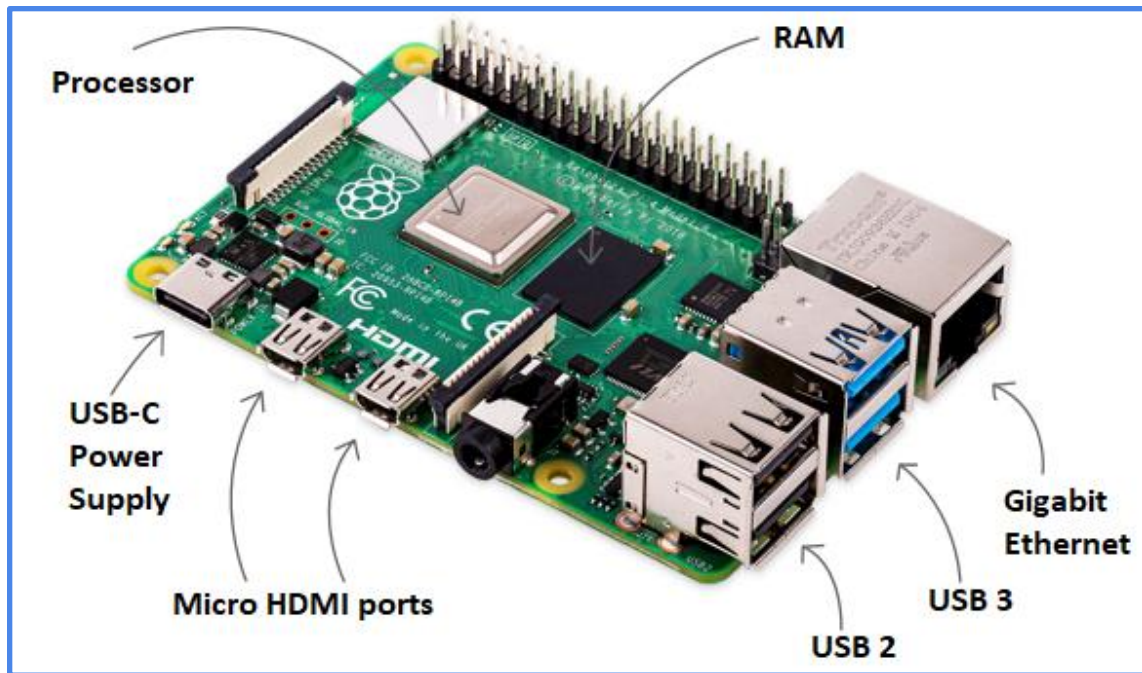


Fig. 11. Raspberry Pi

The Raspberry Pi is a SOC “System on Chip” not a microcontroller. The SOC is a single chip that can do more complex tasks from the microcontroller. It can contain many CPUs, GPUs, or even many Microcontrollers. In addition, the SOC differs from the Microcontroller in its ability to have an operating system on it [4]. The fourth edition of Raspberry Pi comes with two USB-2 ports, two USB-3 ports, one Gigabit-Ethernet port, one USB-C port, two Micro-HDMI ports, one processor, Raspberry Pi Operating System, and RAM with three choices 2GB, 4Gb, or 8GB. Also, Raspberry Pi supports the fifth generation of Bluetooth [5].

3.1.3.3.2 Skywire Adapter

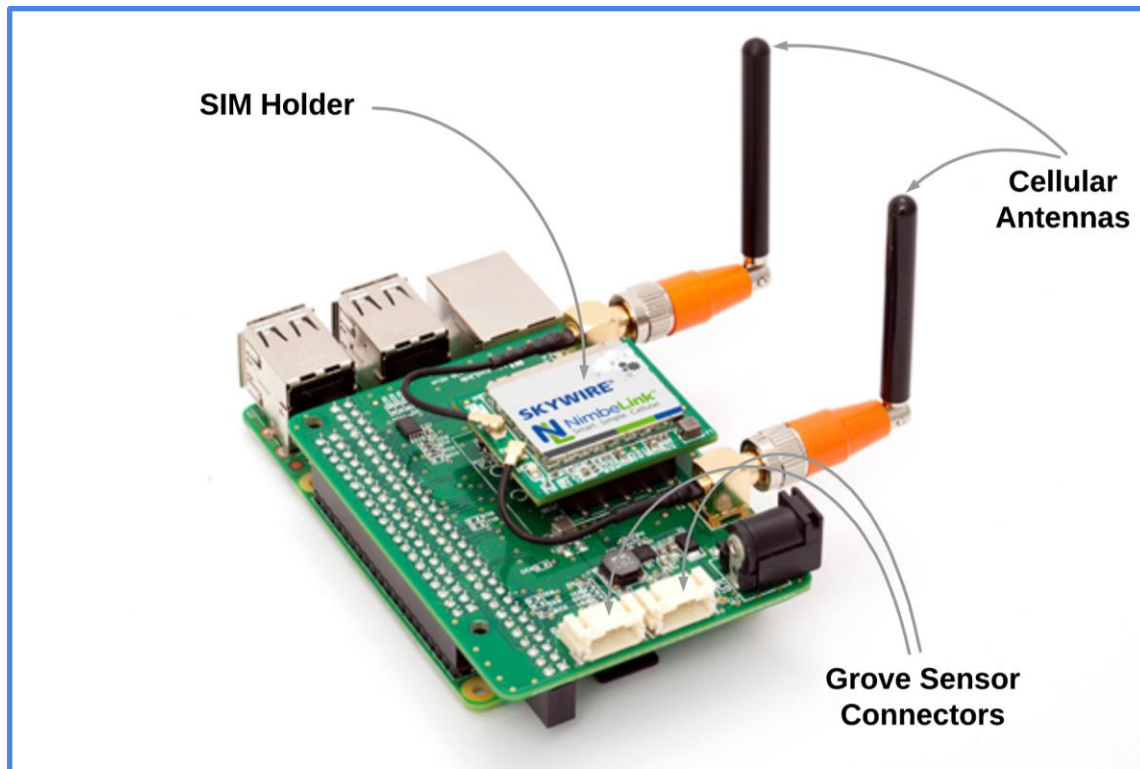


Fig. 12. Skywire Adapter

Mainly, our system depends on the internet. We can connect to the internet by a cellular network or Wi-Fi network. We prefer to connect to the internet via a cellular network and we are going to do this by connecting the Raspberry Pi to a cellular modem. Raspberry Pi Skywire Adapter is a cellular modem compatible with Raspberry Pi chip. Skywire Adapter contains two cellular antennas, 5V DC power input, and two grove sensor connectors. Also, it supports both UART and USB connections. And the most important thing is the application notes and code samples are available on the internet [6].

3.1.3.3 Sun Founder

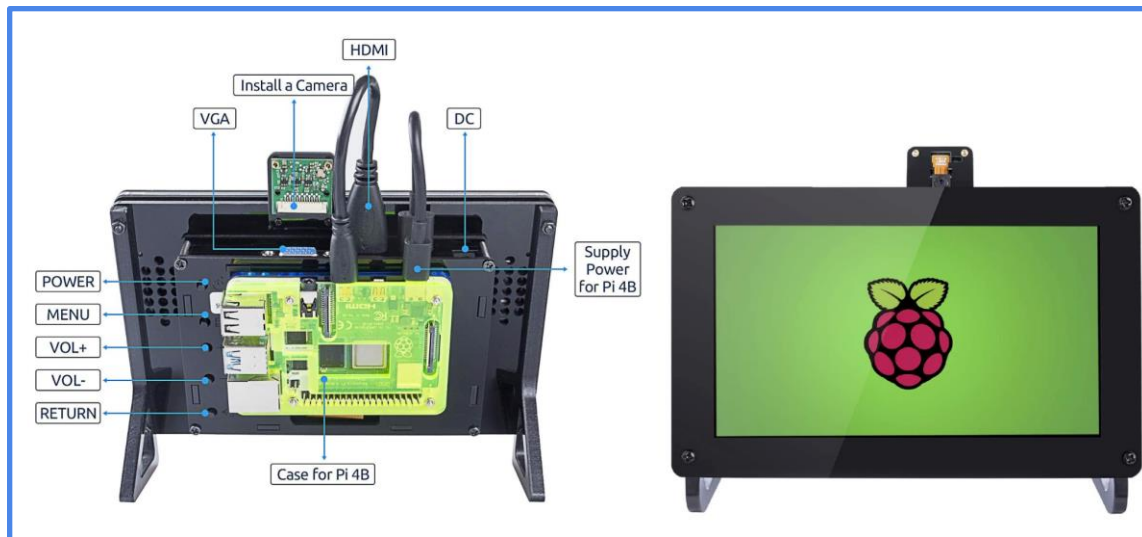


Fig. 13. Sun Founder

Sun Founder is a 7 Inch LCD monitor designed to be compatible with the Raspberry Pi chip. The backend of the Sun Founder contains a case to install the Raspberry Pi chip on it. The Sun founder connects to the Raspberry Pi chip through an HDMI cable and an USB-C cable. Sun Founder comes with a high resolution of 1024x600 pixels and five buttons to control it [7].

3.1.3.4 Ardac Elite

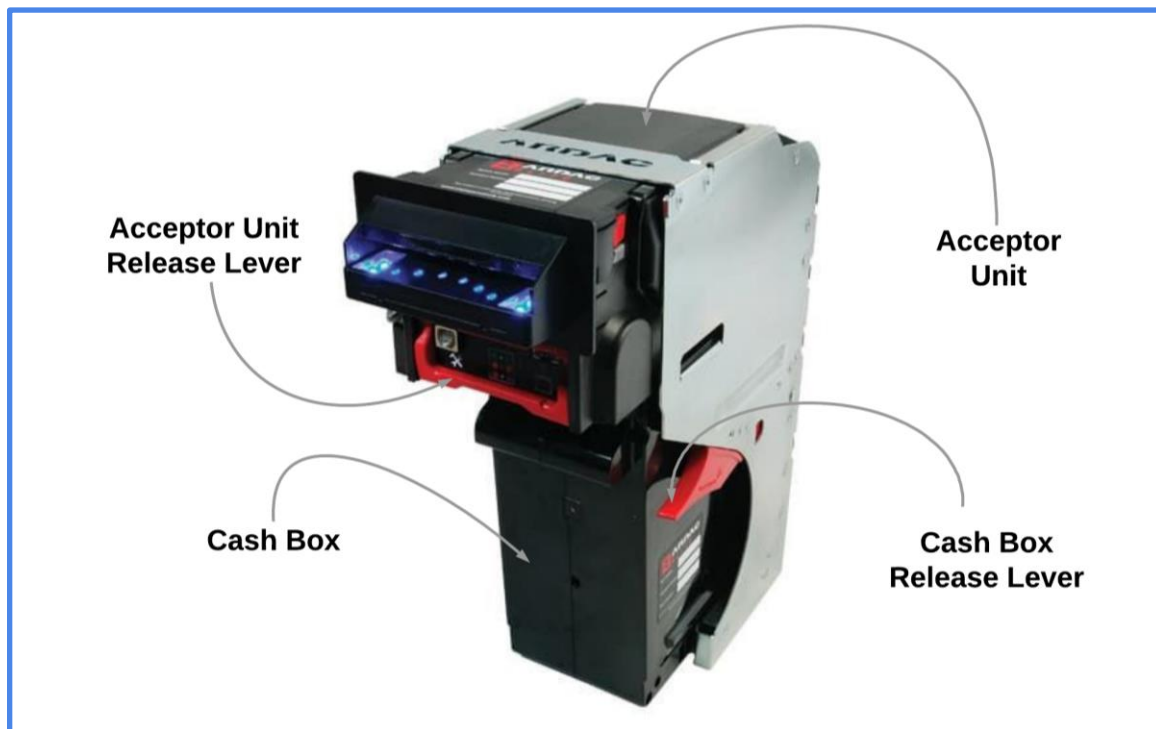


Fig. 14. Ardac Elite

Ardac Elite is a bill acceptor. It could accept any valid banknotes where it can accept 50 different currencies with an acceptance rate of 97%. It's easy to use and allows you to insert the banknotes at any angle and at any direction. And the most important thing is there is a technical manual describing in detail how to use and program Ardac Elite. The Ardac Elite consists of four main parts: the acceptor unit which recognizes the entered banknote and accepts or rejects it, the acceptor unit release lever, the cash box to save the accepted banknotes, and the cash box release lever [8].

If we decide to use the Ardac Elite bill acceptor we must know how to make a connection between the Ardac Elite and the Raspberry Pi chip. That's because the Ardac Elite must report the state of recognizing the entered banknote to the Raspberry Pi chip.

3.1.3.3.5 Doxie Go SE

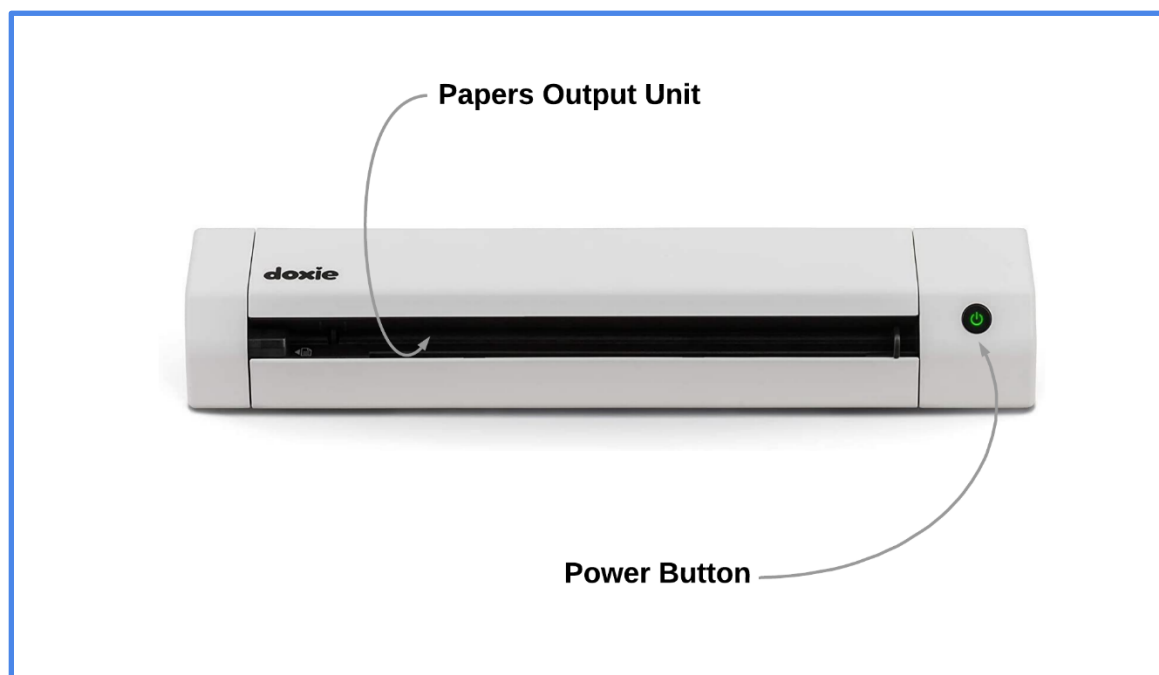


Fig. 15. Doxie Go SE

Doxie-Go-SE is a portable scanner designed to scan the A4 papers but it also can scan any paper with a size smaller than the A4 paper size. Doxie-Go-SE comes with a rechargeable battery with the ability to scan 400 papers per one charge. Doxie-Go-SE can work without a computer and can store up to 4000 papers [9].

If we decide to use the Doxie-Go-SE scanner we must program an intelligent method based on image recognition technology to recognize the banknote after receiving its image from the scanner. That takes us to another challenge which is we must know how to make a connection between the Doxie-Go-SE and the Raspberry Pi chip. That's to give the Doxie-Go-SE the ability to send the image of scanning the entered banknote, to the Raspberry Pi chip.

3.1.3.3.6 Equipment and Cables






HDMI to MICRO HDMI Cable	
USB-2 to USB-C Cable	
Ethernet Cable	
Connecting Wires	
Bred Board	

Table 4. Equipment and Cables

3.1.4 DATABASE DESIGN

3.1.4.1 Production Database

For the production, our database will only contain three entities: user, moneybank, and banknote. The user has three attributes username, password, and moneybank_id. The valid user should have one and only one moneybank. He's will get the moneybank_id when he buys the moneybank. The moneybank has four attributes moneybank_id, total_ils, total_usd, and total_jod. The moneybank could have one or many users and zero or many banknotes. The banknote has two attributes type and value. The banknote should have one and only one moneybank. The type of the banknote should be either ils, usd, or jod.

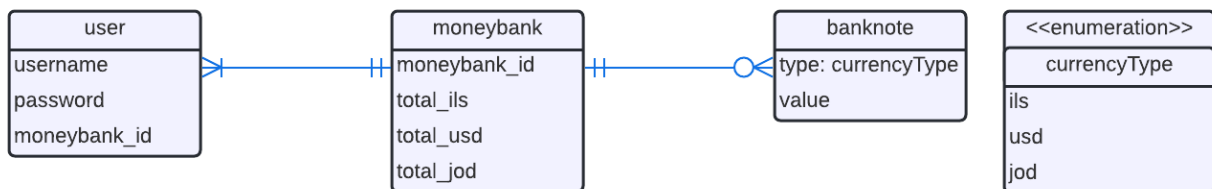


Fig. 16. Production Database

3.1.4.2 Prototype Database

For the prototype, we are going to use the same production database except that the user and the moneybank don't have the moneybank_id attribute. since we currently have only one moneybank, we don't need the moneybank_id attribute.

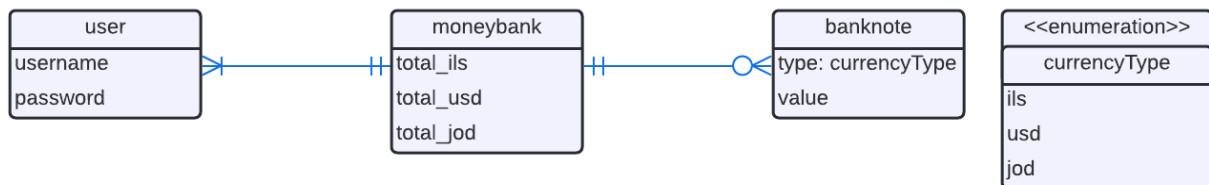


Fig. 17. Prototype Database

3.2 Design Specifications, Standards and Constraints

3.2.1 SPECIFICATIONS

3.2.1.1 Mobile Application Specifications:

- An application which consists of 6 pages, a welcome page, a login page, a signup page, a home page, a currency converter page, and a details page.
- It is on Google Play and programmed using Android.

3.2.1.2 Money Bank Specifications:

- A box of (25 cm, 25cm, 40cm) of wood, consists of 3 joint boxes.
- The upper box has the hardware components.
- The middle box has an entry area for paper money withdrawal.
- The Bottom box is a cache where money is stored.

3.2.2 STANDARDS

- Raspberry Pi: Raspberry Pi 4 model B 4 GB.
- Camera: Raspberry Pi Camera 5MP 1080P OV5647 CSI Webcam Camera Module With 15cm Cable for Raspberry Pi 4 Model B + Accessories.
- Sensor: HC-SR501 Adjust IR Pyroelectric Infrared PIR Motion Sensor Detector Module for Arduino for raspberry pi kits + Case.
- Servo Motor: Dexter Industries Servo Motor for Raspberry Pi (Set of 4 Small).
- LCD: LCD2004+I2C 2004 20x4 2004A Blue screen HD44780 Character LCD /w IIC/I2C Serial Interface Adapter Module for Raspberry Pi.

3.2.3 CONSTRAINTS

- We have replaced the **Ardac Elite (bill acceptor)** with a camera, a sensor, and a motor because it is very expensive and not available in the homeland.

- We have replaced the **Sun Founder collection** with an LCD because there is no need to display the whole system of Raspberry Pi, instead, the value of money is the only data to display.
- We have not used **Skywire Adapter**, instead, we have used an Ethernet cable connected directly with the Raspberry Pi.

3.3 Design Alternatives

3.3.1 ALTERNATIVE HARDWARE DESIGN

The following two images, illustrate the two approaches that we missioned in the previous sections to implement our project. The easiest way to implement our project is implementing it with the Separating approach since we don't need to care about how our system will recognize the banknotes, the bill acceptor does this. Unfortunately, we couldn't go with the Separating approach for two reasons. The first one, the bill acceptor is not available to buy it in our homeland. The second one, we couldn't find an obvious way to integrate the bill acceptor with raspberry pi.

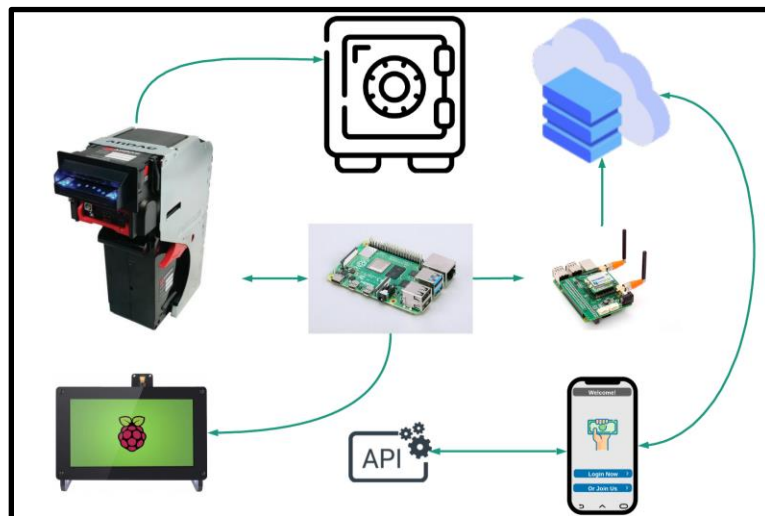


Fig. 18. System Design with the Separating Approach

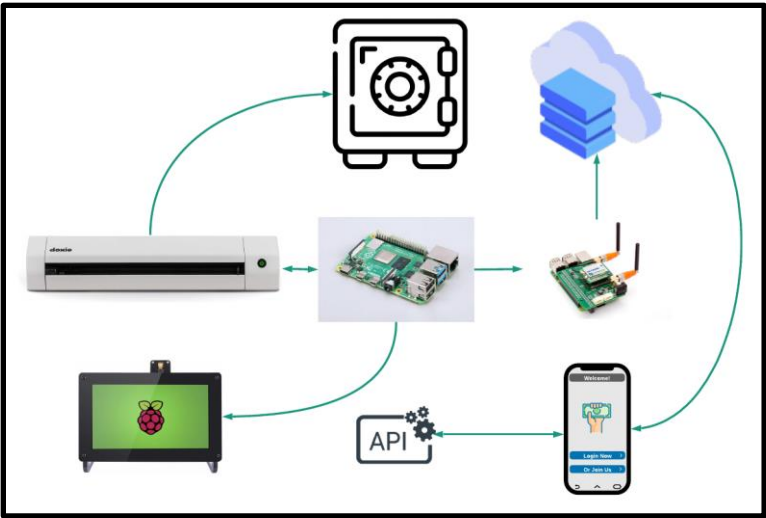






Fig. 19. System Design with Merging Approach

So, we implemented our system using the second approach which is the Merging approach. but we didn't use the portable scanner. Instead of that, we used the raspberry pi camera. which is designed to work with raspberry pi. also, we improved our system and add new features to it. The added features required us to add new hardware components. The following table shows the old components with their alternative and purpose.

New Components	Alternative of	Required for
		the feature of displaying the savings inside the money bank
		the feature of scanning the deposited banknote



motion sensor 		the feature of detecting the motion of deposit a banknote.
servo motor 		the feature of accepting and saving the money

Table 5. Alternative Components

The following figures illustrate the new big picture view and the alternative hardware design of our system.

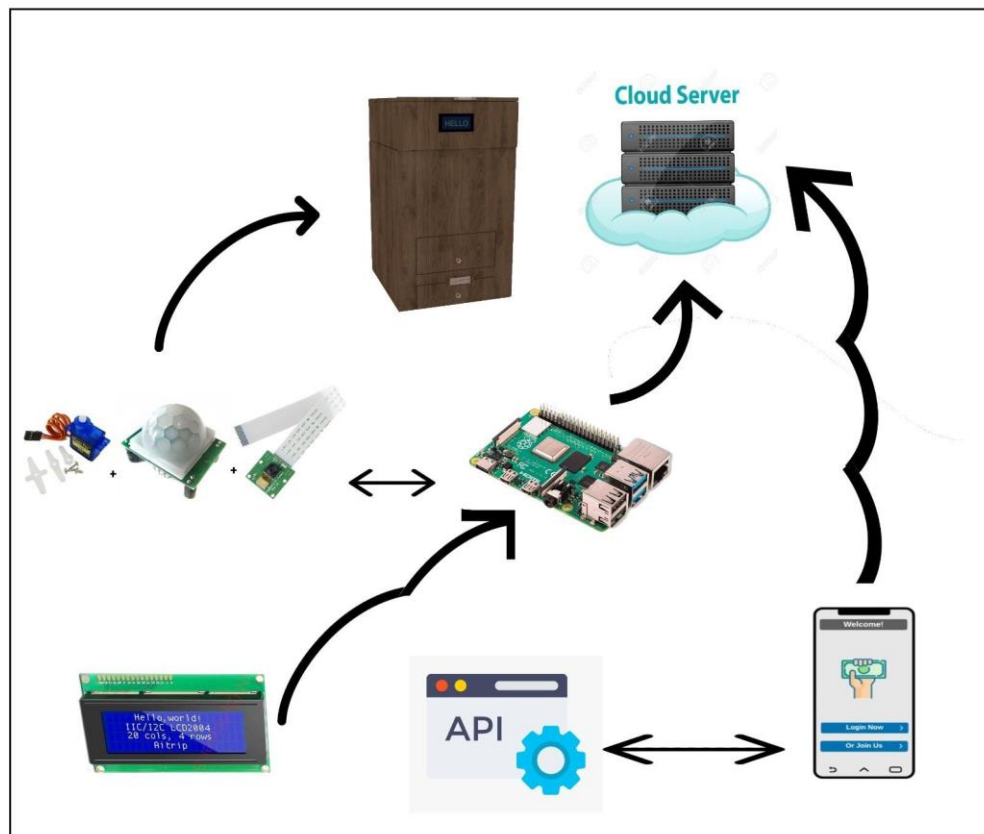


Fig. 20. New Big Picture View



Fig. 21. Alternative Hardware Design

3.3.2 DESCRIPTION OF ALTERNATIVE COMPONENTS

3.3.2.1 Raspberry Pi Camera:

The Raspberry Pi Camera Board plugs directly into the CSI connector on the Raspberry Pi. It's able to deliver a crystal clear 5MP resolution image, or 1080p HD video recording at 30fps! Latest Version 1.3! Custom designed and manufactured by the Raspberry Pi Foundation in the UK, the Raspberry Pi Camera Board features a 5MP (2592x1944 pixels) Omni vision 5647 sensor in a fixed focus module. The module attaches to Raspberry Pi, by way of a 15 Pin Ribbon Cable, to the dedicated 15-pin MIPI Camera Serial Interface (CSI), which was designed especially for interfacing to cameras.



Fig. 22. Raspberry Pi Camera

3.3.2.2 LCD 2004A 20X4 with I2C

This is a high-quality I2C LCD2004 display. It's in elegant blue with white contents displayed. It can display 4 rows with 20 characters for each. With IIC/I2C interface, it only takes two I/O ports thus saving more for other usages. You can adjust the contrast by the potentiometer at its back. If you don't want the backlight, you can also unplug the jumper cap at the LCD back. With this shield, you can display whatever you want by programming the Raspberry Pi board



Fig. 23. LCD 2004A 20X4 with I2C

3.3.2.3 Sensor HC-SR501

The PIR sensor stands for Passive Infrared sensor. It is a low-cost sensor which can detect the presence of Human beings or animals. This HC-SR501 PIR sensor module has three output pins Vcc, Output and Ground as shown in the pin diagram above. Since the output pin is 3.3V TTL logic it can be used with any platforms like Arduino and Raspberry Pi.



Fig. 24. Sensor HC-SR501

3.3.2.4 Servo Motor

Servo motor is a type of DC motor that, upon receiving a signal of a certain frequency, can rotate itself to any angle from 0-180 degrees. Its 90-degree position is generally referred to as a 'neutral' position, because it can rotate equally in either direction from that point.



Fig. 25. Servo Motor

3.4 System Analysis and Optimization

3.4.1 SYSTEM REQUIREMENTS

Our system consists of four main connected systems: The money bank, the mobile application, the cloud server, and the API. In this section, we are going to talk about their requirements individually.

3.4.1.1 Mobile Application Requirements

3.4.1.1.1 Mobile Application Functional Requirements:

- The mobile application should have the ability to send and receive data from the cloud server.
- The mobile application should have the ability to receive data from the API.
- The mobile application should have the ability to calculate the real-time value of the different types of money inside the money bank.
- The mobile application should have the ability to display the details of the money inside the money bank.

3.4.1.1.2 Mobile Application Non-Functional Requirements:

- Ease-of-use: The mobile application should have a familiar and easy-to-use user Interface.
- Accuracy: The mobile application should be accurate in calculating the real-time value of the money inside the money bank.
- Security: The mobile application should verify the user via his phone number when he creates an account.
- Performance: The mobile application should have a high performance and it should not lag while using it.

3.4.1.2 Money Bank Requirements

3.4.1.2.1 Money Bank Functional Requirements:

- Consists of four main parts: recognition unit, display unit, storage unit, and sending unit.
- The money bank should have the ability to recognize the following banknotes:

 20 ILS	 50 ILS	 100 ILS	 200 ILS
 5 JOD	 10 JOD	 20 JOD	 50 JOD
 1 \$	 2 \$	 5 \$	 10 \$
 20 \$	 50 \$	 100 \$	

Table 6. Acceptable Banknotes

- The money bank should reject the unknown banknotes and only accept the defined banknotes.
- The money bank should have the ability to calculate and display the total amount of money inside it in the display unit.
- The money bank should have a reset button to reset it.
- The money bank should have the ability to send data to the cloud server.

3.4.1.2.2 Money Bank Non-Functional Requirements:

- Ease-of-use: The money should be easy-to-use.
- Accuracy: The money bank should be accurate in recognizing the banknotes.
- Performance: The money bank should work in high performance and should not contain any bugs.

3.4.1.3 Cloud server Requirements

3.4.1.3.1 Cloud Server Functional Requirements:

- The cloud server should store all the system data.
- The cloud server should have the ability to send and receive data from the mobile application.
- The cloud server should have the ability to receive data from the money bank.

3.4.1.3.2 Cloud Server Non-Functional Requirements:

- Security: The cloud server should be safe to store the data.

- Scalability: The cloud server should be ready to scale the memory of our system in it, if necessary.
- Availability: The cloud server should always be running well and avoid failure statuses.
- Operational Cost: The cloud server cost must be within 100\$ per month.

3.4.1.4 API Requirements

3.4.1.4.1 API Functional Requirements:

- The API should provide the mobile application with the real-time value of the currencies.
- The API should receive an HTTP request from the mobile application.

3.4.1.4.2 API Non-Functional Requirements:

- Availability: The API should always be running well and avoid failure statuses.
- Reliability: The API should provide up-to-date currencies values correctly.

3.4.2 USER REQUIREMENTS

- The user should be able to store his money inside the money bank.
- The user should be able to create an account and connect it to his money bank on our application.
- Multiple users could create different accounts and connect them to the same money bank.
- The registered user should have the ability to log in or log out from the application at any time he wants.
- The user should be able to see the total money inside his money bank from the display unit on his money bank and from his mobile via our application.
- The user should be able to see the real-time value of the total banknotes inside his money bank in any available currency he wants.
- The user should be able to see in detail what is inside the money bank.

3.4.3 USE CASE DIAGRAMS

3.4.3.1 Use case diagram for user as actor

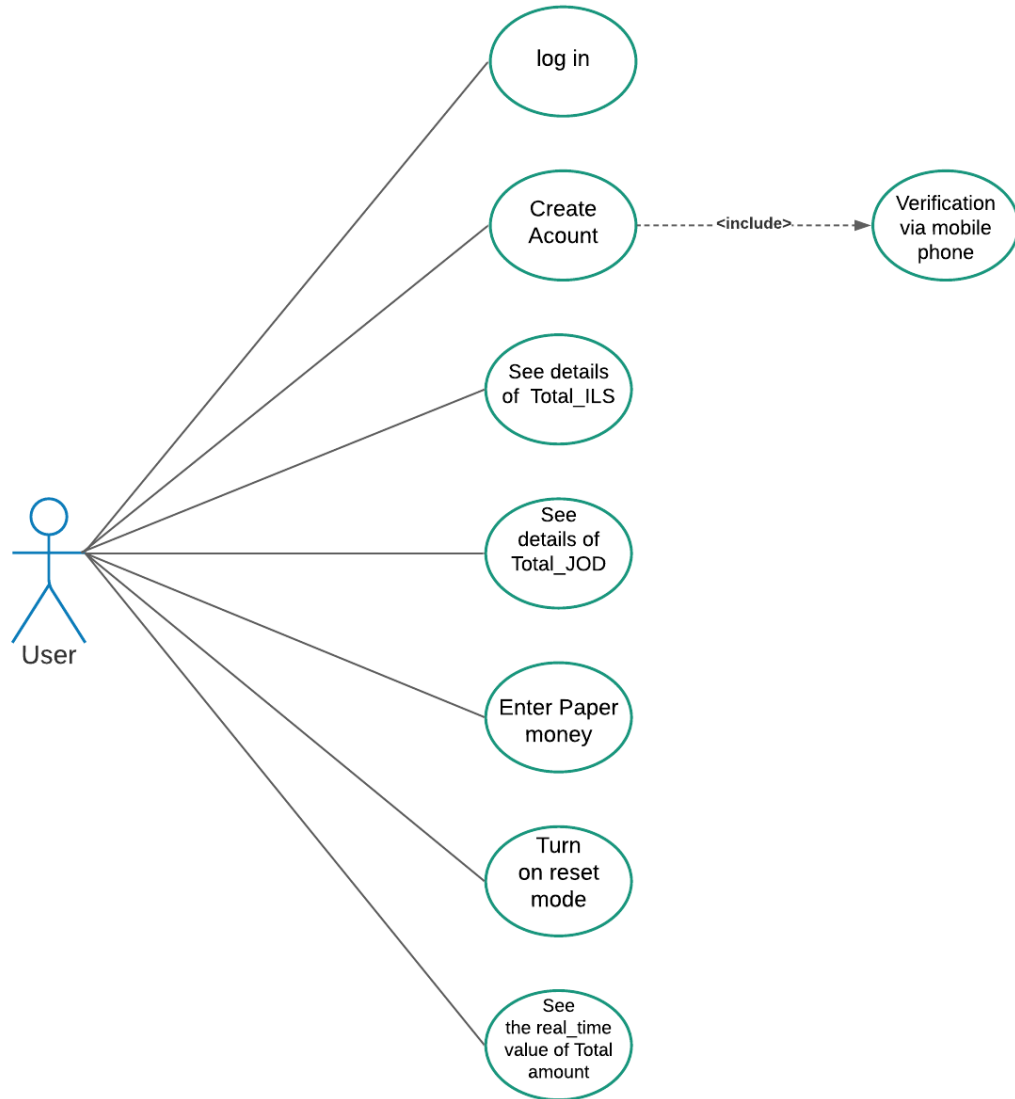


Fig. 26. Use Case Diagram for User as Actor

3.4.3.2 Use case diagram for money bank as actor

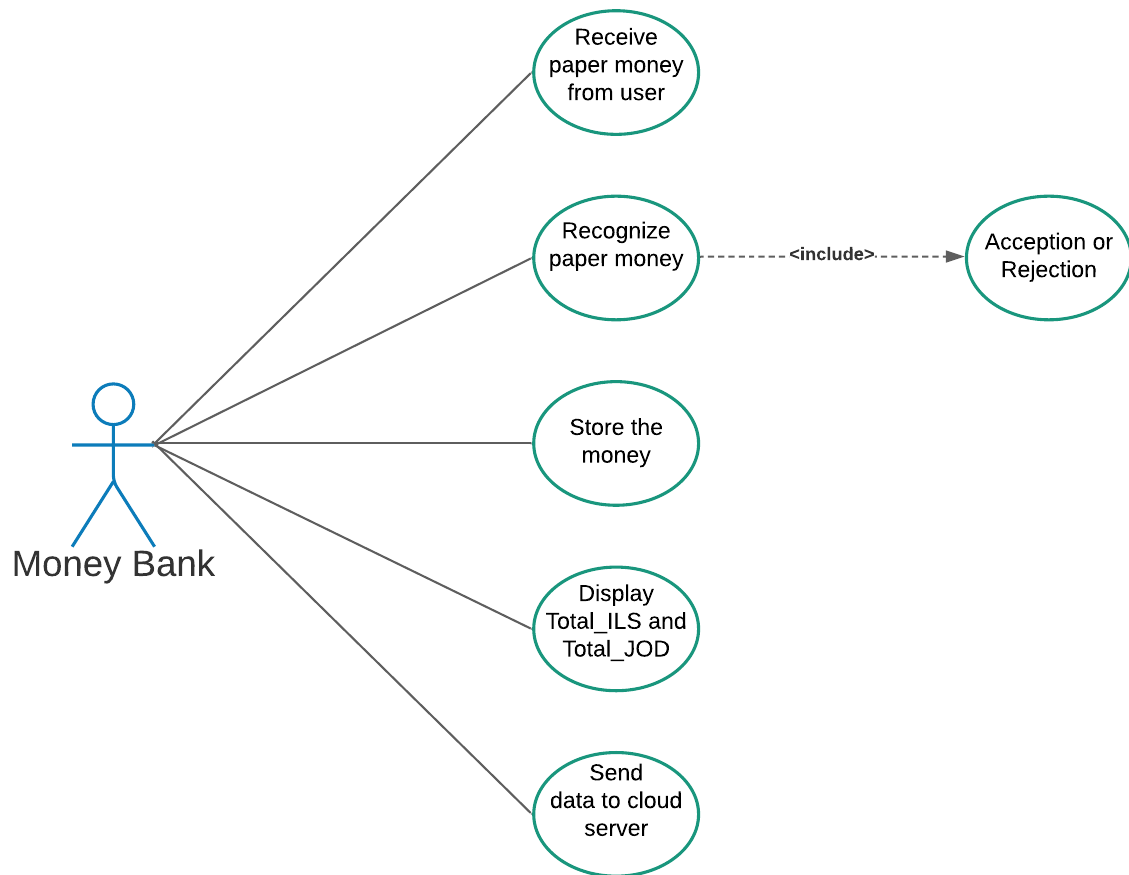


Fig. 27. Use Case Diagram for Money Bank as Actor

3.4.3.3 Use case diagram for cloud server as actor

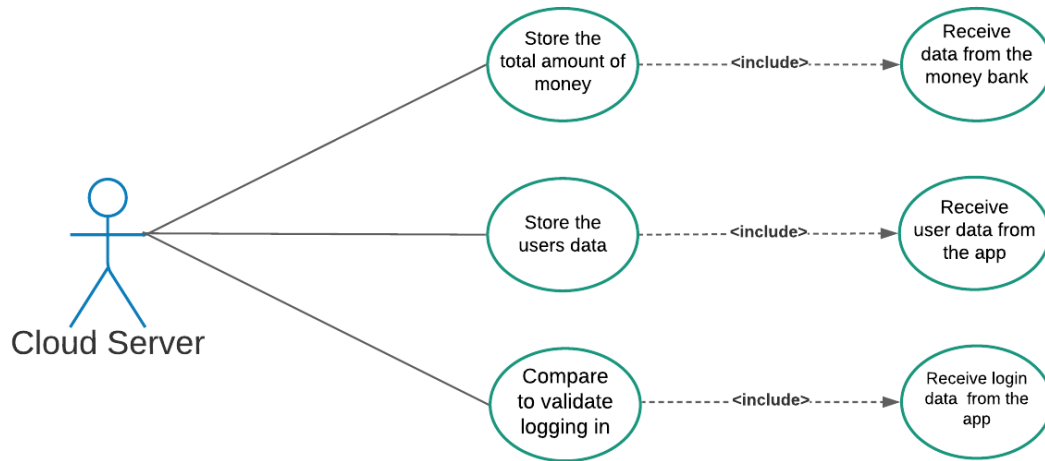


Fig. 28. Use Case Diagram for Cloud Server as Actor

3.4.3.4 Use case diagram for mobile application as actor

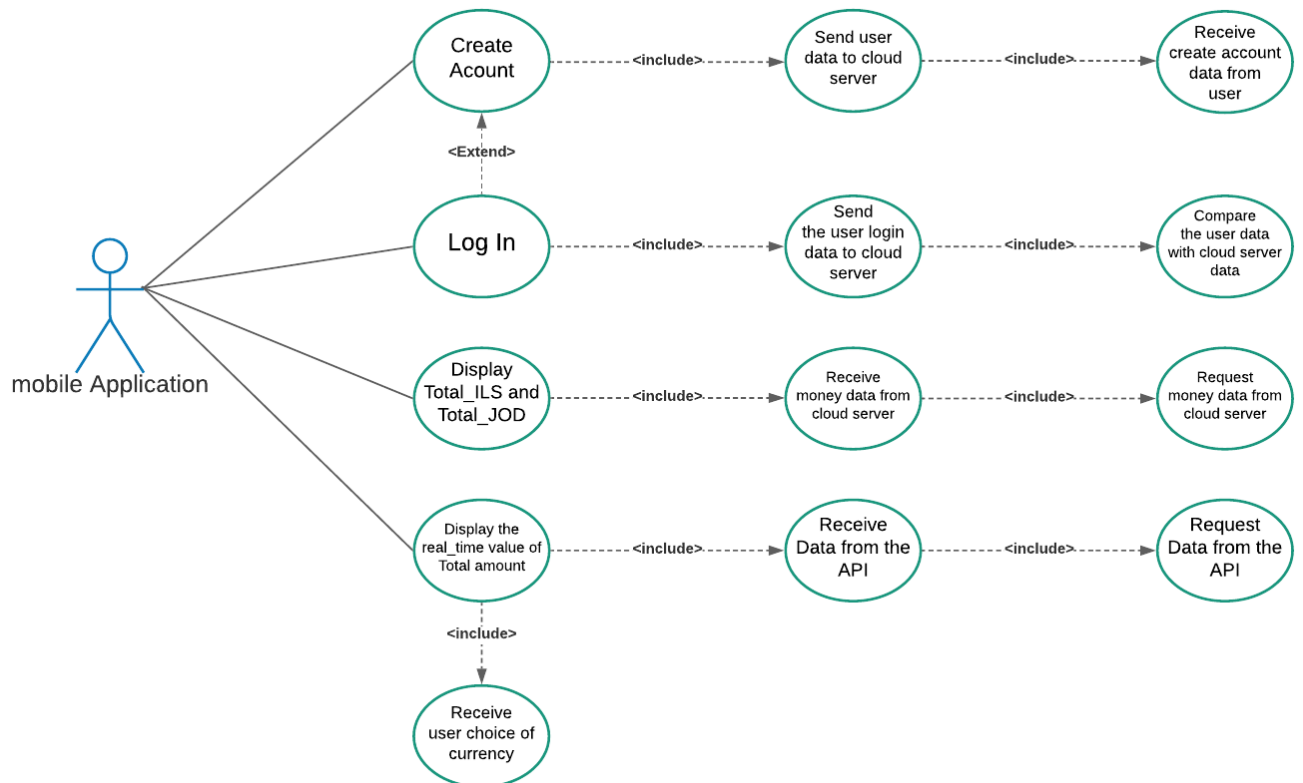


Fig. 29. Use Case Diagram for Mobile Application as Actor

3.4.3.5 Use case diagram for API as actor

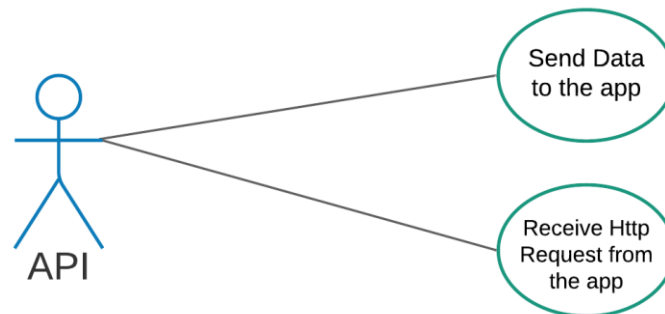


Fig. 30. Use Case Diagram for API as Actor

3.4.4 USE CASE DIAGRAMS DESCRIPTION

3.4.4.1 The interactions that the user could do in the system

Use Case:	Log In
Actor:	User
Precondition:	The user should have an account
Description:	The user logs into the mobile application using his Username and Password.

Table 7. Use Case: Log In

Use Case:	Create Account
Actor:	User
Precondition:	No Precondition
Description:	The user creates an account in the application and he must pass four parameters: Username, Password, PhonNumber, and MoneyBankID.

Table 8. Use Case: Create Account

Use Case:	See in detail the total amount of money on the display screen.
Actor:	User
Precondition:	No Precondition
Description:	The user can see the total amount of ILS money and JOD money on a display screen which is the front part of the money bank.

Table 9. Use Case: See in Detail the Total Amount of Money on the Display Screen.

Use Case:	See in detail the total amount of money on the mobile application.
Actor:	User
Precondition:	Log In
Description:	On the home page, the mobile application displays in detail the total amount of money in the money bank.

Table 10. Use Case: See in Detail the Total Amount of Money on the Mobile Application.

Use Case:	See the real-time value of total money on the mobile application.
Actor:	User
Precondition:	Log In
Description:	On the Currency Exchange Page, the user chooses the currency in which he wants the total amount calculated. The mobile application fetches the real-time value of the currency from the API. Then, the application calculates the total amount of money and presents it on the Currency Exchange Page.

Table 11. Use Case: See the Real-time Value of Total Money on the Mobile Application.

Use Case:	Enter Paper Money
Actor:	User
Precondition:	No Precondition.
Description:	The user enters paper money into the money bank.

Table 12. Use Case: Enter Paper Money

Use Case:	Turn on reset mode
Actor:	User
Precondition:	No Precondition.
Description:	The money bank contains a button for reset. If the user presses this button, The money bank resets the total amount to zero and then displays on its screen "Reset Successfully".

Table 13. Use Case: Turn on Reset Mode

3.4.4.2 The interactions that the money bank could do in the system:

Use Case:	Receive paper money from user
Actor:	Money Bank
Precondition:	The user entered a paper money.
Description:	The money bank receives paper money from the user.

Table 14. Use Case: Receive Paper Money from User

Use Case:	Recognize paper money
Actor:	Money Bank
Precondition:	The money bank receives paper money from user
Description:	After the money bank receives paper money from the user, the money bank recognition unit determines if the paper money is accepted or rejected. If the money bank rejects the entered paper money it displays on its screen "Invalid or unacceptable Money".

Table 15. Recognize Paper Money

Use Case:	Store the money
Actor:	Money Bank
Precondition:	The money bank recognizes paper money and accepts it.
Description:	After the money bank accepts the and recognizes the paper money, it stores the money inside it.

Table 16. Use Case: Store the Money

Use Case:	Display Total_ILS and Total_JOD on the Display Screen
Actor:	Money Bank
Precondition:	The money bank stores the paper money.
Description:	After the money Bank stores the money, it updates the value of Total_ILS and Total_JOD and then displays them on the money bank display unit.

Table 17. Use Case: Display total_ILS and total_JOD on the Display Screen

Use Case:	Send data to cloud server
Actor:	Money Bank
Precondition:	Display Total_ILS and Total_JOD on the Display Screen
Description:	After displaying the updated value of the total amount of paper money, the updated data will be sent to the server cloud by the sending unit.

Table 18. Use Case: Send Data to Cloud Server

3.4.4.3 The interactions that the cloud server could do in the system:

Use Case:	Receive data from the money bank
Actor:	Cloud Server
Precondition:	The money bank sends data to the cloud server
Description:	The cloud server receives the data after sending it from the money bank.

Table 19. Use Case: Receive Data from the Money Bank

Use Case:	Store the total amount of money
Actor:	Cloud Server
Precondition:	Receive data from the money bank
Description:	The cloud server stores the data that it receives from the money bank.

Table 20. Use Case: Store the Total Amount of Money

Use Case:	Receive user data from the mobile application
Actor:	Cloud Server
Precondition:	the mobile application sends the user data to the cloud server
Description:	The cloud server receives the user data after sending it from the application.

Table 21. Use Case: Receive User Data from the Mobile Application

Use Case:	Store the users' data
Actor:	Cloud Server
Precondition:	Receive user data from the mobile application
Description:	The cloud server stores the data that it receives from the application.

Table 22. Use Case: Store the Users' Data

Use Case:	Receive login data from the application
Actor:	Cloud Server
Precondition:	The mobile application sends the user login data to the cloud server.
Description:	The cloud server receives the user login data from the application.

Table 23. Use Case: Receive Login Data from the Application

Use Case:	Compare to validate logging in
Actor:	Cloud Server
Precondition:	Receive login data from the app
Description:	The cloud server compares the login data that it receives from the application. If the data is valid the user can log in, if not the user can not log in.

Table 24. Use Case: Compare to Validate Logging in

3.4.4.4 The interactions that the application could do in the system:

Use Case:	Create an account and send the user data to the cloud server
Actor:	Mobile Application
Precondition:	The user should enter the required data to create an account.
Description:	The application sends the user data to the cloud server to store them.

Table 25. Use Case: Create an Account and Send the User Data to the Cloud Server

Use Case:	Login
Actor:	Mobile Application
Precondition:	The user should enter the required data to log in.
Description:	The application sends the user login data to the cloud server to compare it there.

Table 26. Use Case: Login

Use Case:	Display Total_ILS and Total_JOD
Actor:	Mobile Application
Precondition:	The user should be logged in.
Description:	The mobile application requests the total money data from the cloud server, receives it, and then displays it.

Table 27. Use Case: Display total_ILS and total_JOD

Use Case:	Display the real_time value of Total amount
Actor:	Mobile Application
Precondition:	The API should Send Data to the application after the application requests the data from the API.
Description:	The application displays the real time value of Total amount money.

Table 28. Use Case: Display the real_time Value of Total Amount

3.4.4.5 The interactions that the API could do in the system:

Use Case:	Receive Http request from the application
Actor:	API
Precondition:	The application sends a data request from the API
Description:	The API receive Http request from the application

Table 29. Use Case: Receive Http Request from the Application

Use Case:	Send data to the application
Actor:	API
Precondition:	Receive Http request from the app
Description:	The API send data to the application

Table 30. Use Case: Send Data to the Application

3.4.5 SEQUENCE DIAGRAM

3.4.5.1 The possible sequential interactions between the user, the money bank, and the cloud server.

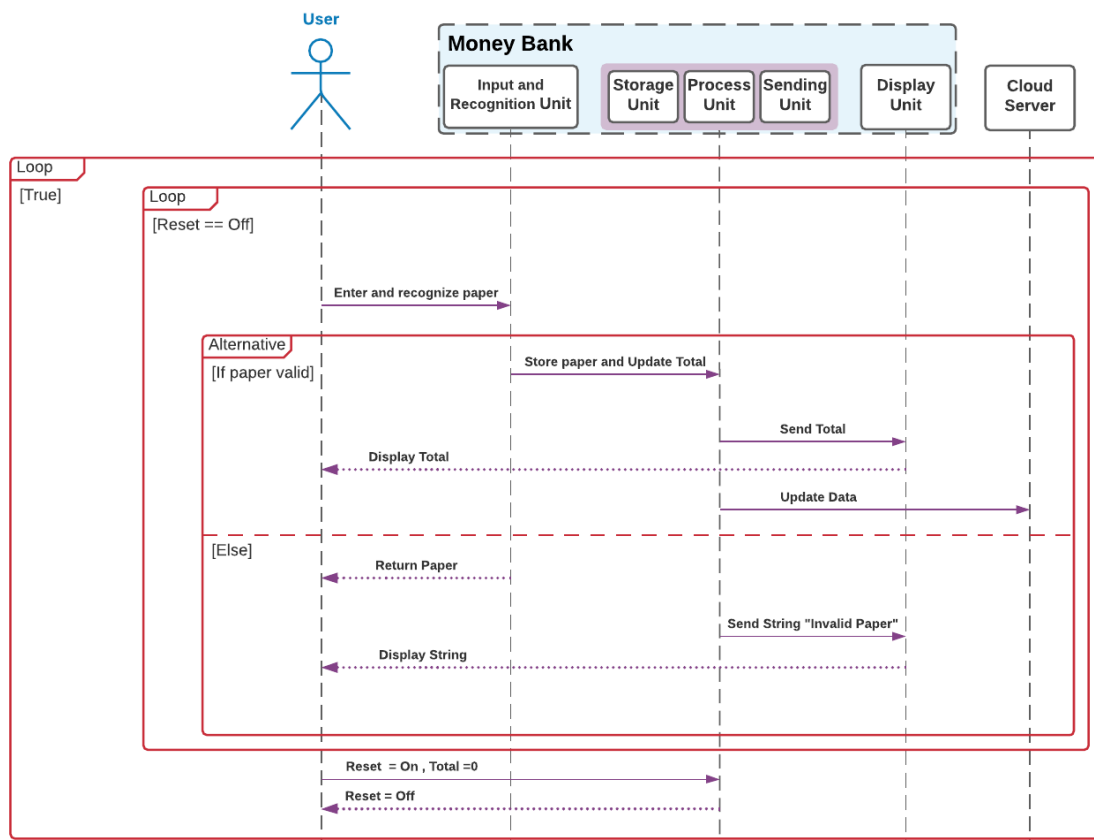


Fig. 31. Sequence Diagram1

3.4.5.2 The possible sequential interactions between the user, the mobile application, the cloud server, and the API.

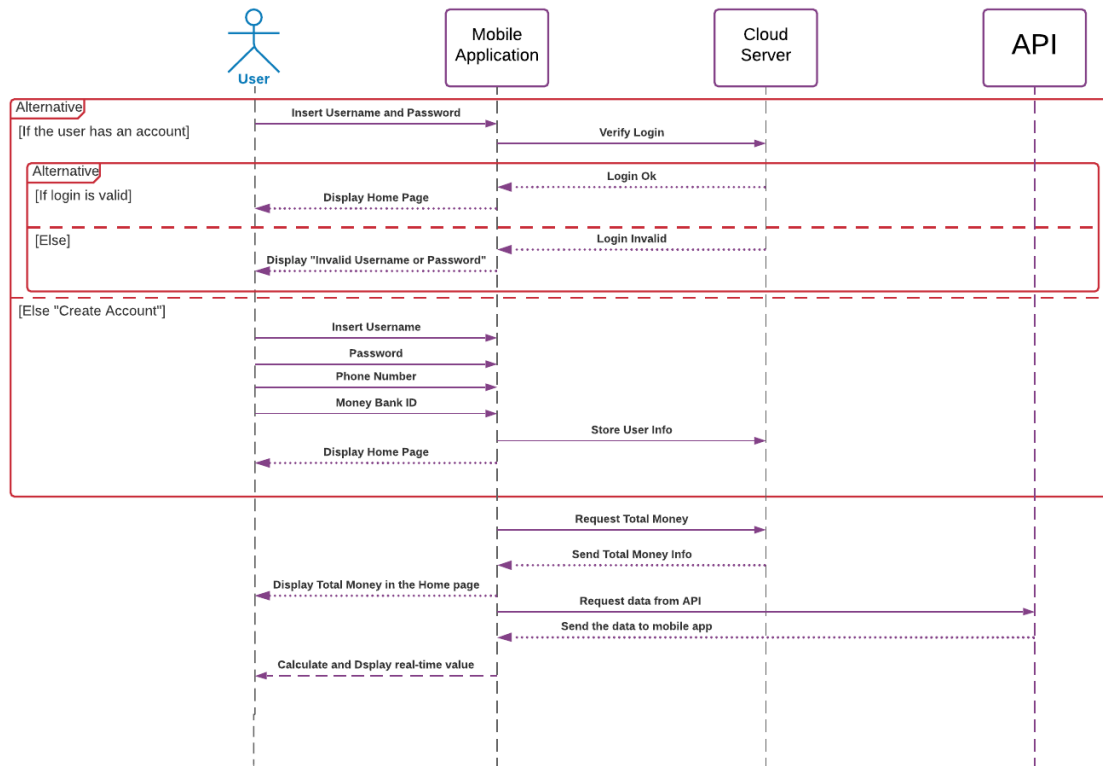


Fig. 32. Sequence Diagram2

3.5 Software Design

3.5.1 SOFTWARE DESIGN OF MOBILE APPLICATION

3.5.1.1 Mobile Application User Interface Flowchart

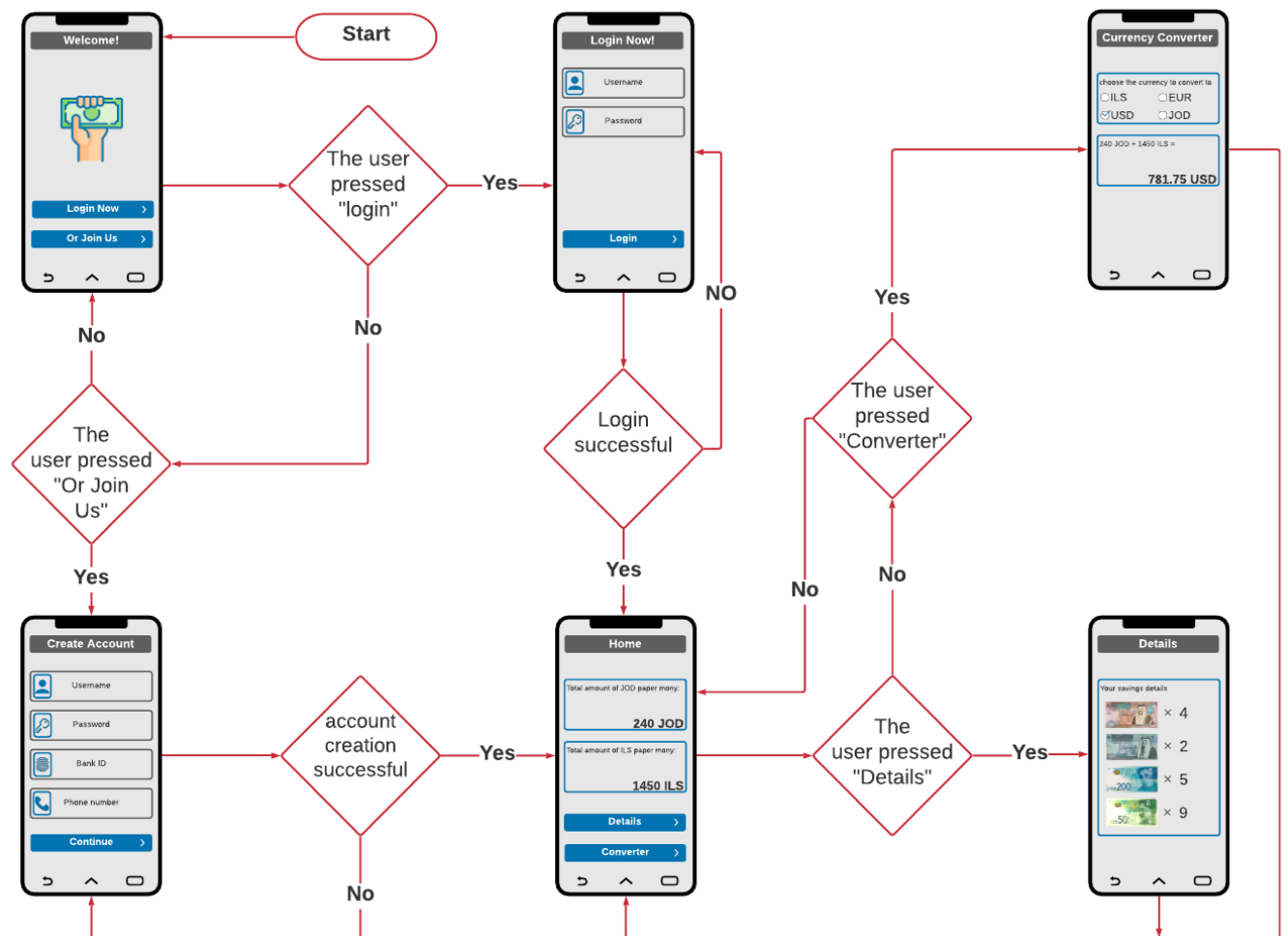


Fig. 33. User Interface Flowchart

3.5.1.2 Fetching the real-time values of currencies from the API

The point of this algorithm is to fetch the data that we need from the API. Since we are going to implement our application on Android Studio, the best choice is to use the Volley library. Volley is an

HTTP library that makes networking for Android apps easier and faster. It automatically schedules the network requests, customizable for our needs, and contains debugging and tracing tools.

In the following code, firstly, we define a String to hold the URL of the API that we are going to use. We take our API from “https://fixer.io/”. Which is a platform that provides APIs for foreign exchange. We decided to go with the free plan and not pay for this API. Unfortunately, the free plan is limited with the EURO as the base currency. That means if we want to automatically change the base currency we must pay. Fortunately, we are programmers. so, later on, we are going to implement a function called `changeBase()` to determine for example the value of one ILS in JOD through knowing about the value of one EUR in ILS.

After defining the URL String, we must override two methods, the first one is **onResponse** which is called when we are connected to the API correctly and here we code to fetch the data from the API. The second one is **onErrorResponse** and here we code anything we want while there is a problem connecting the API.

```
private void getCurrenciesValues() {
    String url =
    "http://api.exchangeratesapi.io/v1/latest?access_key=ba2c4a2b7fed27f2040053098cd691a3&format=1";
    RequestQueue queue = Volley.newRequestQueue(this);
    StringRequest stringRequest = new StringRequest(Request.Method.GET, url,
        new Response.Listener<String>() {
            @Override
            public void onResponse(String response) {
                try {
                    JSONObject jsonObject = new JSONObject(response);
                    String ils = jsonObject.getJSONObject("rates").getString("ILS");
                    String jod = jsonObject.getJSONObject("rates").getString("JOD");
                    String usd = jsonObject.getJSONObject("rates").getString("USD");
                    VALUE_OF_ONE_EUR_IN_ILS = Double.parseDouble(ils);
                    VALUE_OF_ONE_EUR_IN_JOD = Double.parseDouble(jod);
                    VALUE_OF_ONE_EUR_IN_USD = Double.parseDouble(usd);
                } catch (JSONException e) {
                    e.printStackTrace();
                }
            }
        }, new Response.ErrorListener() {
            @Override
            public void onErrorResponse(VolleyError error) {
                Toast.makeText(MainActivity.this, "ERROR", Toast.LENGTH_LONG).show();
            }
        });
}
```

3.5.1.3 Change Base Currency

As we said, the API that we use does not offer the ability of changing the base currency on its free plan. In the previous code, we brought some data from the API which are the values of one EURO in some other currencies. Then we stored these data in the following three variables. The Following function uses these variables to generate new variables which are the values of one ILS in some other currencies.

```
public void changeBase (){
    VALUE_OF_ONE_ILS_IN_EUR = 1 / VALUE_OF_ONE_EUR_IN_ILS;
    VALUE_OF_ONE_ILS_IN_USD = VALUE_OF_ONE_EUR_IN_USD / VALUE_OF_ONE_EUR_IN_ILS;
    VALUE_OF_ONE_ILS_IN_JOD = VALUE_OF_ONE_EUR_IN_JOD / VALUE_OF_ONE_EUR_IN_ILS;
}
```

3.5.1.4 Currency Converter

The currencyConverter method takes outputCurrency as attribute which is the user's choice of the currency that he wants to display output in. First, the method changes all the input to be in ILS currency. And then, according to the user's choice, the function returns the output to be displayed.

```
public double currencyConverter(String outputCurrency){
    ALL_INPUT_IN_ILS = totalILS + (totalJOD*(1/VALUE_OF_ONE_ILS_IN_JOD));
    switch (outputCurrency){
        case "USD":
            return ALL_INPUT_IN_ILS * VALUE_OF_ONE_ILS_IN_USD;
        case "EUR":
            return ALL_INPUT_IN_ILS * VALUE_OF_ONE_ILS_IN_EUR;
        case "JOD":
            return ALL_INPUT_IN_ILS * VALUE_OF_ONE_ILS_IN_JOD;
        default:
            return ALL_INPUT_IN_ILS;
    }
}
```

3.5.2 SOFTWARE DESIGN OF HARDWARE COMPONENTS


As we know the raspberry pi is a full computer since it contains all the computer basic components. the raspberry pi operating system is a Linux-based OS. So, we wrote the code using python. Python is the most popular scripting language. it's compatible with a lot of frameworks and supports a lot of packages.

In this section we start talk about the packages and modules that we used to design the software for the hardware components. Then, we illustrate the software architecture diagram of the hardware components. Finally, we present all the modules that built for each component and feature.

3.5.2.1 Materials of Designing the Software for the Hardware Components

The following table shows the prebuilt packages and modules that we needed to write the code for each component and feature.

Component/Feature	Prebuilt Package/Module	our Build
 <p>Lcd</p>	<p>Lcddriver.py</p> <p>Sys</p> <p>i2c_lib.py</p> <p>time</p>	<p>lcd.py</p>
 <p>Camera</p>	<p>PiCamera</p> <p>datetime</p>	<p>camera.py</p>
 <p>Sensor</p>	<p>RPi.GPIO</p> <p>time</p>	<p>sensor.py</p>
 <p>Motor</p>	<p>RPi.GPIO</p> <p>time</p>	<p>motor.py</p>
 <p>Database</p>	<p>firebase_admin</p> <p>google.cloud</p> <p>google.cloud.firestore</p> <p>time</p>	<p>database.py</p>

 <p>Recognizer</p>	<p>glop cv2</p>	<p>recognizer.py</p>
--	---------------------	----------------------

3.5.2.2 Software Architecture Diagram of Hardware Components

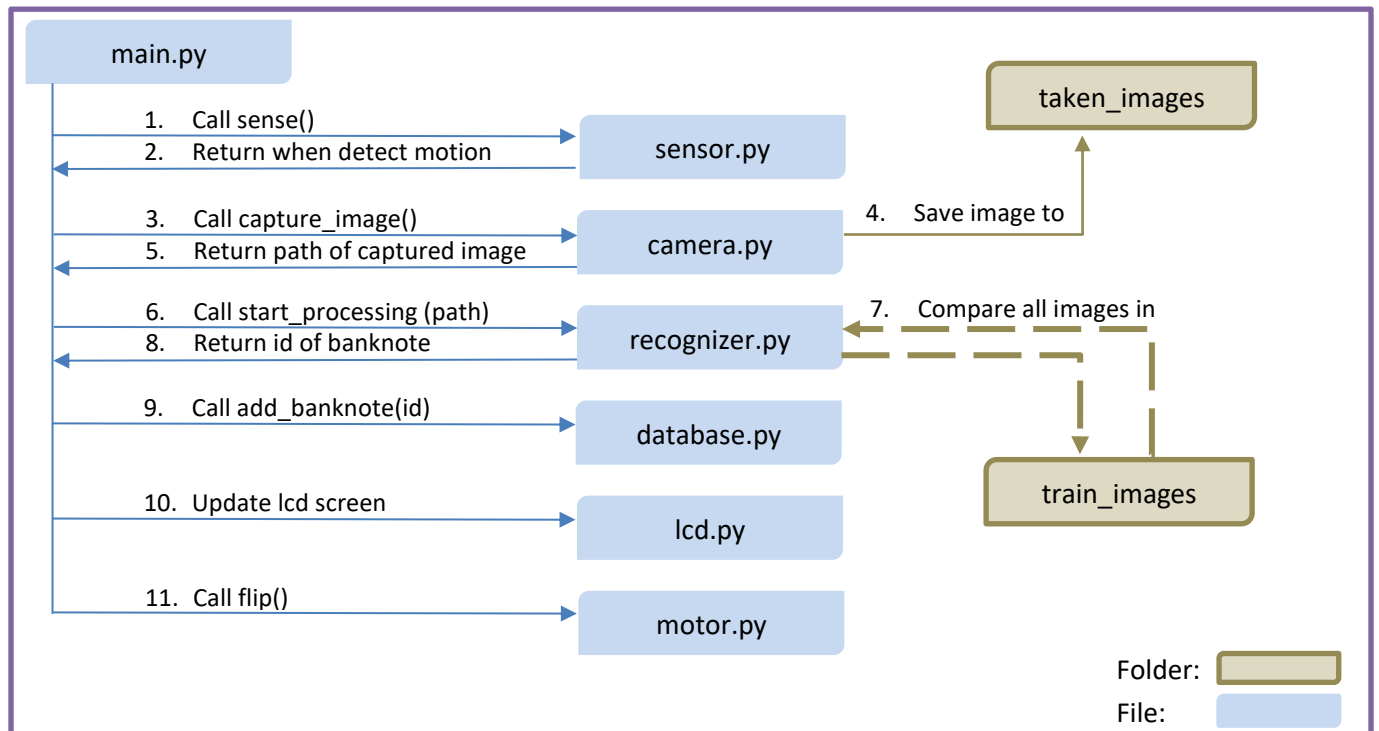


Fig. 34. Software Architecture Diagram of Hardware Components

3.5.2.3 Software Modules of Hardware Components

3.5.2.3.1 *main.py module*

```
import sensor
import camera
import recognizor
import motor
from database import Database
from lcd import LcdDisplay
from threading import Thread
from time import sleep

def main():
    lcd = LcdDisplay.getInstance()
    db = Database.getInstance()
    while(True):
        sensor.sense()
        captured_image = camera.capture_image()
        currency_id = recognizor.start_processing(captured_image)
        if currency_id != "null":
            lcd.setLine1(currency_id)
            db.add_banknote ( currency_id )
            motor.flip()
            Thread(target=db.add_banknote, args=(currency_id,)).start()
            Thread(target=motor.flip).start()
            lcd.setLine2("try again")
            sleep(2)
        else:
            lcd.setLine2("try again")

if __name__ == "__main__":
    main()
```

3.5.2.3.2 *sensor.py module*

```
import RPi.GPIO as GPIO
from time import *
def sense():
    GPIO.setmode(GPIO.BOARD)
    SENSOR_PIN = 7
    GPIO.setup(SENSOR_PIN, GPIO.IN)
    while True:
        if GPIO.input(SENSOR_PIN):
            sleep(2)
            GPIO.cleanup()
        return
```

3.5.2.3.3 *camer.py module*

```
from picamera import PiCamera
from datetime import datetime
def capture_image():
    image_name = datetime.now().strftime("%d_%m_%Y____%H_%M_%S")
    path = ("/home/pi/Desktop/moneybank/taken_images/" + image_name + ".png")
    camera = PiCamera()
    camera.resolution=(2592,1944)
    camera.framerate = 15
    camera.start_preview(fullscreen=True)
    camera.capture(path)
    camera.stop_preview()
    camera.close()
    return path
```

3.5.2.3.4 recognizer.py module [10]

```
import glob
import cv2
def start_processing(image_path):
    test_image = cv2.imread (image_path)
    test_image = cv2.cvtColor (test_image, cv2.COLOR_BGR2RGB)
    sift = cv2.SIFT_create()
    kp2,des2 = sift.detectAndCompute (test_image, None)
    max_points, winner = 0, "no_one"
    for f in glob.iglob("/home/pi/Desktop/moneybank/train_imgs2/*"):
        train_image = cv2.imread(f)
        train_image = cv2.cvtColor(train_image, cv2.COLOR_BGR2RGB)
        kp1,des1 = sift.detectAndCompute (train_image, None)
        bf = cv2.BFMatcher()
        matches = bf.knnMatch(des1,des2,k=2)
        good=[]
        for m,n in matches:
            if m.distance < 0.30*n.distance:
                good.append([m])
        print(f)
        print(len(good))
        if(len(good)>max_points):
            max_points = len(good)
            winner = f
    print("winner: "+winner)
    print("points: " +str(max_points))
    if winner == '/home/pi/Desktop/moneybank/train_imgs2/twenty_ils.png':
        return "ils_20"
    elif winner == '/home/pi/Desktop/moneybank/train_imgs2/fifty_ils.png':
        return "ils_50"
    elif winner == '/home/pi/Desktop/moneybank/train_imgs2/hundred_ils.png':
        return "ils_100"
    elif winner == '/home/pi/Desktop/moneybank/train_imgs2/two_hundred_ils.png':
        return "ils_200"
    return "null"
```

3.5.2.3.5 database.py module

```
import firebase_admin
from firebase_admin import credentials
from google.cloud import firestore
from google.cloud.firestore import Increment
from firebase_admin import firestore
class Database(object):
    __instance = None
    @staticmethod
    def getInstance():
        if Database.__instance == None:
            Database()
        return Database.__instance
    def __init__(self):
        if Database.__instance != None:
            raise Exception("This class is a singleton!")
        else:
            Database.__instance = self
            # initializing firebase
            self.cred = credentials.Certificate('//home//pi//Desktop//moneybank//money-bank-6a53f-firebase-adminsdk-3qvpd-9590961837.json')
            self.firebase_admin = firebase_admin.initialize_app(self.cred)
            self.db = firestore.client()
            # initializing lcd
            #self.lcd = LcdDisplay.getInstance()
```

```
def add_banknote(self, id):
    ref = self.db.collection(u'banknotes').document(f'{id}')
    ref.update({ 'amount' : Increment(1)})
    ref = self.db.collection(u'total').document(f'{id[:3:]}' )
    ref.update({ f'{id[:3:]}' : Increment(int(id[4:])) })
    self.lcd.setLine2(self.lcd.line2+"====")
def get_total(self):
    return{
        "ils": self.db.collection("total").document("ils").get().to_dict()["ils"],
        "jod": self.db.collection("total").document("jod").get().to_dict()["jod"],
        "usd": self.db.collection("total").document("usd").get().to_dict()["usd"]
    }
```

3.5.2.3.6 lcd.py module [11]

```
import lcddriver
from database import Database
class LcdDisplay:
    __instance = None
    @staticmethod
    def getInstance():
        if LcdDisplay.__instance == None:
            LcdDisplay()
        return LcdDisplay.__instance
    def __init__(self):
        if LcdDisplay.__instance != None:
            raise Exception("This class is a singleton!")
        else:
            LcdDisplay.__instance = self
            self.lcd = lcddriver.lcd()
            self.lcd.lcd_clear()
            self.db = Database.getInstance()
            self.line1 = ""
            self.line2 = ""
            self.line3 = "total:"
            self.line4 = self.db.get_total()
    def setLine1(self, string):
        self.line1 = string
        self.refresh_total(string)
        self.refresh_lcd()
    def setLine2(self, string):
        self.line2 = string
        self.refresh_lcd()
    def refresh_total(self, currency_id):
        self.line4.update({ currency_id[:3:] :
            (self.line4.get( currency_id[:3:] ) +
             int(currency_id[4:])) })
    def refresh_lcd(self):
        self.lcd.lcd_clear()
        self.lcd.lcd_display_string(self.line1 , 1)
        self.lcd.lcd_display_string(self.line2 , 2)
        self.lcd.lcd_display_string(self.line3 , 3)
        self.lcd.lcd_display_string((lambda x : str(x['ils'])+'ils '+
                                     str(x['jod'])+'jod '+
                                     str(x['usd'])+'usd')
                                     (self.line4) , 4)
```

3.5.2.3.7 *motor.py* module

```
import RPi.GPIO as GPIO
from time import sleep
from lcd import LcdDisplay
def flip():
    lcd = LcdDisplay.getInstance()
    GPIO.setmode(GPIO.BOARD)
    GPIO.setup(11, GPIO.OUT)
    pwm=GPIO.PWM(11, 30)
    pwm.start(0)
    pwm.ChangeDutyCycle(1.5)
    sleep(0.75)
    pwm.ChangeDutyCycle(10)
    sleep(0.75)
    pwm.stop()
    GPIO.cleanup()
    lcd.setLine2(lcd.line2+"====")
```

CHAPTER 4: RESULTS AND DISCUSSIONS

4.1 Results

- Raspberry pi is slow in image processing.
- The money bank currently recognizes ILS currencies. It can be programmed to recognize different types of currencies. It is the same process, but it takes a lot of time.
- updating the database and flipping the motor by a separated two threads decreased the time for one round of detecting motion, capturing image, recognizing image, updating LCD, updating database, and flipping the motor.
- The front end of the money bank contains a display screen that displays the total amount of money inside it and the status of the entered banknotes.
- The data in the mobile application is automatically updated when the system recognizes a new entered paper money by the user.
- The currency convertor in the mobile application works very well.

4.2 Discussions

- Why not upgrade the algorithm that is being used in our project to be used in different projects such as detecting and counting the paper money of different currencies and values on an image?
- Why not upgrade the algorithm that is being used to be more accurate and to be self-learning?

CHAPTER 5: PROJECT MANAGEMENT

5.1 Tasks, Schedule, and Milestones

5.1.1 Tasks:

- Buy main components and get info about (Raspberry Pi).
- Turn the components on and get info about its OS which is based on Linux.
- How to control the Raspberry Pi with a computer which is Windows.
- Buy a camera, get info about its configurations and test it.
- Buy a sensor, programming it for camera dynamic start-up purposes.
- Buy a servo motor and configure it.
- Make the servo motor work dynamically along with sensor results.
- Buy an LCD, configure it and test it.
- Test the whole system and display the results on LCD.
- Use a specific algorithm in image processing to recognize specific currency.
- Test the system.
- Change the algorithm with another more efficient one that works with many currencies.
- Test the whole system and demand it.
- Design the interfaces of the mobile app.
- Use the API in the app and test it.
- Connect between the hardware and software using a cloud server.
- Test the whole system and approve it.

5.1.2 SCHEDULE AND MILESTONES:

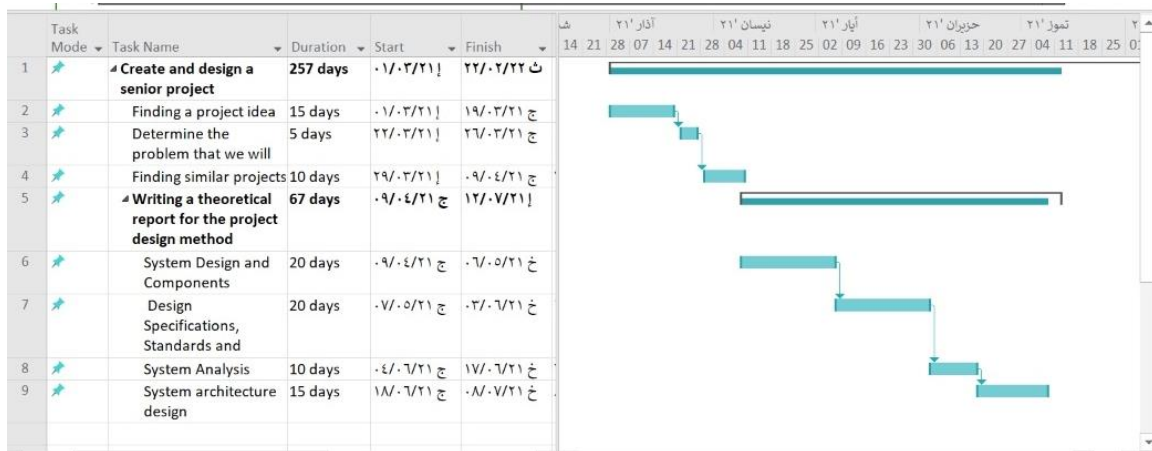


Fig. 35. Schedule and Milestones1

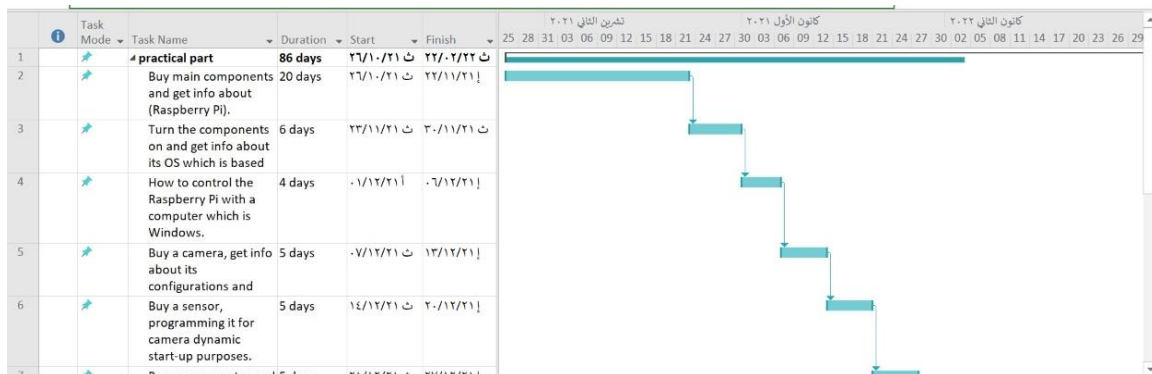


Fig. 36. Schedule and Milestones2

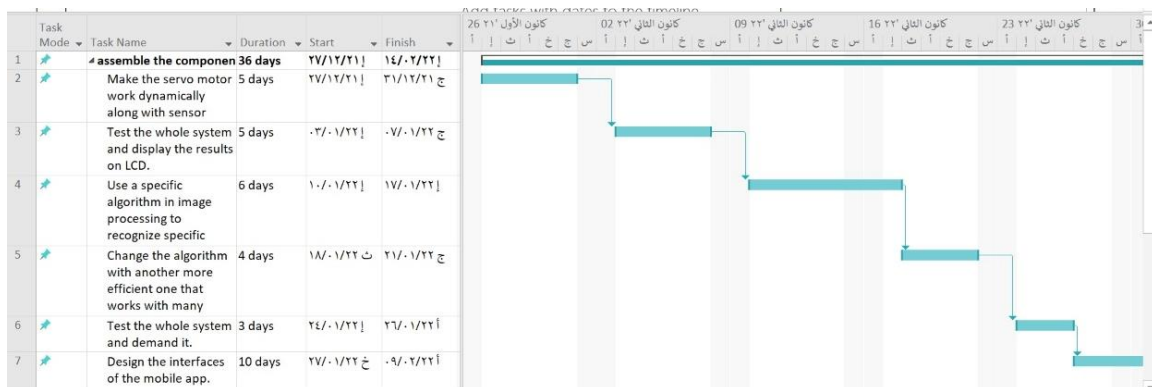


Fig. 37. Schedule and Milestones3

5.2 Resources and Cost Management

5.2.1 RESOURCES:

- We have managed to get started with the project initialization along with the supervisor and we have chosen the topic.
- We have seen existing systems that are similar to our project to some extent.
- We have collected related information from the Internet to go deeply in the research process especially from Google Scholar.
- We have looked for related projects from electronic stores and looked for the relative components that may be used in the project.
- We have basically got information about money banks from YouTube.
- We have downloaded some algorithms from some research and tested them.
- We have downloaded packages and libraries for programming.

5.2.2 COST:

- Raspberry Pi: 130\$
- Camera: 18\$
- Motor: 11\$
- Sensor: 3\$
- LCD: 28\$
- Wooden Box: 20\$
- LEDs: 10\$
- Wires and Cables: 5\$

In addition to many other requirements, the total cost is about 250\$.

5.3 Lessons Learned

We have gathered the requirements from scratch that have no meaning alone. Then we have selected the minimum cost for each component that can work properly and achieve our needs. We have also managed most of the works online. We have distributed the work for the team, everyone with a specific task that can be done. We have used the algorithm to make our project not a traditional one but more intelligent.

CHAPTER 6: IMPACT OF THE ENGINEERING SOLUTION

6.1 Economical, Societal and Global

For economical purposes, the money bank can be used for many users many times. It also can be owned by an institute or company for specific purposes. It is lower cost than having traditional ones which break each time you may get benefit of.

For societal purposes, the money bank can be owned and used within a specific group of people and they can be notified of each transaction on it. Also, it can be used for donation purposes so they can benefit from its features to demonstrate specific goals.

For global purposes, researchers can be done to upgrade and improve the efficiency and features of the money bank. It also can be used in global universities and educational centers to demonstrate the efficiency. We also wish to use it in a good manner that would reach it globally.

6.2 Environmental and Ethical

The environment is affected by using special algorithms that imitate humans recognizing the money. We have used components to reach the human attitude in an intelligent way. We imitate the human in which he can see or calculate the money.

For ethics, the money bank can save money in an ethical way by scheduling the transactions that would not be lost. So, the credibility in the project is high.

6.3 Other Issues

The money bank with a mobile application encourages people to use it, especially that you can be notified of many things related to that bank.

CHAPTER 7: CONCLUSIONS AND RECOMMENDATIONS

7.1 Summary of Achievements of the Project Objectives

We have learnt how to build a project from scratch. We have engaged with hardware and software fields so we have improved our skills to join between these topics. We have also learnt how to make use of different components so that we can achieve specific targets that would help the society in many issues. It was never for us having a project with different fields that would create useful features.

We have engaged with team spirit and we have shared experiences with each other. In addition, we have learnt time management and how to distribute the tasks within the team in a correct way. Also, we have got a great knowledge in how we can deal with problem solving for a real problem. Thinking out of the box to get the best result with a lower cost and a few times is also a great stage we have come through.

7.2 New Skills and Experiences Learnt

We have learned many technical skills from the project. For example, we have improved the programming skills with different programming languages such as Python, Java and Android. Also, we have got a knowledge of frameworks and how to use them to improve the code. In addition, how to make a configuration for any hardware component and make many features from them in code. Also, how to import libraries and packages to the code so we can use many tools from them to make great ideas. Also, how to design interfaces, views and many other things using special applications. For instance, how to connect components with each other.

In fact, we have come through many experiences that made us create the whole system. Such as, problem solving, data structures and algorithms. Also, how to get benefit from specific servers by retrieving data using APIs. Also, how to use image processing to recognize custom objects. And how to implement a mobile app that makes a successful system.

7.3 Recommendations for Future Work

- Using Raspberry Pi with 8 GB of RAM will surely reduce the execution time.
- Add a new camera that works by thermo image capturing in order to recognize the fake paper money.
- Add a deposit feature that allows the user to deposit money from the money bank by establishing a mechanical system that helps in depositing money from the money bank automatically.
- Add a feature to the mobile application which is controlling the deposit process from the app.

Smart Money Bank with a Mobile Application

- Add a new feature that uses machine learning to recognize new currencies that are not defined in the system databases.
- Add a reset button on the money bank layout to reset the total money for the money bank and for the app when depositing the whole cache money from the box.

REFERENCES

- [1] "Connecting a bill acceptor to Arduino.," [Online]. Available: <https://www.youtube.com/watch?v=md7iS5ku1b0>.
- [2] "Digital Piggy Bank with Automatic LCD Display,," [Online]. Available: https://www.amazon.com/Automatic-Capacity-Counting-Christmas-Birthday/dp/B0817MHBVG/ref=sr_1_12?crid=1CCSBMW5O0U40&dchild=1&keywords=piggy+bank&qid=1625888854&srefix=biggy+ban%2Caps%2C341&sr=8-12.
- [3] "Arduino Currency Counter using IR and Color Sensor," [Online]. Available: <https://circuitdigest.com/microcontroller-projects/arduino-currency-counter-using-ir-and-color-sensor>.
- [4] "Difference between MCU and SoC," [Online]. Available: <https://www.geeksforgeeks.org/difference-between-mcu-and-soc/>.
- [5] "Raspberry Pi Organization," [Online]. Available: <https://www.raspberrypi.com/products/raspberry-pi-4-model-b/>.
- [6] "Raspberry Pi Skywire Adapter," [Online]. Available: <https://nimbelink.com/products/raspberry-pi-skywire-adapter/>.
- [7] "SunFounder Raspberry Pi," [Online]. Available: https://www.amazon.com/Raspberry-Inch-Monitor-HDMI-SunFounder/dp/B01J51CXU4/ref=as_li_ss_tl?ie=UTF8&linkCode=sl1&tag=rasplcdkits-20&linkId=4760e4e7dfd50f16275ab2c04610e129&language=en_US.
- [8] "Bill Acceptor Ardac Elite," [Online]. Available: <http://dglpro.eu/eng-componente-acceptoare-de-bancnote.html>.
- [9] "Portable Scanner - Doxie Go SE," [Online]. Available: <https://www.amazon.com/dp/B073H7V4DM?tag=aboutcom02thebalancesmb20&linkCode=ogi&th=1&psc=1&ascsubtag=4157744%7Cn98dff0b2fa4a49ac9016f93a23d6192300>.
- [10] "OpenCV Feature Matching — SIFT Algorithm," [Online]. Available: <https://medium.com/analytics-vidhya/opencv-feature-matching-sift-algorithm-scale-invariant-feature-transform-16672eafb253>.

- [11] "Raspberry Pi with I2C 2004 LCD," [Online]. Available:
<https://arduino geek.wordpress.com/2014/04/23/raspberry-pi-with-i2c-2004-lcd/>.