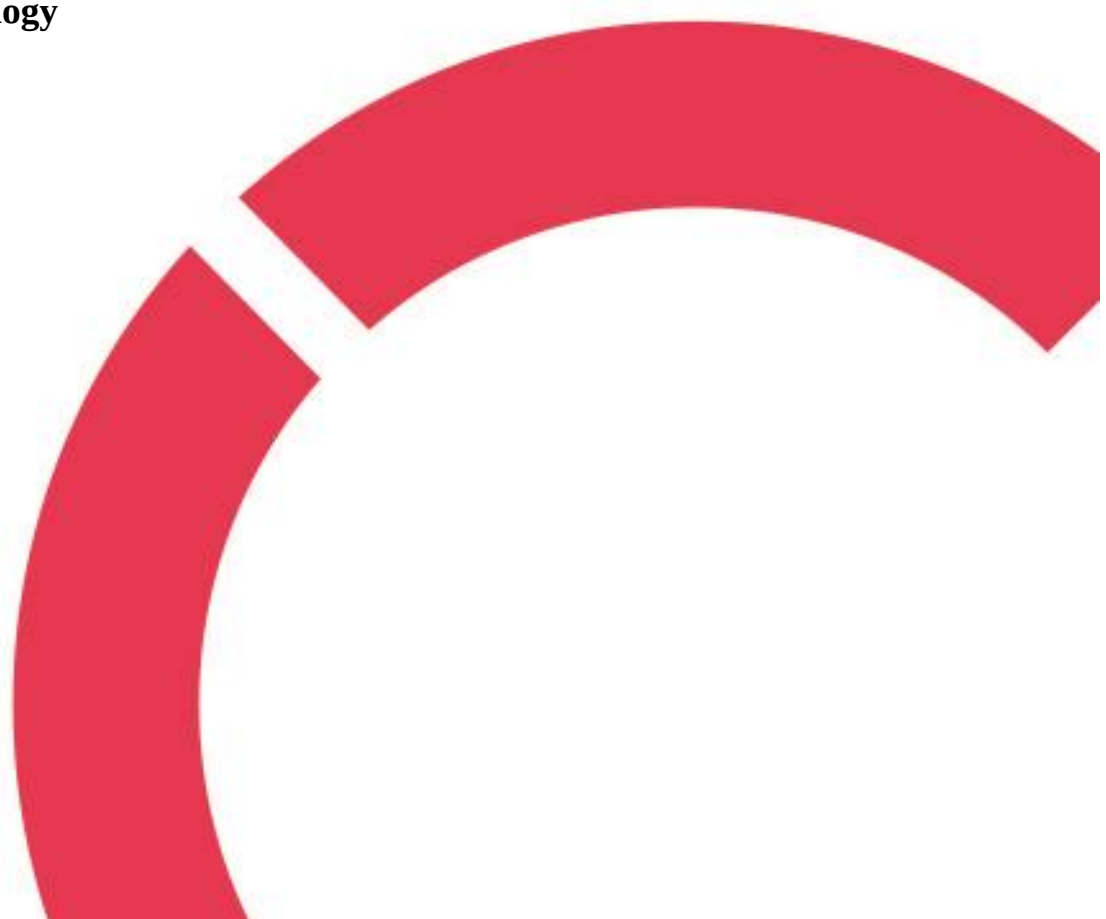
MANH MAN LE

DESIGN AND CREATE MOBILE APPLICATION
Case: Pizza Application

Thesis
CENTRIA UNIVERSITY OF APPLIED SCIENCES
Information Technology
December 2022



ABSTRACT

Centria University of Applied Sciences	Date December 2022	Author Manh Man Le
Degree programme Information Technology		
Name of thesis DESIGN AND CREATE A MOBILE APPLICATION		
Centria supervisor Kauko Kolehmanen		**Pages**
Instructor representing commissioning institution or company Kauko Kolehmanen		

During the last 10 years, because of the development of technology, mobile phones have been popular in many industrial fields, especially the service sector.

The aim of this thesis was to gain insight into the way to build an Android mobile app from the beginning to the final product. It is mainly divided into 4 parts: building user interface, creating database, developing Android, and testing.

"Kotipizza 383" is the mobile application which was developed in this project. The application still needs to be improved in the future, but it has been successfully deployed and meets all the requirements set for it in the beginning.

Key words

Android Studio, AVD, Java file, Manifest file, User interface

CONCEPT DEFINITIONS

AVD	Android Virtual Device
APK	Android Package Kit
ER	Entity-relationship diagram
JSON	JavaScript Object Notation (JSON)
XML	Extensible Markup Language
UI	User Interface
UX	User Experience
OS	Operating System

ABSTRACT
CONCEPT DEFINITIONS

FIGURE

1 INTRODUCTION

Nowadays, smartphone is an indispensable item for everyone. Because of the convenience, agility and accuracy, many industries are enhancing the customer experience with mobile apps, and the restaurant industry is no exception. Those are the reasons why mobile development is becoming important. Mobile development is the term for IT engineers to design and create a mobile application by choosing a programming language (Java, Python, Apk). From the parameters of Statcounter website, Android OS is currently the leading operating system in the world. Until March 2021, Android devices account 76.57 percent of the market, more than three times of iOS, which has only 22.61 percent. Android devices have become more diverse. At the beginning, it was only for smartphones. However, nowadays, there are lots of tablets, televisions, which are also using Android OS. The growth in numbers of Android devices has a very promising future. Therefore, it is a good idea to create a mobile application in Android OS.

The idea of this thesis topic came from the development of smartphones and the increase of smartphone users. Currently, most of the restaurants focus on developing mobile applications to reach more ranges of customers. Beside software for waitress or manager, mobile application for customers is also important. It helps to improve business popularity and works as another source of advertising and marketing for the business. Moreover, for customers, it helps them in choosing dishes, booking reservation, and using other services easily. The purpose of this thesis was to present how to create "Kotipizza 383" mobile application from beginning to final product.

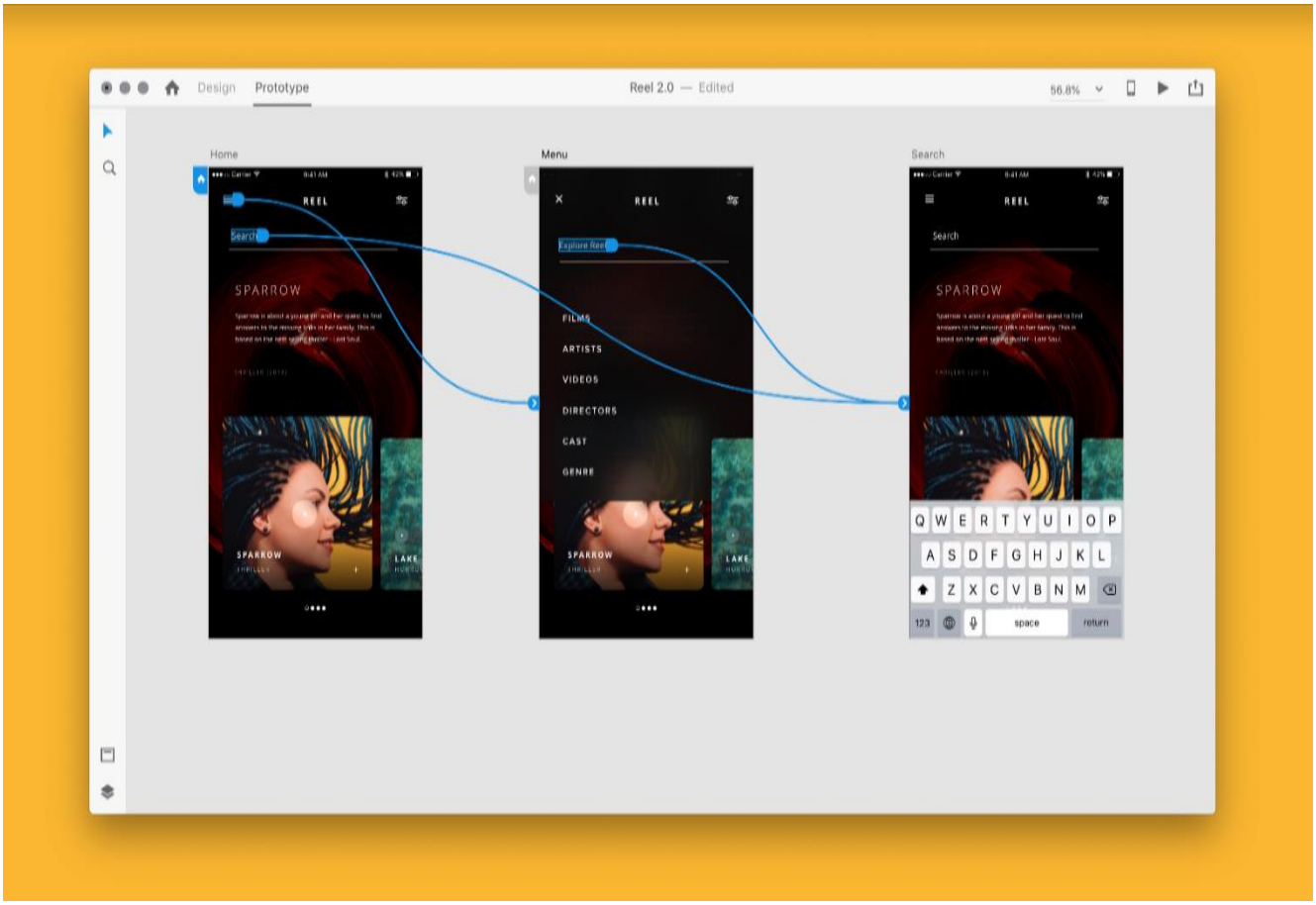
Developing mobile applications is not an easy job. Just like creating a website, developers need to start from the user interface, then write in programming language (Java) to make sure the application runs correctly. This thesis is divided into three parts: design UI, creating database and coding Android. Besides the introduction of theories about user interface as well as Android OS, this thesis also guides the steps to create UI and Android apps. Additionally, other support applications, such as: Adobe XD, Adobe Photoshop for UI and Android Studio for Android Development are also provided so that people can learn more about the benefits it offers.

2 USER INTERFACE DEVELOPMENT

Based on research from Forrester Research, a website or a mobile application with a good user interface increases conversion rates by up to 200%. That is the reason most companies pay attention to UI/UX design to improve the quality of the application, as well as to make the application easy to use for users. While UX Design stands for User experience Design, which is more analytical and technical, UI Design is the abbreviation for User Interface Design that focus on graphic design (Lamprecht 2019). User interface is the working space between users and machine. It conveys messages from designers, service providers, and products to users. It includes all elements that users can see, for example: layout, colour, text, pictures and so on. Designers need to produce a good user interface which satisfies the following factors: intuitive, efficient, and enjoyable. (Moreno 2014.)

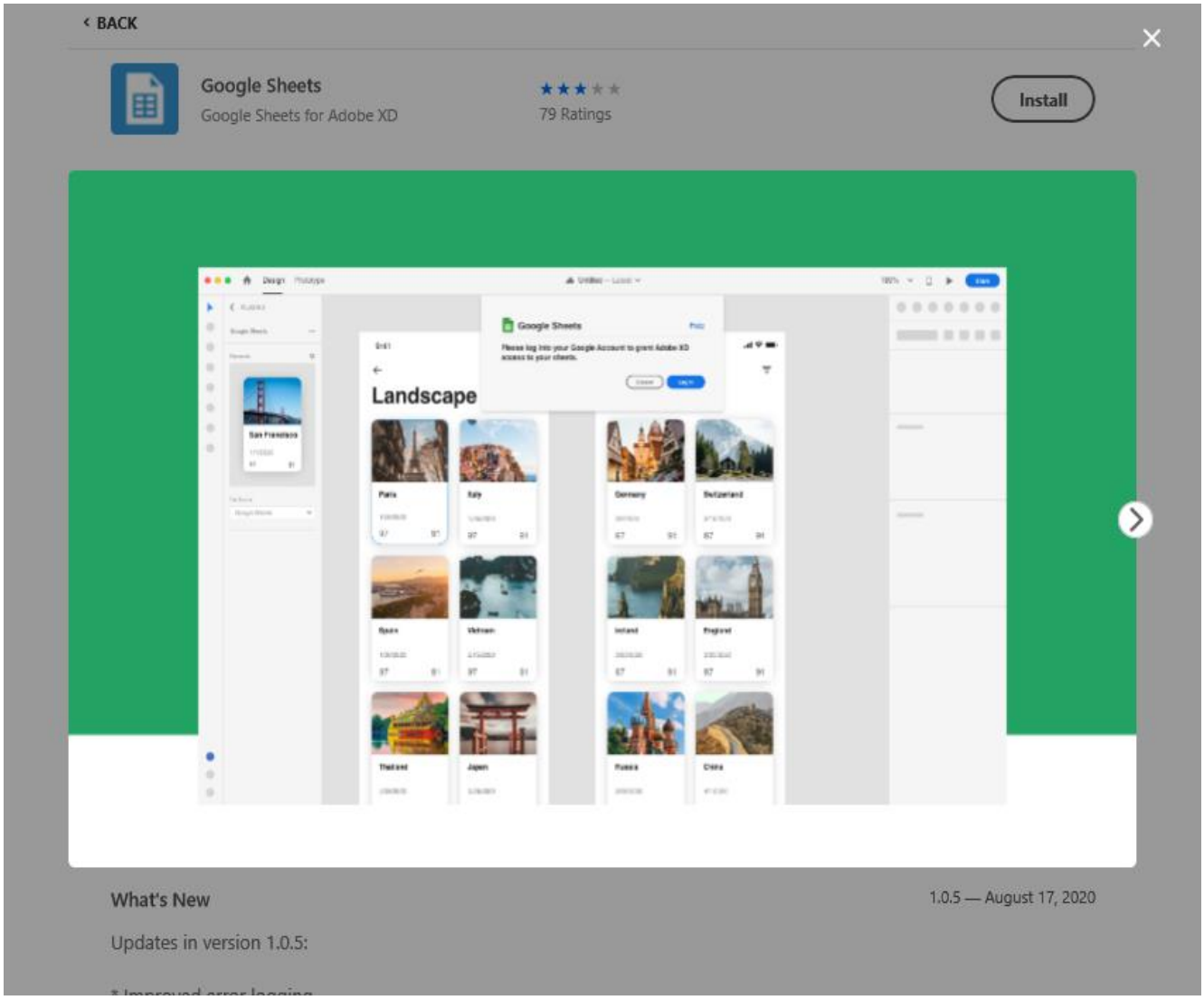
2.1 Adobe XD

Adobe XD stands for Adobe Experience Design. Although a little late to the race for UI/UX design tool, Adobe XD has grown in popularity and become one of the most used free UI/UX design tools after its release in 2016. Even though Adobe XD is free, the device only needs to have the Adobe CC subscription, it still has full of necessary features for UI/UX design, that is 3D transform components and repeat grid. Especially, Prototype is one of the most important features in Adobe XD. While Sketch needs to have plug in from another software to make a prototype, Adobe XD already has it. After designing UI/UX, user can switch to Prototype mode to use a lot of interesting features: overlays, wires, preview, fixed elements, time-based triggers, auto - animate, and more (Pimento 2018).



Picture 1. Example of prototype mode (Pimento 2018)

Adobe XD plugin is still an important function over years. That is because it brings the features and functionality of third-party tools right into XD (Gregory 2020). It helps designers to manage, share and collab each other to reduce working time and increase work efficiency. At the time being, there are more than 250 plugins to supercharge workflow, for example: Google sheets, CloudApp and Web export. Google sheets allow users to import their document from Google Sheets document straight to Adobe XD (Gregory 2020). Also, Users can apply CloudApp to upload and share instantly their design with others without leaving Adobe XD (Gregory 2020). Yet for designers who are not proficient with writing code, Web export can help them export HTML and CSS files from XD files to share and design specs with developers. (Vitale 2020.)



Picture 2. Google sheets plugin in Adobe XD.

2.2 Adobe Photoshop CS6

Adobe Photoshop is the best photo editing and manipulation software on the market. It was developed in 1987 by Thomas and John Knoll. In 1989, the program was sold to Adobe Systems, which marketed it as Photoshop. The core purpose of Photoshop is editing images with multiple layers and import the images in various file formats. In addition, Photoshop has some abilities to edit or render text and vector graphics, 3D graphics and video, as well as create a website, one page at a time, sliced and optimized and even with animated GIFs. (Bauer 2013.)

Picture 3. Example of Adobe Photoshop

2.3 Design process

Design process is a way of figuring out what is needed to do. While designing, the designers first have to understand their product, then they can follow design processes so that product features and functions will be performed optimally. There are four steps in the design process. The first step is creating user flow. Creating Wireframes is the second step. The third step is creating mock-ups. And creating an app prototype is final step. These steps will be described in detail as below.

2.3.1 User-flow

Optimizing the path which the user takes when interacting with the product is an essential task for the user experience. Good navigation helps users understand how to use the app easily. To do that, creating user flows is an important part of the design process. User flow is a visual representation of how the application works. It begins from the user's entry point through a set of steps to get a result, like purchasing a product. The user flow describes the intended usage for the app that shows in one direction. Designers should focus on conversation paths, account for all types of users and keep content concise. In a large picture of user flow, designers can ensure that the goals of the flow are always being set on the top (Alexander 2018). By satisfying the user-flow requirements, the user-flow of the application was created as shown in Picture 4.

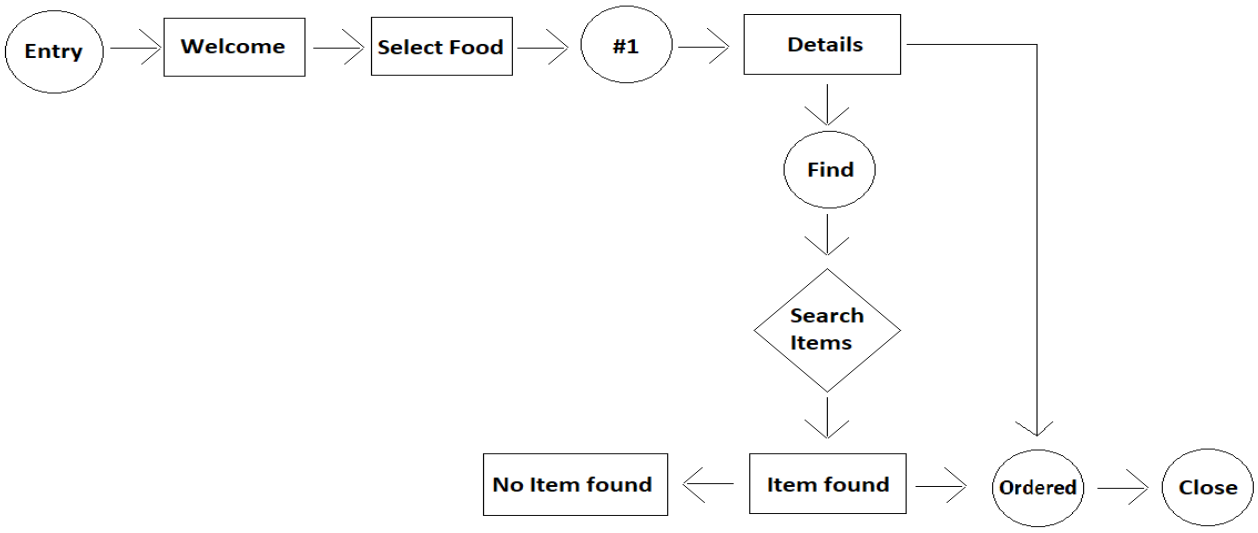


Figure 1. User-flow

2.3.2 Wireframes

Wireframe is often used to lay out content and functionality on a page and used in the process of establishing the basic structure of the website before the visual design. In short, it can be understood that Wireframes are black and white layouts outlined in specific sizes and positions corresponding to each page element, page feature, transition areas or each navigation for each page of the website or apps. Wireframe provides an overview at the earliest stage, that is used to review negotiations with customers. Wireframes ensure the full content and functionality of the page based on precise positioning and

the needs of users and businesses. This is also the biggest advantage that wireframe structure brings to users. (Beegel 2014.)

Wireframes push usability first and look at it objectively: conversion paths, link naming, navigation and, locations, etc. At the same time, wireframes can point out holes in the architecture of a website. This will help to control possible risks and promptly have more effective back-up plans and solutions. Wireframes ensure that the multi-dimensional aspects of the website's creativity and branding are combined in one step. This allows customers to provide feedback earlier in the process. Moreover, Wireframe helps users do more calculations, making everything clear and specific. (Beegel 2014.)

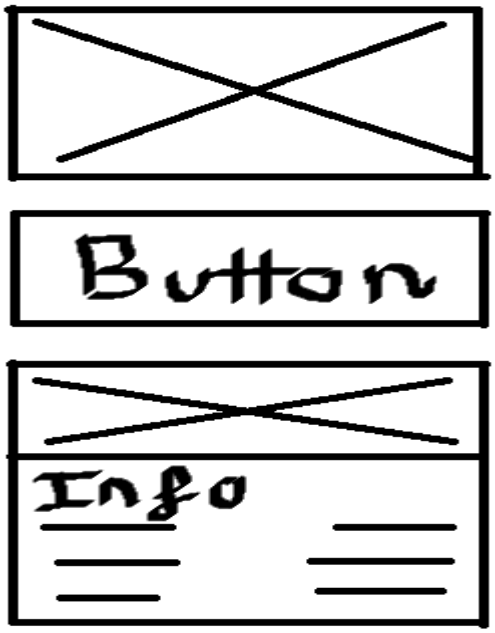
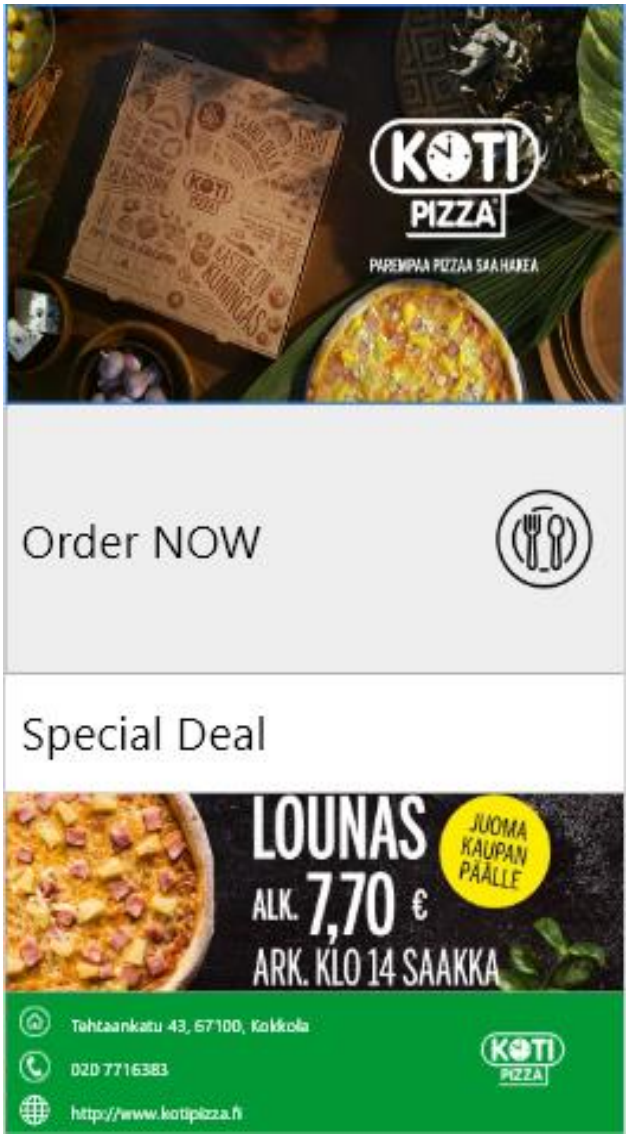


Figure 2. Wireframes

2.3.3 Mock-ups

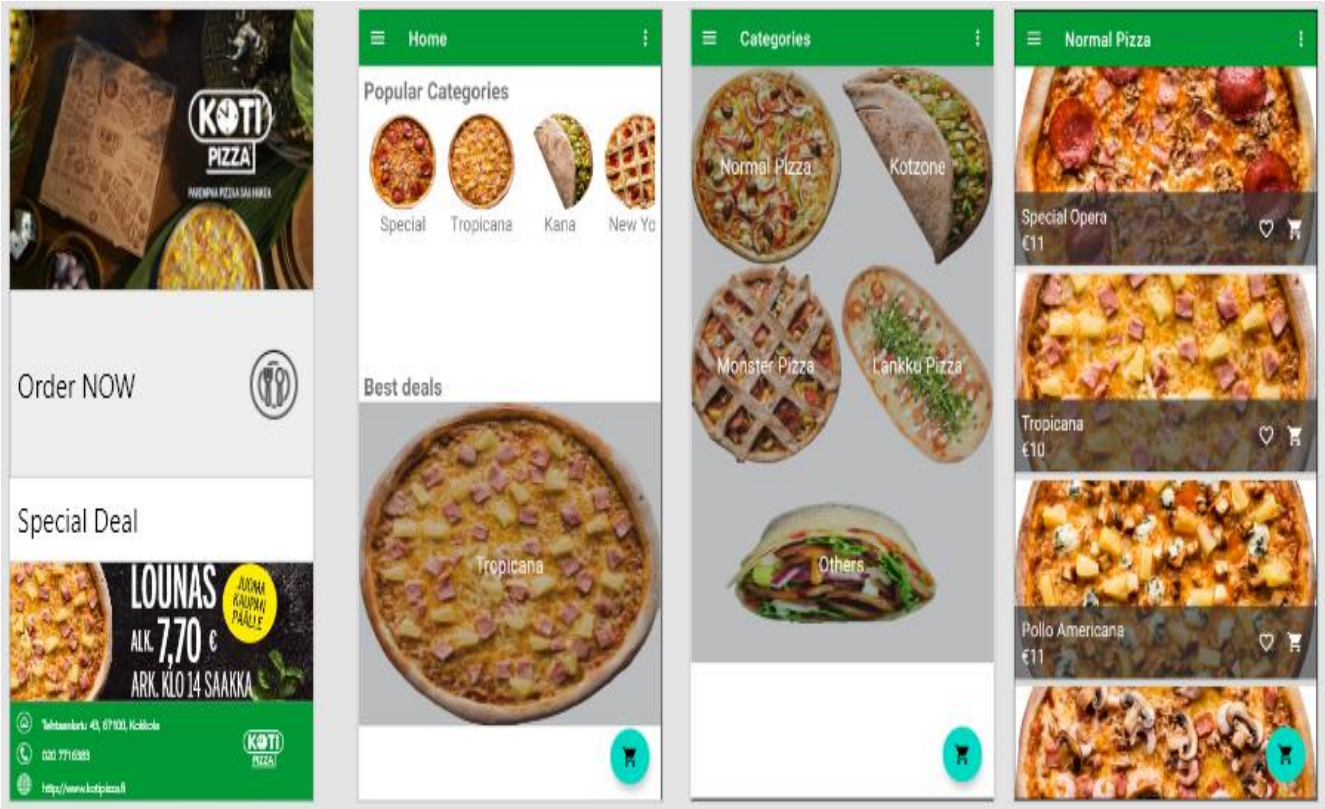
In simple terms, mock-up is an example model of an object or device created based on a specific design in full scale or size. After creating the user flow and sketches, editing the image, and designing a mock-up is a very important step. It gives the closest look to the final product. Therefore, in design activities, mock-up is used to "realize" design patterns, helping designers collect feedback from users and customers. Thereby, it helps to save time and money in testing the design, to detect errors and come up with reasonable solution. (Cao 2015.)



Picture 4. Mock-Up of the main page

2.3.4 Prototypes

In UI/UX design, the prototype of an interface is used to perform tests with users. It includes clickable images and an effective way for designers to approve their work and test their design before changing the design into code, creating a product that is officially used. The prototype of the interface will represent solutions that can solve the specific problems of the user. And to know if that solution is suitable or not, the simplest way is to observe the user's actions and reactions when interacting and experiencing with that prototype. (Lazarova 2018.)



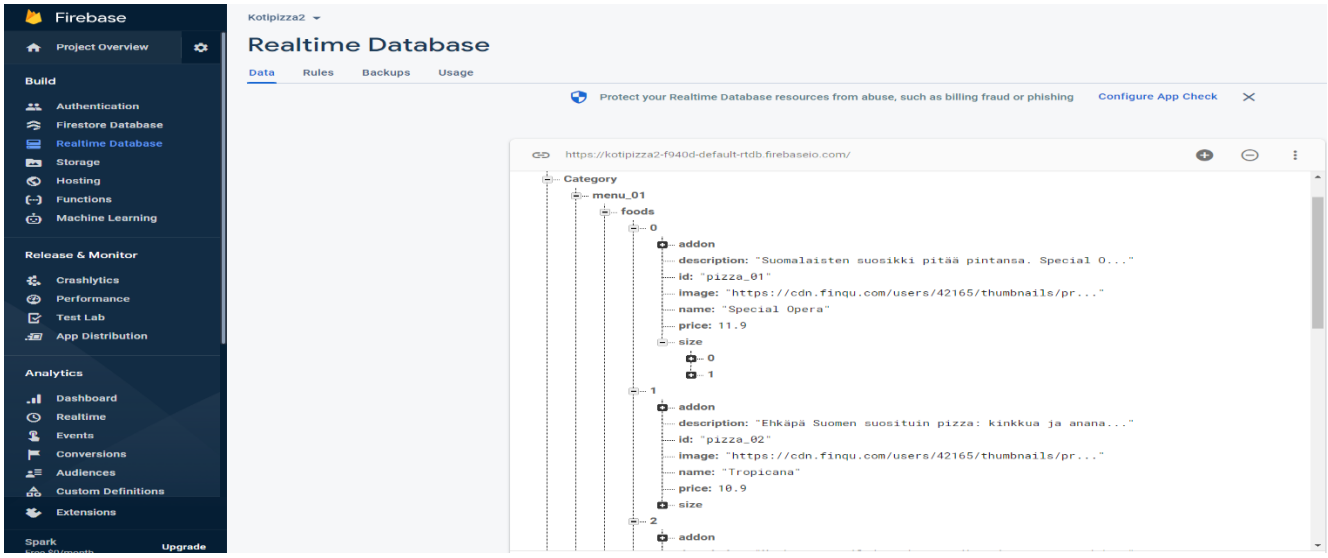
Picture 5. The Complete User Interface

3 CREATING DATABASE

If data is books, database can be seen as a library that contains a variety of books on different topics. Database is an organized collection of related information, or data that is stored electronically in a computer system. Data can be organized in rows and columns in the form of a table, which helps users can easily access, retrieve, manage, and update. (Peterson 2022.)

3.1 Firebase

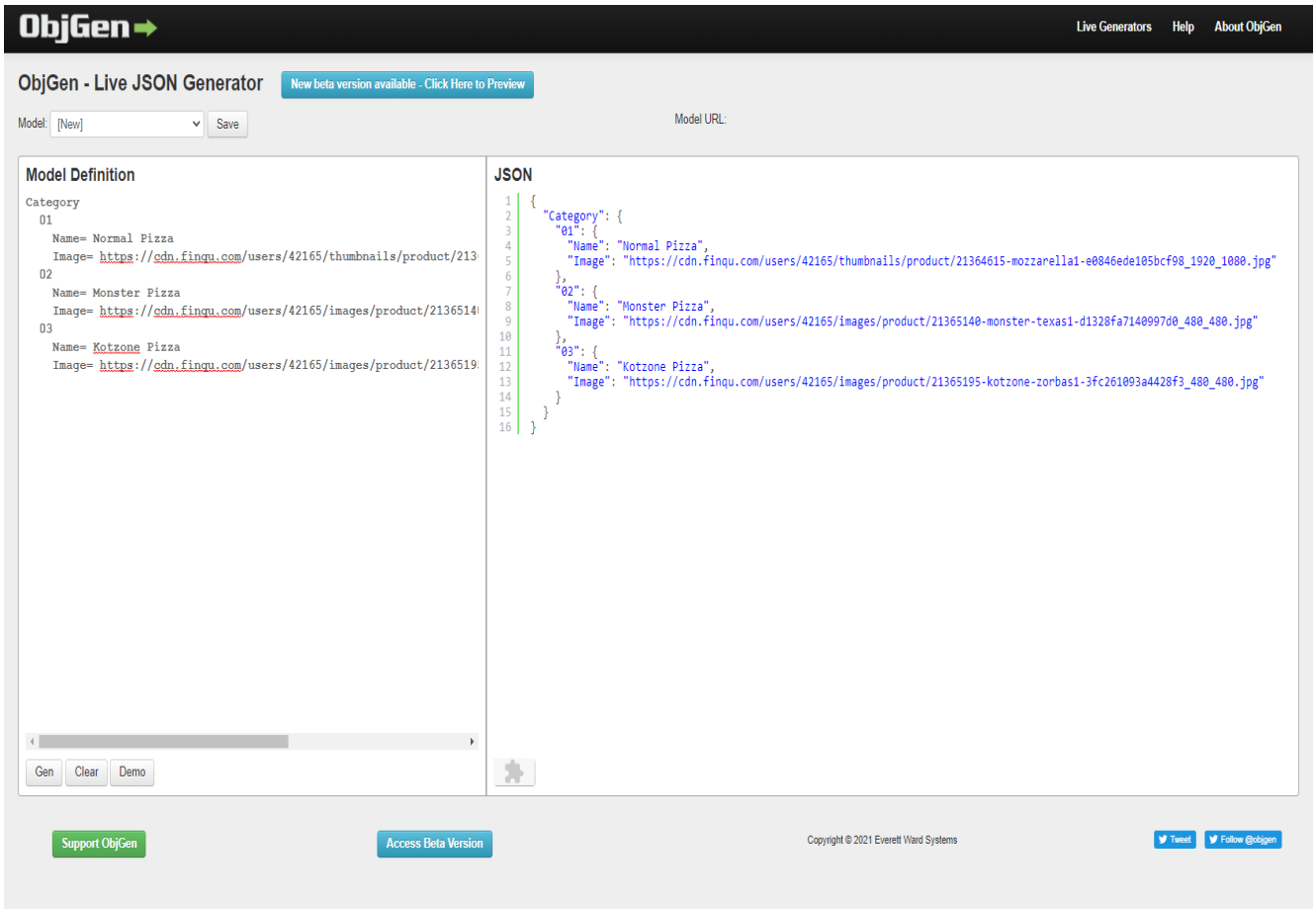
In April 2012, Firebase was developed for the first time by James Tamplin and Andrew Lee. It is known as Envolve, which is a platform that specializes in providing APIs to integrate chat features into websites (Kumar, 2019). After being acquired by Google in 2014, Firebase produced a lot of features for high-quality mobile and web applications development such as: Firestore Database, Cloud Storage, Cloud functions, Authentication, Hosting, and Machine Learning. However, Realtime Database still takes a leading role in backend development. It is a cloud-based NoSQL database as JSON format, which provides an API that helps to synchronize application data across different platforms (include iOS, Android, Web, and more) in real time. Even when the devices are offline, Firebase apps remain responsive by storage all data to disk and will synchronize with current server stat after reconnection. (Tanna and Singh 2018).



Picture 6. Realtime Database in Firebase

3.2 JavaScript Object Notation (JSON)

JavaScript Object Notation (also known as JSON) was discovered by Douglas Crockford in 2007, and it is a very popular lightweight data interchange format. JSON was derived from JavaScript, but JSON is language-independent data format. All the popular programming languages (such as: C#, PHP, Java, C++, and Ruby) support JSON data format. JSON objects are written in key/value pairs, in which the key should be strings and value must be a valid JSON data type (string, number, object, array, boolean or null). In comparison with XML, which is another data interchange format, JSON is compact, technically safer, and portable. Moreover, JSON file is easier to write and read, even if programmers have never seen it before. JSON can be easily generated by using Objgen website (Sriparasa and Josepd 2018). The picture 8 below shows how to create categories in the project by using Objgen website.



Picture 7. Generate JSON by using Objgen website

3.3 Database Design

Entity-relationship diagram, or ER-diagram for short, is a way to represent data in a graphical form. A basic ER model is composed of entity types (which classify the things of interest, includes their attributes) and specifies relationships that can exist between entities (instances of those entity types). In this project, the database includes six tables, which are represented for six entities: Categories, Food, Add on, Cart Item, Customer, and Order. Each table has different relationship to the others. There are three different relationships in ER diagram. The first one is one-to-one, which means two entities can have only one relationship with each other. The second one is one-to-many, which means an element of the first entity may be linked to many elements of the second entity, but a member of the second entity is linked to only one element of the first entity. The last one is many-to-many, which occurs when multiple records in a table are associated with multiple records in another table. (Gordon 2017.)

As can be seen from the figure 1 below, each table or each entity has its own attributes, such as food entity has name, image, description, size, quantity, price. They need to have relationship with the other table to create a diagram. After having one-to-one relationship with categories, the food table has one-to-many relationship with add on table, which means, one pizza can have more than one topping. The cart item has one-to-many relationship with food and add on, which means users can have many different pizzas in their cart. The relationship between customer table and cart item table as well as order table are one-to-one, one customer can have only one cart item and finish by only one order. If they want to have another order, they can start the app again. CustomerID and OrderID are primarykey in customer table and cart item table, which will be used in order table to confirm the order.

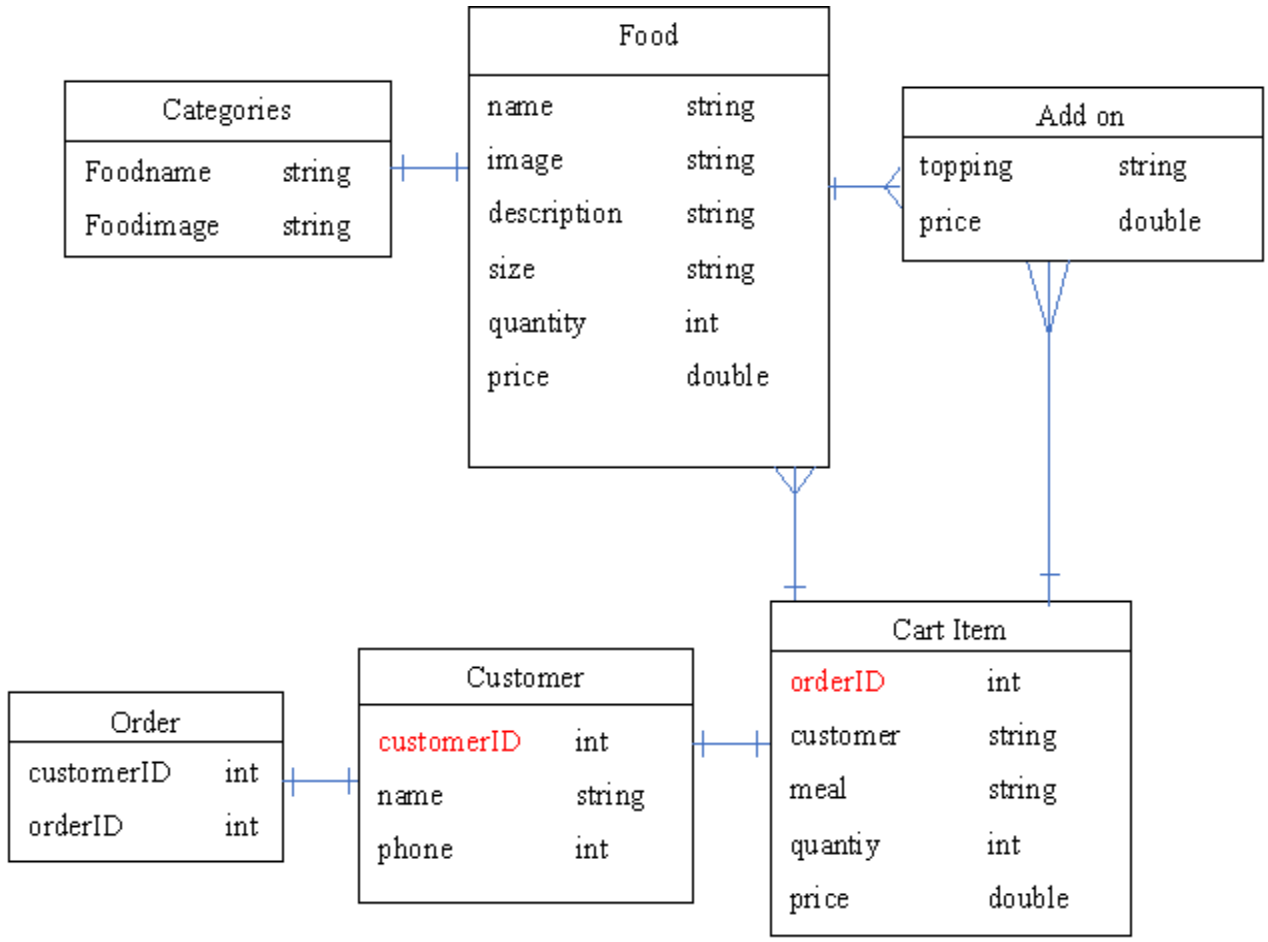


Figure 1. ER diagram

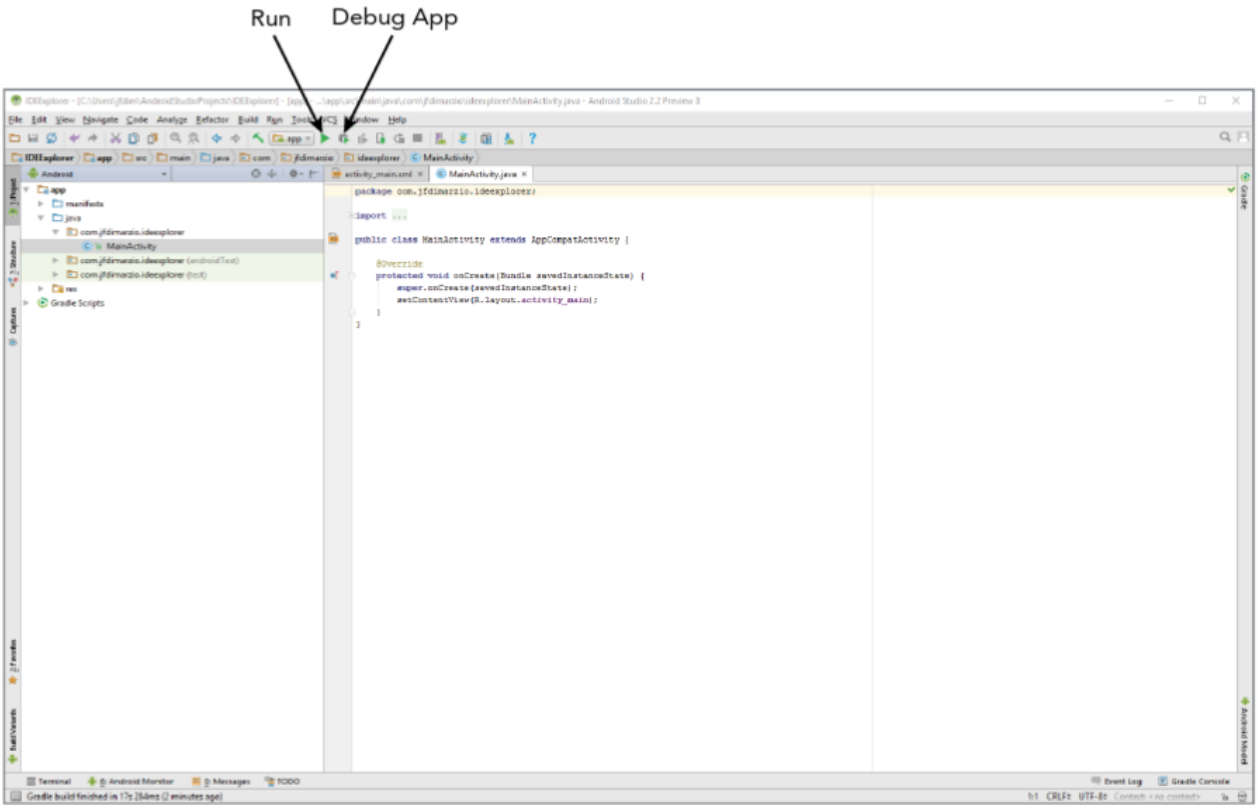
4 ANDROID DEVELOPMENT

Nowadays, Android is a formidable mobile operating system, is one of the most popular mobile operating systems in the world. Android software development is the process to create applications for smartphones, tablet, and television running Android operating system. In Android software development kit (SDK), Kotlin, Java, and C++ languages can be used to write Android apps. After having mock-ups and database from the last two steps, developers can start coding, changing the design into code. (DiMarzio 2016.)

4.1 Tools

4.1.1 Android studio

There are lots of softwares that are used to create and design a mobile application, such as Zoho Creator, Swiftic, and Vuforia, etc. However, in this project, Android Studio was the software that is used because it is free, using less memory capacities than the others, easily access to everyone and there are various command lines on this software website, which help programmers to build mobile applications. Besides, Android Studio helps write code faster with a smart flexible autocomplete as well as find, preview, and replace text as developers type in a file or across all projects. With Android Studio, programmers can easily browse and open the whole project, switch between codes and user interface to check for any errors. For example, in Picture 9 below, the project is displayed on the left side, whole interface codes are in the middle and on the right side there is the user interface of these codes. (Mullis 2017.)



Picture 8. Example for Android Studio (DiMarzio 2016)

4.1.2 Android Virtual Devices

Besides code editing features, Android Studio provides an emulator software, which is Android Virtual Devices, to allow developers to run the application by simulating real device capabilities. (Nilanchala 2013). With Android Virtual Devices (AVD) on Android Studio, developers have many options about Android version, devices, resolution, or density. Developers can create as many AVDs as they need, based on the types of devices they want to test for. They can not only run all devices at the same time, switch between them, but also stop whenever they want. By following instructions, developers can create an AVD from AVD manager graphical interface. Firstly, by going to Window and choosing AVD Manager, Virtual Devices window will appear on screen. And then, the New button can be selected to create a Virtual Device. The name of the device can be given before selecting Target Android Platform from the drop-down list. Finally, an AVD can be created by click “Create AVD” as the last step. (Nilanchala 2013.)

Picture 9. Example for AVD (Alexander 2018)

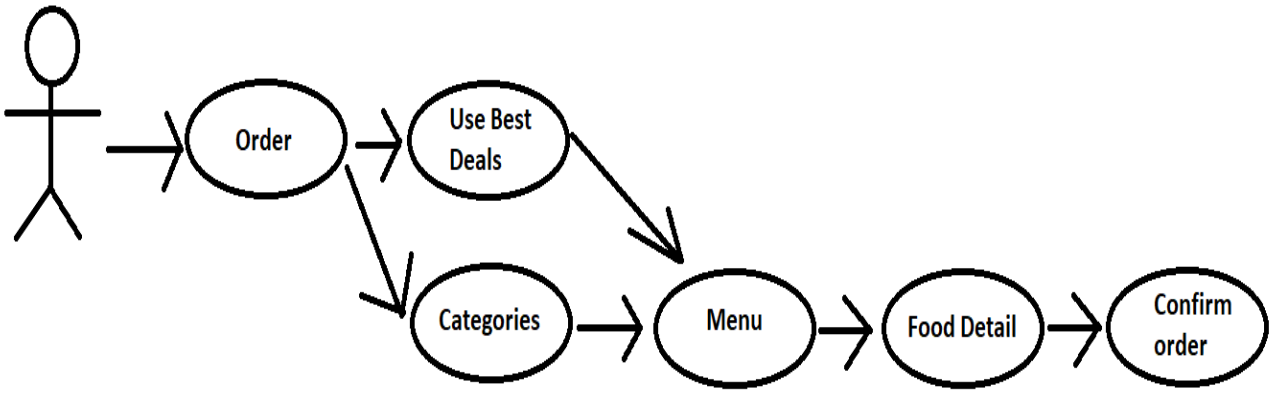
4.2 Development Process

4.2.1 Quality Function Deployment

There are several requirements for this application when it is fully developed. The expected outcome should show that the UI of the application will be friendly. Users will be able to use the apps easily and do not need instructions. The apps need to have full menu choices for customers. There is no problem opening or using the application. Furthermore, there will be optional requirements that may be available in the future, for instance: searching for food, adding favourite food, and rating food, etc.

4.2.2 Use Case Diagram

Use case modeling in the Unified Modeling Language (UML) is a popular text-based tool for systems analysis and design (Gemino, Parker 2009). The functions of the application can be seen from picture 11 below. It shows how the application works and which subject to perform at each stage. Users have access to all features of the application. Users have two different options (Best deals, and categories). Users can use best deals if they want to know suggestion foods from restaurant. Otherwise, they can choose categories to have more options.

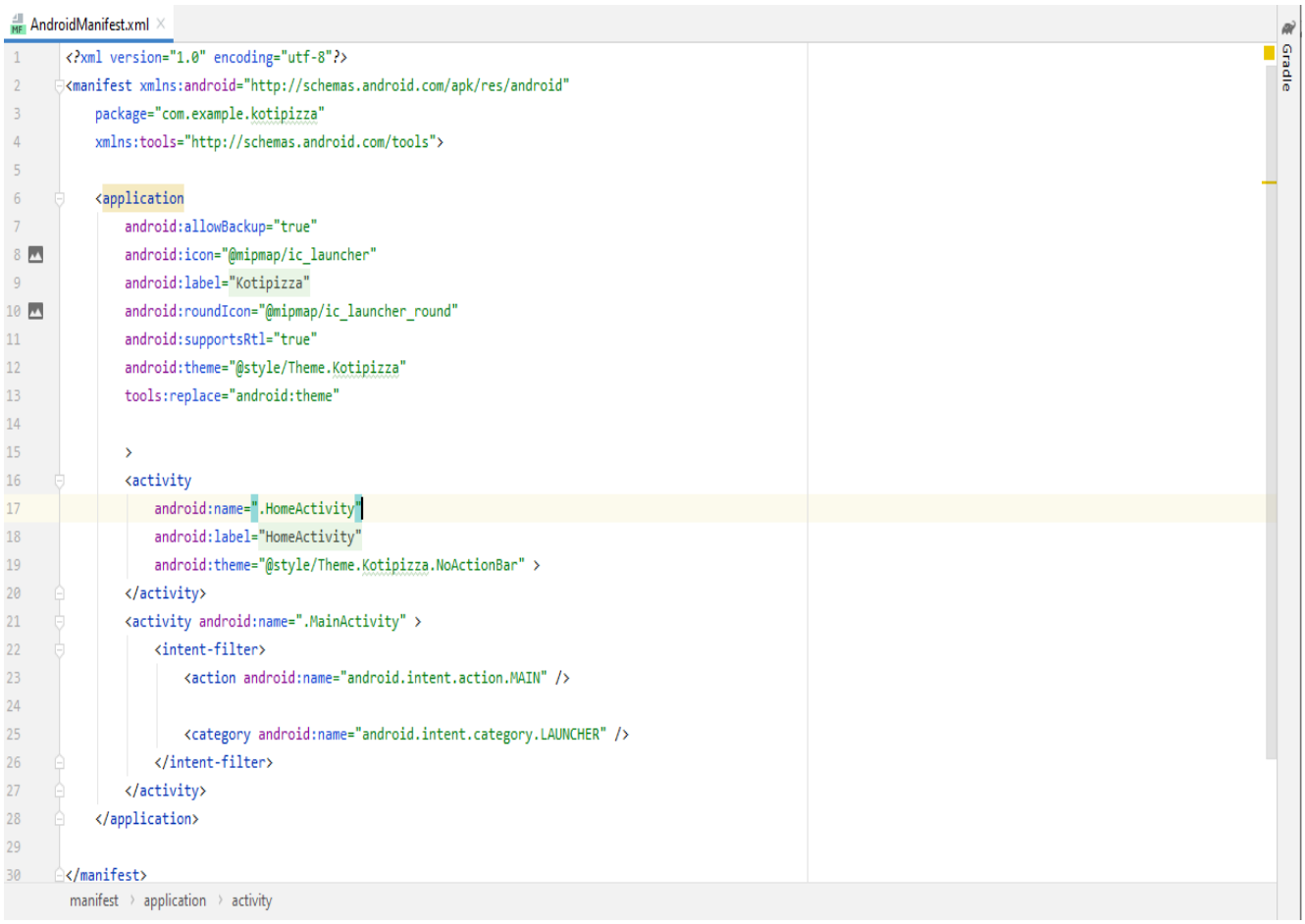


Picture 10. Use case Diagram

4.2.3 Manifest Files

Android Manifest is an important file which is formatted XML. Without manifest file, the system cannot run any of its code. It is used to configure properties for apps: characteristics, theme, assets, activities, and permissions (Wallace 2017). This is a place to help summarize all information about the application in how the application will use the internet, read the external memory card or use GPS. In the same way, it not only shows the number of screens that the application will have but also the number of services that will be available in varied broadcast receivers. Based on these definitions, the system will allow the application to use special functions of the system, and at the same time alert the user of important information in application before they decide to install and use it.

Each app component must be declared a corresponding XML element in the manifest file, otherwise the system cannot start. Developers can use <activity> tag for each subclass of Activity, <service> tag for each subclass of Service, <receiver> tag for each subclass of BroadcastReceiver, and <provider> tag for each subclass of ContentProvider. The name of subclass must be specified with the name attribute, using the full package designation. As picture 12 below, <activity> tag was used to declare name, label, and theme in HomeActivity file, and MainActivity file.



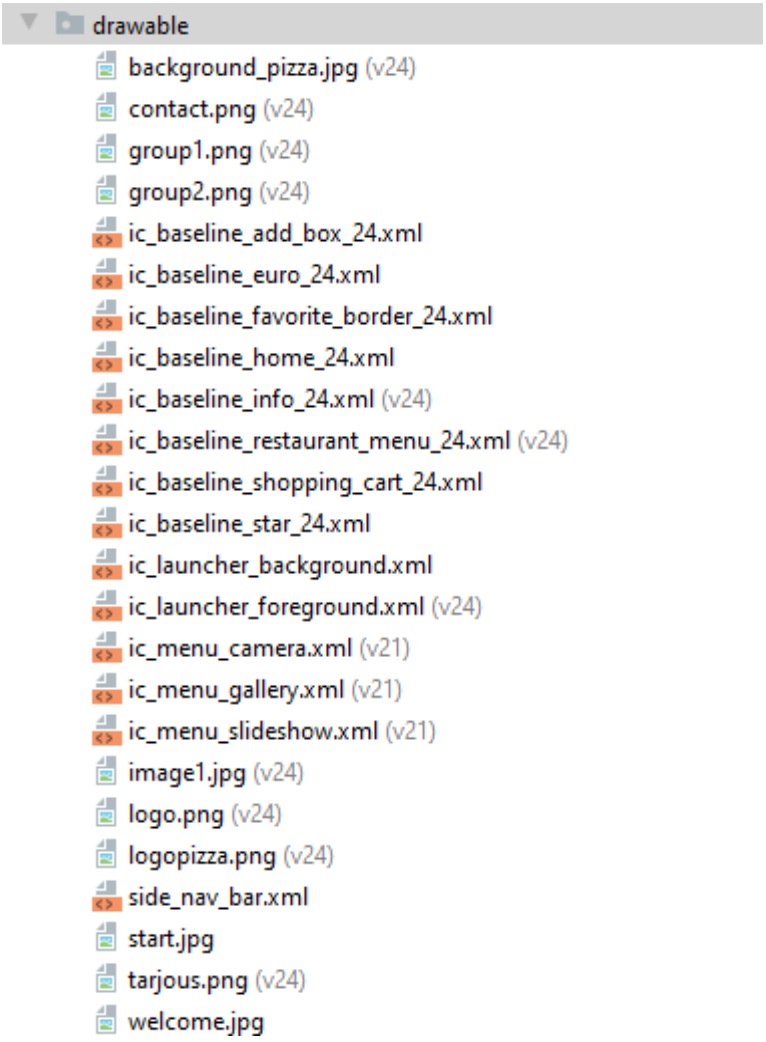
Picture 11. Manifest file

4.2.4 Drawable Files

In Android Studio, the Drawable class and its subclasses are used to display shapes, colors, borders, gradients which can be applied to views inside the Activity. There are at least 17 types of drawables, BimapDrawable is the most popular one. Other types of drawables such as Shape Drawables, StateList Drawables, LayerList Drawables, NinePatch Drawables, Vector Drawables are also common. As the picture 17 below, all the drawable files are in the /res/drawable folder and can be accessed as follows:

-In XML file: @drawable/drawable_name.

-In Java: R.drawable.drawable_name. (Wallace 2017)



Picture 12. Drawable files

4.2.5 Layout

The graphical user interface (GUI) is an important part of the application, is a standard of user-centered design in software programming. Through GUI, users can interact with devices via icons or visual indicators such as buttons, image buttons, check boxes, switches, menus, cursors, touchscreen, or even voice-command interaction capabilities, instead of command line. Each GUI screen is designed by xml file. It contains tags that define how the data should be structured and stored and transported over the internet. (Teske 2019.)

For example, creating food list: the user interface of this layout, which name is layout_food_item, was created. It consists of relative layout component, linear layout, component, text view, and image view.

Different properties like gravity, orientation, weightsum, and etc are used to adjust the interface. The values for the elements have been defined in the strings folder of the resource files as strings.xml. The following is a code snippet of the XML file that is used in the layout_food_item.xml.

```
<RelativeLayout
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical">

    <ImageView
        android:id="@+id/img_food_image"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:scaleType="centerCrop"
        />
    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_alignParentBottom="true"
        android:background="@color/border_color"
        android:orientation="horizontal"
        android:padding="10dp"
        android:weightSum="10">

        <LinearLayout
            android:layout_width="0dp"
            android:layout_height="wrap_content"
            android:layout_weight="8"
            android:gravity="center_vertical"
            android:orientation="vertical"
            >
        <TextView
            android:id="@+id/txt_food_name"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:text="Name of food"
            android:textColor="@android:color/white"
            android:textSize="20sp"/>

            <TextView
                android:id="@+id/txt_food_price"
                android:layout_width="match_parent"
                android:layout_height="wrap_content"
                android:text="€0"
                android:textColor="@android:color/white"
```

Picture 13. Layout_food_item.xml source code

4.2.6 Activitiy Files

An Activity object in Android contains a single UI focused on a specific task or feature that your application offers to the user (Wallace 2017). Any app, no matter how small it is (in terms of code and scalability), it has at least one Activity class. An activity is a combination of 2 parts: an xml file which is seen as the design defined its interface and a Java file that contains the source code to run the application. After creating the layouts of the user interface, activities files that contain source code (Java or Kotlin) must been created to make the apps work. As a picture 14 below, in the MainActivity.java file, the btnPickup image button was coded to go to the next page.

MainActivity.java

```
package com.example.kotipizza;

import ...

public class MainActivity extends AppCompatActivity implements View.OnClickListener {

    ImageButton btnStart;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        ImageButton b = null;
        b = (ImageButton) findViewById(R.id.btnPickup);
        b.setOnClickListener(this);
    }

    @Override
    public void onClick(View v) {
        switch (v.getId()) {
            case R.id.btnPickup:
                Intent intent = new Intent( packageContext: this, HomeActivity.class);
                startActivity(intent);
                break;


        }
    }
}
```

Picture 14. Activity files

Any Android activity goes through a certain life cycle during its life inside the Android app. This life cycle is illustrated below. It starts with Activity launched. Then it goes through three different steps to reach Activity running mode. From here, three more steps can be driven through until the Activity Shutdown at the last. Besides that, App process killed stage is also showed in this circle in Picture 15.

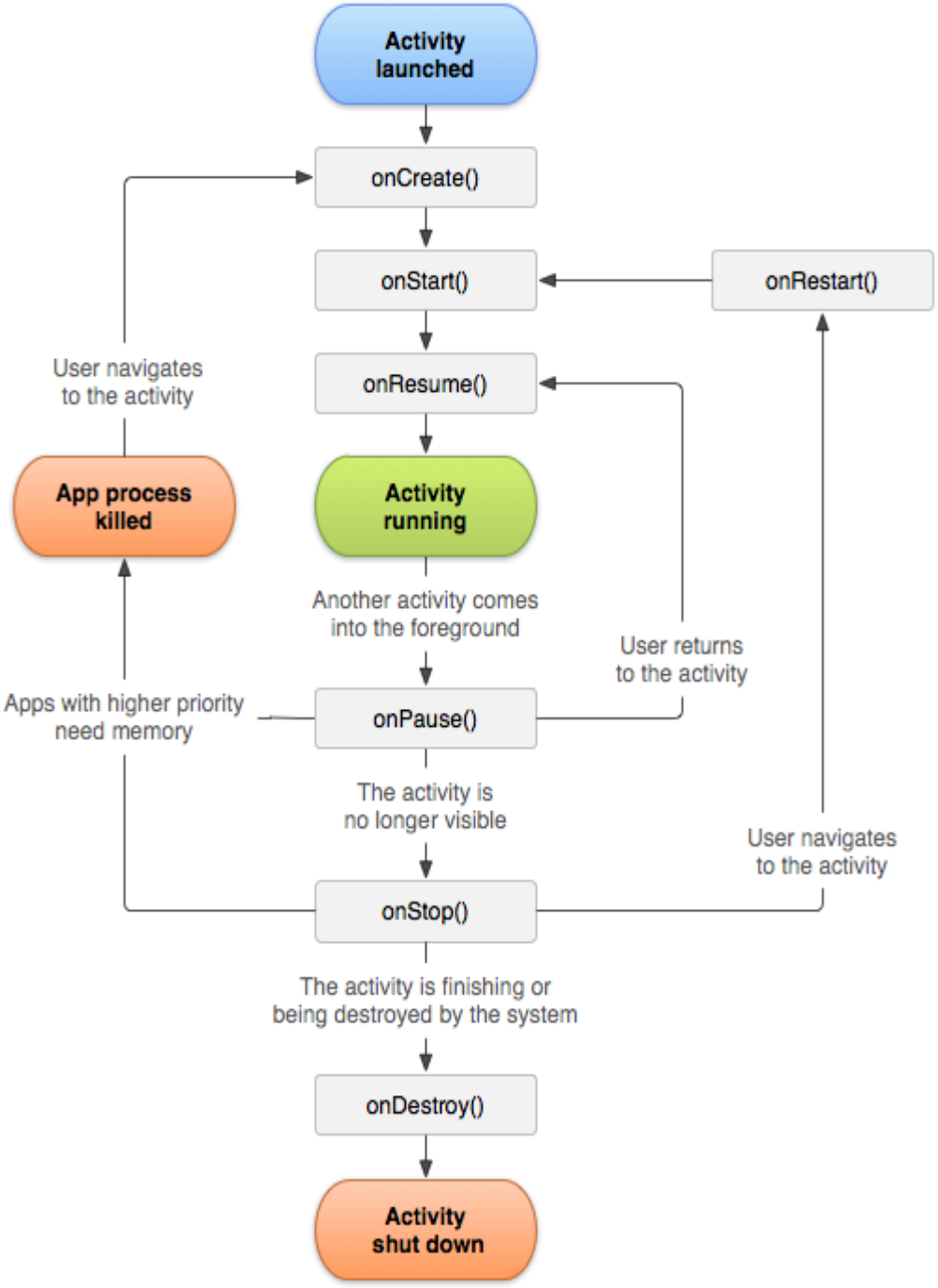


Figure 3. How activity file run

4.2.7 Adapter

An Adapter object acts as a bridge between an AdapterView and the underlying data for that view. The Adapter provides access to the data items. The Adapter is also responsible for making a View for each item in the data set. An Adapter is responsible for getting data from a dataset and creating View objects based on that data. The generated View objects are then used to attach to any Adapter Views that are bound to the Adapter. (Wallace 2017.)

Adapter View can display large datasets very efficiently. For example, ListView and GridView can display millions of elements without any significant lag while still using very low CPU and memory. They only render View objects that are already on the screen or that are moving onto the screen. This way, the memory consumed by an Adapter View can be fixed and becomes independently in the size of the dataset. They also allow developers to reduce inflater layout operations and reuse existing View objects that have moved off the screen (Wallace 2017).

After getting the layout of food list, which was mentioned in chapter 4.2.5, all the data, such as picture, name, price, image, only display when the layout matched with the adapter. In that case, RecyclerView.Adapter was used to manage data and update data to display in view. Each RecyclerView needs a data source. As picture 15 below, FoodModel class was made to represent the data model, which will be displayed by the RecyclerView.

```
public class FoodModel {
    private String name, image, id, description;
    private Long price;
    private List<AddonModel> addon;
    private List<SizeModel> size;

    //For cart
    private List<AddonModel> userSelectedAddon;
    private SizeModel userSelectedSize;

    public FoodModel() {
    }

    public String getName() { return name; }

    public void setName(String name) { this.name = name; }

    public String getImage() { return image; }

    public void setImage(String image) { this.image = image; }
```

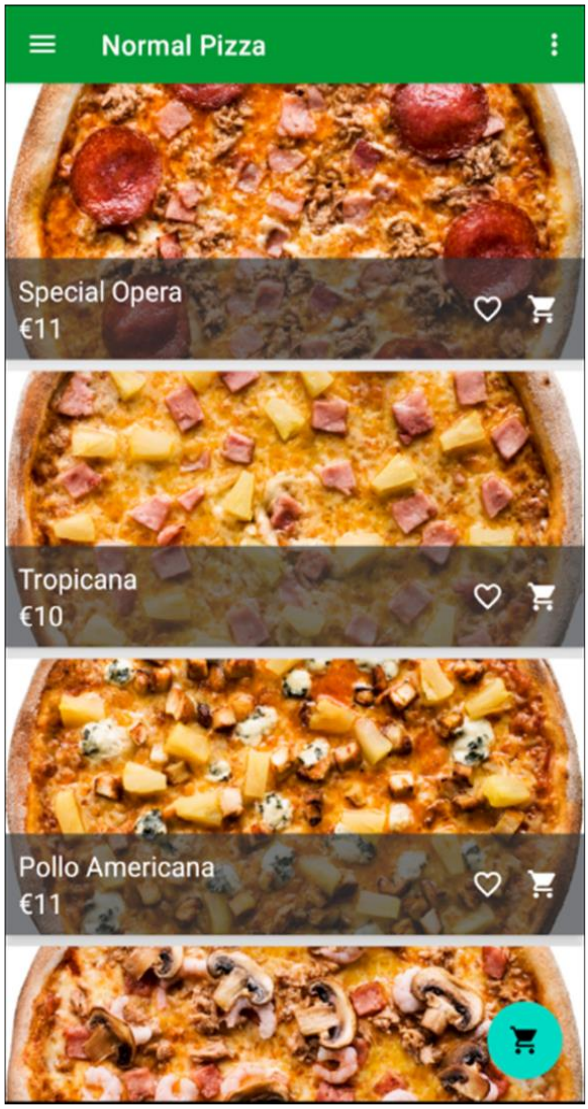
Picture 15. FoodModel class

There are 2 important parts in RecyclerView.Adapter: onCreateViewHolder and onBindViewHolder. The picture 16 below shows that onCreateViewHolder was used to create Viewholer, which is a function that determines which layout will be used at the current line displayed on the screen. Function on BindViewHolder is used to transfer data to ViewHolder. Price, name, price of food, and cart item picture were the data that has been attached

```
public class MyFoodListAdapter extends RecyclerView.Adapter<MyFoodListAdapter.MyViewHolder> {

    private Context context;
    private List<FoodModel> foodModelList;
    private CompositeDisposable compositeDisposable;
    private CartDataSource cartDataSource;

    public MyFoodListAdapter(Context context, List<FoodModel> foodModelList) {
        this.context = context;
        this.foodModelList = foodModelList;
        this.compositeDisposable = new CompositeDisposable();
        this.cartDataSource = new LocalCartDataSource(CartDatabase.getInstance(context).cartDAO());

    }

    @NonNull
    @Override
    public MyViewHolder onCreateViewHolder(@NonNull ViewGroup parent, int viewType) {
        return new MyViewHolder(LayoutInflater.from(context)
        .inflate(R.layout.layout_food_item,parent, attachToRoot: false));
    }

    @Override
    public void onBindViewHolder(@NonNull MyViewHolder holder, int position) {
        Glide.with(context).load(foodModelList.get(position).getImage()).into(holder.img_food_image);
        holder.txt_food_price.setText(new StringBuilder("€")
        .append(foodModelList.get(position).getPrice()));
        holder.txt_food_name.setText(new StringBuilder("")
        .append(foodModelList.get(position).getName()));

        //Event
        holder.setListener((view, pos) -> {
            Common.selectedFood = foodModelList.get(pos);
            EventBus.getDefault().postSticky(new FoodItemClick( success: true,foodModelList.get(pos)));
        });

        holder.img_cart.setOnClickListener(view -> {
            CartItem cartItem = new CartItem();
            cartItem.setFoodId(foodModelList.get(position).getId());
            cartItem.setFoodName(foodModelList.get(position).getName());
            cartItem.setFoodImage(foodModelList.get(position).getImage());
```

Picture 16. Adapter

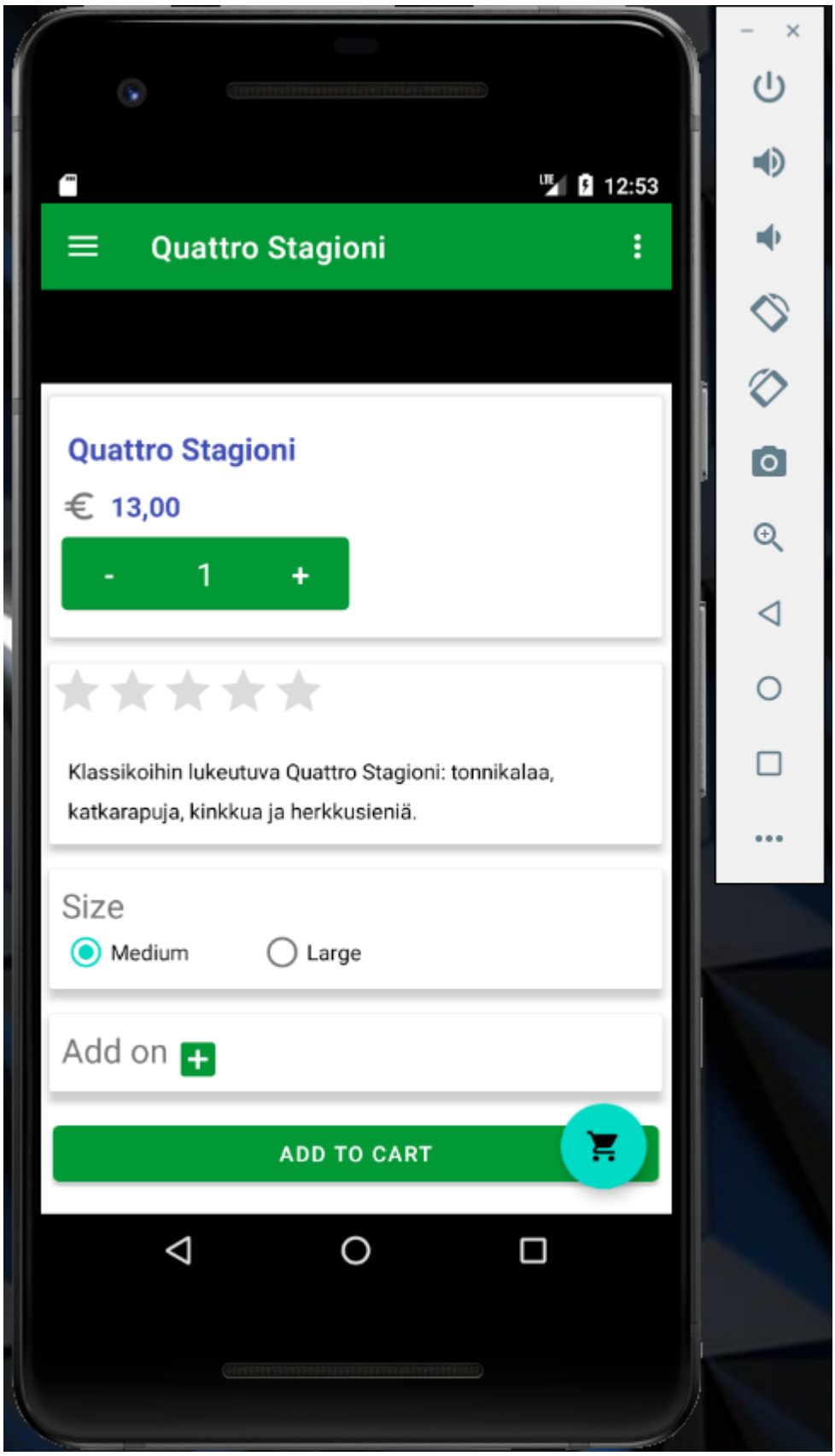
After combining layout_food_item xml file and MyFoodListAdapter jave file, the food list page has been completed as picture 17. It can be seen that Normal pizza list has showed with several types of pizza. Together with the pizza names and pictures, the price of each pizza also mentioned clearly right under its name. Users can click on heart icon to save the pizza in their favourite list to make the order easier for the next time, and shopping cart icon helps they add the item directly to the final order list.



Picture 17. Normal pizza page

5 TESTING

The test was carried out several times on Android phones based on the requirements. The app was installed on several Android devices and also virtual emulators. The main page, categories or cart item page worked perfectly as expected. If users want to make orders, besides the suggestions from best deals, they can choose categories to have more choices. In categories, there are four different kinds of pizza and kebab roll. After that, they can choose pizza or make their own one by choosing custom pizza and add topping on that. As in Picture 18, the selected pizza will be showed with the price and description in raw materials and size meanwhile customer can choose the size option like Medium or Large. Add to cart button can be used to add the food to the cart. By using cart symbol in the right corner, the user can have a whole look of their selection where they can reconsider to add more item or remove any of the extra or wrong choice. Thereafter, the user can fill in their personal information for example name and phone number. The end page of the final step will show the complete order information which consists of customer information and their food selection as well as their food receiving option. Payment function still needs to improve later to help customer complete the order at the restaurant. The test was successful and satisfactory. To make the testing broader and more reliable, a small project had been done at a specific Kotipizza 383 restaurant which will be described as below.



Picture 18. Results of Quatro stagioni pizza page

5.1 Discussion

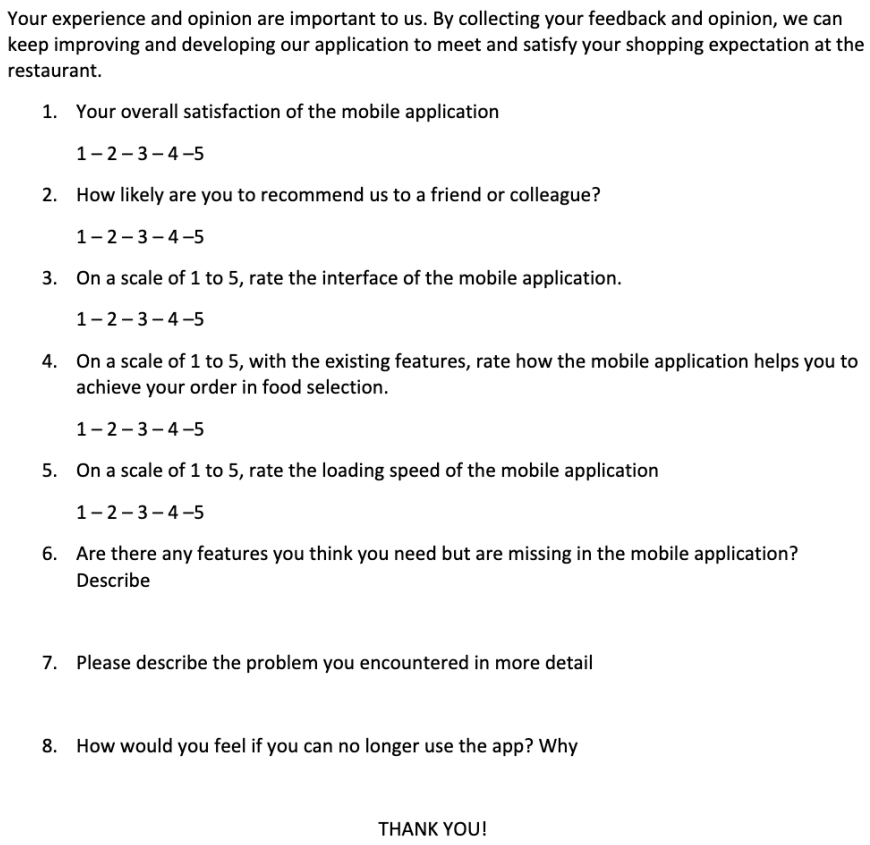
In this part of the thesis, the process of the project of getting customer experience and feedback on how the product, that is pizza application in Android operating system mobile for a specific Kotipizza restaurant, was used by making a survey, will be discussed along with the result of the research as well as the author learning process. The project began by doing visits to the Kotipizza 383 restaurant and having discuss with restaurant manager in September 2022. The idea was to produce online ordering customer feedback system of "Kotipizza 383" and get a chance to try the idea in practice of the mobile application development for pizza restaurant. In the beginning of the project, it was decided that each Kotipizza 383 restaurant customer feedback would be evaluated by the research questions that are found in Picture 19. Due to the lack of workforce and the schedule of the author, the project was done in one week. In addition, the project was running in English language so that all the participation will be English speaking customers who had visited and willing to join the project. Thanks for the help of restaurant owners and employees, many customers had visited were author, restaurant owners and employees' friends and relatives.

The project had happened in Kotipizza 383 after lunch time and from fourteen o'clock until eighteen o'clock every day during one-week time. At the cashier table of the restaurant, there will be a small note that encourages the customer to participate in the project. The note stated that: "Online mobile order helps cashier going smoothly." In the meantime, the cashier employees also explain and introduce shortly to customers about the project and its purposes. A form of feedback was designed as a leaflet putting next to the note at the cashier. The customers can write their own feedback during the time waiting for the food and leave the filled form feedback at the table for waiter to collect or return to the cashier by themselves. The reason of the project had happening after lunch time was to avoid rushing lunch break time and to let restaurant employees have enough time to introduce the project and collect the customer feedback.

5.1.1 The survey

The survey was designed by questionnaires. It included eight questions consisting of five rating questions and three free word answering questions. Five rating questions were about the overall feelings of customer about the application features, design, and speed. Three other free answering questions were

aimed to get customer personal experience to put it differently in the mobile application development plan based on that the application can be improved in the future.



Your experience and opinion are important to us. By collecting your feedback and opinion, we can keep improving and developing our application to meet and satisfy your shopping expectation at the restaurant.

1. Your overall satisfaction of the mobile application

   1 – 2 – 3 – 4 –5

2. How likely are you to recommend us to a friend or colleague?

   1 – 2 – 3 – 4 –5

3. On a scale of 1 to 5, rate the interface of the mobile application.

   1 – 2 – 3 – 4 –5

4. On a scale of 1 to 5, with the existing features, rate how the mobile application helps you to achieve your order in food selection.

   1 – 2 – 3 – 4 –5

5. On a scale of 1 to 5, rate the loading speed of the mobile application

   1 – 2 – 3 – 4 –5

6. Are there any features you think you need but are missing in the mobile application? Describe

7. Please describe the problem you encountered in more detail

8. How would you feel if you can no longer use the app? Why

THANK YOU!

Picture 19: Survey questionnaires

5.1.2 The result

The total number customers had participated into the project was 28 customers. Many of them have had experience of shopping online before through mobile application and happily to response the survey. The first question got overall answer of 4,1 points which indicates that the mobile application was easy to use and has helped customers to achieve their goals. The second questions came out surprisingly with 5 points rate which means that any customers having mobile with Android operating system wish to recommend this application to their friend and colleague whoever want to making order without visiting the restaurant in person.

The interface question got the rate of 4,1 meanwhile, the question of customer food selection satisfactory got the rate of 4. This can be understood that with clear interface and well-organized food infor-

mation, it made easily and faster for customer to recognized and select their favourite pizza. The sixth question which is about missing features of this mobile application was not being answered by all the participants. The answers were focus on making the purchase online and other services like home delivery or student discount. The seventh question came out with very less completed response, most of them answered that there is no problem in food selection. There is still a minor number of participants answering that it took them awhile to switch between the Best Deals to the main Menu to look for the food that they want. Finally, the last question was commented interesting by some of customers. They highly recommend continuing and developing the application to avoid the rush and pressure for the restaurant cashier taking order in general and for customers for having time to think and select their food in private.

5.2 Learning outcomes

Although the project was challenging because of the narrow of timeframe and human resource, it was rewarding with the result to get restaurant customers who has never ordered food online instead of visiting the restaurant in person, experience ordering through mobile application. The research methods used, go hand in hand when making the application testing for Kotipizza 383 restaurant. Real testing with the users plays a great role in software development due to the efficiency and accuracy of the executed output. It provides the application developers the assurance of producing a reliable and quality software that is easy to use and understand by the end users. Despite the difficulties in language barrier, the author and others were able to transform the message to the end users and get the feedback from them.

6 CONCLUSION

As has been said, along with the development of smartphones, ordering and shopping online is becoming popular in the Food and Beverage Industry. Instead of having to move to stores or restaurants, waiting to be served, customers can stay at home and order almost everything on their phone. The "Kotipizza 383" app is designed to make it easier for users to order from a restaurant.

This bachelor's thesis has drawn up the basic idea of a mobile application development background and introduce many other useful applications to support the work of software development in Android operating system. A good foundation was achieved in a limited time but more development is required for the application to be used properly in the real situation. The current test can be developed to work with new features and fixes in the software, because when there is a various range of customers, the test cases will function with minimal changes. Although there are small bugs that need to be upgraded and fixed in the future, the application still meets the requirements set forth. In brief, working on this project was a good experience while programming in Android. Various usages of Android Software Development Kit on the Eclipse Development Environment had been researched and dug deeper during the project. Dealing with several different app techniques was instructive, challenging and meaningful.

REFRENCES

Alexander, 2018. The biggest WTF in design right now. Available: https://uxdesign.cc/the-biggestwtf-in-design-right-now-87139f367d66. Accessed: 21st July 2022.

Babich, N. 2019. The 4 Golden Rules of UI Design. Available: https://xd.adobe.com/ideas/process/ui-design/4-golden-rules-ui-design/ Accessed: 27th March 2022

Bauer, P. 2013. Photoshop CC for Dummies, 2013. Accessed: 20th October 2022.

Beegel, Justin, et al. Infographics for Dummies, 2014. Accessed: 20th October 2022.

Jerry, C. 2015. The What, Why, and How of Mockups. Available: https://designmodo.com/mockups/. Accessed: 28th July 2022.

DiMarzio, Jerome, and J. F. DiMarzio. 2016. Beginning Android Programming with Android. Accessed: 28th July 2022

Gordon, K. 2017. Modelling Business Information: Entity relationship and class modelling for Business Analysts. Available at: https://ebookcentral-proquest-com.ezproxy.centria.fi/lib/cop-ebooks/detail.action?docID=4875459/ Accessed: 28th November 2022

Gregory, E. 2020. Create Jaw-Dropping Designs with these 7 Best Adobe XD Plugins. Available at: https://www.getcloudapp.com/blog/adobe-xd-plugins/ Accessed: 4th May 2022

Kumar, A. 2018. Mastering Firebase for Android Development: Build Real-Time, Scalable, and Cloud-enabled Android Apps with Firebase. Accessed: 28th April 2022

Moreno, H. 2014. The Gap between UI and UX Design – Know the Difference. Available at: http://snip.ly/mjj7s#http://www.onextrapixel.com/201 4/04/24/the-gap-between-ui-and-ux-design-know-the- difference/ Accessed: 28th April 2022

Mullis, A. 2017. Android Studio tutorial for beginners. Available: https://www.androidauthority.com/android-studio-tutorial-beginners-637572/ Accessed: 24th April 2022

Nilanchala, P. 2013. What is Android Virtual Device. Available: https://stacktips.com/tutorials/android/what-is-avd/ Accessed: 27th April 2022

Peterson, R. 2022. What is Database? Definition, Meaning, Types with Examlphie. Available: https://www.guru99.com/introduction-to-database-sql.html/ Accessed: 5th July 2022

Pimento, J. 2018. The Evolution of Prototyping Features (and How to Use Them) in Adobe XD. Available at: https://medium.com/thinking-design/prototyping-with-adobe-xd-681d61f73ddd/ Accessed: 1st May 2022

Sriparasa, S and Joseph, B. 2018. JavaScript and JSON Essentials: Build Light Weight, Scalable, and Faster Web Applications with the Power of JSON, 2nd Edition. Accessed: 24th April 2022

Shetye, S. 2014. Introduction to Java for Android Application Development. Available: https://blog.udemy.com/java-for-android/ Accessed: 24th March 2022

Tanna, M and Singh, H. 2018. Serverless Web Applications with React and Firebase: Develop real-time applications for web and mobile platforms. Accessed: 28th April 2022

Teske, 2019. What Is GUI (Graphical User Interface). Availalble: https://www.lifewire.com/what-isgui-graphical-user-interface-4682595/ Accessed: 19th May 2022

Vitale, A. 2020. 13 essential Adobe XD plugins. Available at: https://www.creativebloq.com/features/best-adobe-xd-plugins/ Accessed: 5th May 2022

APPENDIX 1/1 MyBestDealsAdapater

MyBestDealsAdapter.java

```
20
21  public class MyBestDealsAdapter extends LoopingPagerAdapter<BestDealModel> {
22
23      @BindView(R.id.img_best_deal)
24      ImageView img_best_deal;
25      @BindView(R.id.txt_best_deal)
26      TextView txt_best_deal;
27
28      Unbinder unbinder;
29
30      public MyBestDealsAdapter(Context context, List<BestDealModel> itemList, boolean isInfinite) {
31          super(context, itemList, isInfinite);
32      }
33
34      @Override
35      protected View inflateView(int viewType, ViewGroup container, int listPosition) {
36          return LayoutInflater.from(context).inflate(R.layout.layout_best_deal_item,container, attachToRoot: false);
37      }
38
39      @Override
40      protected void bindView(View convertView, int listPosition, int viewType) {
41
42          unbinder = ButterKnife.bind( target: this, convertView);
43          //Setdata
44          Glide.with(convertView).load(itemList.get(listPosition).getImage()).into(img_best_deal);
45          txt_best_deal.setText(itemList.get(listPosition).getName());
46
47      }
48  }
```

APPENDIX 1/2 MyCategoriesAdapter

```
1   package com.example.kotipizza.Adapter;
2
3   import ...
27
28  public class MycategoriesAdapter extends RecyclerView.Adapter<MycategoriesAdapter.MyViewHolder> {
29
30      Context context;
31      List<CategoryModel> categoryModelList;
32
33
34      public MycategoriesAdapter(Context context, List<CategoryModel> categoryModelList) {
35          this.context = context;
36          this.categoryModelList = categoryModelList;
37      }
38
39      @NonNull
40      @Override
41      public MyViewHolder onCreateViewHolder(@NonNull ViewGroup parent, int viewType) {
42          return new MyViewHolder(LayoutInflater.from(context)
43                  .inflate(R.layout.layout_category_item,parent, attachToRoot: false));
44      }
45
46      @Override
47      public void onBindViewHolder(@NonNull MyViewHolder holder, int position) {
48          Glide.with(context).load(categoryModelList.get(position).getImage())
49                  .into(holder.category_image);
50          holder.category_name.setText(new StringBuilder(categoryModelList.get(position).getName()));
```

```
52          //Event
53          holder.setListener((view, pos) -> {
54              Common.categorySelected = categoryModelList.get(pos);
55              EventBus.getDefault().postSticky(new CategoryClick( success: true, categoryModelList.get(pos)));
56
57          });
58      }
59
60      @Override
61      public int getItemCount() {
62          return categoryModelList.size();
63      }
64
65      public class MyViewHolder extends RecyclerView.ViewHolder implements View.OnClickListener {
66          Unbinder unbinder;
67          @BindView(R.id.img_category)
68          ImageView category_image;
69          @BindView(R.id.txt_category)
70          TextView category_name;
71
72          IRecyclerClickListener listener;
73
74          public void setListener(IRecyclerClickListener listener) { this.listener = listener; }
77
78          public MyViewHolder(@NonNull View itemView) {
79              super(itemView);
80              unbinder = ButterKnife.bind( target: this, itemView);
```

```
89
90
91      @Override
92      public int getItemViewType(int position) {
93          if(categoryModelList.size() == 1)
94              return Common.DEFAULT_COLUMN_COUNT;
95          else
96          {
97              if (categoryModelList.size() % 2 == 0)
98                  return Common.DEFAULT_COLUMN_COUNT;
99              else
100                 return (position > 1 && position == categoryModelList.size()-1) ? Common.FULL_WIDTH_COLUMN:Common.DEFAULT_COLUMN_COUNT;
101         }
102
103     }
104 }
105
```

APPENDIX 1/3 MyFoodListAdapater

```java
package com.example.kotipizza.Adapter;

import ...


public class MyFoodListAdapter extends RecyclerView.Adapter<MyFoodListAdapter.MyViewHolder> {

    private Context context;
    private List<FoodModel> foodModelList;
    private CompositeDisposable compositeDisposable;
    private CartDataSource cartDataSource;

    public MyFoodListAdapter(Context context, List<FoodModel> foodModelList) {
        this.context = context;
        this.foodModelList = foodModelList;
        this.compositeDisposable = new CompositeDisposable();
        this.cartDataSource = new LocalCartDataSource(CartDatabase.getInstance(context).cartDAO());

    }

    @NonNull
    @Override
    public MyViewHolder onCreateViewHolder(@NonNull ViewGroup parent, int viewType) {
        return new MyViewHolder(LayoutInflater.from(context)
        .inflate(R.layout.layout_food_item,parent, attachToRoot: false));
    }
```

```java
    @Override
    public void onBindViewHolder(@NonNull MyViewHolder holder, int position) {
        Glide.with(context).load(foodModelList.get(position).getImage()).into(holder.img_food_image);
        holder.txt_food_price.setText(new StringBuilder("€")
        .append(foodModelList.get(position).getPrice()));
        holder.txt_food_name.setText(new StringBuilder("")
        .append(foodModelList.get(position).getName()));

        //Event
        holder.setListener((view, pos) -> {
            Common.selectedFood = foodModelList.get(pos);
            EventBus.getDefault().postSticky(new FoodItemClick( success: true,foodModelList.get(pos)));
        });

        holder.img_cart.setOnClickListener(view -> {
            CartItem cartItem = new CartItem();
            cartItem.setFoodId(foodModelList.get(position).getId());
            cartItem.setFoodName(foodModelList.get(position).getName());
            cartItem.setFoodImage(foodModelList.get(position).getImage());
            cartItem.setFoodPrice(Double.valueOf(String.valueOf(foodModelList.get(position).getPrice())));
            cartItem.setFoodQuantity(1);
            cartItem.setFoodExtraPrice(0.0); // Because default we not choose size + addon so extra price is 0
            cartItem.setFoodAddon("Default");
            cartItem.setFoodSize("Default");
```

```
            compositeDisposable.add(cartDataSource.insertOrReplaceAll(cartItem)
            .subscribeOn(Schedulers.io())
            .observeOn(AndroidSchedulers.mainThread())
            .subscribe(() ->{
                Toast.makeText(context, text: "Add to Cart success", Toast.LENGTH_SHORT).show();
                //Here we will send a notify to HomeActivity to update counter in cart
                EventBus.getDefault().postSticky(new CounterCartEvent( success: true));

            },throwable -> {
                Toast.makeText(context, text: "[Cart error]"+throwable.getMessage(),Toast.LENGTH_SHORT).show();
            }));

        });

    }

    @Override
    public int getItemCount() {
        return foodModelList.size();
    }

```

```
    @Override
    public int getItemCount() {
        return foodModelList.size();
    }

    public class MyViewHolder extends RecyclerView.ViewHolder implements View.OnClickListener {
        private Unbinder unbinder;
        @BindView(R.id.txt_food_name)
        TextView txt_food_name;
        @BindView(R.id.txt_food_price)
        TextView txt_food_price;
        @BindView(R.id.img_food_image)
        ImageView img_food_image;
        @BindView(R.id.img_fav)
        ImageView img_fav;
        @BindView(R.id.img_quick_cart)
        ImageView img_cart;


        IRecyclerClickListener listener;

        public IRecyclerClickListener getListener() { return listener; }

        public void setListener(IRecyclerClickListener listener) { this.listener = listener; }

        public MyViewHolder(@NonNull View itemView) {
            super(itemView);
            unbinder = ButterKnife.bind( target: this, itemView);
            itemView.setOnClickListener(this);
        }

        @Override
        public void onClick(View view) { listener.onItemClickListener(view,getAdapterPosition()); }
    }
}
```

APPENDIX 2 Home Activity

```java
package com.example.kotipizza;

import ...

public class HomeActivity extends AppCompatActivity implements NavigationView.OnNavigationItemSelectedListener {

    private AppBarConfiguration mAppBarConfiguration;
    private DrawerLayout drawer;
    private NavController navController;
    private CartDataSource cartDataSource;

    @BindView(R.id.fab)
    CounterFab fab;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_home);

        ButterKnife.bind( target: this);

        cartDataSource = new LocalCartDataSource(CartDatabase.getInstance(this).cartDAO());

        Toolbar toolbar = findViewById(R.id.toolbar);
        setSupportActionBar(toolbar);
        FloatingActionButton fab = findViewById(R.id.fab);
        fab.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                Snackbar.make(view, text: "Replace with your own action", Snackbar.LENGTH_LONG)
                        .setAction( text: "Action", listener: null).show();
            }
        });
        drawer = findViewById(R.id.drawer_layout);
        NavigationView navigationView = findViewById(R.id.nav_view);
        // Passing each menu ID as a set of Ids because each
        // menu should be considered as top level destinations.
        mAppBarConfiguration = new AppBarConfiguration.Builder(
                R.id.nav_home, R.id.nav_menu, R.id.nav_food_detail, R.id.nav_food_list)
                .setDrawerLayout(drawer)
                .build();
```

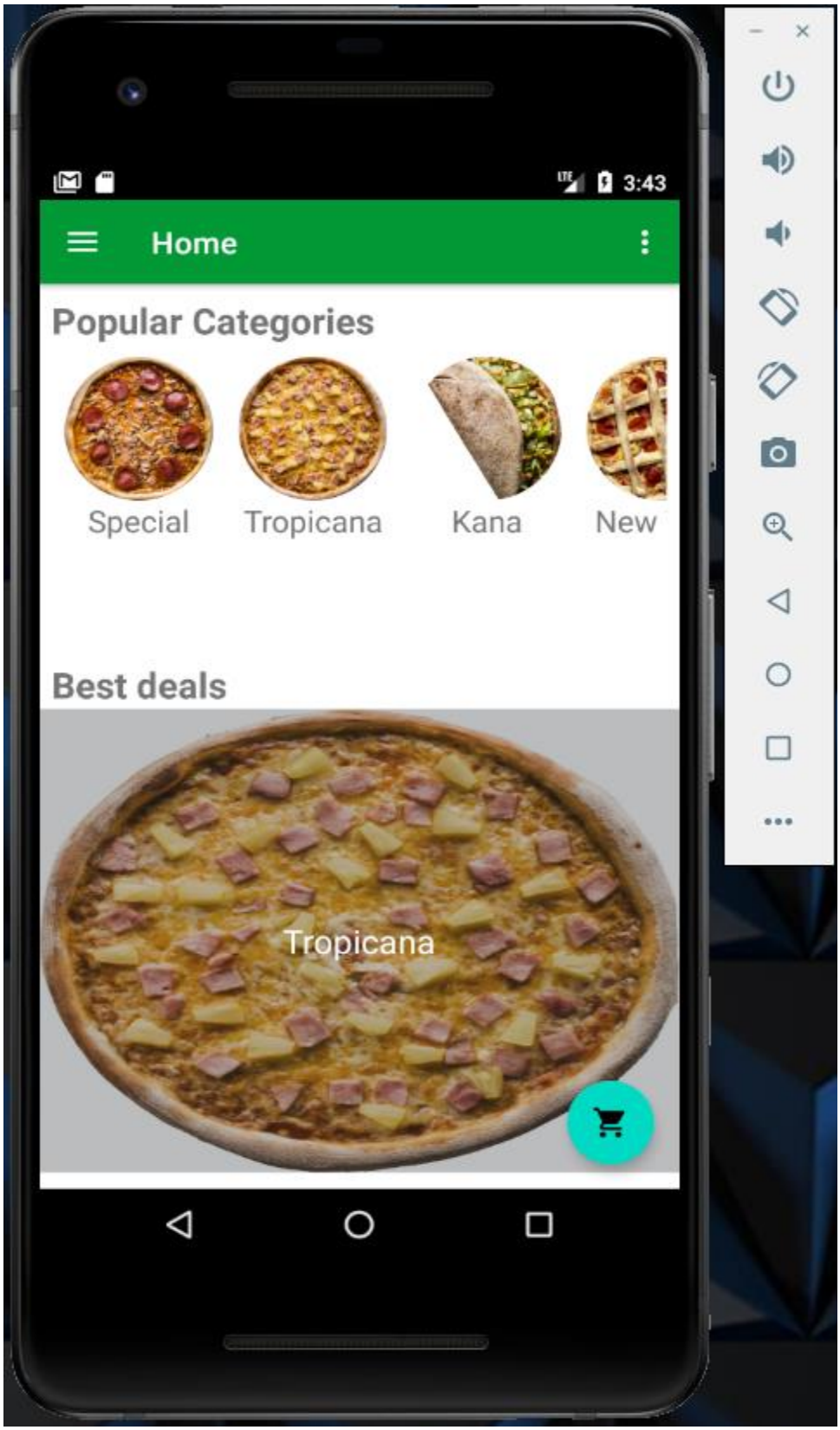
APPENDIX 3/1 CartItem

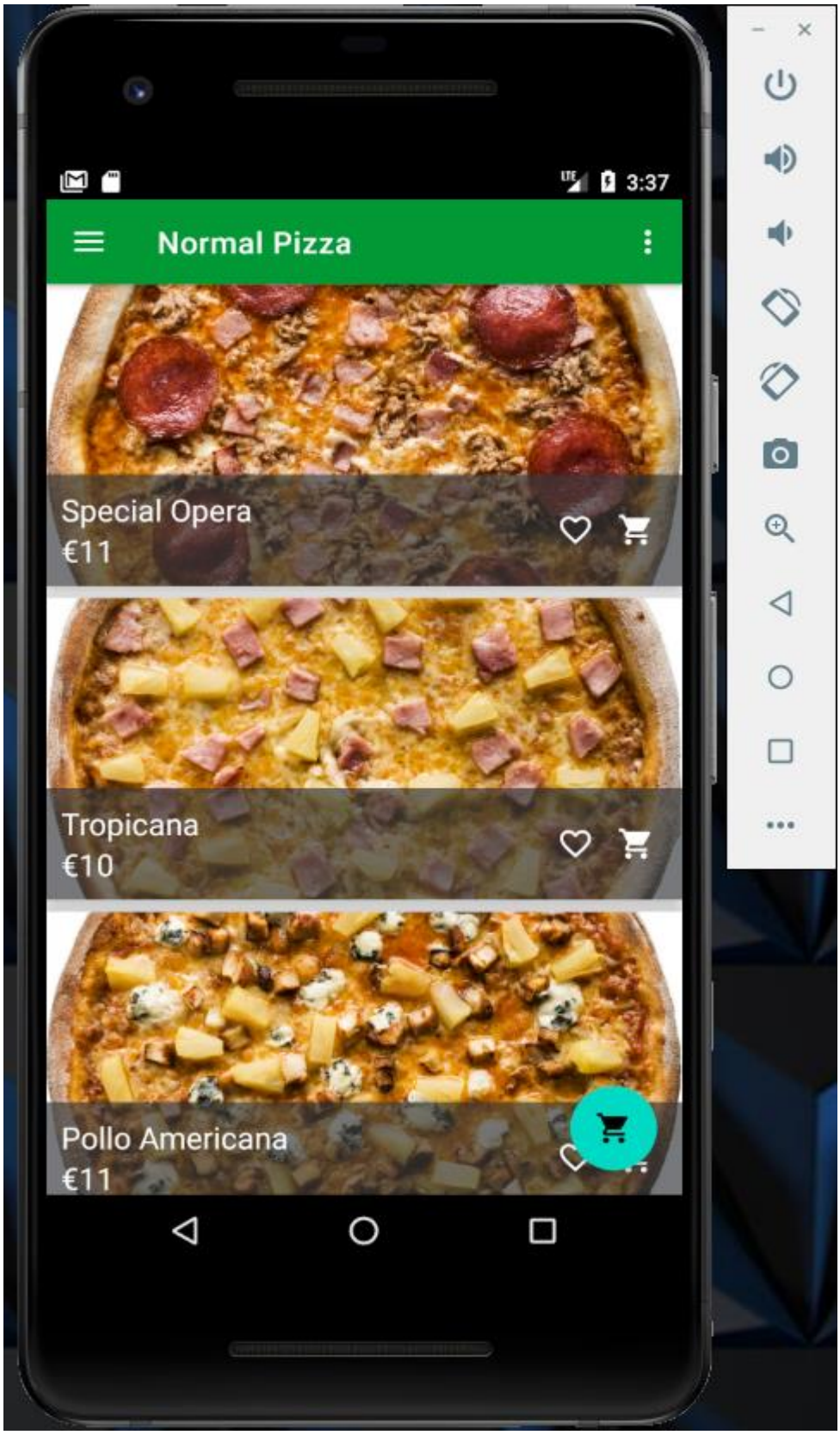
```java
@Entity(tableName = "Cart", primaryKeys = {"uid", "foodId", "foodAddon", "foodSize"} )
public class CartItem {
    @NonNull
    @ColumnInfo(name = "foodId")
    private String foodId;

    @ColumnInfo(name = "foodName")
    private String foodName;

    @ColumnInfo(name = "foodImage")
    private String foodImage;

    @ColumnInfo(name = "foodPrice")
    private Double foodPrice;

    @ColumnInfo(name = "foodQuantity")
    private int foodQuantity;

    @ColumnInfo(name = "foodExtraPrice")
    private Double foodExtraPrice;

    @NonNull
    @ColumnInfo(name = "foodAddon")
    private String foodAddon;

    @NonNull
    @ColumnInfo(name = "foodSize")
    private String foodSize;
```

APPENDIX 3/2 LocalCartDataSource

```
13      public LocalCartDataSource(CartDAO cartDAO) { this.cartDAO = cartDAO; }
16
17      @Override
18      public Flowable<List<CartItem>> getAllCart(String uid) { return cartDAO.getAllCart(uid); }
21
22      @Override
23      public Single<Integer> countItemInCart(String uid) { return cartDAO.countItemInCart(uid); }
26
27      @Override
28      public Single<Long> sumPriceInCart(String uid) { return cartDAO.sumPriceInCart(uid); }
31
32      @Override
33      public Single<CartItem> getItemInCart(String foodId, String uid) {
34          return cartDAO.getItemInCart(foodId,uid);
35      }
36
37      @Override
38      public Completable insertOrReplaceAll(CartItem... cartItems) {
39          return cartDAO.insertOrReplaceAll(cartItems);
40      }
41
42      @Override
43      public Single<Integer> updateCartItems(CartItem cartItem) {
44          return cartDAO.updateCartItems(cartItem);
45      }
46
47      @Override
48      public Single<Integer> deleteCartItem(CartItem cartItem) {
49          return cartDAO.deleteCartItem(cartItem);
```

APPENDIX 4 Results




3:37
Normal Pizza
Special Opera
€11
Tropicana
€10
Pollo Americana
€11