ipeka
INTEGRATED
CHRISTIAN SCHOOL

“Augment-Eat”

SDD Assessment Task 2

Danyl Stephan Kok

Isaac Edwin Winoto

Nicholas Riykco Widjaja

12A

Table of Contents

1. Research Paper
    a. Problem Identification
        i. Needs of the Client
        ii. Boundaries of the Problem
    b. Process of Data Collection Method
        i. Online Survey
        ii. Interview
            1. Go Curry
            2. Bari-Uma Ramen
        iii. OnlineResearch
    c. Conversion of Data to Information
    d. Proposed Solution
        i. Programming Language
            1. C#
        ii. CASE Tools Used
            1. Unity
            2. Vuforia
            3. Android Studios
            4. Visual Studio
        iii. Paradigm
        iv. Software Development Approach
2. Software Documentation
    a. System Flowchart
    b. Structure Chart
    c. Context Diagram
3. The Actual Source Code
    a. Source Code
    b. Data Dictionary

1. RESEARCH PAPER

a. Problem Identification

How do restaurants effectively sell and present their food? For the majority of restaurants in the world, a physical menu is used to display the variety of food and drinks offered in the restaurant, together with their respective prices. As such, the menu becomes a huge factor when customers want to decide what to eat. It is a critical component to the restaurant's success, serving its primary purpose to “advertise” the offered food in the restaurant. However, we believe there are still some issues with how the menu system works, which can be further improved through technology to enhance user experience and advertise the restaurant’s products.

A huge issue of the physical menu in today’s world is its limitations to display information. Most menus provide the name of the dish and a small description of the ingredients in the dish. Several pictures are also usually attached into the menu, but not every item list has an image. The data we collected through surveys and interviews below show that menu with specific images result in consumers choosing those types of food more. Furthermore, especially in restaurants that serve only specific groups of people (ex. Korean BBQ, French), the difference in language and food vocabulary can also be an issue.

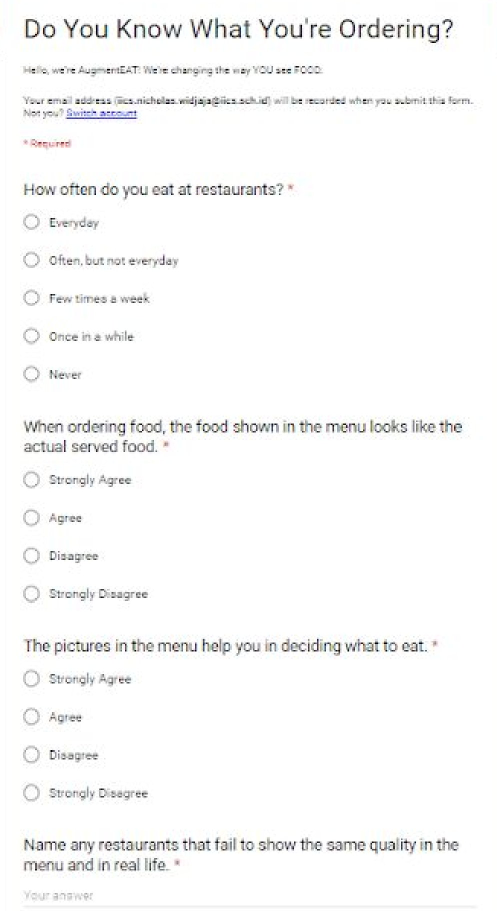
A more important issue that we discovered based on our surveys from a paper-based menu is the difference between the images in the menu and the actual output from the kitchen. Often times, customers find that the actual presented food is displayed differently, in size and quality, to the images shown in the menu. Furthermore, the image provided by the restaurant only limits the viewer to see the food from one angle. Customers would feel more satisfied if the food in the menu could be displayed as similar to the actual food. A tech feature that would allow the food to be viewed in 360 degrees in the menu would further boost the customer experience.

b. Process of Data Collection Method

After identifying these issues, our group used several data collection methods to identify the problems related to the attractiveness, effectiveness, and customer satisfaction of the paper-based menu.

Online Survey

Our online-based survey focused on a small sample space of key personnel. These personnel include IICS Senior High teachers and students, along with parents who voiced their opinions on the survey given regarding their purchasing behavior and the problems they may encounter in a paper-based menu. Below are the questions that we asked our key personnel before designing and thinking of our solution.



Do You Know What You're Ordering?

Hello, we're AugmentEAT! We're changing the way YOU see FOOD:

Your email address (iics.nicholas.widjaja@iics.sch.id) will be recorded when you submit this form. Not you? Switch account

* Required

How often do you eat at restaurants? *

○ Everyday

○ Often, but not everyday

○ Few times a week

○ Once in a while

○ Never

When ordering food, the food shown in the menu looks like the actual served food. *

○ Strongly Agree

○ Agree

○ Disagree

○ Strongly Disagree

The pictures in the menu help you in deciding what to eat. *

○ Strongly Agree

○ Agree

○ Disagree

○ Strongly Disagree

Name any restaurants that fail to show the same quality in the menu and in real life. *

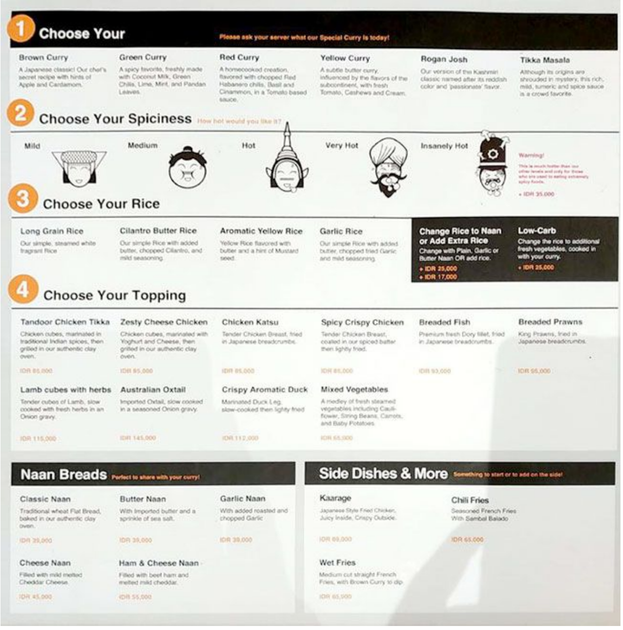
Your answer

i. Face-to-Face Interview

We also conducted a face-to-face interview with surrounding restaurants in Lippo Mall Puri, such as Go-Curry and BariUma Ramen. These interviews aim to identify any complications that the restaurants may have with customer choice and paper-based menu. We also asked their opinions on possible improvements to the menu and what they thought if emerging technologies, that could be our possible solutions, were added to the menu.

- **Go-Curry:**

Go-Curry's menu gives no image reference to their menu that serves Indian, Thai, and Japanese curry, which all might be unfamiliar to many Indonesian customers. Due to the nature of their menu (with only words and icons), we asked them whether there were any complaints from the customers regarding the menu. Although Go-Curry stated that there aren't a lot of complaints with the menu system, they believed that images would further resolve the issue of portion size that customers usually get confused with. Whenever customers ask the food size, they say it is "cukup untuk satu orang," or just right for one person. However, we still consider this description vague, as how much a person can eat is something relative. Being able to visualize the portion, especially in proportion to the plate and table mat, would help the users decide which dish would be best for them.



1 Choose Your Please ask your server what our Special Curry is today!

Brown Curry
A Japanese classic! Our chef's secret recipe with hints of Apple and Cardamom.

Green Curry
A spicy favorite, freshly made with Coconut Milk, Green Chilis, Lime, Mint, and Pandan Leaves.

Red Curry
A homecooked creation, flavored with chopped Red Habanero chilis, Basil and Cinammon, in a Tomato based sauce.

Yellow Curry
A subtle butter curry, influenced by the flavors of the subcontinent, with fresh Tomato, Cashews and Cream.

Rogan Josh
Our version of the Kashmiri classic named after its reddish color and 'passionate' flavor.

Tikka Masala
Although its origins are shrouded in mystery, this rich, mild, tumeric and spice sauce is a crowd favorite.

2 Choose Your Spiciness How hot would you like it?

Mild

Medium

Hot

Very Hot

Insanely Hot

Warning!
This is much hotter than our other levels and only for those who are used to eating extremely spicy foods.
+ IDR 35,000

3 Choose Your Rice

Long Grain Rice
Our simple, steamed white fragrant Rice

Cilantro Butter Rice
Our simple Rice with added butter, chopped Cilantro, and mild seasoning.

Aromatic Yellow Rice
Yellow Rice flavored with butter and a hint of Mustard seed.

Garlic Rice
Our simple Rice with added butter, chopped fried Garlic and mild seasoning.

Change Rice to Naan or Add Extra Rice
Change with Plain, Garlic or Butter Naan OR add rice.
+ IDR 25,000
+ IDR 17,000

Low-Carb
Change the rice to additional fresh vegetables, cooked in with your curry.
+ IDR 25,000

4 Choose Your Topping

Tandoor Chicken Tikka
Chicken cubes, marinated in traditional Indian spices, then grilled in our authentic clay oven.
IDR 85,000

Zesty Cheese Chicken
Chicken cubes, marinated with Yoghurt and Cheese, then grilled in our authentic clay oven.
IDR 85,000

Chicken Katsu
Tender Chicken Breast, fried in Japanese breadcrumbs.
IDR 85,000

Spicy Crispy Chicken
Tender Chicken Breast, coated in our spiced batter then lighty fried.
IDR 85,000

Breaded Fish
Premium fresh Dory fillet, fried in Japanese breadcrumbs.
IDR 93,000

Breaded Prawns
King Prawns, fried in Japanese breadcrumbs.
IDR 95,000

Lamb cubes with herbs
Tender cubes of Lamb, slow cooked with fresh herbs in an Onion gravy.
IDR 115,000

Australian Oxtail
Imported Oxtail, slow cooked in a seasoned Onion gravy.
IDR 145,000

Crispy Aromatic Duck
Marinated Duck Leg, slow-cooked then lighty fried
IDR 112,000

Mixed Vegetables
A medley of fresh steamed vegetables including Cauli-flower, String Beans, Carrots, and Baby Potatoes
IDR 65,000

Naan Breads Perfect to share with your curry!

Classic Naan
Traditional wheat Flat Bread, baked in our authentic clay oven.
IDR 39,000

Butter Naan
With imported butter and a sprinkle of sea salt.
IDR 39,000

Garlic Naan
With added roasted and chopped Garlic
IDR 39,000

Cheese Naan
Filled with mild melted Cheddar Cheese.
IDR 45,000

Ham & Cheese Naan
Filled with beef ham and melted mild cheddar.
IDR 55,000

Side Dishes & More Something to start or to add on the side!

Kaarage
Japanese Style Fried Chicken, Juicy Inside, Crispy Outside.
IDR 69,000

Chili Fries
Seasoned French Fries With Sambal Balado
IDR 65,000

Wet Fries
Medium cut straight French Fries, with Brown Curry to dip
IDR 65,000

- **Bari-Uma Ramen:**

We interviewed the product manager of Japanese food chain Bari-Uma Ramen, who provided his personal opinion about the menu system in his restaurant. According to him, the menu relies heavily on images to influence consumers' decisions. He stated that the restaurant sets the standard that the actual product should be very similar to what is displayed in the menu. According to him, pictures play a crucial role in influencing consumer's decisions, as they make certain products more eye-catching to the viewer. He added that to increase this attractiveness, Bari-Uma Ramen is entering a transition period to update their menus to meet customer needs moving forward. When we asked whether an AR-based menu, one of our potential solutions, could help boost his company, the manager responded that it would help revolutionize the menu concept, but would take time for both restaurants and consumers to adapt.

Play audio

The above QR code links to an online mp3 file of our interview with Bari-Uma Ramen.

ii. Augmented Reality Familiarity Survey

Shortly after, we randomly asked visitors in Lippo Mall Puri about their familiarity with Augmented Reality. About 63% of respondents, most of which realized what AR was when we mentioned some AR-based applications like Snapchat and Pokemon Go, were familiar with AR. This data speaks to why an AR-based menu may prove both innovative and relevant for users here in Jakarta.

iii. Online Research

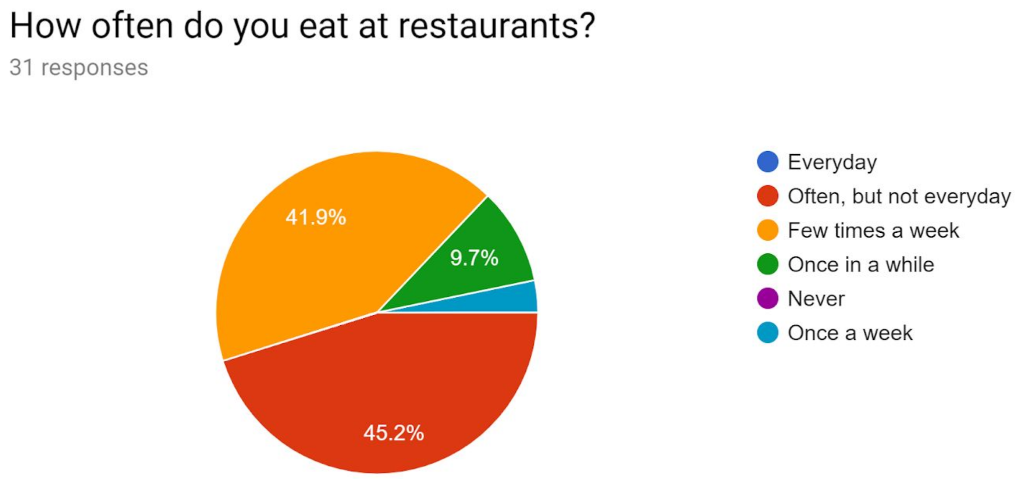
To add on, we also conducted online research to find out whether customers around the world believe that the actual food presented is similar to the food displayed on menus. We researched the stats that links customer choice and satisfaction with the layout, design and images of restaurants menus. We also researched the frequency of our customers visiting restaurants and customer ordering behavior.

According to the "*Ask Your Target Market*" survey conducted by Anne Pilon, 41% of correspondents eat out every week, and more than 75% eat out every month. Furthermore, 59% of correspondents state that they have to scan through the menu and images before deciding what to order in restaurants.

c. Conversion of Data to Information

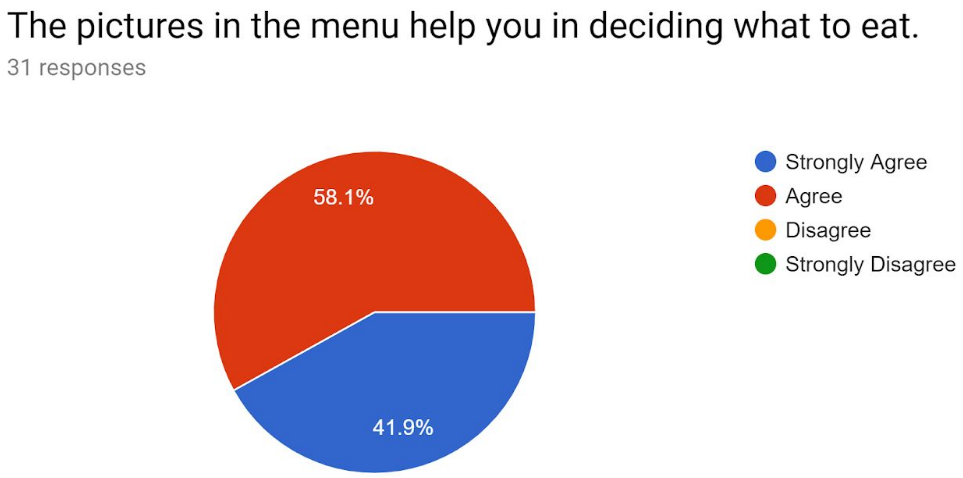
Our process of data collection has allowed us to determine the following conclusions:

- **Product Usage**



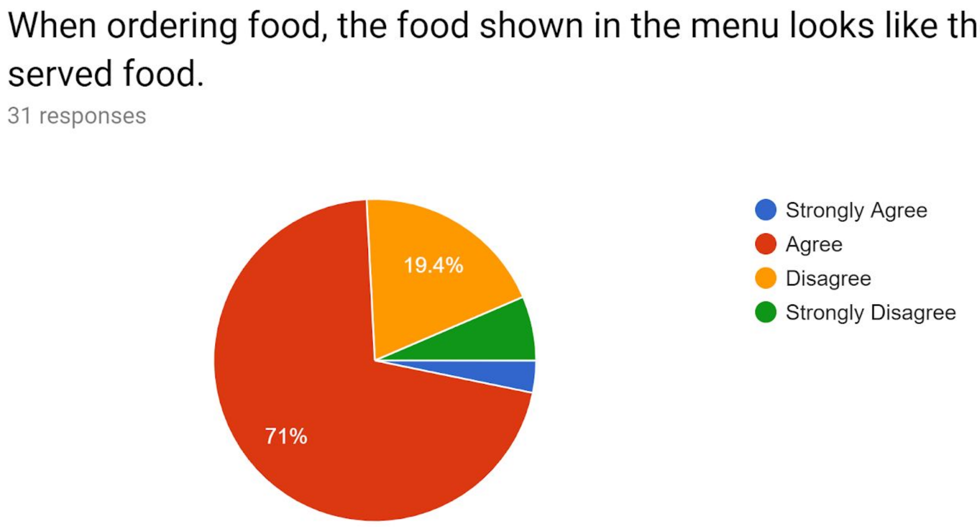
How often do you eat at restaurants?

31 responses



The pictures in the menu help you in deciding what to eat.

31 responses



When ordering food, the food shown in the menu looks like the actual served food.

31 responses

Our data obtained from online surveys shows that 41.9% of our correspondents eat out a few times a week, and more than 86% often eat in restaurants. Also, more than 58% state that the pictures in the menu play a deciding factor in what they eat. These two sets of data show us that almost 9 out of 10 consumers picks up a menu sometime during the week, and a majority of their food selections is influenced by the attractiveness of the images and display of the menu. Thus, it is crucial that the menu presents itself as best as it can for all consumers.

Our online survey also goes on to depict that more than 20% of correspondents believe that the images in the menu are different with the actual presented food. Although 20% seems to be a fairly low percentage, to put it into perspective, it shows us that 1 out of 5 consumers are not satisfied with the accuracy of the menu. This brings up the need for a food model that consumers can see beforehand that is similar to the actual food. We believe this can be done through today's advanced technologies, including Augmented Reality, which allows a 3D image to be superimposed once a placemat is scanned.

Since 63% of correspondents that were interviewed have used and understood what AR is, it would not be too difficult to adapt and use a new technology like AR in the restaurants. It is true that it would require some process for AR to be embraced by consumers, but it is likely to be effective enough to tackle our problems.

d. Proposed Solution

In consideration with the problems with paper-based menus that we discovered above, we decided to develop a menu-based augmented reality application that allows customers to view dishes in 3D and 360 degrees. By using augmented reality, a tech feature that superimposes a computer generated image or object that can be displayed on the user's view of the real world, a 3D model of the food item the users choose can pop up on the users' screen. We designed this software solution specifically to aid the practicality of physical menus and improve user experience with this new, cutting-edge technology aiming to solve the problems listed above.

This application, named as "Augment-Eat" by its developers, solves the problem where images in the menu may be limited and may not paint an accurate picture of the actual product. It allows us to depict an accurate 3D model of the dish where the customers are able to view it from different angles. The fact that our app allows users to see the food in its original size before it is served is also very advantageous, as users can estimate the portion of the food serving before ordering. Finally, with this menu-based application, restaurants are able to manually add new menus to their restaurant list and input adequate descriptions for each menu more efficiently.

<u>Specification Requirement for Software Solution (Menu-Based AR Mobile Application)</u>

- Able to work on all Android devices running operating system equivalent or greater than an Android Lollipop Level 22.
- Able to select from different restaurants
- Able to select the dish that the users want to superimpose in Augmented Reality
- Categorize the dishes into three sub categories: Food, Drink, Set Meals
- Show information and details regarding the description and pricing of the dish
- Able to portray every dish in Augmented Reality
- Make sure the size of the food is exact to what it is in actual size
- A help menu regarding the restaurant and menu list and AR-Camera operations.
- An end user agreement fully explained in the help menu and at the beginning of the application.

i. Programming Language - C#

For our software solution, we decided to use C# as our object-oriented backend programming language. This is primarily due to the Unity game engine, the main environment where we will develop our application and the user interface, only supporting C#. C# works with Microsoft Net. platform and is a supported language of Microsoft Visual Studios.

Although it was just our first time using C#, it was not difficult to adapt to it, as it is similar to other programming languages and very easy to understand. Especially in app development, C# is the perfect application because it allows us to see the whole app as the superclass, and dive into each feature and page as the subclasses in the application. The scalability and updatability of this language also aids us for our low budget and 3-month timespan.

ii. CASE Tools Used

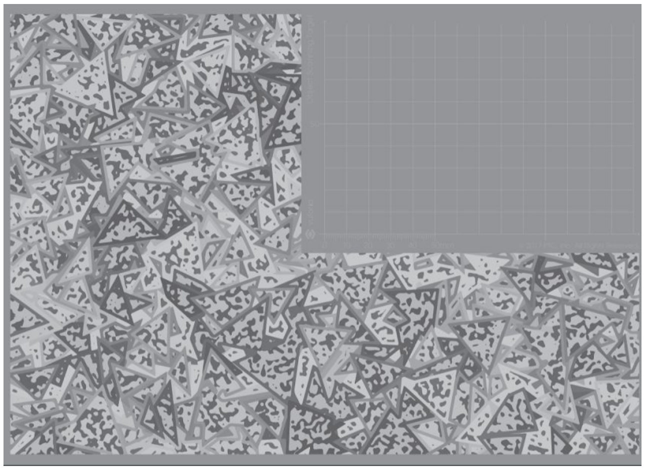
1. Unity

Unity is the engine platform to develop the UI. It is the environment in which we can apply Vuforia and Android Studios to create the Augmented Reality application. Although it is usually used for game development, in this project, we add project assets, buttons, sliders, libraries, and checkboxes among a variety of resources to further develop our user interface. Furthermore, Unity has a propriety of GUI tools that can be implemented in our application without the need to manually code them.

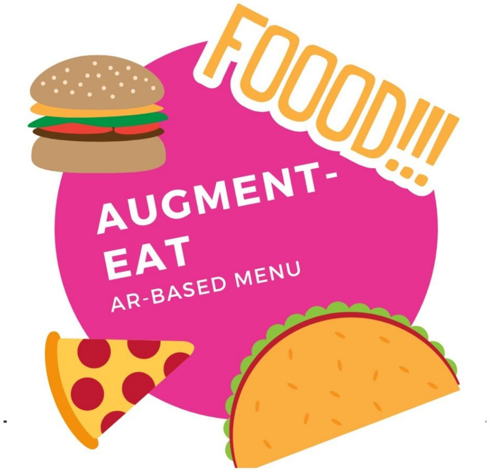
2. Vuforia Tools

Without Vuforia, there is no AR. Vuforia provides the necessary support for all our Augmented Reality endeavors. After creating a license key and adding the necessary image targets, we are able to include Vuforia plugins in Unity to enable our access to the AR camera.

Another useful tool that we took advantage of was Vuforia Object Scanner. The scanner allows us to make any object, 2D or 3D, into a target image. The following image below is our tracking image for the software solution. Through green dot points that scan parts of the object, we can render it into a 3D file with “.od” extension to be uploaded into our Vuforia Developer Target Manager.



With Vuforia, we are able to personalize a tracking image. Vuforia will automatically analyze our tracking image and determine key tracking points that the AR camera will be able to analyze and project our 3D Model into the tracking image. Furthermore, Vuforia is able to analyze the augmentability of our tracking image. The following image below represents the tracking image that we use for our application.



3. Android Studios

Android Studios serves as our integrated development environment to develop Android applications. Through an Android Studios SDK (software development kit) plugin within the Unity game engine, we are able to directly build the application within Unity itself. Furthermore, as we develop the application in Unity in Windows, the only supported option is to use Android

SDK to build our application. In the future, other CASE Tools such as Xamarin can be used to build our application on an iOS and Linux platform.

4. Visual Studio

Visual Studio is our integrated development environment to create our C# script. Visual Studio offers a plethora of tools that enable us to debug, test, analyze, and fulfill the needs of the software development process. Different scripts can be used to perform specific functions from operating menu buttons or adding event listeners to our AR objects. Furthermore, Visual Studios will automatically give feedback or error regarding the scripts of our code.

iii. Paradigm

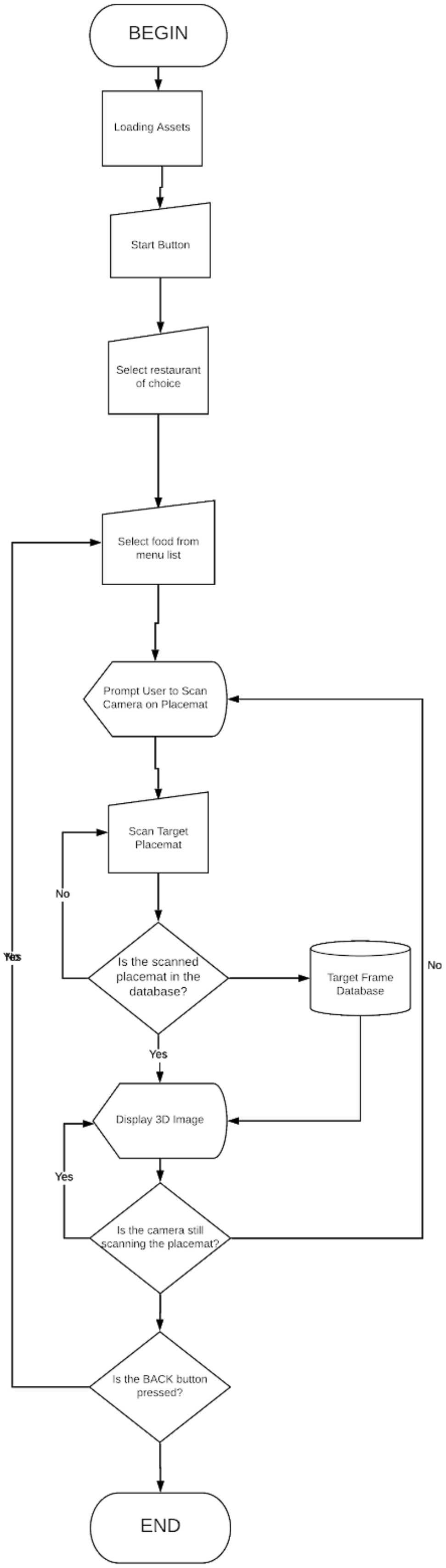
It would be a burden for us if we implemented the project using imperative or logic paradigms. Especially for graphical applications, object oriented programming offers the best advantages to a rather large system with many components, including the loading page, start menu, restaurant list, menu widget, and AR camera features. The inheritance property of OOP allows us to make small changes to certain features that we want to alter while not messing up the whole system and parent functions. If we used an imperative paradigm, one mistake could allow the whole system to not function. Troubleshooting, including the addition and removal of restaurants, food menus, and prices can be more effectively done with the OOP. The object oriented paradigm, when used in C# for Unity, is implemented by setting various scenes in one project, as can be seen later in the source code. Following our system flowchart, each part of our flowchart is represented by one or two scenes that each have different codes relating to their behaviors. Problem solving and debugging become way more effective through OOP.

iv. Software Development Approaches

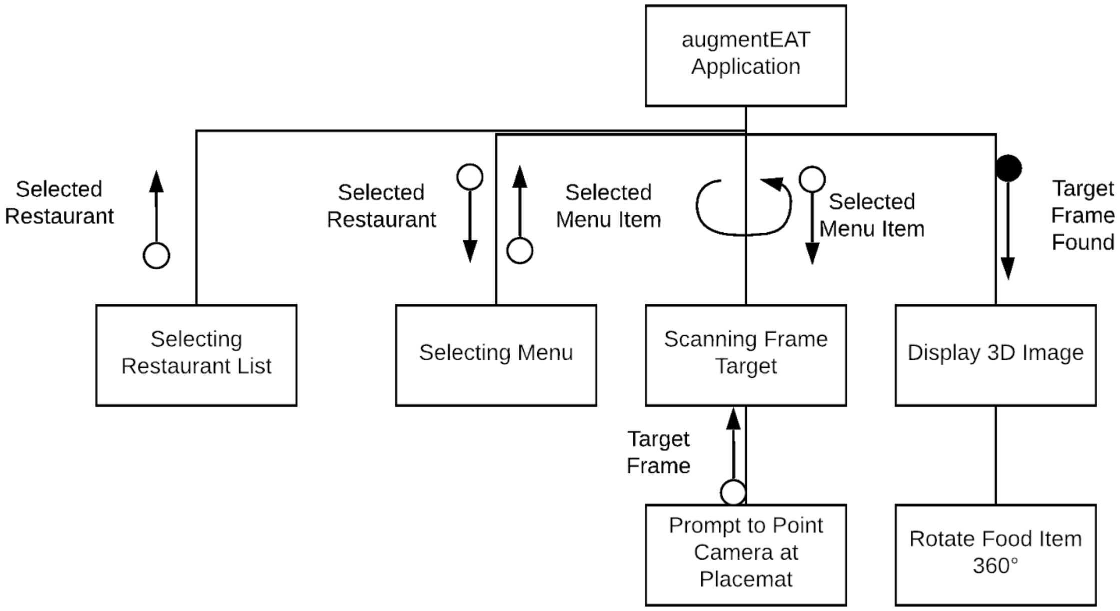
Based on the time constraint and nature of this project, we decided that a RAD approach would fit our project best. The RAD approach suits us because this project was undertaken with a lack of formal stages and fast development time. To add on, this Augmented Reality project was that it used a variety of CASE tools in order to develop the system, which is another characteristic of RAD. The reusing of code for C# from Stack Overflow and the assets that we imported for Unity also show why our project was undertaken with RAD. Libraries of code, combined with the Graphical User Interface IDE that we used, cohere with all the CASE tools we used to create a working Augmented Reality project.

2. SOFTWARE DOCUMENTATION

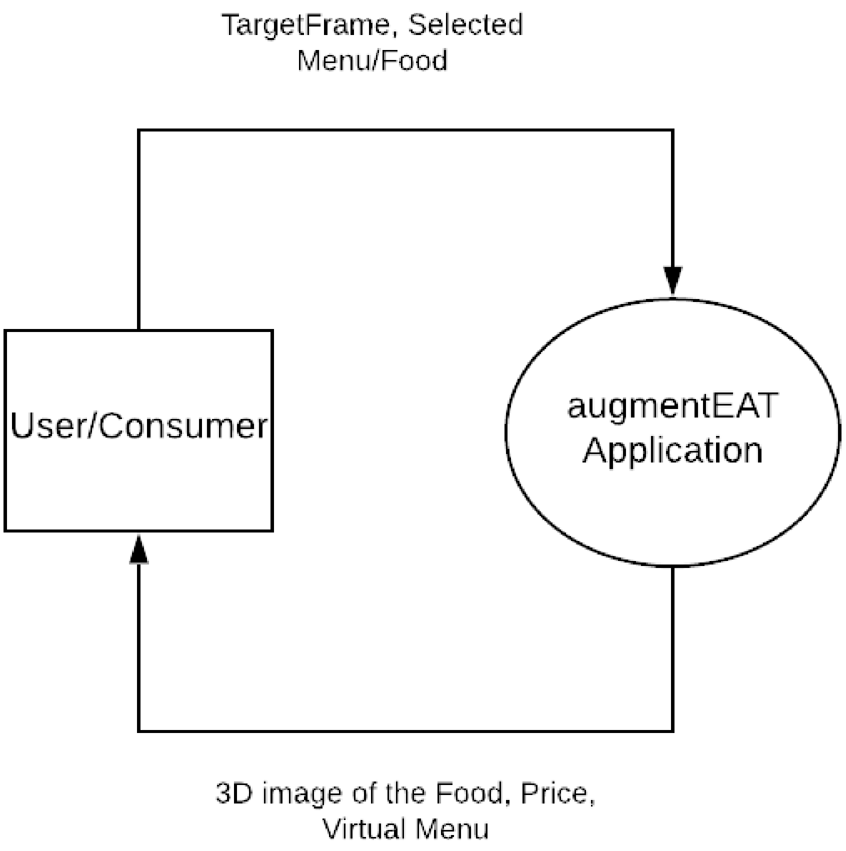
a. System Flowchart


BEGIN
Loading Assets
Start Button
Select restaurant of choice
Select food from menu list
Prompt User to Scan Camera on Placemat
Scan Target Placemat
No
Yes
Is the scanned placemat in the database?
Target Frame Database
No
Yes
Display 3D Image
Yes
Is the camera still scanning the placemat?
Is the BACK button pressed?
END

b. Structure Chart



c. Context Diagram


TargetFrame, Selected Menu/Food
User/Consumer
augmentEAT Application
3D image of the Food, Price, Virtual Menu

3. The Actual Source Code

a. Source Code

Source code legend:

Color	Indication
Green	Comments
Blue	Class Name
Purple	URL

ChangeScene (to change scene from one page to another)

```
using System.Collections;
//contains necessary classes and interfaces used in C#
using System.Collections.Generic;
using UnityEngine;
//indicates that C# is using Unity

public class changescene : MonoBehaviour
//this allows us to move from one scene to another
{
   public void changemenuscene(string scenename)
   //scene name refers to the scene we want to use
    {
        Application.LoadLevel(scenename);
        //this will load the next scene that we choose
    }
}
```

OpenWeb (links code to a web page)

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class OpenWeb : MonoBehaviour
//this allows a button to be linked to a web page
```

```
{
    public void btntwo()
   //a function that the button calls and triggers to open URL
    {
        Application.OpenURL
       (“https://iics.danylstephan.wixsite.com/augment-eat”);
        //load the url that we input in config
    }
}
```

UIManager (to manage and use the User Interface buttons)

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.UI; //to work with the UI
using DG.Tweening; //library used to call RectTransform function

public class UIManager : MonoBehaviour
/* this script allows the developer to move objects to selected
locations on the 2D and 3D plane using the dotween engine */
{
    public RectTransform HelpMenu, TermsandLicenseGuideline,
RestaurantList, MenuGuideline, ARCAMGuideline;
    // map the objects based on their fields to be transformed
    // Start is called before the first frame update
    void Start()
    {
        HelpMenu.DOAnchorPos(Vector2.zero, 0.25f);
        // change the position of the HelpMenu to the screen
        /* vector2 refers to the 2d plane and the subsequent
number is the X,Y plane */
        //0.25f is a float referring to speed of moving object
    }

    public void TermsandLicenseButton()
    /* set the function name as TermsandLicenseButton and is
applicable for all buttons */
    {
        HelpMenu.DOAnchorPos(new Vector2(-500, 0), 0.25f);
       // adjust HelpMenu so that it is unseen in the screen
        TermsandLicenseGuideline.DOAnchorPos(new Vector2(0, 0),
0.25f);
        /* change the position to show the
TermsandLicenseGuideline on the screen */
    }

    public void CloseTermsandLicenseButton()
    /* set the function name as CloseTermsandLicenseButton and is
applicable for all objects */
    {
        HelpMenu.DOAnchorPos(new Vector2(0, 0), 0.25f);
        //adjust HelpMenu so that it is seen in the screen
        TermsandLicenseGuideline.DOAnchorPos(new Vector2(1500,
0), 0.25f);
        /* change the position of the TermsandLicenseGuideline to
be unseen */
```

```
        }

        public void RestaurantListButton()
        {
            HelpMenu.DOAnchorPos(new Vector2(-500, 0), 0.25f);

            RestaurantList.DOAnchorPos(new Vector2(0, 0), 0.25f);
            /* change the position to show the RestaurantList on the
            screen */

        }

        public void CloseRestaurantListButton()
        {
            HelpMenu.DOAnchorPos(new Vector2(0, 0), 0.25f);
            RestaurantList.DOAnchorPos(new Vector2(3000, 0), 0.25f);
    // change the position of the RestaurantList to be unseen
        }
        public void MenuGuidelinesButton()
        {
            HelpMenu.DOAnchorPos(new Vector2(-500, 0), 0.25f);
            MenuGuideline.DOAnchorPos(new Vector2(0, 0), 0.25f);
    // change the position to show the MenuGuideline on the screen
        }

        public void CloseMenuGuidelinesButton()
        {
            HelpMenu.DOAnchorPos(new Vector2(0, 0), 0.25f);
            MenuGuideline.DOAnchorPos(new Vector2(4500, 0), 0.25f);
    // change the position of the MenuGuideline to be unseen
        }

        public void ARCameraGuidelines()
        {
            HelpMenu.DOAnchorPos(new Vector2(-500, 0), 0.25f);
            ARCAMGuideline.DOAnchorPos(new Vector2(0, 0), 0.25f);
    // change the position to show the ARCAMGuideline on the screen
        }

        public void CloseARCameraGuidelines()
        {
            HelpMenu.DOAnchorPos(new Vector2(0, 0), 0.25f);
            ARCAMGuideline.DOAnchorPos(new Vector2(6000, 0), 0.25f);
    // change the position of the ARCamGuideline to be unseen
        }
        // Update is called once per frame
        void Update()
        {

        }
    }
```

LoadingBar (loading the assets in the background)

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class LoadingBar : MonoBehaviour
{
    public GameObject loadingScreen;
    //creates a field for the Loading Screen
    public Slider slider;
    //creates a field for the slider
    public progressText;
    //text to show loading percentage progress
        public void LoadLevel(string scenename)
        //declare which scene we want to load
    {
        StartCoroutine(LoadAsynchronously(scenename));
        //call the coroutine of the AsyncOperation
    }

    IEnumerator LoadAsynchronously(string scenename)
    {
        AsyncOperation operation = SceneManager.LoadScene(scenename);
        /* load the scene of our choosing in the background as the
loading bar operation continues */

        loadingScreen.SetActive(true);
        //activates the loading screen
        while (!operation.isDone)
        // if operation is not done, the progress bar still executes
        {
            float progress = Mathf Clamp01(operation progress / .9f);
            //clamps a value between 0 and 1 for the operations
            slider.value = progress;
            //slider or the bar represents the progress of the scene
loading
            progressText.text = progress * 100 + "%";
            //text shows percentage loaded
            yield return null;
            //wait until next frame before repeating again
        }
    }
}
```

Panel_Opener (it is the “information” button in the restaurant list)

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class PanelOpener : MonoBehaviour
//<summary> open a panel if a button is clicked
{
    public GameObject Panel;
    //set Panel as base class that Unity can reference
    public void OpenPanel()
    //set event name to OpenPanel
    {
        if (Panel!=null)
        /* checks if the Panel is active or not, operator returns
true if it is not true */
        {
            bool isactive = Panel.activeSelf;
            //returns whether the panel is active or not
            Panel.SetActive(!isactive);
            //set the panel to active
        }
    }
}
```

MenuUIManager (to move and use the buttons in the Menu Interface)

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.UI; //to work with the UI
using DG.Tweening; //library used to call RectTransform function

public class UIMANAGER_MENU : MonoBehaviour
{
    public RectTransform DishMenu, DrinksMenu, SetDishMenu;
    // map the objects based on their fields to be transformed
    // Start is called before the first frame update
    void Start()
    {
        DishMenu.DOAnchorPos(Vector2.zero, 0.25f);
//change the position of the DishMenu screen to the camera
    }


    public void Dishmenu()
    {
        // display DishMenu, hide DrinksMenu and SetDishMenu
        DishMenu.DOAnchorPos(new Vector2(0, 0), 0.25f);
```

```
            DrinksMenu.DOAnchorPos(new Vector2(-2500, 0), 0.25f);
            SetDishMenu.DOAnchorPos(new Vector2(-2500, 0), 0.25f);
        }
        public void DrinksButton()
        {
            // display DrinksMenu, hide DishMenu and SetDishMenu
            DishMenu.DOAnchorPos(new Vector2(2500, 0), 0.25f);
            DrinksMenu.DOAnchorPos(new Vector2(0, 0), 0.25f);
            SetDishMenu.DOAnchorPos(new Vector2(-2500, 0), 0.25f);
        }

        public void SetDishButton()
        {
            // display SetDishMenu, hide DrinksMenu and DishMenu
            DishMenu.DOAnchorPos(new Vector2(-2500, 0), 0.25f);
            SetDishMenu.DOAnchorPos(new Vector2(0, 0), 0.25f);
            DrinksMenu.DOAnchorPos(new Vector2(-2500, 0), 0.25f);
        }

        // Update is called once per frame
        void Update()
        {

        }
    }
```

LeanRotate

```
using UnityEngine;

namespace Lean.Touch
{
    //This script allows you to transform the current GameObject
    [HelpURL(LeanTouch.HelpUrlPrefix + "LeanRotate")]
    public class LeanRotate : MonoBehaviour
    {
        [Tooltip("Ignore fingers with StartedOverGui?")]
        public bool IgnoreStartedOverGui = true;

        [Tooltip("Ignore fingers with IsOverGui?")]
        public bool IgnoreIsOverGui;

        [Tooltip("Allows you to force rotation with a specific amount
of fingers (0 = any)")]
        public int RequiredFingerCount;

        [Tooltip("Does rotation require an object to be selected?")]
        public LeanSelectable RequiredSelectable;

        [Tooltip("The camera we will be used to calculate relative
rotations (None = MainCamera)")]
        public Camera Camera;

```

```
          [Tooltip("Should the rotation be performanced relative to the
finger center?")]
          public bool Relative;

#if UNITY_EDITOR
          protected virtual void Reset()
          {
               Start();
          }
#endif

          protected virtual void Start()
          {
               if (RequiredSelectable == null)
               {
                    RequiredSelectable = GetComponent<LeanSelectable>();
               }
          }

          protected virtual void Update()
          {
               // Get the fingers we want to use
               var fingers =
LeanSelectable.GetFingers(IgnoreStartedOverGui, IgnoreIsOverGui,
RequiredFingerCount, RequiredSelectable);

               // Calculate the rotation values based on these fingers
               var twistDegrees = LeanGesture.GetTwistDegrees(fingers);

```

```
            if (twistDegrees != 0.0f)
            {
                if (Relative == true)
                {
                    var twistScreenCenter =
LeanGesture.GetScreenCenter(fingers);

                    if (transform is RectTransform)
                    {
                        TranslateUI(twistDegrees,
twistScreenCenter);
                        RotateUI(twistDegrees);
                    }
                    else
                    {
                        Translate(twistDegrees, twistScreenCenter);
                        Rotate(twistDegrees);
                    }
                }
                else
                {
                    if (transform is RectTransform)
                    {
                        RotateUI(twistDegrees);
                    }
                    else
                    {
                        Rotate(twistDegrees);
                    }
                    }
               }
          }

          protected virtual void TranslateUI(float twistDegrees, Vector2
twistScreenCenter)
          {
               // Screen position of the transform
               var screenPoint =
RectTransformUtility.WorldToScreenPoint(Camera, transform.position);

               // Twist screen point around the twistScreenCenter by
twistDegrees
               var twistRotation = Quaternion.Euler(0.0f, 0.0f,
twistDegrees);
               var screenDelta   = twistRotation * (screenPoint -
twistScreenCenter);

               screenPoint.x = twistScreenCenter.x + screenDelta.x;
               screenPoint.y = twistScreenCenter.y + screenDelta.y;

               // Convert back to world space
               var worldPoint = default(Vector3);

               if
(RectTransformUtility.ScreenPointToWorldPointInRectangle(transform.parent
as RectTransform, screenPoint, Camera, out worldPoint) == true)
               {
                    transform.position = worldPoint;
               }
          }

          protected virtual void Translate(float twistDegrees, Vector2
twistScreenCenter)
          {
               // Make sure the camera exists
               var camera = LeanTouch.GetCamera(Camera, gameObject);

               if (camera != null)
               {
                    // Screen position of the transform
                    var screenPoint =
camera.WorldToScreenPoint(transform.position);

                    // Twist screen point around the twistScreenCenter by
twistDegrees
                    var twistRotation = Quaternion.Euler(0.0f, 0.0f,
twistDegrees);
                    var screenDelta   = twistRotation *
((Vector2)screenPoint - twistScreenCenter);

                    screenPoint.x = twistScreenCenter.x + screenDelta.x;
                    screenPoint.y = twistScreenCenter.y + screenDelta.y;

                    // Convert back to world space
                    transform.position =
camera.ScreenToWorldPoint(screenPoint);
               }
```

```
            else
            {
                Debug.LogError("Failed to find camera. Either tag
your cameras MainCamera, or set one in this component.", this);
            }
        }

        protected virtual void RotateUI(float twistDegrees)
        {
            transform.rotation *= Quaternion.Euler(0.0f, 0.0f,
twistDegrees);
        }

        protected virtual void Rotate(float twistDegrees)
        {
            // Make sure the camera exists
            var camera = LeanTouch.GetCamera(Camera, gameObject);

            if (camera != null)
            {
                var axis =
transform.InverseTransformDirection(camera.transform.forward);

                transform.rotation *=
Quaternion.AngleAxis(twistDegrees, axis);
            }
            else
            {
                Debug.LogError("Failed to find camera. Either tag
your cameras MainCamera, or set one in this component.", this);
            }
        }
    }
}
```

LeanTranslate

```
using UnityEngine;

namespace Lean.Touch
{
    /* This script allows you to translate the current GameObject
relative to the camera. */
    [HelpURL(LeanTouch.HelpUrlPrefix + "LeanTranslate")]
    public class LeanTranslate : MonoBehaviour
    {
        [Tooltip("Ignore fingers with StartedOverGui?")]
        public bool IgnoreStartedOverGui = true;

        [Tooltip("Ignore fingers with IsOverGui?")]
        public bool IgnoreIsOverGui;

        [Tooltip("Ignore fingers if the finger count doesn't match? (0
= any)")]
        public int RequiredFingerCount;

```

```
          [Tooltip("Does translation require an object to be
selected?")]
          public LeanSelectable RequiredSelectable;

          [Tooltip("The camera the translation will be calculated using
(None = MainCamera)")]
          public Camera Camera;

#if UNITY_EDITOR
          protected virtual void Reset()
          {
               Start();
          }
#endif

          protected virtual void Start()
          {
               if (RequiredSelectable == null)
               {
                    RequiredSelectable = GetComponent<LeanSelectable>();
               }
          }

          protected virtual void Update()
          {
               // Get the fingers we want to use
               var fingers =
LeanSelectable.GetFingers(IgnoreStartedOverGui, IgnoreIsOverGui,
RequiredFingerCount, RequiredSelectable);
```

```

               // Calculate the screenDelta value based on fingers
               var screenDelta = LeanGesture.GetScreenDelta(fingers);

               if (screenDelta != Vector2.zero)
               {
                    // Perform the translation
                    if (transform is RectTransform)
                    {
                         TranslateUI(screenDelta);
                    }
                    else
                    {
                         Translate(screenDelta);
                    }
               }
          }

          protected virtual void TranslateUI(Vector2 screenDelta)
          {
               // Screen position of the transform
               var screenPoint =
RectTransformUtility.WorldToScreenPoint(Camera, transform.position);

               // Add the deltaPosition
               screenPoint += screenDelta;

               // Convert back to world space
               var worldPoint = default(Vector3);
```

```

            if
(RectTransformUtility.ScreenPointToWorldPointInRectangle(transform.parent as RectTransform, screenPoint, Camera, out worldPoint) == true)
            {
                transform.position = worldPoint;
            }
        }

        protected virtual void Translate(Vector2 screenDelta)
        {
            // Make sure the camera exists
            var camera = LeanTouch.GetCamera(Camera, gameObject);

            if (camera != null)
            {
                // Screen position of the transform
                var screenPoint =
camera.WorldToScreenPoint(transform.position);

                // Add the deltaPosition
                screenPoint += (Vector3)screenDelta;

                // Convert back to world space
                transform.position =
camera.ScreenToWorldPoint(screenPoint);
            }
            else
            {
                    Debug.LogError("Failed to find camera. Either tag
your camera as MainCamera, or set one in this component.", this);
               }
          }
     }
}
```

b. Data Dictionary

AugmentEat Data Dictionary					
Name	**Data type**	**Length**	**Scope**	**Purpose**	**Example**
scenename	String	50	Local	Defines the next scene that is passed by the changemenuscene parameter to the LoadLevel function. This is declared inside the Unity engine	RestoList (restaurant list page)
vector2(X,Y)	Array of Integers	-2,147,483,648 to 2,147,483,648	Local	Move the object of our choosing to an X,Y location only on the 2D plane	Vector2(0, 0) (set the game object to 0,0 location)
HelpMenu	String	50	Local	HelpMenu is a field that is defined elsewhere inside the Unity Engine. Meaning, it is a variable that stores an object.	HelpMenu==helpmenu (HelpMenu is our variable name whereas helpmenu is our actual game object)
TermsandLicenseGuideline	String	50	Local	TermsandLicenseGuideline refers to a field that is defined elsewhere inside the Unity Engine. Meaning, it is	TermsandLicenseGuideline == Terms and License Guideline

				a variable that stores an object.	
RestaurantList	String	50	Local	RestaurantList is a field that is defined elsewhere inside the Unity Engine. Meaning, it is a variable that stores an object	RestaurantList == Restaurant List
MenuGuideline	String	50	Local	MenuGuideline is a field that is defined elsewhere inside the Unity Engine. Meaning, it is a variable that stores an object	MenuGuideline == Menu Guideline
ARCamGuideline	String	50	Local	ARCamGuideline is a field that is defined elsewhere inside the Unity Engine. Meaning, it is a variable that stores an object	ARCamGuideline == AR Camera Guideline
IgnoreStartedOverGui	Boolean	1	Local	It determines whether or not fingers have to be started over GUI	True

IgnoreIsOverGui	Boolean	1	Local	It determines whether or not the fingers is over GUI	True
RequiredFingerCount	Integer	2	Local	It gives the number of required fingers	1
Relative	Boolean	1	Local	It determines whether the rotation is performed relative to the finger center	True
LoadingScreen	string	50	Local	Create a field where the loading screen is defined elsewhere in the Unity engine.	LoadingScreen == Loading Screen
Slider	string	50		Create a field where the Slider is defined elsewhere in the Unity engine.	Slider == Slider
ProgressText.text	string	50		Create a field where the ProgressText is defined elsewhere in the Unity engine. It is made up of integer and characters.	98%

operations	string	50	Local	Loading the next scene of our own choosing	One scene → another scene (difficult to show example)
progress	float	9	Local	Gives value of loading operations between 0 to 1	0.45
DishMenu	String	50	Local	DishMenu is a field that is defined elsewhere inside the Unity Engine. Meaning, it is a variable that stores an object.	DishMenu == dishmenu (HelpMenu is our variable name whereas helpmenu is our actual game object)
DrinksMenu	String	50	Local	DrinksMenu is a field that is defined elsewhere inside the Unity Engine. Meaning, it is a variable that stores an object.	DrinksMenu==drinks menu (DrinksMenu is our variable name whereas drinksmenu is our actual game object)
SetDishMenu	String	50	Local	SetDishMenu is a field that is defined elsewhere inside the Unity Engine. Meaning, it is a variable that stores an object.	SetDishMenu==setdishpmenu (SetDishMenu is our variable name whereas helpmenu is our actual game object)

fingers	Array of Integers and Boolean	4	Local	Get the fingers we want to use	(True, true, 2, true)
screenPoint	Array of camera and integers	2	Local	Screen position of the transform	(Camera, position)
worldPoint	Array of 3D points	500	Local	Converts back to world space	A 3D position, difficult to show example