CHAPTER-1

INTRODUCTION

1.1 Overview

The purpose of the project is to build an application program to reduce the manual work for managing the above information. The main objective of this app is to manage the details of item category, food, order, shopping cart. To provide efficient authorization and security. To achieve integration of all records of shopping cart. To achieve the good representation of food items and restaurant details. Better synchronization of data. This project is designed using Kotlin language in android studio.

1.2 Problem Statement

The technology we recommend is an easy-to-use online meal ordering system for customers. It overcomes the disadvantages of traditional queueing systems. Our system is both a convenient way to order food from restaurants and a mess service. The procedure of taking a customer's order is made easier with this technology. Customers may place orders fast utilising the online meal ordering system, which generates an online menu. Customers can also use a meal menu to keep track of their orders. Users can also rate the food goods using this system's feedback feature. In addition, based on the user's ratings, the proposed system can recommend hotels and meals, and the hotel staff will be notified of any quality adjustments.

1.3 Motivation

My motivation for creating this app stemmed from the fact that my family works in the fast food industry, and I dislike waiting in lines or having to call ahead to place an order, especially during peak lunch or dinner hours. In addition, I value my current knowledge of the Java and JSP programming languages, as well as understanding how strong and dynamic they are when it comes to web design and application development. Because I found them to be highly beneficial when working on the technologies, I used Kotlin, android stuio to develop this application.

1.4 Technologies

This whole project designed based on food ordering system which is used by citizens. This project uses Kotlin language for field validations.

Gradle’s and wrappers are used for assigning the SDKs and other runtime platform. This project also can be implemented in different gadgets like mobile phones. The whole project has a simple looking UI design.

1.5 Application of Mobile Application Development

- **Simplicity –** It is the key to good performing application UI/UX design is a crucial factor that should be implemented in every application to draw the attention of audience. Simple UI designs with simple or login applications will catch the user's attention.
- **Best performance** – The best performance is decided by loading speed of the applications. Also, security plays a major role here.
- **Different modes of work** – Most of the mobile apps are offline and some of the apps are online. But both will have a particular traffic base to access them. Thus, users can change the app mode based on their network.
- **Customization** – when comes to organize the applications, the users may have the choice to design there UI according to their wishes. The mobile app is having more customization features like a web app.
- **Notifications** – Another main application of mobile app development is, it is having pop-up notification. Any new messages from the app or regarding the app will appear in front and user can easily remember the new messages.
- **Branding** – A mobile app is a wide platform for marketing your business easily. Most of the apps are generating ads which result in branding for a particular business.
- **Real time users** – A mobile app is popular because it is useful for real time activities. Food ordering and many applications make the audience more engaging with it. To get a data intensive application, it should be fun, engaging at the same time, it should be useful because users are giving their valuable time to access your apps. Hence the app should worth the time of using it.

CHAPTER-2

REQUIREMENT SPECIFICATION

2.1 SOFTWARE SPECIFICATION

Operating System: Windows Vista 7/8/10
Software: Android Studio
Language: Kotlin

2.2 HARDWARE SPECIFICATION

Processor: X86 Compatible processor with 1.7GHz clock
Speed Ram: 4GB or greater Hard
Disk: 400 GB min
Monitor: VGA/SVGA
Keyboard: 104 keys standard
Mouse: 2/3 button optical / mechanical
Smart Phone

2.3 USER CHARACTERISTICS

Every user

- Should be comfortable with the basic working of the computer.
- Must have basic knowledge of English.
- Must have skills of Android Studio and Kotlin.

Chapter 3

System Design

3.1 Proposed System

In this paper, we created a app, which is popular in today's world where the app is simple and has set of rules, when the user downloads and opens the applications, user can register by pressing "don't have an account?/sign up" and user can fill the details for further process.
User can see set of restaurants and food items, can add the items to cart by pressing "Add to cart". It also has the option to place the order.

3.2 Features of System

- User can add the restaurants to his favourite list
- User can give the ratings to the restaurant.
- Order history of users is available in profile.
- User can change the profile details.

3.3 Design

How Food is Ordered?

- User can login by entering username and password.
- If user does not have account then he can register by entering details.
- User can search or select the restaurant.
- User can select and add food items to the cart.
- Swipe right to see his profile, order history, faqs.
- User can place order after adding food items to cart.

3.4 Database

Room database is room is persistence library that provides an abstraction layer over the SQLite database to allow more robust database with the help of room we can easily create the database and perform CRUD operations very easily.
components of room database:

• **Entity**: Entity is a model class that is annotated with @Entity.This class is having variables that will be our columns and the class is our table

• **Database**: It is an abstract class where we will be storing all our database entries which we can call Entities.

● **DAO**: the full form of DAO is Database Access Object which is an interface class with the help of it we can perform different operations in our database.

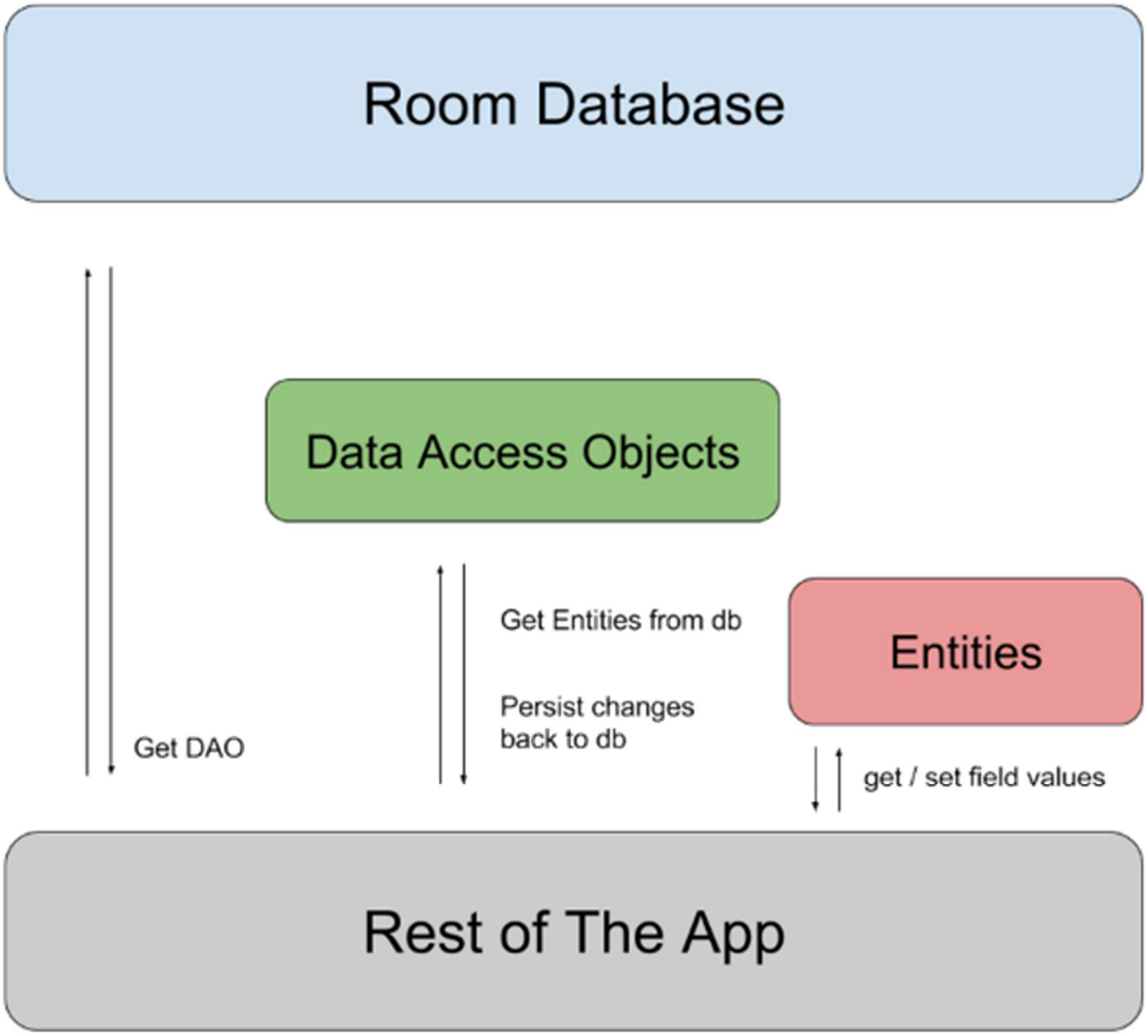

Fig 3.4.1: Room Database

Chapter 4

Implementation

4.1 Module Description

Login Module: Login page is an entry page to application that requires user identification and authentication, it is performed by entering a mobile number and password combination.

Register Module: If the user does not have an account then he can create his account by clicking register button and entering the name, email_id, mobile number, password and all these details are stored in database.

Forgot password Module: If the user forgot's his password then he can click on the forgot password button and change the password and these details are automatically updated to his profile.

Restaurant Module: User can search the restaurant by its name or user can choose from the default menu shown on the screen then he can move on to the food items module.

Food Item Management Module: The user can select the food items based on his choice and he can remove the food items if user is not required further.

Order Module: The user can add the food items to the cart. Then he can move on to the cart page and user can place and confirm the order.

4.2 Source Code:

Manifest file:

Include permissions for running activities.

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.kartikey.foodrunner">

    <uses-permission android:name="android.permission.ACCESS_NETWORK_STATE" />
    <uses-permission android:name="android.permission.INTERNET" />

    <application
```

```
android:allowBackup="true"
android:icon="@mipmap/ic_app_logo"
android:label="@string/app_name"
android:networkSecurityConfig="@xml/network_security_config"
android:resizeableActivity="false"
android:roundIcon="@mipmap/ic_app_logo_round"
android:supportsRtl="true"
android:theme="@style/AppTheme">

<activity
    android:name=".activity.OrderPlacedActivity"
    android:configChanges="orientation|screenSize"
    android:screenOrientation="portrait" />
<activity
    android:name=".activity.DashboardActivity"
    android:configChanges="orientation|screenSize"
    android:screenOrientation="portrait" />
<activity
    android:name=".activity.CartActivity"
    android:configChanges="orientation|screenSize"
    android:screenOrientation="portrait" />
<activity
    android:name=".activity.RestaurantMenuActivity"
    android:configChanges="orientation|screenSize"
    android:screenOrientation="portrait" />
<activity
    android:name=".activity.LoginRegisterActivity"
    android:configChanges="orientation|screenSize"
    android:screenOrientation="portrait" />
<activity
    android:name=".activity.SplashActivity"
    android:configChanges="orientation|screenSize"
    android:screenOrientation="portrait">
    <intent-filter>
        <action android:name="android.intent.action.MAIN" />

        <category android:name="android.intent.category.LAUNCHER" />
```

```
        </intent-filter>
    </activity>

</application>

</manifest>
```

RestaurantDao.kt:

Database access for object files of restaurant database.

```
package com.kartikey.foodrunner.database

import androidx.room.Dao
import androidx.room.Delete
import androidx.room.Insert
import androidx.room.Query

@Dao
interface RestaurantDao {
    @Insert
    fun insertRestaurant(restaurantEntity: RestaurantEntity)

    @Delete
    fun deleteRestaurant(restaurantEntity: RestaurantEntity)

    @Query("SELECT * FROM restaurants")
    fun getAllRestaurants(): List<RestaurantEntity>

    @Query("SELECT * FROM restaurants WHERE restaurant_id = :restaurantId")
    fun getRestaurantById(restaurantId: String): RestaurantEntity

}
```

RestaurantDatabase.kt:

Restaurant Database.

```
package com.kartikey.foodrunner.database
```

```
import androidx.room.Database
import androidx.room.RoomDatabase

@Database(entities = [RestaurantEntity::class], version = 1)
abstract class RestaurantDatabase : RoomDatabase() {
    abstract fun restaurantDao(): RestaurantDao

}
```

RestaurantEntity.kt:

Entity class file which is stored in database.

```
package com.kartikey.foodrunner.database

import androidx.room.ColumnInfo
import androidx.room.Entity
import androidx.room.PrimaryKey

@Entity(tableName = "restaurants")
data class RestaurantEntity
    (
    @ColumnInfo(name = "restaurant_id") @PrimaryKey var restaurantId: String,
    @ColumnInfo(name = "restaurant_name") var restaurantName: String
)
```

Login activity.kt:

Access to the user's account.

```
package com.kartikey.foodrunner.activity

import android.content.Context
import android.content.Intent
import androidx.appcompat.app.AppCompatActivity
import android.os.Bundle
import android.view.MenuItem
```

```
import com.kartikey.foodrunner.fragment.LoginFragment
import com.kartikey.foodrunner.R

class LoginRegisterActivity : AppCompatActivity() {

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_login_register)

        val sharedPreferences = getSharedPreferences(
            getString(R.string.shared_preferences),
            Context.MODE_PRIVATE
        )

        if (sharedPreferences.getBoolean("user_logged_in", false)) {
            val intent = Intent(this@LoginRegisterActivity, DashboardActivity::class.java)
            startActivity(intent)
            finish();
        } else {
            openLoginFragment()
        }
    }

    fun openLoginFragment() {

        val transaction = supportFragmentManager.beginTransaction()
        transaction.replace(
            R.id.frameLayout,
            LoginFragment(this)
        )
        transaction.commit()
        supportActionBar?.title = "DashboardActivity"

    }

    override fun onBackPressed() {
        val currentFragment = supportFragmentManager.findFragmentById(R.id.frameLayout)
```

```
        when (currentFragment) {
            !is LoginFragment -> {
                openLoginFragment()
            }
            else -> {
                super.onBackPressed()
            }
        }
    }

    override fun onOptionsItemSelected(item: MenuItem): Boolean {
        val id = item.itemId
        when (id) {
            android.R.id.home -> {
                openLoginFragment()
            }
        }
        return super.onOptionsItemSelected(item)
    }

}
```

Gradle file:

Plugins Used are:

```
apply plugin: 'com.android.application'
apply plugin: 'kotlin-android'
apply plugin: 'kotlin-android-extensions'
apply plugin: 'kotlin-kapt'

android {
    compileSdkVersion 30
    buildToolsVersion "30.0.0"

    defaultConfig {
        applicationId "com.kartikey.foodrunner"
        minSdkVersion 18
```

```
        targetSdkVersion 30
        versionCode 1
        versionName "1.0"

        testInstrumentationRunner "androidx.test.runner.AndroidJUnitRunner"
    }

    buildTypes {
        release {
            minifyEnabled false
            proguardFiles    getDefaultProguardFile('proguard-android-optimize.txt'),    'proguard-
rules.pro'
        }
    }
}

dependencies {
    implementation fileTree(dir: "libs", include: ["*.jar"])

    def room_version = "2.2.5"
    implementation "androidx.room:room-runtime:$room_version"
    kapt "androidx.room:room-compiler:$room_version"

    implementation "org.jetbrains.kotlin:kotlin-stdlib:$kotlin_version"
    implementation 'androidx.core:core-ktx:1.3.0'
    implementation 'androidx.appcompat:appcompat:1.1.0'
    implementation 'androidx.constraintlayout:constraintlayout:1.1.3'
    implementation 'com.squareup.picasso:picasso:2.71828'
    implementation 'com.android.support:design:28.0.0'
    implementation 'com.android.volley:volley:1.1.1'
    implementation 'androidx.legacy:legacy-support-v4:1.0.0'
    testImplementation 'junit:junit:4.12'
    androidTestImplementation 'androidx.test.ext:junit:1.1.1'
    androidTestImplementation 'androidx.test.espresso:espresso-core:3.2.0'

}
```

Chapter 5

Results

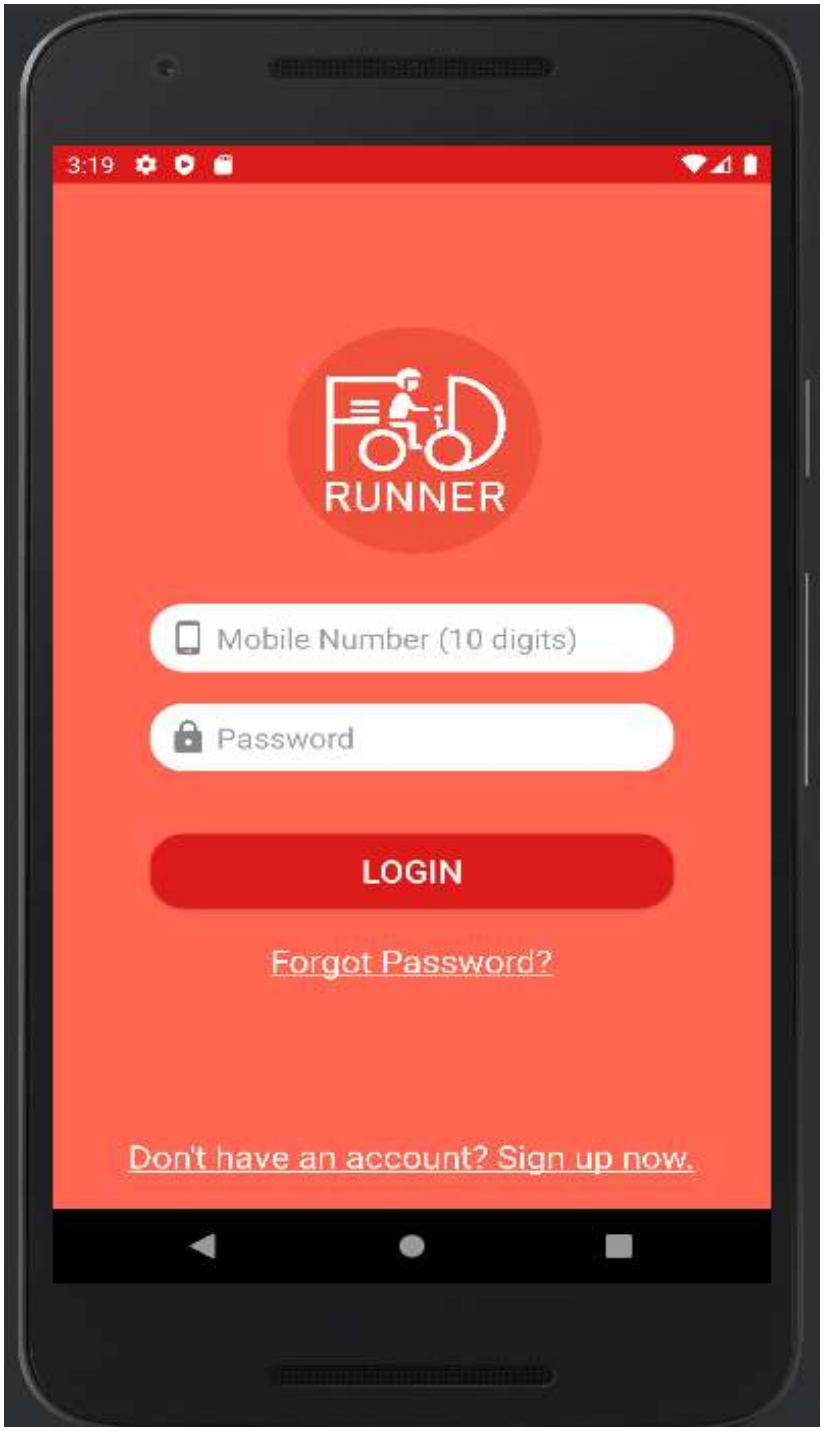

Fig 5.1 - Login Page

Fig 5.2 - Register Page

Fig 5.3 - User Profile Page

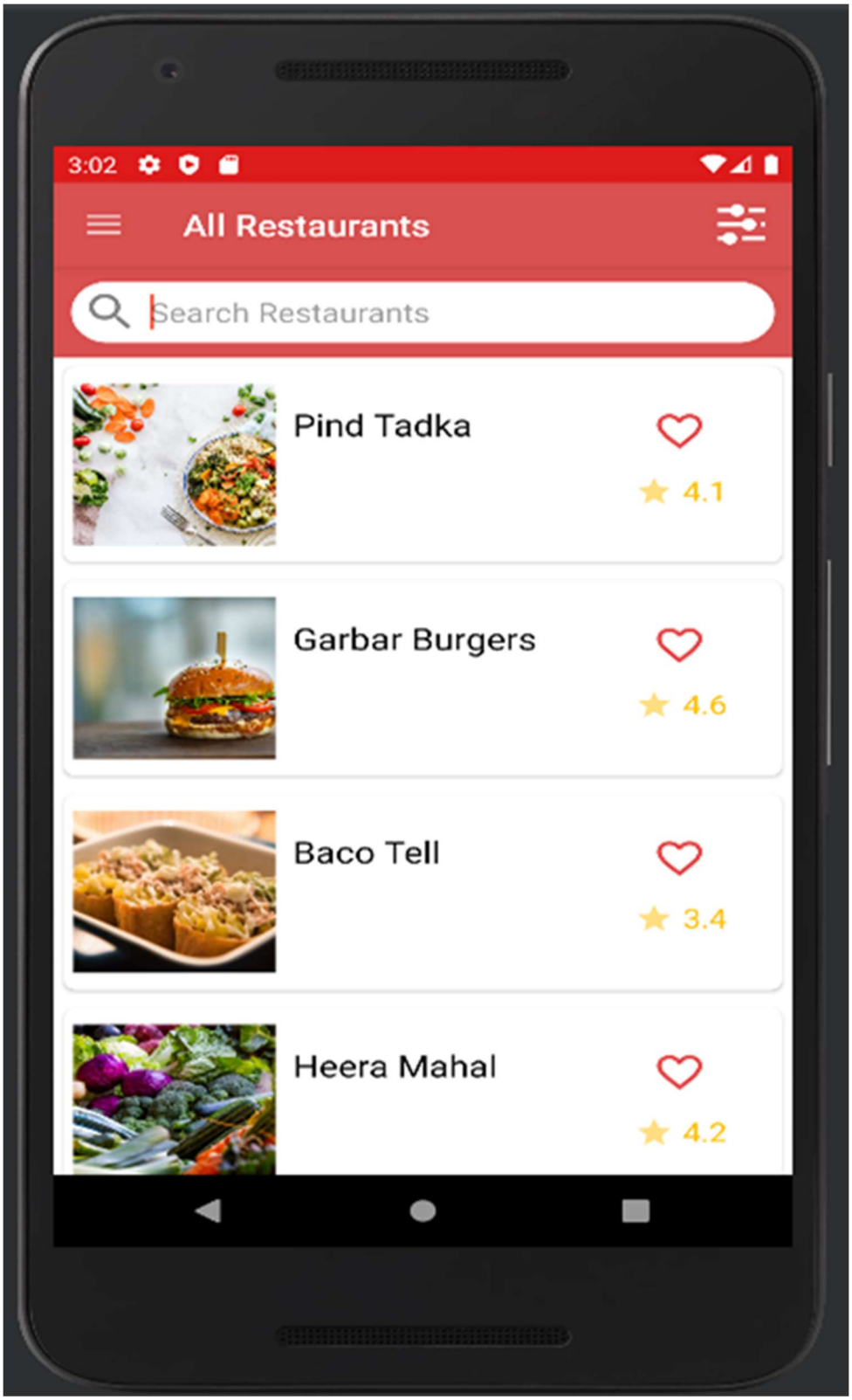

Fig 5.4 - Restaurants Pag

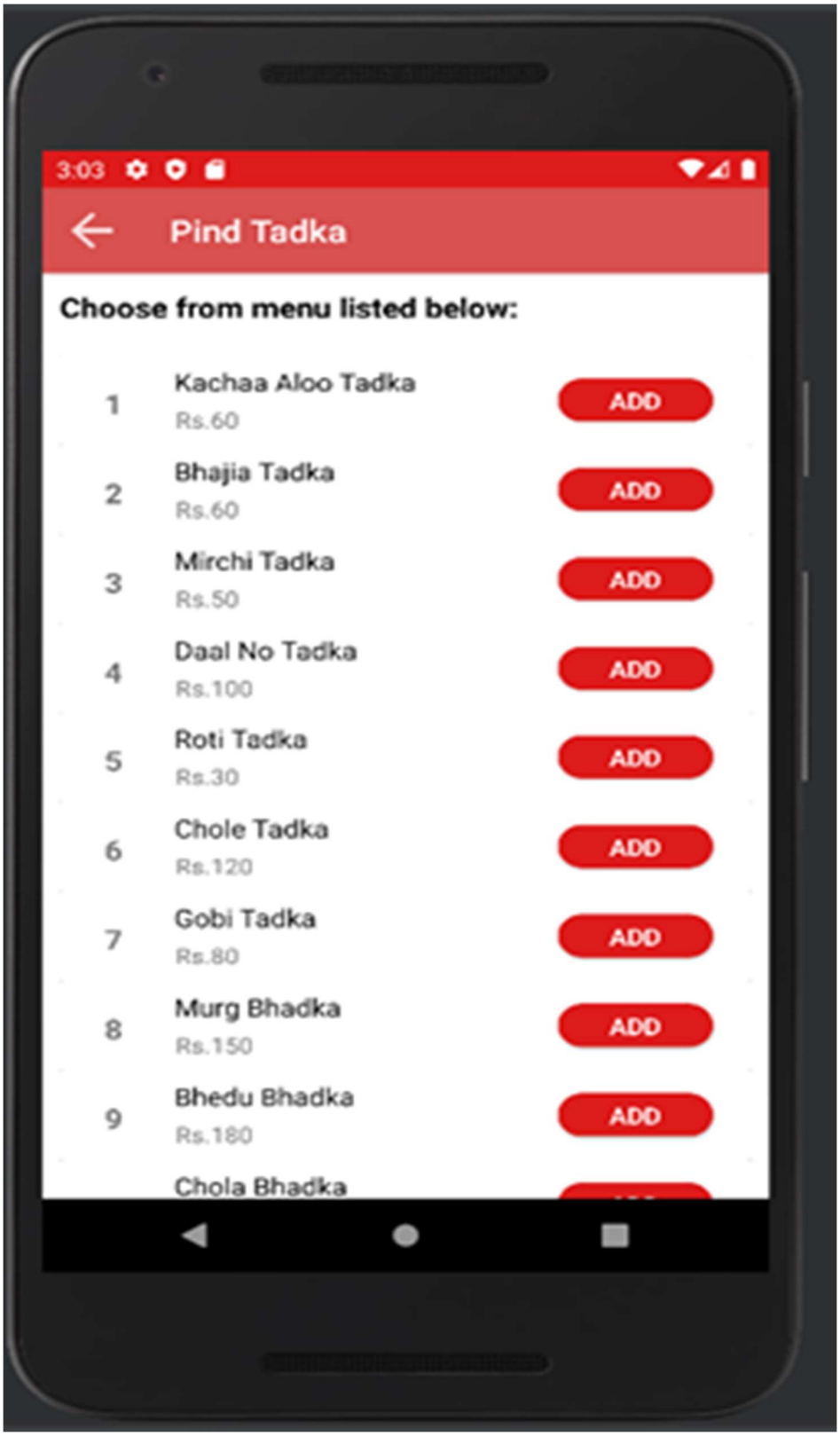

Fig 5.5 - Menu Page of Restaurant

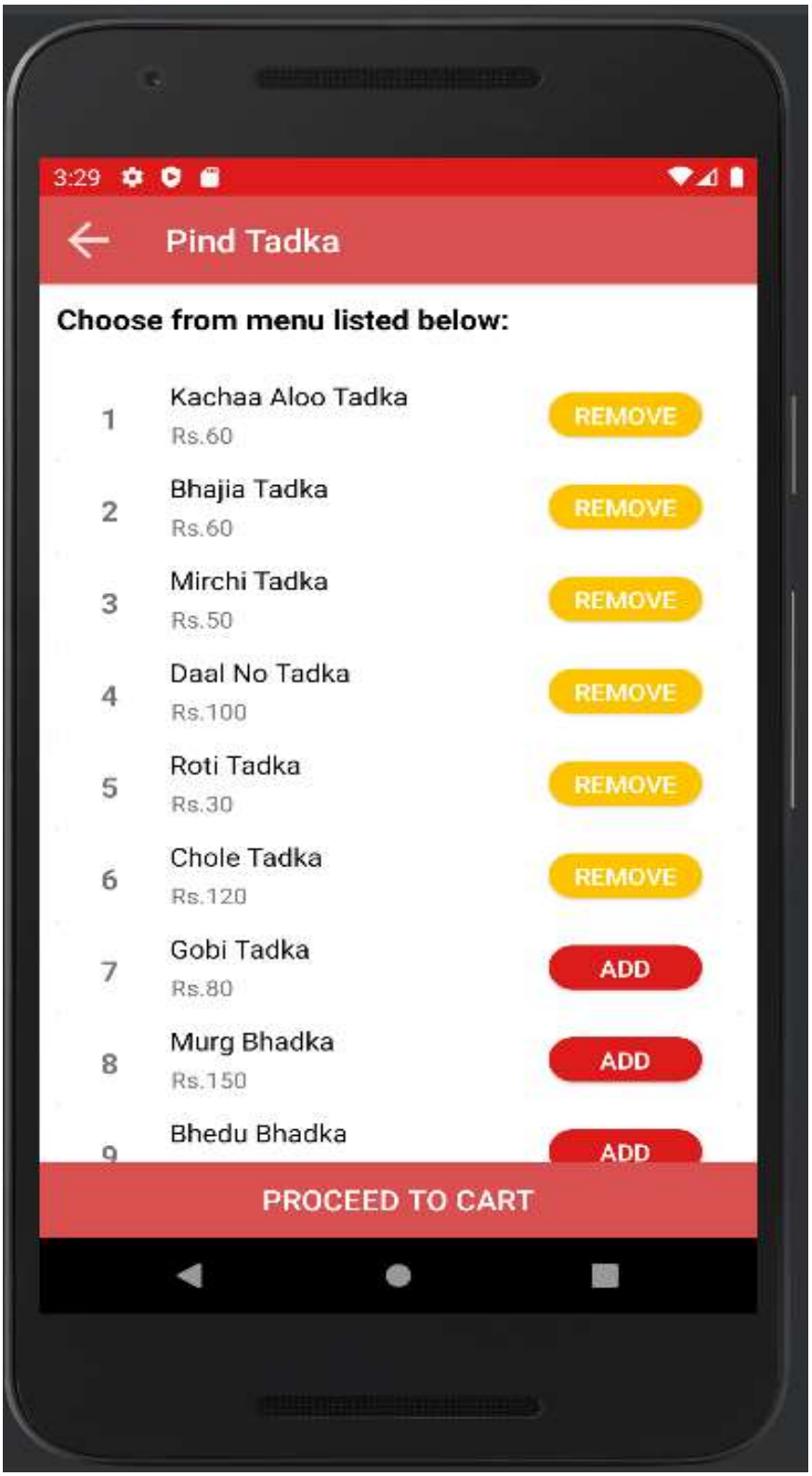

Fig 5.6 - Proceed to cart

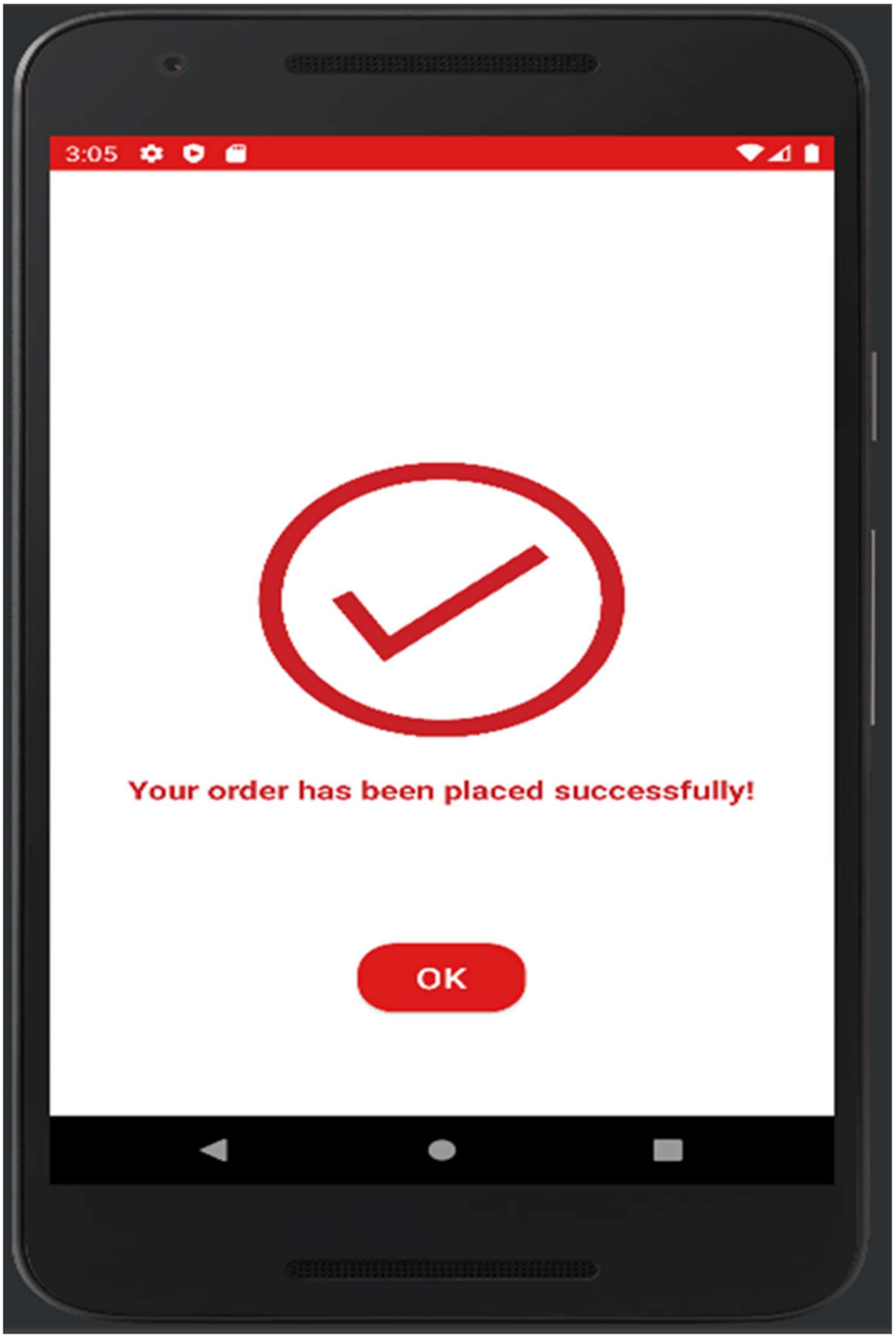

Fig 5.7- Order Confirmed Page

Chapter 6

Conclusion And Future Enhancement

6.1 Conclusion

An online food ordering system is developed where the customers can make an order for the food and avoid the hassles of waiting for the order to be taken by the waiter. Using the application, the end users register online, read the E-menu card and select the food from the e-menu card to order food online. Once the customer selects the required food item the chef will be able to see the results on the screen and start processing the food. This application nullifies the need of a waiter or reduces the workload of the waiter. The advantage is that in a crowded restaurant there will be chances that the waiters are overloaded with orders and they are unable to meet the requirements of the customer in a satisfactory manner. Therefore by using this application, the users can directly place the order for food to the chef online.

6.2 Future Enhancement

With this platform we developed, we hoping to reduce time-wastage, avoid misunderstandings, provide easy data flow, customer pleasure, and less hard work. We believe that we have accomplished our goals and satisfied with the code we developed.

There is more for the future work that can be done, can be implemented to other platform so the developers can aim to wider market, and can be done in various language so the application can suite better for each specific count.

6.3 References

1. Android programming for Beginners by Smarthead
2. https://www.youtube.com/watch?v=BCSlZIUj18Y
3. https://developer.android.com/studio/intro
4. Programming Kotlin is written by Venkat Subramaniam.
5. https://kotlinlang.org/