

SCHOOL *of* BUSINESS AND TECHNOLOGY

Department of Engineering and Aviation Sciences

**Design of Autonomous Food Delivery Vehicle**

**Eli Nbede**

Advisor: Dr. Zhang

July. 29, 2020

Design of Autonomous Food Delivery Vehicle

By

Eli Nbede

Submitted to the Department of Engineering and Aviation Sciences in partial fulfillment of the requirements for the degree of Bachelor of Science in Engineering at the

UNIVERSITY OF MARYLAND EASTERN SHORE

Date

The author hereby grants the University of Maryland Eastern Shore permission to reproduce and distribute publicly, paper and or electronic copies of this document in whole or in parts.

Authors Eli Nbede

Signature

Date

Department of Engineering and Aviation Sciences

Certified by

List of Contents

[1. Introduction 6](#_Toc47015819)

[1.1. Background/Motivation 6](#_Toc47015820)

[1.2. Objective 7](#_Toc47015821)

[1.3. Design Requirements 7](#_Toc47015822)

[1.4. Design Constraints 8](#_Toc47015823)

[1.5. Design Method (Approach) 8](#_Toc47015824)

[2. PROJECT DESCRIPTION 8](#_Toc47015825)

[2.1. . System Description 8](#_Toc47015826)

[2.2 System Functions 9](#_Toc47015827)

[3. Implementation Plan 10](#_Toc47015828)

[3.1. Tasks 10](#_Toc47015829)

[3.2. Implementation of Task 4. 12](#_Toc47015830)

[1.1.1. Implementation of Subtask 3.0: Pantheon Website 12](#_Toc47015831)

[1.1.2. Implementation of Subtask 4.2: Modeling 14](#_Toc47015832)

[3.1.2 Implementation of Subtask 3.1a 15](#_Toc47015833)

[3.1.2 Implementation of Subtask 3.2 Design Menu 18](#_Toc47015834)

[3.1.2 Implementation of Subtask 3.2a communication 20](#_Toc47015835)

List of Figures

[Figure. 1. Starship Robot designed by Starship Technologies 7](#_Toc47015931)

[Figure. 2. Electrical System Diagram 9](#_Toc47015932)

[Figure. 3. Pantheon Website Imager 12](#_Toc47015933)

[Figure. 4. Pantheon Website Imager Message 13](#_Toc47015934)

[Figure. 5. Pantheon Website Imager Map 13](#_Toc47015935)

[Figure. 6. Client Application Flow Chart 15](#_Toc47015936)

[Figure. 7. Image of Registration Page on android studio 16](#_Toc47015937)

[Figure. 8. Flow Chart of the Working Process of the App 16](#_Toc47015938)

[Figure. 9. Existing Account with Email 17](#_Toc47015939)

[Figure. 10. Existing Account with Phone Number 18](#_Toc47015940)

[Figure. 11. Phone Login & Firebase Imager 18](#_Toc47015941)

[Figure. 12. Menu 19](#_Toc47015942)

[Figure. 13. Food Detail 19](#_Toc47015943)

[Figure. 14. MSD Messaging System 21](#_Toc47015944)

[Figure. 15. Communication between server & client app flow chart 21](#_Toc47015945)

[Figure. 16. 21](#_Toc47015946)

List of Tables

[Table 1. PROJECT TIMELINE AND DELIVERY PLAN 11](#_Toc47015991)

1. Introduction

Autonomous self-driving vehicles are growing exponentially in popularity within new-age technology. There are only a few level 5 self-driving vehicles that require no human activation to drive. These self-driving cars are being used to transport goods and people around the world. Furthermore, the electric car is preferred more often because it offers economic, global, and environmental benefits. The car will be planed and designed to deliver food to people autonomously.

## Background/Motivation

On the campus of the University of Maryland Eastern Shore, there are many students that worry about their next meal and how they are going to get hold of it in a timely fashion. A college student has numerous responsibilities on their table, which limits their ability to obtain food at an ideal time. Throughout an average college student's day, they might have at least 3 to 6 classes per day, extracurricular activities if they are an athlete or in some type of group, and they must study to stay on top of their grades. Furthermore, most colleges have freshmen who are not allowed to have cars. And if any student who drives a car must pay a lot of money for parking and fees. This limits the number of college students with cars and increases the number of college students walking around campus. “Among the 214 National Universities that reported these data to U.S. News in an annual survey, the average percentage of students who brought cars to campus in the 2016-2017 academic year was 46.8 percent.”

When college students are in their dorm after a long day of classes and stressful lecturing. They still must do homework and study for their classes. At that point, they are tired of walking and going to get some food that is at least .4 miles away. The combination of the gruesome schedule of college students makes and the tireless walking around campus makes it hard for college students to eat food when it is offered.

What if it was possible to send Campus Universities dining services to students around their dorm room or any location they might possibly be located. This would allow campus living students to have the option to not worry about receiving food without stopping their homework and walking half a mile to their cafeteria or dining services. Autonomous Food Delivery System can be created to deliver food around the campus to any recipient, preferably college students living on campus.

At the University of Maryland Eastern Shore students are offered food at the locations including Students Service Center, Engineering & Aviation Sciences Complex, Hawk’s Nest, Waters Hall. In the Students Service Center, they offer students plateau dining and oasis staff dining. The UMES campus offers food, but they have no form of delivery options for students to receive food on campus. Furthermore, college students often argue and complain about the food they receive from the cafeteria. This leads them to eat the food that the university provides or any other dining services other than the cafeteria options. There should also be an option for students to choose the food they want to receive via delivery options. Instead of dreading the taste of the food they hate, they can designate their specified food to come to their location.

The idea of creating an autonomous food delivery car was sparked by the Starship Delivery Robot. As shown in Figure1, The Starship Robot is a six-wheeled ground robot that can navigate streets and sidewalks, where they offer on-demand package delivery for consumers and businesses. The Starship Robot was created and operated by Starship Technologies in July 2014.



1. Starship Robot designed by Starship Technologies

## Objective

Design an Autonomous Electric Car that can deliver dining services around Campus Universities.

## Design Requirements

1. Carry up to eight pounds of food
2. Can travel up to a 2-mile radius around campus
3. Will deliver one meal at a time
4. Opening and closing of the food compartment
5. Navigates to different locations on campus
6. Speed of 5-10 mph (ca. 16 km/h)
7. Drive on the sidewalk
8. Detects crosswalk
9. Detects Objects
10. Phone Application
    1. Payment System
    2. Ordering System
    3. Food Tracking
    4. SMS
    5. Client App
    6. The client receives information from customer order

## Design Constraints

1. Tires might not be able to withstand certain road conditions
2. Not recommended driving in the rain, snow, and thunderstorms
3. The system is incapable of going upstairs or ramp
4. The car is incapable of driving on roadways

## Design Method (Approach)

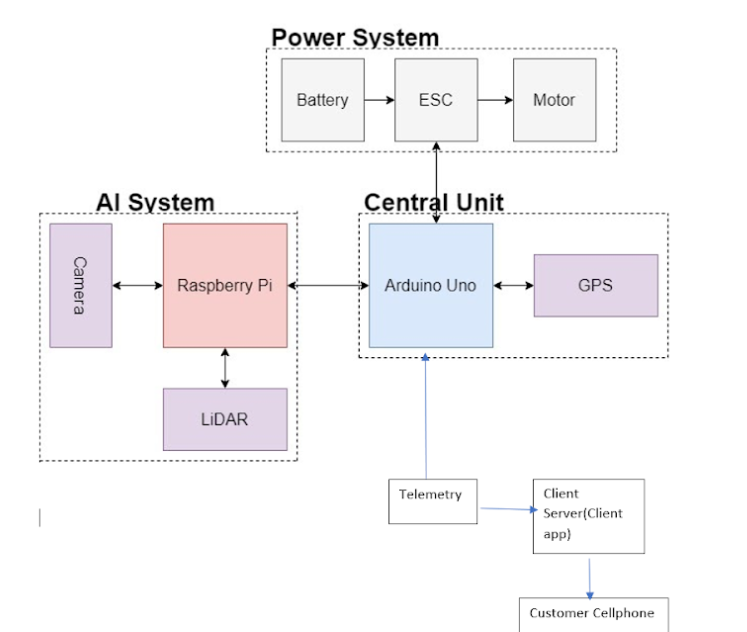
The first step of designing our vehicle is to build the structure of the car, that includes the frame or chassis, this is where our electrical components, Arduino microcontroller, and Raspberry Pi we are placed at. The next step will be to connect the motor and ESC to the Arduino and our battery.  This is how the vehicle will get the power to operate. The next step will be to configure the GPS and compass with our vehicle. After that, we will configure the LIDAR with a Raspberry Pi so that the Raspberry Pi can give the Arduino commands based on the information it receives from the LIDAR i.e. (object avoidance). Once the LIDAR is completed it can be placed in the front of the vehicle. The GPS and compass will both be placed on the outside of the highest part of the vehicle to receive a signal. The next step will be to interface the application “Mission Planner”. This is the application that we will adopt, the waypoints/directions that our vehicle will travel will be placed on here. The use of telemetry will connect the Mission Planner to our vehicle so that they can seamlessly work together. Once the electrical components are all finished, we can close them in under the frame. Next, the food compartment can be set up. a cup will be designed and 3D printed. Also, a small frame will also be built to protect the food from rolling around inside the compartment.

1. PROJECT DESCRIPTION

## . System Description

The core system of the electric car is the Raspberry Pi that is interfaced with the camera module, LIDAR sensor, and the Arduino. The driving parameters from the camera and LIDAR such as object detection and lane detection are sent to the Arduino. The Arduino uses the parameters from the Raspberry Pi to control the car. Also, the GPS is interfaced with the Arduino to send and receive location parameters from the website location. The Arduino uses those GPS commands to drive the car to the designated location. Arduino will notify the website when a designated location is achieved through telemetry, so it can then send an SMS to the recipient. Refer to figure 2, as it shows the system diagram of the described system.

**System Diagram (or Flow Chart)**



1. Electrical System Diagram

## System Functions

The Arduino acts as the Central Unit in the system that takes the input and output parameters to control the entire system. The Raspberry Pi is a microcomputer that takes sensor inputs from the Raspberry Pi camera and the LIDAR unit for deep learning functions in the self-driving car. The Arduino uses the commands from the Raspberry Pi, and the GPS to determine navigation parameters to control the motor. The Arduino also gains commands from the App to set designated navigation parameters that leads to the enabling of the power system

1. Implementation Plan

## Tasks

Task1: Vehicle Structural Design

Subtask 1.1: CAD of Structure

Subtask 1.2: Implement Structure

Subtask 1.3: Design Lock System

Task 2: Design of Electrical Power System

Subtask 2.1: Identify electrical components

Subtask 2.2: Connect Components

Subtask 2.3: ESC configuration with Arduino

Task 3: APP Design

Subtask 3.1: Create Register User

Subtask 3.1a: Create Sign in Method

Subtask 3.2: Design Menu

Subtask 3.2a: Communication

Subtask 3.2a: Payment System

Subtask 3.3: Design Food Tracking System

Subtask 3.4: Design SMS

Task 4: Design of the Artificial Intelligence System

Subtask 4.1: Design Semantic Segmentation for System

4.1.1: Gather Dataset

4.1.2: Build Semantic Segmentation Model

4.1.3: Train Model

4.1.4: Test and Evaluate Model

Subtask 4.2: Detection

4.2.1: Deploy Pre-Trained Model to Raspberry Pi

4.2.2: Design Motion Control Guidelines

4.3.3: Design Serial Communication Protocol

Task 5: Design LIDAR System

Subtask 5.1: Design Program for LIDAR sensor

Subtask 5.2: Design connection for LIDAR sensor

Subtask 5.3: Send control and steering commands

Task 6: Navigation

Subtask 6.1: Extract Geographic Coordinate Units

Subtask 6.2: Configure GPS on Arduino

Subtask 6.3: Configure Compass on Arduino

Task 7: System Testing, evaluation, and enhancement

Subtask 7.1: Test Design Requirements

Subtask 7.2: Evaluate and apply changes if necessary

Subtask 7.3: Repeat process

Task 1, Task 3, Task 5, Task 6, Task 7

**Timeline/Milestone/Delivery Plan**

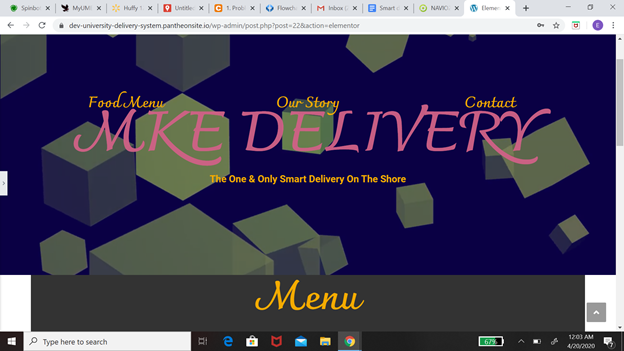
1. PROJECT TIMELINE AND DELIVERY PLAN

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Time | Tasks | | | Comments |
| Montraz Oliver | Eli Nbede | Kevin Harper |
| Week 1-3 | Subtask 1.1 | Subtask 1.1 | Subtask 1.1 |  |
| Week 4 -7 | Subtask 4.1 | Task 3 | Subtask 2.1, 6.1 |  |
| Week 8 - 11 | Subtask 4.1& 5.1 | Task 3 | subtask 5.1 |  |
| Week 12-13 | Subtask 1.2, Subtask 5.2 | Subtask 1.3,  Task 3 | Subtask 1.2, Subtask  2.2 |  |
| week 14-16 | Subtask 4.3-4.4 | Task 3  Subtask 2.3 | subtask 6.2  subtask 2.3 |  |
| week 17-18 | Subtask 4.3-4.4 | Subtask 6.3 | subtask 6.3 |  |
| week 19-22 | Task 7 | Task 7 | Task 7 |  |

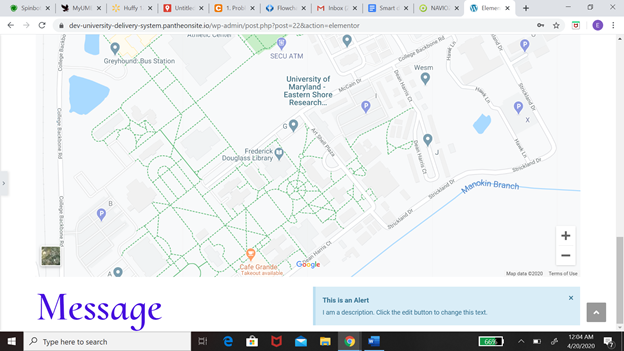
## Implementation of Task 4.

### Implementation of Subtask 3.0: Pantheon Website

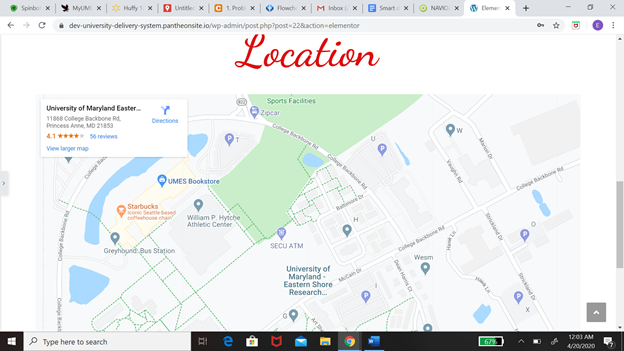
*The task at hand is basically designing a system where customers can order from and the employees are able to receive this order. This message from the customer will come with two key information. One being with the location of the customer and another being in the order history that the customer please. My first solution with the problem at hand was to create a website using Pantheon. One of the main reasons I chose Pantheon was because it is user-friendly. Which would allow us to come to the situation with no experience with this application. The approach I took from there was to design the function of the website, which includes the menu, the contact information, location, and basket. The menu was based on the Chick-fil-A menu that is already online so all I needed was a plug-in but one of the problems I ran into while doing that was there was more than one restaurant on campus. So how would I be able to implement multiple menus on this website? After that, I worked on contact information which would include the number. From there I want to create the location and check out the basket. I still have some dilemma for the program, which includes how to install more than one menu at a time and have the GPS system tracks the food order. I was never able to add the Message system to the website before moving away from this idea.* Figure 3,4,5 shows the logins from Google API mad that I created of the campus which Would allow users to share the location. This is link to locate the website [*https://dev-university-delivery-system.pantheonsite.io/wp-admin/post.php?post=22&action=elementor*](https://dev-university-delivery-system.pantheonsite.io/wp-admin/post.php?post=22&action=elementor)

**

1. Pantheon Website Imager

**

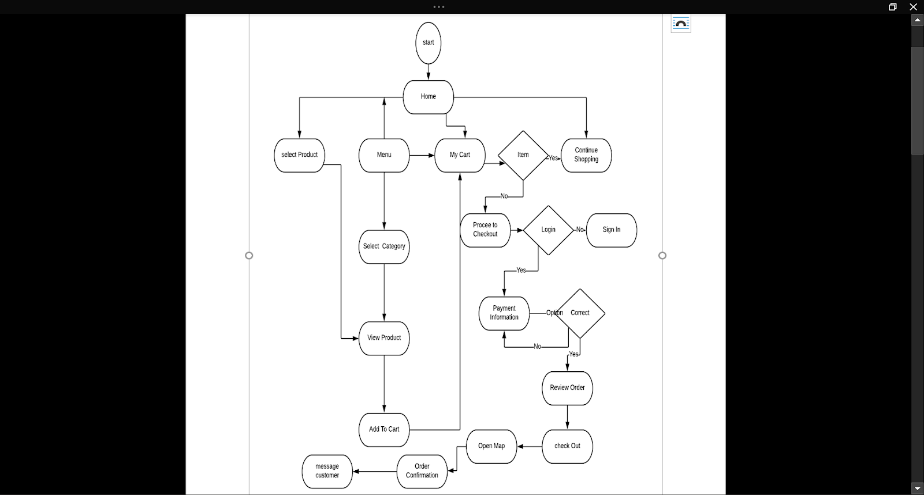
1. Pantheon Website Imager Message

**

1. Pantheon Website Imager Map

### Implementation of Subtask 4.2: Modeling

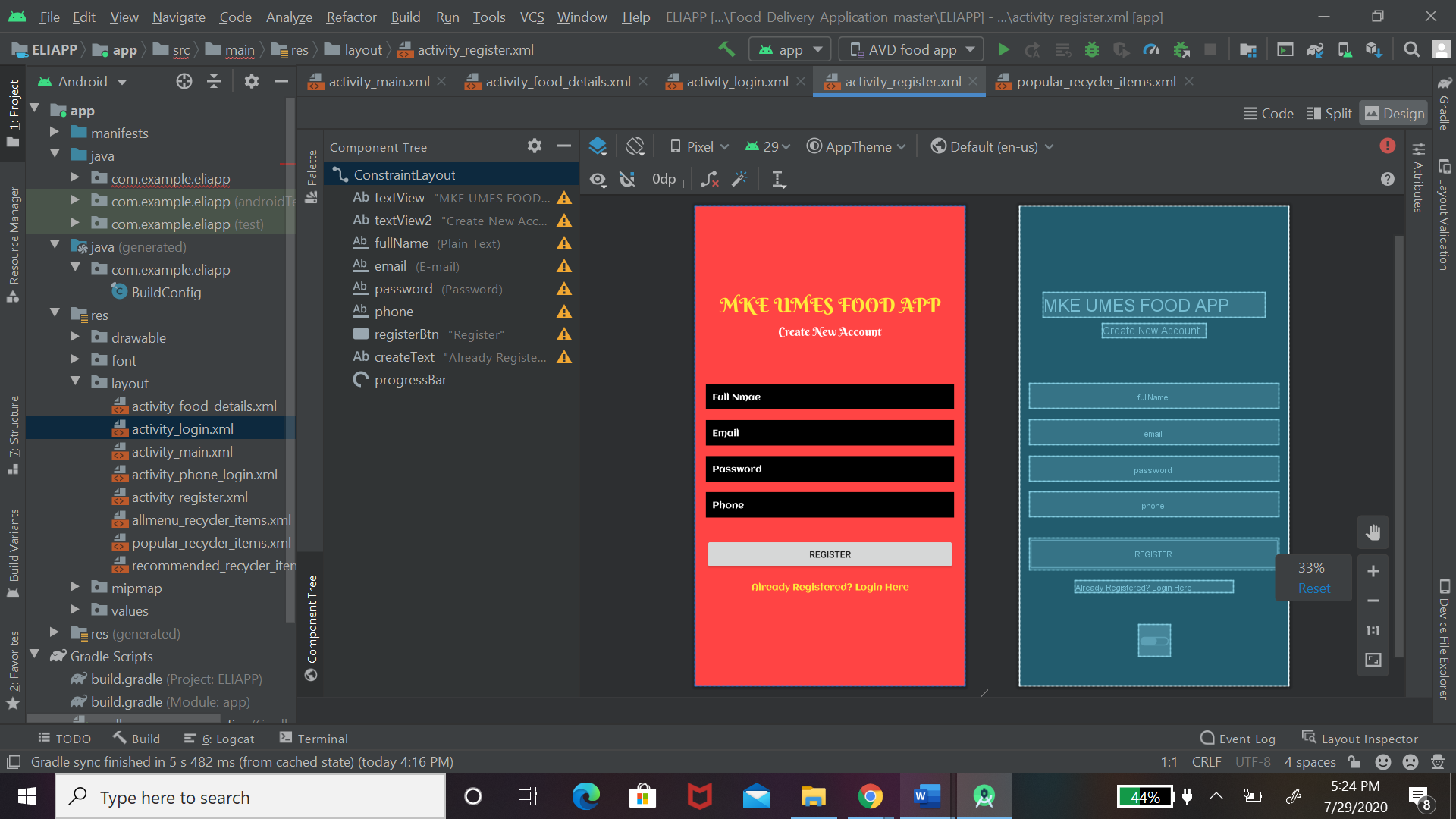
*The next path after leaving the website was to create a cell phone app. One of the reasons why I decided to work with a cell phone app was that it is way more versatile.  When it comes to the solving problem I am having with the messaging system and including the map on the website. The next step to the solution would be to create an application just for the customer. The features on this application would include register, login into the account, home menu, placing orders, allowing payment sharing, location, messaging system. I will be going into more detail about every single feature on this. At first, I need to explain the steps and procedures the App will take to solve these problems. Now I'm going to explain my flow chart, which basically shows my thinking process for creating the application. When you first open the claim application it going to take you to the registration page.*  All the registration screen is going to ask you for your personal information like your full name, email, password, phone number and it’s also give you the option if you have an account to move on to the login page*. If you already have an account with us you will be able to sign in to the app with your phone number or your email and password. If not, you will have the option to sign up. After your login is successfully the app will take you to the homepage. The information that will be included on the homepage would be an all menu items button, shopping cart button, popular this week button and you will be able to search the app from the homepage. Each option will take you to a different page. If you decide to click the popular this week button page you would be able to select the food you want to order from and once you do that it would take you to the information about the food .* If you’re not satisfied with your choice you will have the option to cancel the order and return to the menu and from there you can select a different option*. The next part of this app would be the payment, which would include information like credit or debit card information and the name of the person who is buying the food. If you have a promo you can also include that into payment. After that it’s going to take you to a map of the campus, and you will pick your location on the map. This would all be included in the messaging system. your receipt and all that information will be sent to the business you order from and you will receive a text saying the order is complete.* Figure 6 will show you how the operation flows through the client application

**

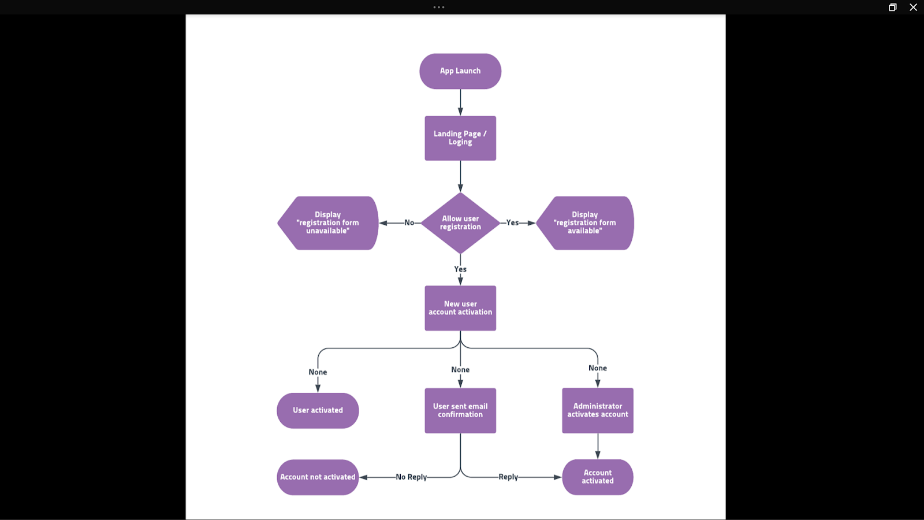
1. Client Application Flow Chart

### 3.1.2 Implementation of Subtask 3.1a

*As we dive into more details about this app. The next stage will be discussing how the register page works. Going into detail about the whole structure and mostly explaining figure 4 and show how the flowchart work. The first aspect of the register page will also be the most important. The reason why is* because this is how the information from the client will be extracted. Once the customer has provided that information, this will be stored in a database. For the database part, I decided to use a Firebase. The motivation behind this decision is that Firebase is already installed on an android studio, and they communicate with each other very well. The first step when you’re trying to connect the two applications is to first create your Firebase account and name it the same thing you name your Android studio project. After that, you must connect your android studio to the Internet so you can access the information for Firebase. This is done by a simple command “<user-permission Android: name= Android. Permission. INTERNET />”. All that is done by asking for permission to the Internet*.* The next step is to create the slot where the user would be able to input the information. This will also come with a button that will allow the user to register the information. Once the button is pushed all the information from the user will be sent to Firebase. There is only one restriction when it comes to the information that is put in. That will be the password information. When users decide to create the password, the password must be six syllables or more. The way I got this to work is by creating an if statement. That says if the password is not more than six or equal, then send that error message saying the password is too weak. The last part of this page already has an account command. Once these commands trigger it will prompt you to the phone login page. Where you can enter your phone number to log into the app or if that is not accessible you can log in with the email and password.  In figure 7,8 it shows how the register page is gone and looked and it also shows the flow chart of the background on the one in the register page

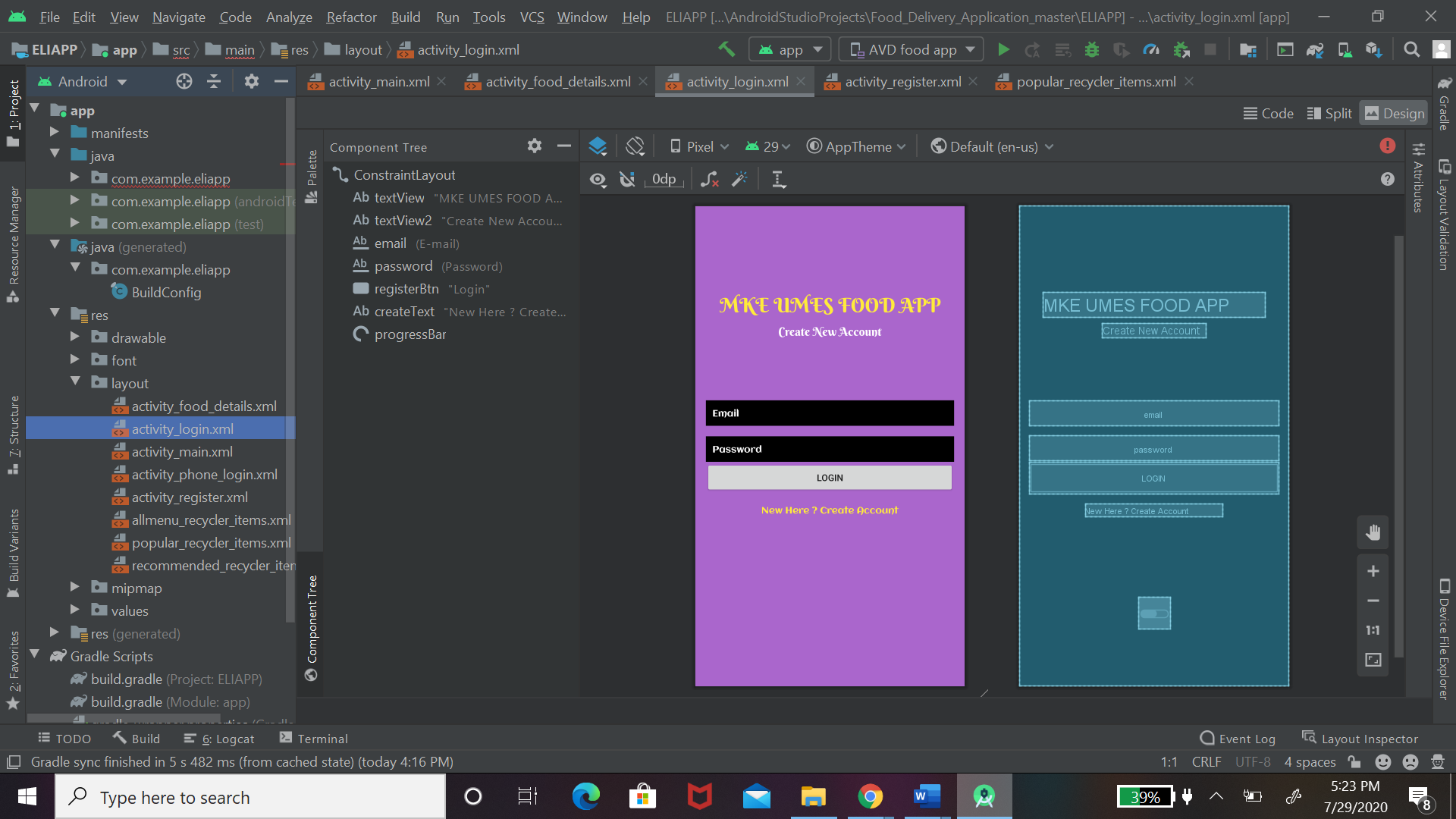


1. Image of Registration Page on android studio

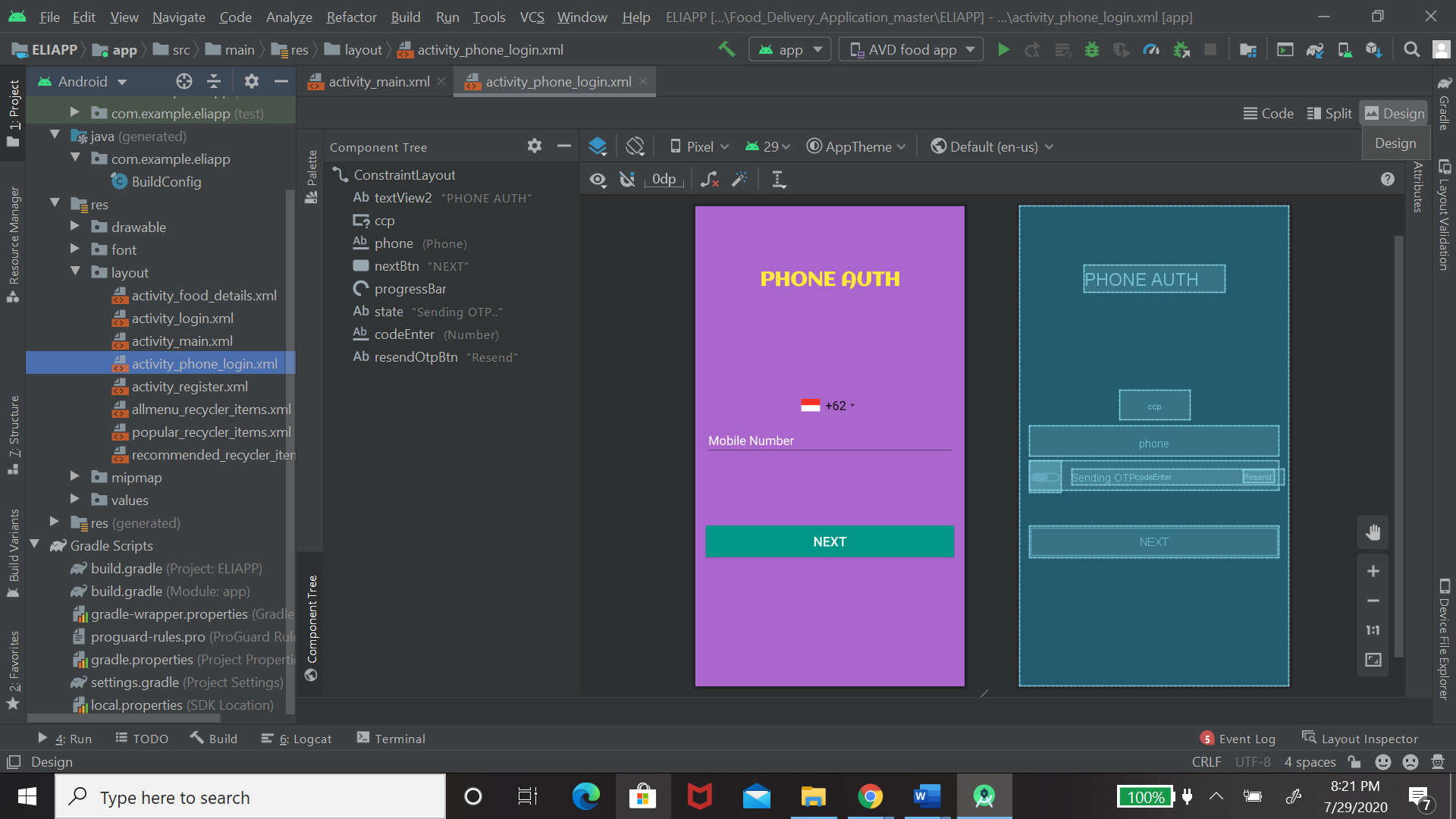
**

1. Flow Chart of the Working Process of the App

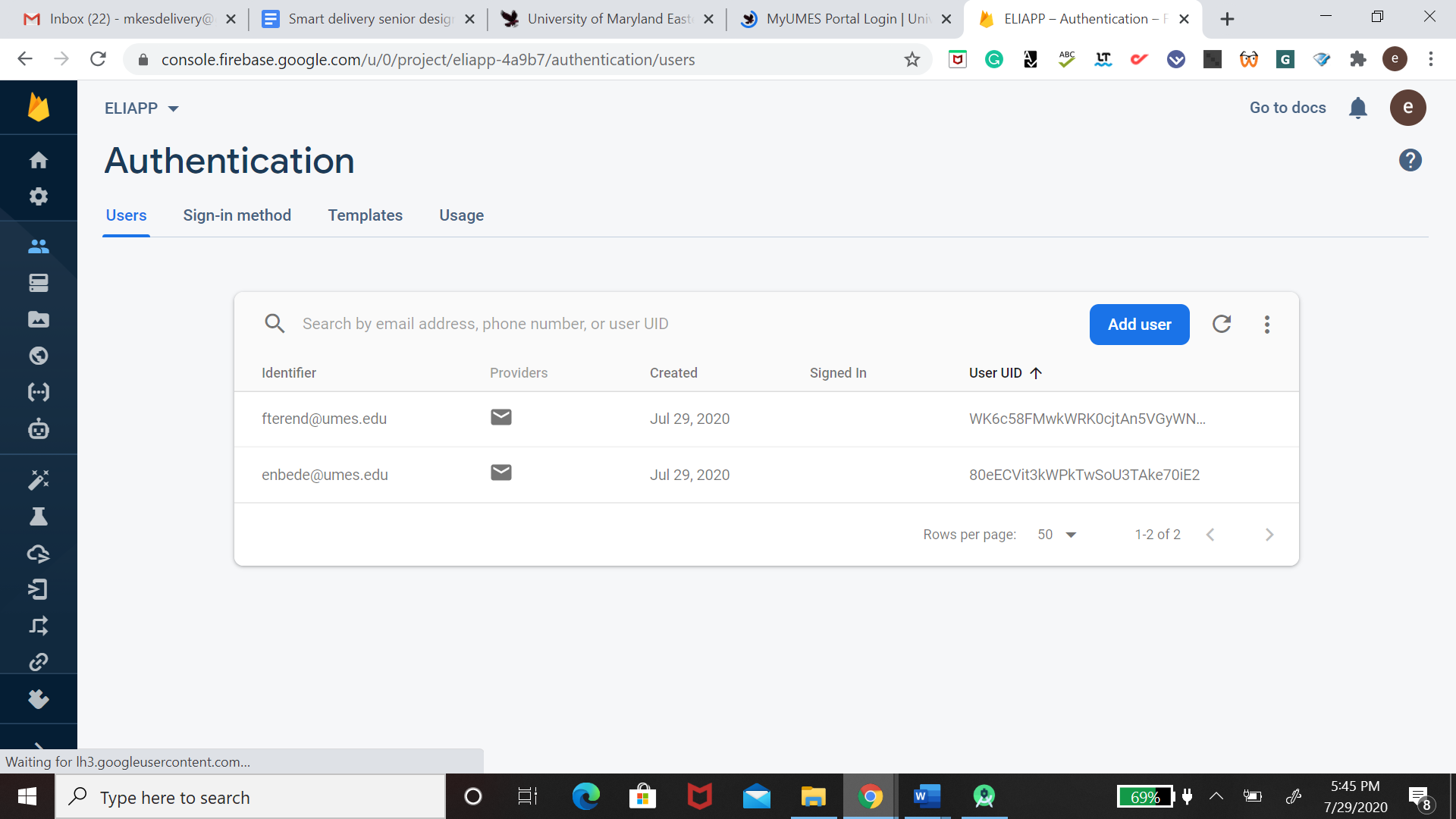
After completing the registration process of the application, the app will take you to the next page which will include your phone number login and email and password login. The purpose of this page is to verify that your information is saved in the Firebase system. Once you register your information all that data will be stored in the Firebase. So all I have to do is to create an on-click listener which will allow the information to check the database. Once the customer puts in the phone number or the email and password, then the click listener will check that information with what is in the storage. If this information matches the application will give access to the next page, we should be on the home page. If the information does not match what is in the storage, an error message will be sent to the user. The user would have two options: the first one would be to try again and the second one will be to get password information. This is also accomplished by one click listener. After the button is pushed to get a password Firebase will send the information to the app. Figure 9 and 10 shows multiple different things: the first one is the login page with your email and password and from there the next one is the login page with your phone number. Last but not least is how your information is stored inside of Firebase.



1. Existing Account with Email



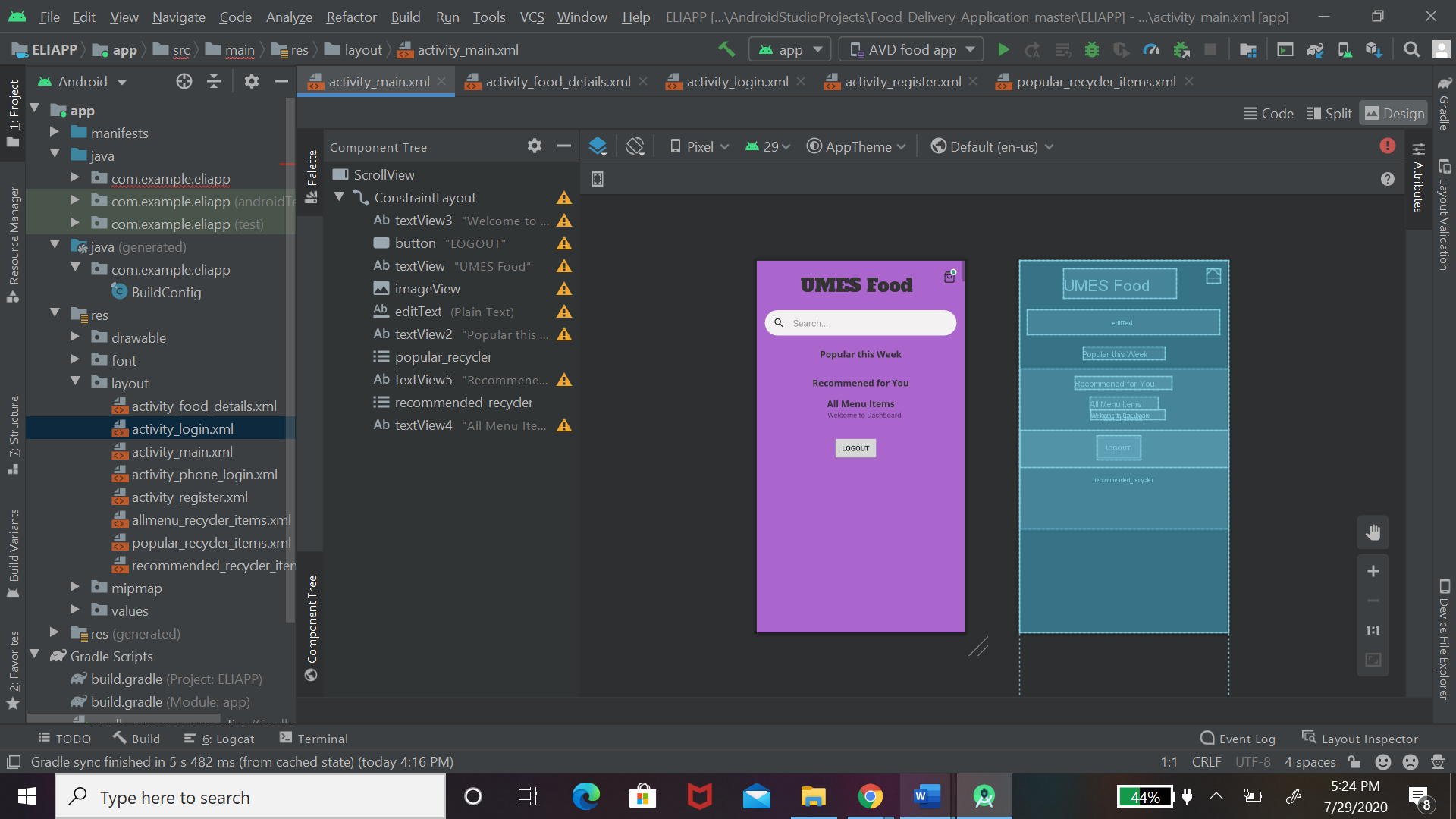
1. Existing Account with Phone Number



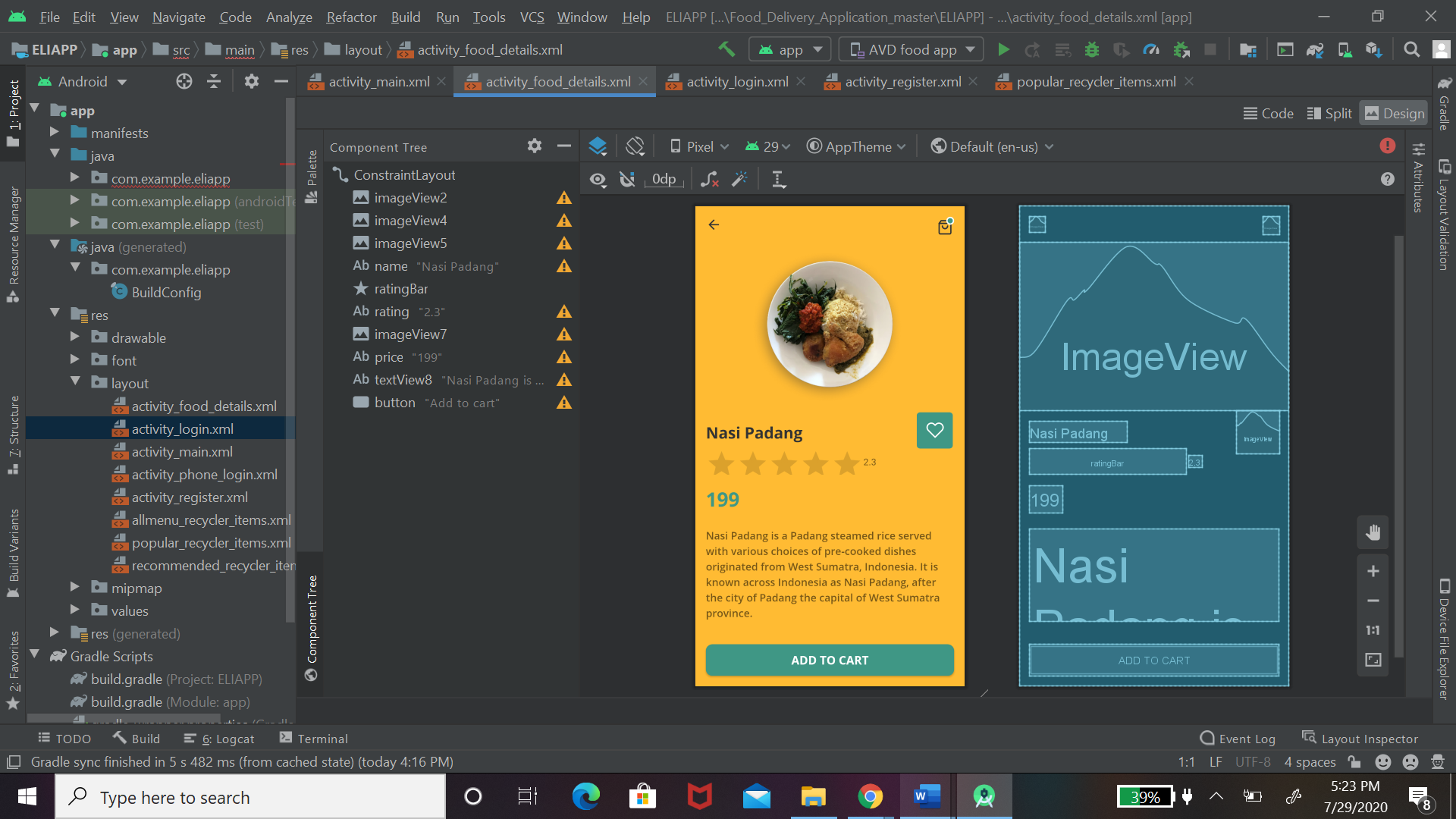
1. Phone Login & Firebase Imager

### 3.1.2 Implementation of Subtask 3.2 Design Menu

The next stage of the project was to create the home menu and food details. So once the customer is successfully logged into the account they will be prompted with the home screen. The home screen would include the menu page, the search bar, and some recommendations for the week. The approach I took to solve this is creating a button for each part of this command except the search bar. The thing about creating the search bar is you don’t physically need to code because the android studio is already adding it to the program. All you have to do is drag the search bar diagram and then it will install the code to your application as a text view file. Once the search bar is clicked it will open with space where you can put in information. The next step is to create the food information. This was not a hard procedure because I was getting the information online so all I needed to do was to add a plug-in to extract the data from online. In figure 12 and 13 you will be able to see how the main menu and food detail would look.



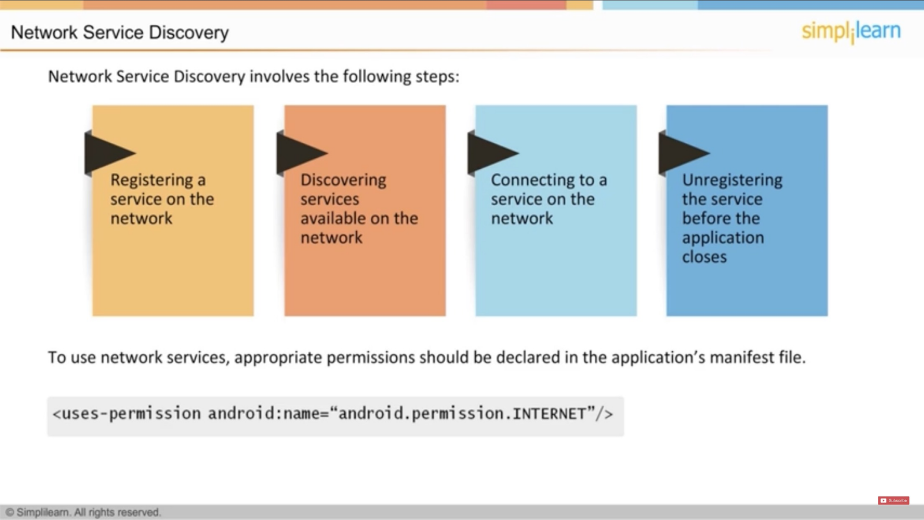
1. Menu



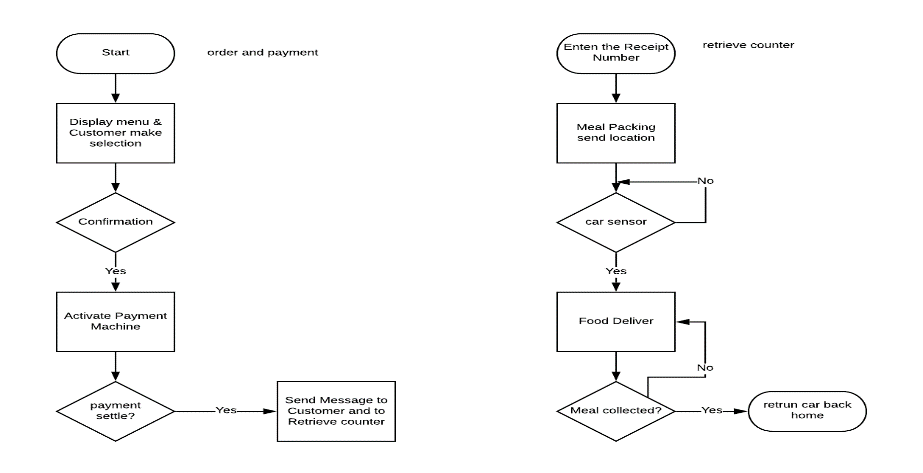
1. Food Detail

### 3.1.2 Implementation of Subtask 3.2a communication

The main part that I am going to dress in this project is how the communication it is going to work between the client application, the server and mission planning. The first thing we must understand is how information is passed through the Internet from one computer to another. So, I will start with the IP address and how it functions. IP address has four numbers separated by dots. For example192.168.0.100 each bracket number is an eight-bit number and that means the number can range from 0 to 255. Now the IP address is two addresses in one. What that means is the IP address consists of host address and network address. The way communication is done between two different devices would be with the IP address and the router. When a package of information is being sent from one device to another, the first step is the device must check and see if they are on the same subnet. The way the check is by looking and seeing if they were on the same IP address. After this step is done the message will be sent to the nearest gateway and all this information will be sent to the nearest router. Once the router has dead bug and confirm that this package of information is safe to be sent to the nearest router that is close to the destination of the second device that the message has been sent to. I am going to explain how this information will work with my solution. My first solution was to use android studio and create my client application. I think after that I will create my server with Raspberry Pi, but the problem I face is after you create your application and your studio is not possible to download that on your phone so if you’re running on a PC it will be pointless. After doing more research I came to a solution of creating my app with MIT app Inventor. Which would be compatible with the server. This will also solve the second part of the problem, which would be how the server is going to communicate with the mission planner. So once that information is sent from MIT app Inventor to the Raspberry Pi server, the information process and send the telemetry. From there that information can be extracted by the mission planner. You may be asking how this will work. The first step is to develop the application which is going to include the signing page, menu page, google map and check out payment method. The next step is to develop a server and I decided to build it on Raspberry Pi. The reason why I chose Raspberry Pi is that it is compatible with MIT app developers. Figure 15 displays how the communication system was through a flow chart and you will see a block diagram. The first step is to register your application and your application will use the network server discovery. To send out the information.



1. MSD Messaging System

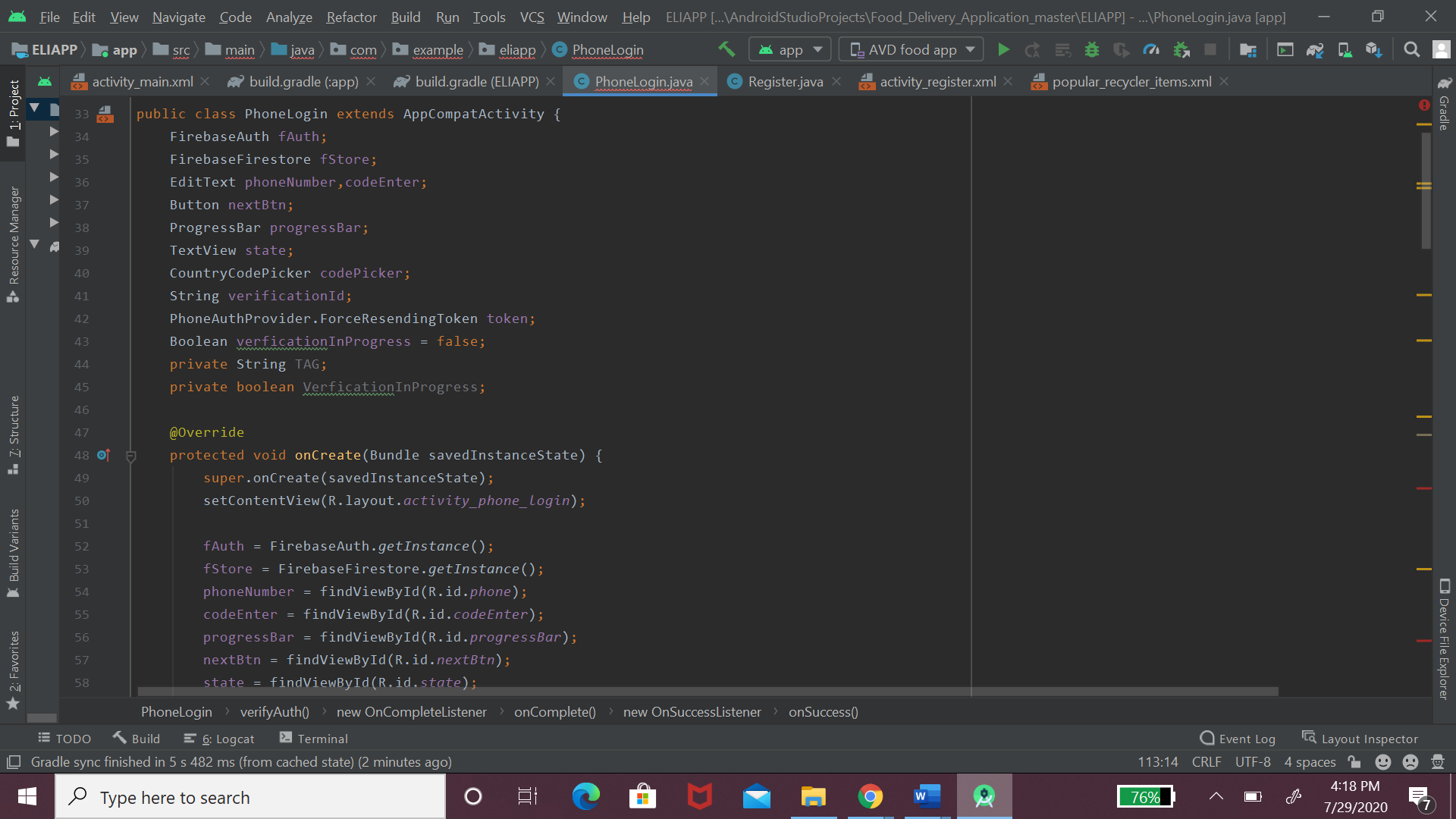
**

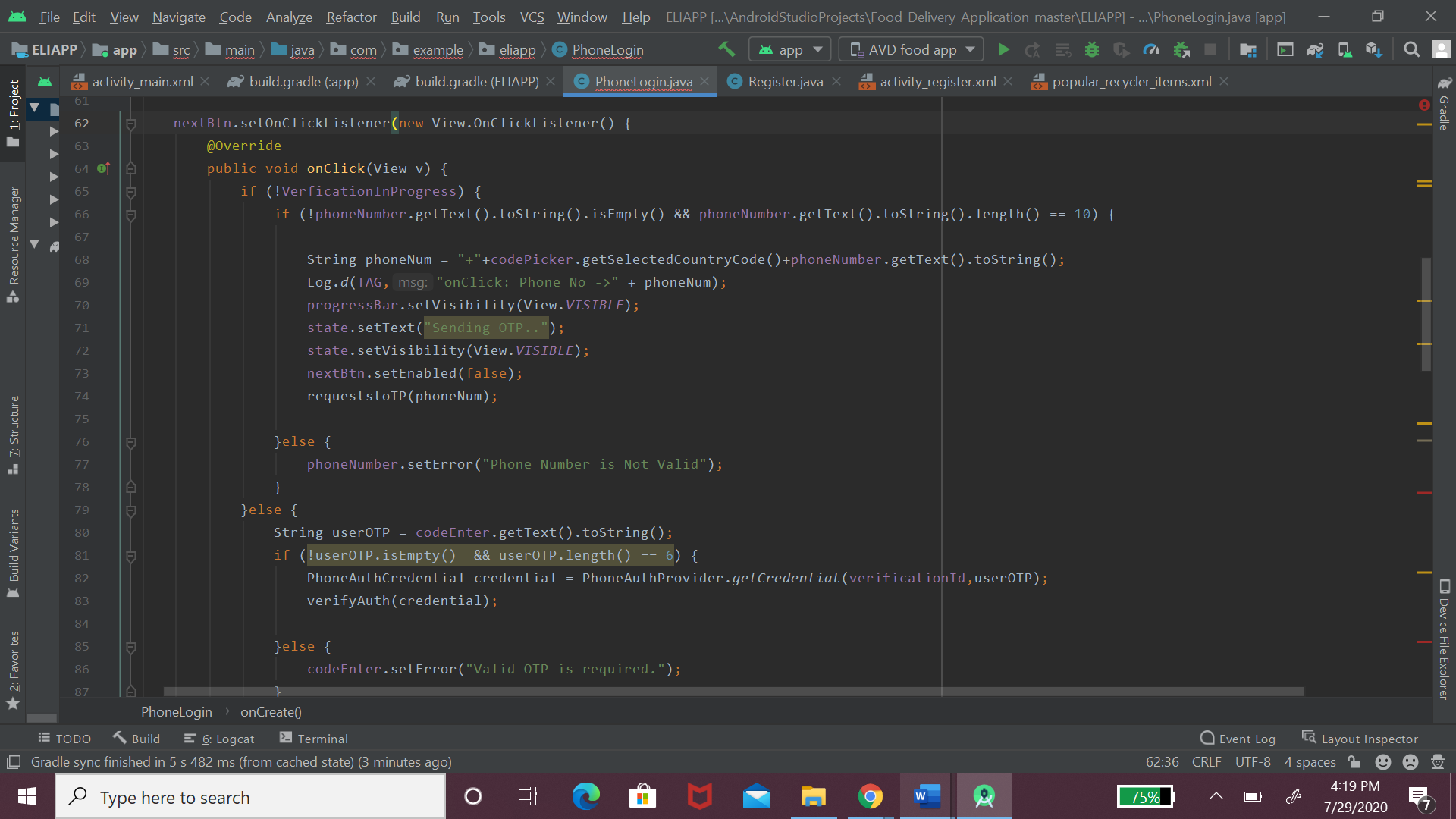
1. Communication between server & client app flow chart

After creating the client application, I was faced with a problem. I would not be able to download the app on my phone. If I can publish this application the client will not be able to use the app so this would defeat the purpose of creating the app. After deep research, I concluded creating a new app with the MIT App Inventor. There are many reasons why I choose the MIT app inventors. The first one is because it is very user-friendly. Unlike the Android studio where I had to hard-code the information, MIT app inventor is coded with block. The next reason why I chose MIT app inventor is that it is compatible with Raspberry Pi wish I would create my service. The way the communication works between these two applications is with CloudDB. The CloudDB component is a non-visible component that allows you to store data on an Internet-connected database server. Once I set up my raspberry server and connected to the Internet. All I would have to do is connect the app by importing the port number and the server name into my application. After this step is complete, I will be able to send and receive information from the server.

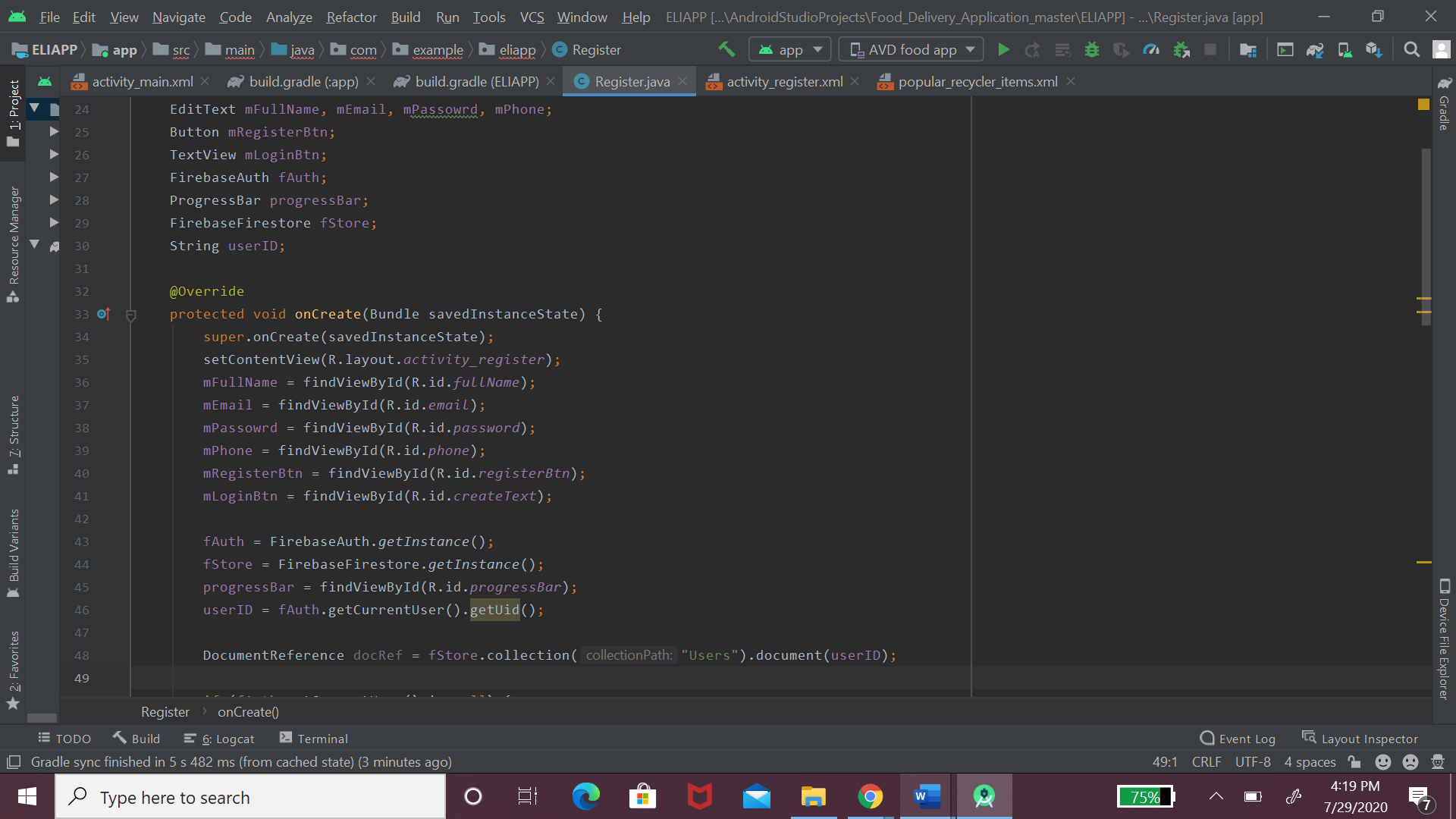
Appendix

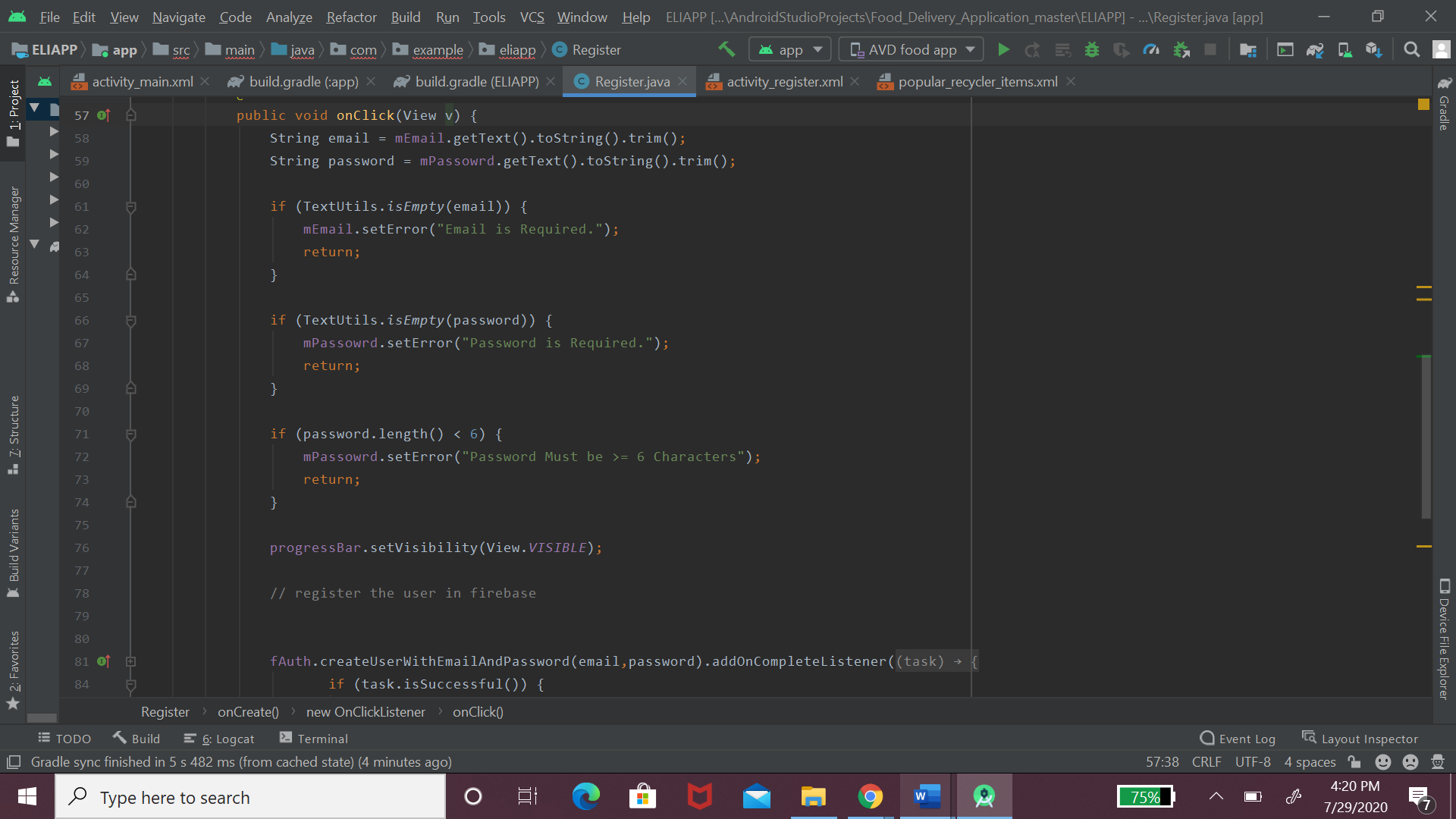
1. Source Code.
2. Source Code of Phone Login



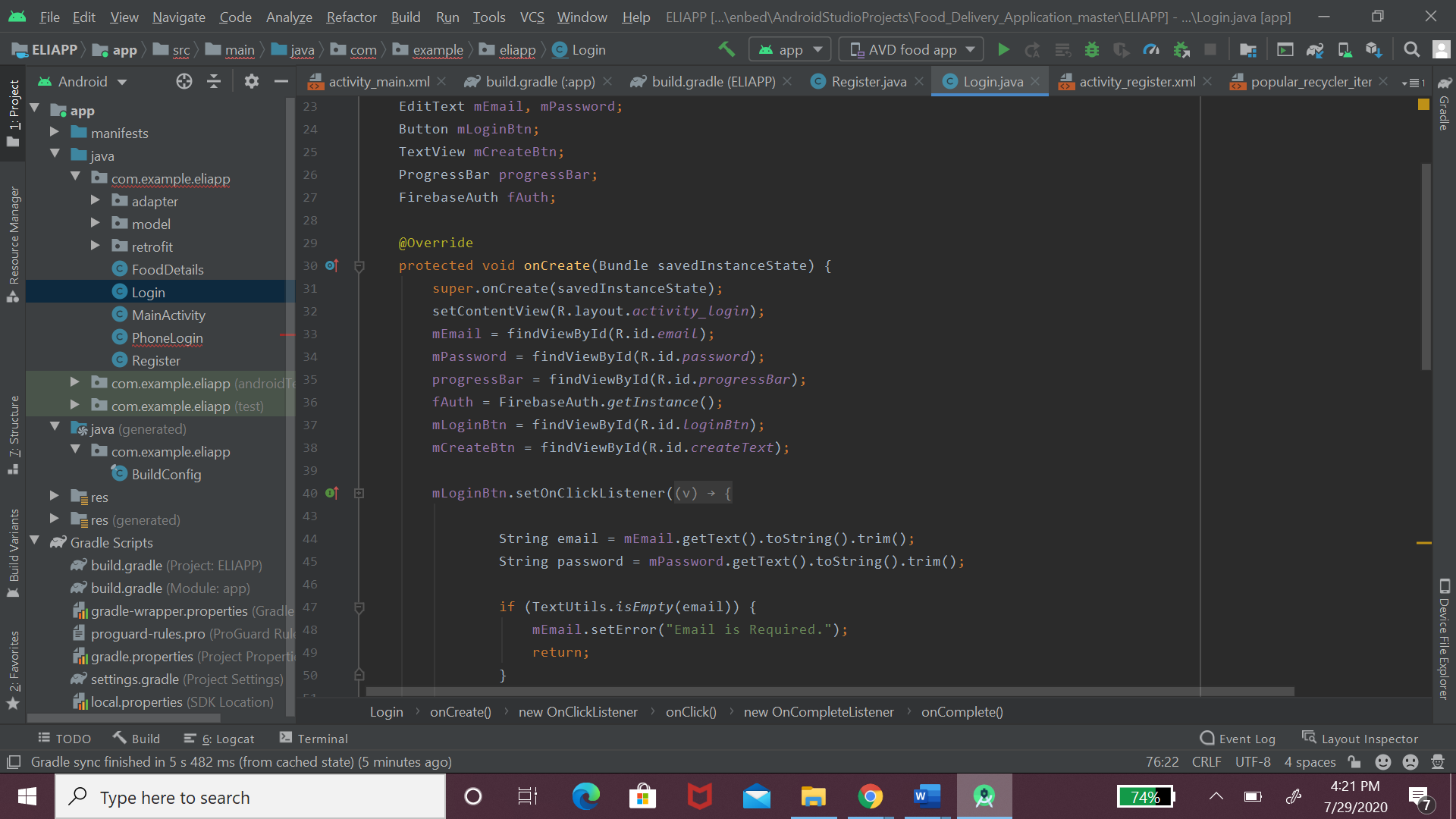


1. Source Code of Register User

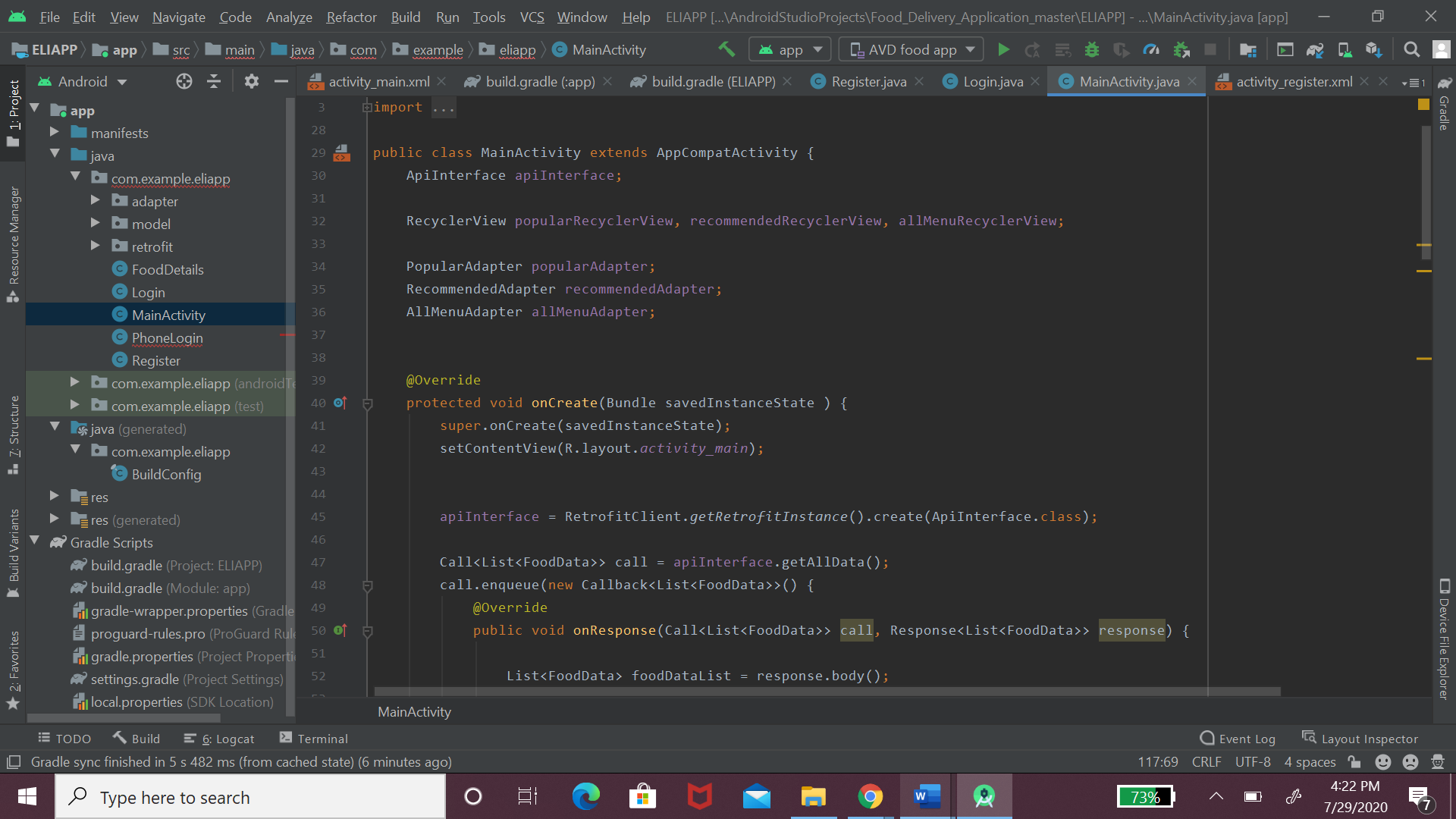


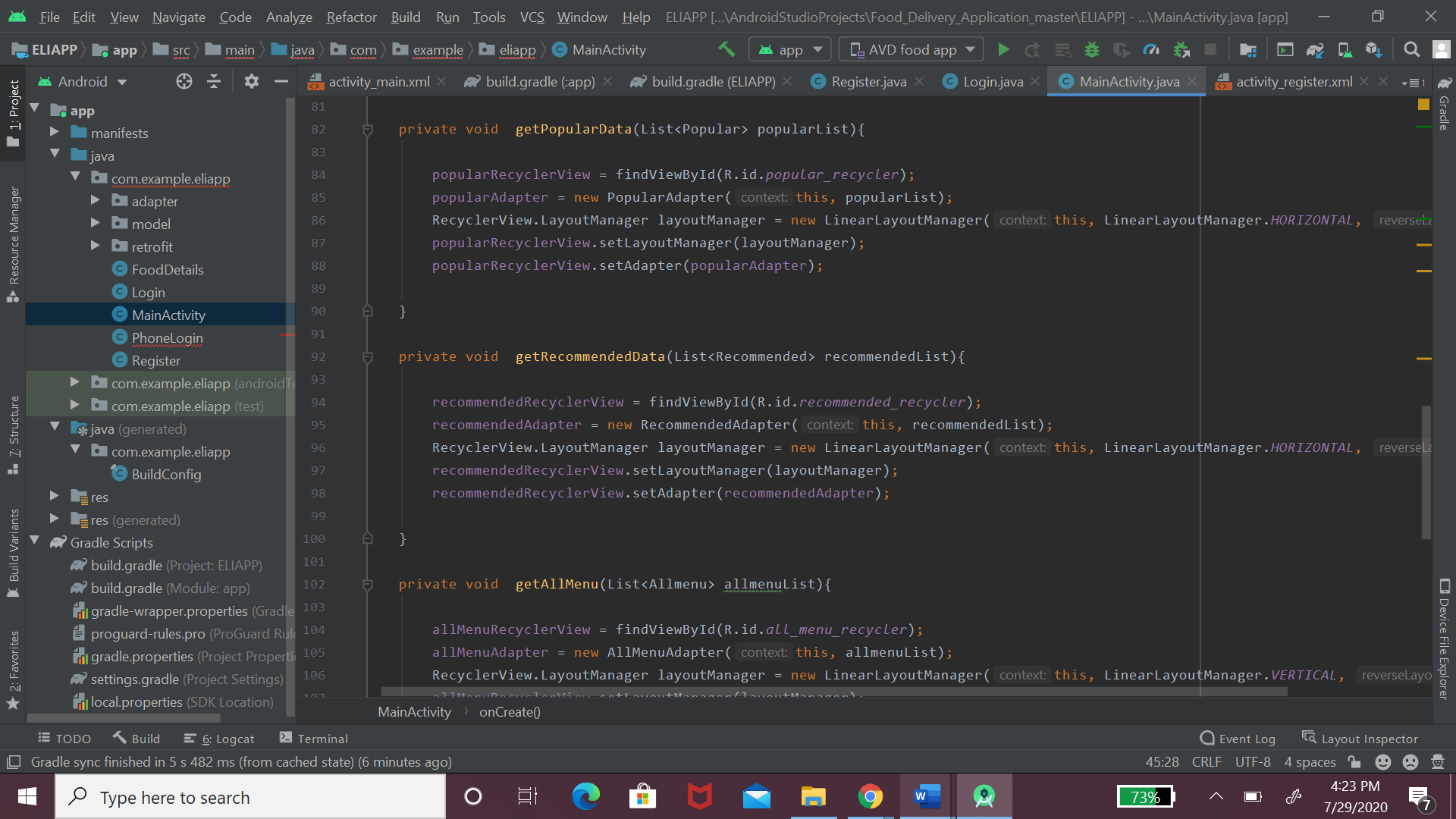


Source Code of Old User

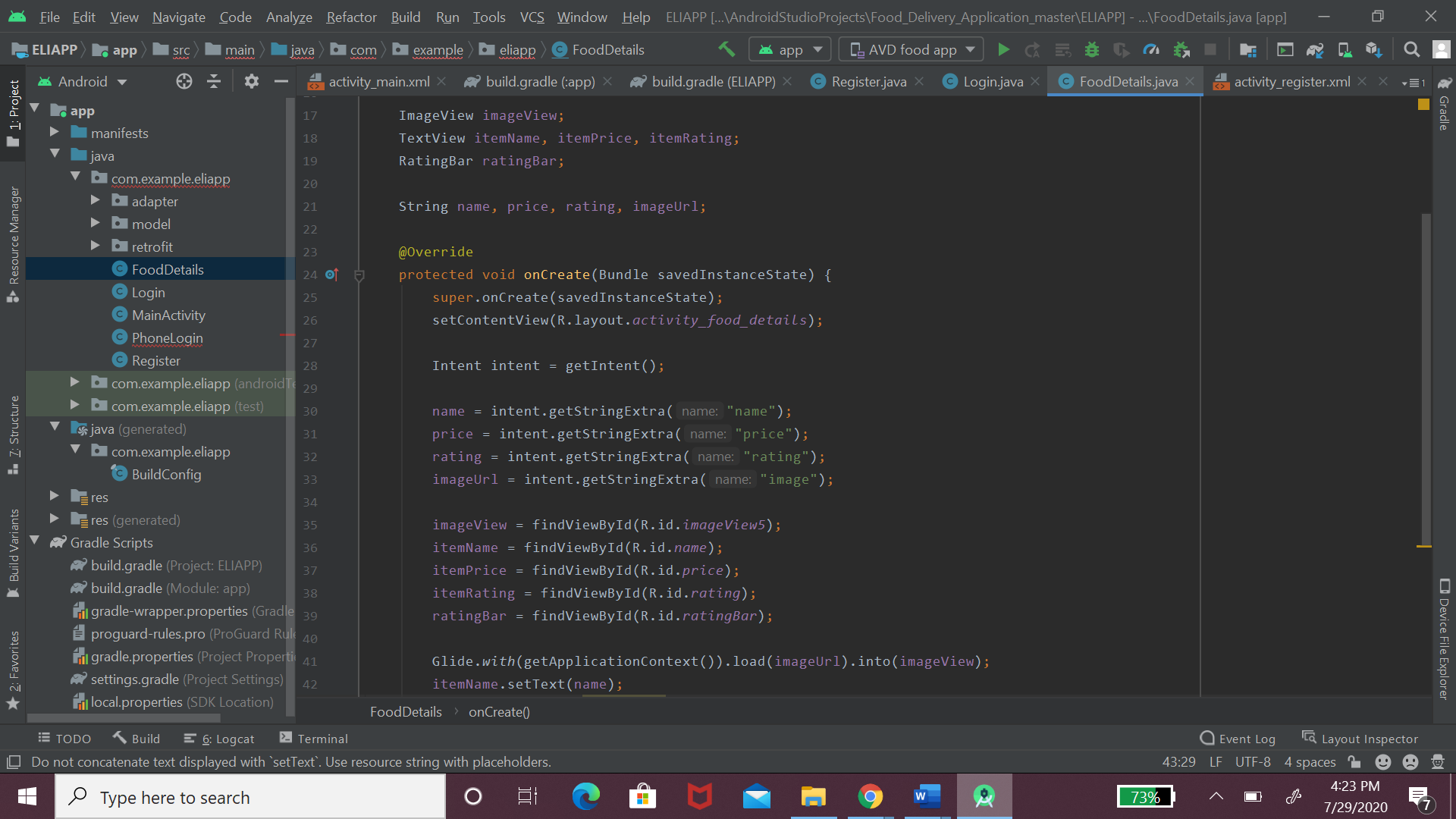


1. Source Code of Main Activity

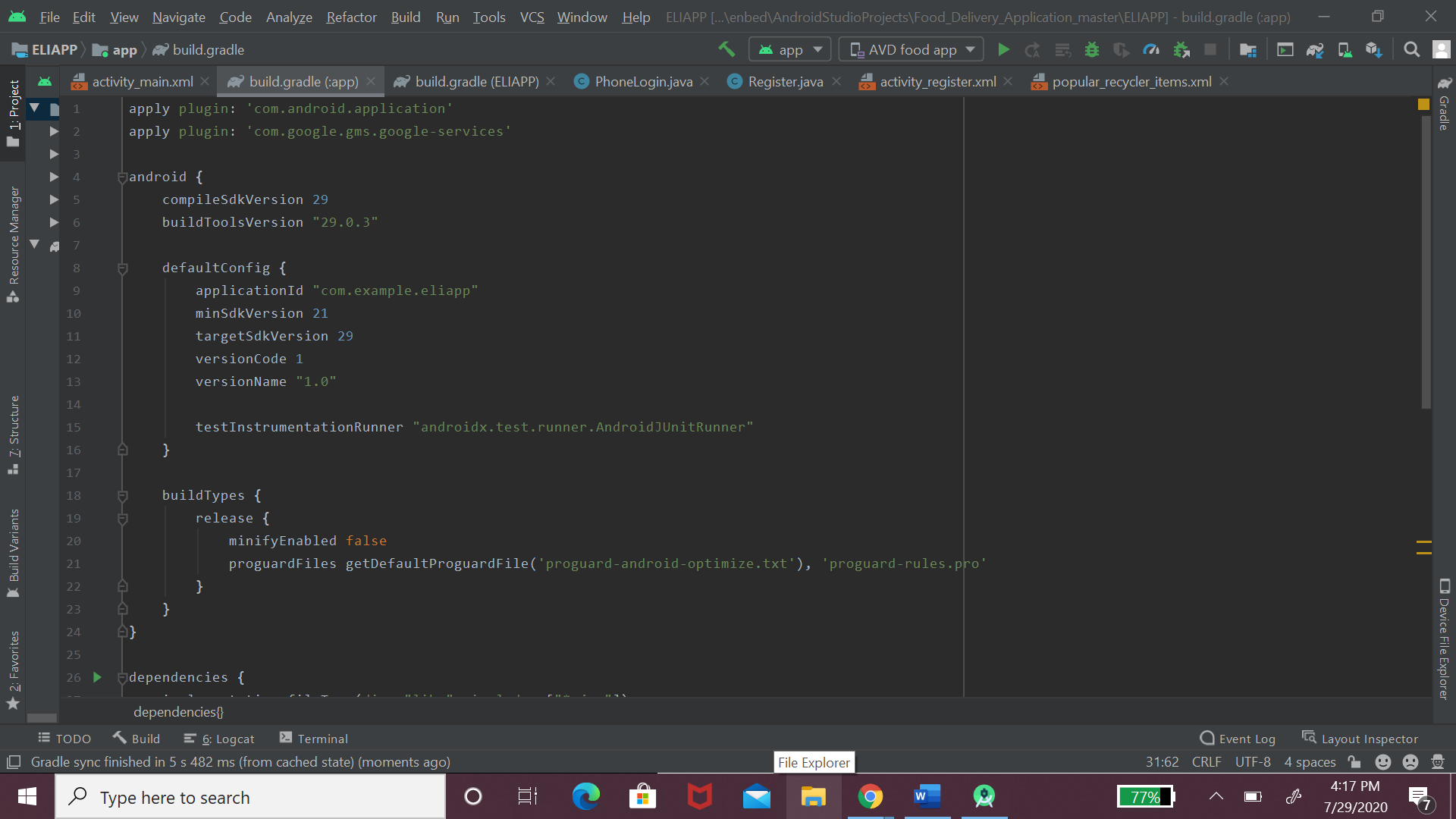




1. Source Code of Food Details



1. Source Code of Build Gradle



REFERENCES

[1] *use network service discover*. (n.d.). Developers. Retrieved July 7, 2020, from https://developer.android.com/training/connect-devices-wirelessly/nsd.

[2] “Http://ai2.Appinventor.mit.edu/Reference/Components/Storage.html.” *Storage*, ai2.appinventor.mit.edu/reference/components/storage.html.

[3] GitHub. 2020. [online] Available at: <https:// <https://github.com/thecodenest> > [Accessed 1 July 3 2020].

nk