

## 1. Introduction:-

### 1. a) objective Of Proposed System:

House of Mandi is a hotel which is client of Yplison IT Soutlion. Objective of this app is to provide web base solution by which privileged customer can give order online and hotel receive the order for the same. Our mission is dedicated to the freshest most authentic Middle Eastern cuisine delivered with the highest level of customer service. It's also manage the details of Food, Item Category, Shopping Cart, Customer and Order. It manage all the information about Food, Delivery Address, Order, and Food. House of mandi project-Food Ordering For VIP Customers Of Specific Restaurant.in the project two person admin and customer and delivery boy have perform task.

### 1. b) Details of Project:

House of Mandi is a hotel which is client. Objective of this app is to provide web base solution by which customer can give order online and hotel receive the order for the same. It's also Manage the details of Food, Item Category, Shopping Card, Customer, Order. It Manage all the information about Food, Delivery Address, Order, Food. House\_ of mandi project-Food Ordering For VIP Customers of Specific Restaurant.

### 1. c) Need of System:

May be have a strategy to increase the additional sales for your existing customer? To do this you can use the project category which gives you the opportunity to report on another dimension Apart from the project type.

customer and manage our profile like change password reset password change profile image .he will food items management like as-Addition , Updating ,and Deletion of food items and food category management. He will receive food orders status management weekly, monthly and yearly reports for completed orders

Feasibility is an important phase in the software development process it enables the developers to have an assessment of the product being developed it refers to the

feasibility study of the product in terms of outcomes of the product, operational required for implementing it. Feasibility study should be performed on the basis of various criteria and parameters.

#### **1. d) Economical Feasibility:**

This is very important aspect to be considered while developing a project. We decided the technology based on minimum possible cost factor.

- All hardware and software cost has to be borne by the organization.
- Overall we have to estimate that the benefits the organization is going to receive from the proposed system will surely overcome the initial costs and the later on running cost for system.
- To develop an easy way to use web based interface where users can search for product view the details of the product and order it without going to market.
- The searching product can be done by product category, manufacturer as well as latest product, view it purchase it become a convenient way for customer.
- Customer can add product to cart to purchase, delete the product from cart before selecting the final submission.
- A user can view the complete specification of the product with various images and also view the customer review the product.
- It minimizes the shopping time of customer, increase the point of choice.
- It also facilitate the service provider to know the current stats of market and take decision which product are selling more now a days and have to keep in store.

## 1.1 Module Description

### Admin role:-

Admin will login and logout admin approval to login customer and manage our profile like change password reset password change profile image .he will food items management like as-Addition , Updating ,and Deletion of food items and food category management. He will receive food orders status management weekly, monthly and yearly reports for completed orders.

- 1) Admin Login and Logout
- 2) Admin Profile Management (Change Password, Forgot Password)
- 3) Approval for Requesting VIP Customers
- 4) Food Items Management (Addition, Updation and Deletion of Food Items)
- 5) Food Category Management (Addition, Updation and Deletion of Food Categories)
- 6) Coupon Management (Addition, Updation and Deletion of Food Coupons)
- 7) Registered VIP Customers Section
- 8) Received Orders Section
- 9) Received Orders Status Management (Changing Status of Orders)
- 10) Weekly, Monthly and Yearly Reports For Completed Orders

### Customer role:

In the customer section customer will sign up by name, email and phone number. When admin approval to login by email link or text msg then the customer will login and logout from the android app portal .customer can manage profile account like-change profile image, reset password change password and delete profile food

- 1) Customer Registration (Verification via Email and SMS: Login Allowed Only on approval of Admin)
- 2) Customer Login and Logout
- 3) Customer Profile Management (Change Password, Forgot Password, Profile Deletion)
- 4) Category-wise Display of Menu Items (categories such as drinks, main course, salads etc.)
- 5) Quantity Selection Option for each Food Item
- 6) 2 options for buying (Buy Now, Add to Cart)
- 7) Order History Section (Placed Order Status)

## 2. Company Profile:-

Ypsilon IT Solutions is a company with commitment and passion for providing value to its customers by enabling technology. At YPSILON, we firmly believe that technology can create value only if it helps organizations achieve their business objectives and sustained competitive advantage. Ypsilon's focus is on helping companies strategize, plan and deploy cutting edge technologies, with the overall purpose of contributing in realization of their business goals. As an IT Consulting and Services firm, YPSILON offers services in the areas of ERP, e-Business, EAI, Data warehousing, Quality Analysis and Testing.

YPSILON employees make a rich pool of talented, experienced and technically proficient professionals, who provide clients with technology expertise, execution capabilities and most importantly high levels of delivery. Ypsilon's management has several years of proven experience in providing technology solutions and shaping IT initiatives for large and mid-size corporations.

Based on our belief that customers can best be served only when technology is amalgamated with the clear understanding of its application in business, YPSILON consultants offer solutions by alloying technology, industry knowledge and valuable experience. Aligning our service delivery and approach for each client and their specific requirements, we strive for highest level of customer satisfaction. This has resulted in a growing list of delighted customers for YPSILON.

External Guide: Prof. Vikash Sharma

**Address:** 8/1, Dr. R. S. Bhandari Marg, Race Course Road, Indore, Race Course Road, Indore, Madhya Pradesh 452001, India

PVHH+62 Indore, Madhya Pradesh, India

Contact No: +91 98933 03390

Website: [www.ypsilonsolutions.com](http://www.ypsilonsolutions.com)

### 3. System Analysis And System Design

#### Scope of the work:-

This android software system will be designed to implement the process of food ordering from a specific restaurant. The different parts of the system are as follows:

(a) Types of Users: Admin and Customers of the constituent Restaurant.

(b) Roles of Users:

Role of Admin: The admin of the app will have master access to the system. He will have full access to the database that will contain approval of customers, OTP sending, OTP verification, delete customer, add, update and delete food category and food items, change order status, payment of vip customer. He can modify the information in the database anytime. Only he will have the access to the encryption key that will allow him the access to important information like transaction history, restaurant details and information of vip customer.

Role of Customers: The customers will log into the app using their email ID's as username and their password. The customers will have access to their own information only that includes view profile, view food item, view food category, place order payment and transaction history along with their personal information.

#### Detail information about system modules

System Features:

Login – There will be separate one-step login procedure for admin and customer in which the user will be required to enter his unique user id, his password.

Payment Gateway – This will be an either a paid gateway of any suitable bank or if the service will be unavailable then we would have option of cash on delivery.

Automatic receipt generation – After the order payment will be over, the receipt of the fee payment will be generated automatically by the app. In case any exception occurs then failsafe features like forwarding the Email that the user receives from the bank to a specified number or uploading the receipt downloaded from the bank website as a proof of payment, will be present to ensure the confirmation of order payment.

Confirmation through e-mail id – As soon as the automated receipt generation is complete a confirmation mail will be sent to the user's registered email then user can enter OTP and till wait for approval of admin to be VIP customer. Both of these will help the user to keep record of the payment and would help the user in confirming the completion of the payment procedure.

Database that will store customer' information , food order , last app open along with the transaction history that will contain all the receipts generated till now for record. There will be different levels of access to this database based on the type of user.

Report Generation: After the process of order payment is complete for all the VIP customer the system can generate a summary report.

## **System Design**

System design is the solution for the creation of a new system. This phase focuses on the detailed implementation of the feasible system. It emphasis on translating design. Specifications to performance specification. System design has two phases of development.

- Logical design
- Physical design

During logical design phase the analyst describes inputs sources, output destinations, databases data stores and procedures data flows all in a format that meets the user requirements. The analyst also specifies the needs of the user at a level that virtually determines the information flow in and out of the system and the data resources. Here the logical design is done through data flow diagrams and database design. The physical design is followed by physical design or coding. Physical design produces the working system by defining the design specifications, which specify exactly what the candidate system must do. The programmers write the necessary programs that accept input from the user perform necessary processing on accepted data and produce the required report on a hard copy or display it on the screen.

### **3.1 Preliminary Investigation**

#### **Literatures Survey:-**

This involves studying the current system to find out how it is working and where the improvements should be made. These studies consider both manual and computer methods. Hence an early step in investigation is to understand situation.

#### **Activities In Requirement Determination:**

- Requirement Investigation:
- Requirement Specification:

#### **1. Requirement Investigation:-**

This activity is at the heart of system analysis. Using a variety of tools and skills analyst study the current system and documents its features for further analysis. Requirement investigation relies on the fact-finding techniques

#### **2. Requirement Specification:-**

The data produced during fact-finding investigation are analyzed to determine requirement specification. This is the description of features for new system.

## 3.2 Feasibility Study.

### Fact Finding Techniques:-

Fact-Finding is the formal process of using research, interviews, questionnaires, sampling and preferences. It is also called information about systems, requirements, and preferences. It is also called **Information Gathering** or **Data Collection**. Tools, such as data and process models, document facts, and conclusion are drawn from facts. If you can't collect the facts, you can't use the tools. Fact-Finding skills must be learned and practiced.

**Different types of Fact-Finding techniques are:**

- \* INTERVIEWS
- \* QUESTIONNAIRE
- \* RECORD REVIEW
- \* OBSERVATION

#### ❖ Interviews :

Interview technique is used to collect the information from individuals groups. Analyst should select responds that are related with the system under study. In this method the interviewer (analyst) faces to face with respondent & records of his/her responses. This interviewer must plane in advance and should fully know the problems under consideration. He must choose a suitable time & place, so that the interviewer may feel at ease during interview.

#### ❖ Questionnaire:

A questionnaire performs containing a sequence of questions to elicit information mostly from a large no of persons. Drafting of questionnaires requires skill. The questions must be clear, simple & to the point. They must be well organized from the point of view of the



respondent and formulated in such a manner as to provide the data in so far as possible in the desired form. A questionnaire may be mailed to individuals who are requested to write the answer of each question and return complemented performs back by post.

#### ❖ **Record view:**

Information related with the system may be present in the form of records like books, magazines, newspaper, historical documents, letters, journals, manuals, government publications. This kind of record review provides very valuable information to the analyst about the system, organization & various procedures & rules.

Record review may be performed in the beginning of study to collect initial information or at the end of the study to compare actual operations.

#### ❖ **Observations:**

If information is not collected from the other fact-finding methods, then observation method is used. In this method analyst observes the flow of documents, way the process is carried out, step followed, the persons involved etc. If the analyst is familiar with the system then he/she knows what to observe and how to gather information. In experienced person may observe unnecessary things, which delays the system study.

### **3.2.1 Technical Feasibility Study**

In this system we use Android, JAVA, and JSP (API)

Platform for programming language. Android, JAVA Platform means the environment which is used to run program. JAVA is platform independent language since no only single operating system can be required by the java. All the ANDROID operating system can execute the .apk file.

Android, Java provides huge functionality that means it provide A huge library.

- ❖ Containing lots of reusable codes.
- ❖ An execution environment that provides services such as security.
- ❖ Portability across operating system.
- ❖ Automatic garbage collection.

### 3.2.2 Operational Feasibility

Our experts at House of Mandi are well aware of the fact that if a new system is not user friendly, it would not have app for food ordering and managing customer profile. the expected results. Keeping this in mind the professionals at House of Mandi do a complete research to assist you in successful completion of any given project. As we lay our foundation on customer satisfaction, we do our every bit to accomplish it. Our experts are well equipped with a wealth of knowledge and skills to provide you excellent services. Our experts always take care that the management and the employees support the project. There are various advantages you can get at House of Mandi, like:

- Our team prepare and implement the project in a way that the project is well accepted by the management as well as by the users.
- Our experts coordinate with the customers and restaurant management and get information about the current system and try to implement a time saving system in an order to be accepted by the employees.
- When we carry out the project, we try to involve the employees as early involvement in the project will help them understand the project in a better way.
- We always try to estimate the expected profitability of the organization after the project is successfully completed.
- We also take utmost care that the overall response increases benefiting the organization.

### 3.3 Software Paradigm Applied

#### Web-Based App Development

Web-based apps are applications or websites that run *within* the mobile web browser, rather than the platform's native UI framework. The difference between an "app" and a "website" in this context is purely semantic: basically an app enables users to complete

a task (e.g. finding movie times), whereas a website is broader in scope and is used mainly for accessing information (e.g. finding out about a movie theater chain).

While I hope at this point that every website you build is responsive or adaptive to multiple screen sizes, if you are building an app you could do it with HTML5, CSS3 and JavaScript and deploy it purely over the web. Libraries such as jQuery Mobile and Sencha Touch make this incredibly easy. The big advantage is that you can bypass each platform's store to deliver your product directly to your customer. Unfortunately the downside to this is also what makes web-based apps unique – that they run inside the web browser. As a result, they can be perceived as being harder to use and less intuitive than a native app. This perception is not always true, however. It is in fact possible to develop web-based apps that are on-par with native apps.

Take Facebook, for example. In September 2012, Facebook CEO Mark Zuckerberg said the following at the Disrupt SF conference:

“When I’m introspective about the last few years I think the biggest mistake that we made, as a company, is betting too much on HTML5 as opposed to native... because it just wasn’t there. And it’s not that HTML5 is bad. I’m actually, on long-term, really excited about it. One of the things that’s interesting is we actually have more people on a daily basis using mobile Web Facebook than we have using our Android or Android apps combined. So mobile Web is a big thing for us.”

While he didn’t dismiss HTML5 entirely, it was pretty clear that he believes that, for now, the Facebook, iOS and Android native apps are superior. In response, the folks at Sencha cloned the native Facebook app within their HTML5 mobile app framework.

Therefore, if done right, it is possible to build great experiences that live within a mobile web browser. Web-based apps are completely platform-agnostic, and the HTML5/CSS3/JavaScript codebase is one that is accessible by many developers and offers a very streamlined approach to app development.

### 3.4 Software Development Life Cycle Paradigm

Making a mobile App is no mystery in today's time however making a successful mobile App is a process which involves quite an extensive pre-planning. There has been a meteoric rise in the App market in the past few years with millions of Apps in the two prominent App stores. In such a case when the competition is so huge, don't you think that you should be really thorough with the basic concepts of App building? Also you have to really make it stand out in function, in its use and in its design to attract the consumer's attention. And to achieve all this you have to get it right the very first time. Since, it is so important for an App to be spot-on the first time hence it is necessary for developers to follow a step by step process to building an App. At Queppelin we build a mobile App for you in eight different phases and each phase has a number of steps, involved. This process of building an App following a detailed one step at a time approach is called Mobile App Development Lifecycle. To understand it further – Mobile App Development Lifecycle is just a representation of the conventional Software Development Lifecycle (SDLC) but from the perspective of a mobile device. So without further ado let us have a look at the eight phases of Mobile Application Development cycle.

*Phase 1: Pre-planning and research*

The first phase is the most important one because it is during this phase that you lay down the necessary ground work for what is to follow next. During this step it is very important to do substantial research and brainstorming before moving on to the next phase. You need to do the homework and have answers to questions like – What is the main aim of this App? Who is the target audience? Which platform should you target first? Is the App going to be free or paid?

Once you have the answer to all these questions then you clearly know that how much time it will take for you to develop the App. Another thing which is a must in this phase is the analysis of the competition. Do a detailed study of your competitor's App to see what features they are offering. Try to figure out the features which are absent in their App so that you can include it in your App, to make it stand out.

Once you have all this information then the next thing which you need to do is to lay down the cost and the time for App development.

### *Phase 2: Mental Prototyping*



Once you are done with the research and have laid down the costs involved then the next phase involves preparing a detailed scope of work. You need to do a mental prototyping of your App and draw your ideas in the form of sketches on a whiteboard. This will be the first visual representation of the ideas which you collected in Phase 1 and it will help you uncover usability issues.

Another thing which needs to be done in this phase is to take the feedback of relevant people to get a perspective of what they think about your idea. Discussing it with them will help you figure out the loopholes and allow you to look for a solution to tackle with them.

*Phase 3: Assessment of Technical feasibility*

Having an understanding of the visuals is not enough because you need to analyze whether the back end systems will support the App's functionality or not. To understand if the idea of your App is technically feasible you need to get access to public data by simply sourcing public APIs. You also need to determine which platform you are building your App for, first. Building an App shall have different requirements depending upon its platform (Android/iOS) as well as its format (tablet/smartphone).



*Phase 4: Building a prototype*

You cannot define the touch experience until and unless you actually touch the App and see how it works and flows. In order for that to happen you must build a prototype and get the experience of the App into the users hand as soon as possible. This will help you see if things are going in the right direction. In this phase you can use rough and not exhaustive wireframes. Including the stakeholders in this process and allowing them to touch the prototype shall help you take their feedback and implement it into your work.



## Phase 5: Designing and development of App



Pic credits: helpweixin.com

Before moving to coding you must design your App. A User Experience designer can create the interaction architecture of the design elements while a User Interface designer can create the look and feel of your App. This in itself is a multi-step process and the end results is visual directions and blueprints which gives envision of the final product. It also informs you on how an interaction should feel, move and flow.

*Phase 6: Building the App using Agile methodologies*

Once the design is ready then it is the time for you to build the App. **Agile methodology** is the best approach for mobile application development as it allows you to make changes, add new features and keep evolving with the changing trends.

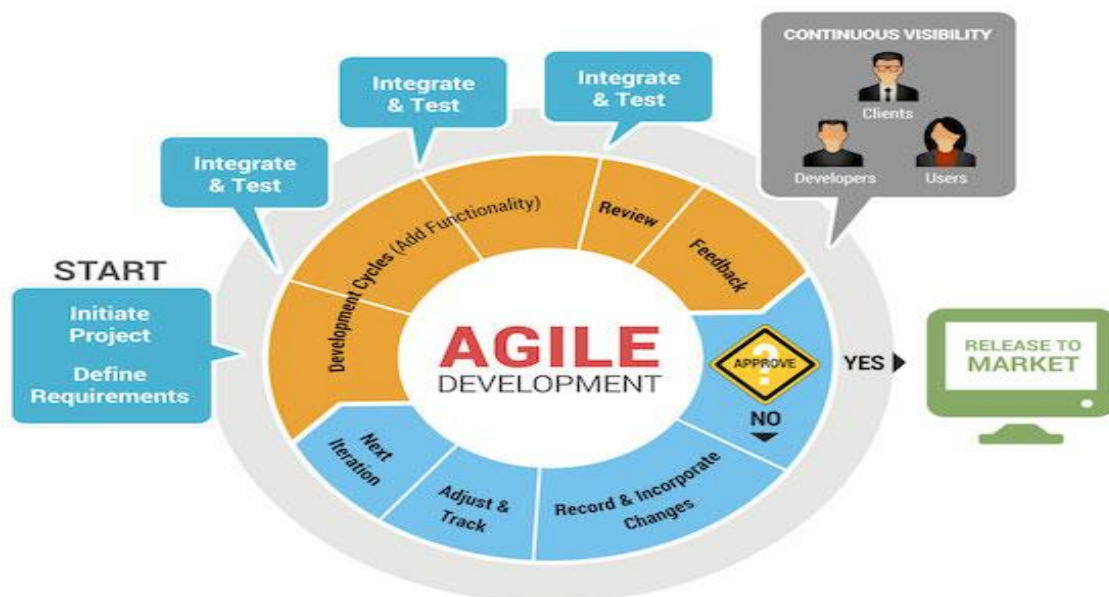
Traditionally, applications were developed using the Waterfall methodology. This is a sequential method where app development is done in phases. The next phase is started only when the previous phase is complete. There is little emphasis of communication between the app developers and the customers. At the end of app development, a test for the app feasibility and marketability is done to analyze whether the app is of any value or not. Sometimes, the app development may take such a long time that by the end of the process the market dynamics have changed and the app is rendered irrelevant. As customer-development interaction in waterfall methodology is limited, developers are unable to gauge the mood of customers. By the time this is assessed it may be too late to make any changes in the app to make it more saleable. The waterfall method allows

little flexibility in app development. It is a risky model not suitable for long and ongoing projects and projects where requirements are at a high risk of changing.

There was a need for a model which ensures that the app we are developing caters to the need of the customers. This gave rise to the Agile Methodology in the 1990s. Agile methodology is an incremental model of app development. The project is developed and released in incremental stages or 'iterations'. It gives ample opportunities to the developers to assess whether their app is moving in the right direction throughout the development cycle. If the developers feel their app needs some modifications due to a change in customer needs, there is scope for the project to be pivoted around at any stage of app development. The result of this "inspect-and-adapt approach" is to greatly reduce the effort and time of the developers and come up with an app with a high probability of success.

### How does the Agile Methodology work?

In the Agile methodology each project is broken up into short-duration 'iterations' or 'sprints'. Each iteration would be of the same duration lasting a week or so. At the end of each iteration, a simple working model of the project would be launched in the market to check its feasibility. Regular team meetings would be held at the end of each iteration for assessing the possible roadblocks. Each iteration offers freedom to the developers to make changes to the project depending upon the market response to the launched version. Any features which are not delivered in the first iteration would be incorporated on a priority level in successive iterations. The Agile model offers freedom and flexibility to developers. It gives ample opportunities for interactions between developers and customers and also between the development teams handling the different aspects of the project like finance, designing, coding and marketing.



Agile method is most suitable for small projects which cater to customers whose needs might be moderately or highly variable. An example to consider is of Instagram app which is very popular these days. When Instagram was conceived, it was found to be very similar in its features to Foursquare. So the developers realized that the project could fail. They pivoted the project and introduced the photography element in their app which has made Instagram so widely used and appreciated. Had this app been developed on the Waterfall methodology, the app would have been launched at the end of a long and tedious **development process** only to find that a similar app already existed in the market. What a waste it would have been to the developers and investors!

### **Some pros and cons of the Agile Methodology**

#### ***The Agile methodology has many advantages:***

- Working software is delivered to the market frequently at the end of an iteration.
- There is more room for making changes and adding features to the app at any stage of its development.
- More stress is laid upon people and interactions rather than on tools and results.
- Continuous developer-customer interaction and interteam interactions lead to good collaborations and better results.

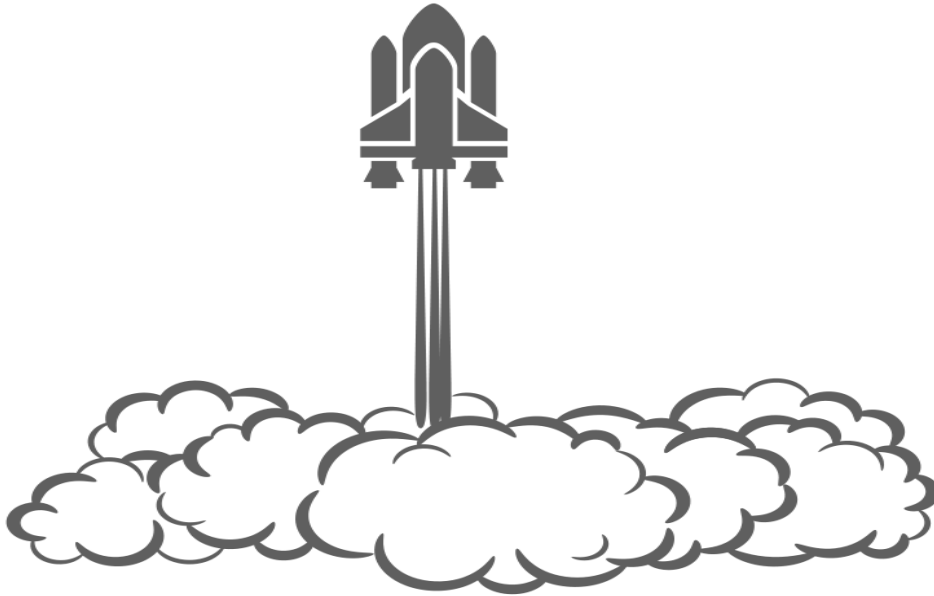
*Phase 7: Testing the mobile App*

Pic credits: [blogs.thinksys.com](https://blogs.thinksys.com)

Congratulations on building your Mobile App. For Phase 7 you need to get some target users to help you test the App.

UAT Testing: For user experience testing you need to put your App in the hands of the users which you are targeting and once it passes the UAT test you know that the solution which you are providing actually works.

Beta Testing: Make your App available for the beta trial by allowing open participation of people to test it. The feedback from these beta users will help you determine if your App's functions work well in the real world environment.

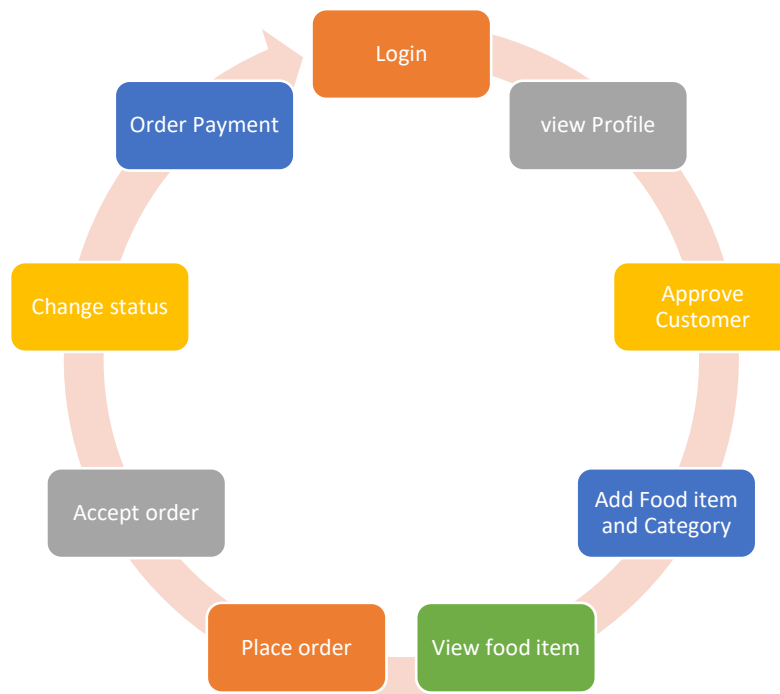
*Phase 8: The launch*

Once Phase 7 is successful, your App is complete and is ready to be submitted to the App stores for approval.

However, this is not the end as every App requires regular updates and new features to be added to it during the mobile application development lifecycle. The development cycle begins once again as soon as the first version of the App is launched.

### 3.4.1 Prototype Paradigm

#### a. Admin paradigm



#### b. Customer Paradigm



## 4. Software and Hardware Requirement Specification: -

### 4.1 Specification

Micro Processor	Quad Core or Above
RAM (SD/DDR)	512 MB or Above
Internal Space	1GB or above.
Device	Qualcomm MSM8226 Snapdragon 400
Operating System	Android
Data Base	MySQL 5.0
IDE/Tools	Android Studio/ Net Beans
Web Server	XAMPP /WAMP/ Apache Tomcat

### 4.2 Runtime Time Requirement

- a) Android OS
- b) 512 MB RAM
- c) 1GB internal space
- d) Internet Connectivity



## 5. System Design

**Systems design** is the process of defining the architecture, modules, interfaces, and data for a system to satisfy specified requirements. Systems design could be seen as the application of systems theory to product development. There is some overlap with the disciplines of systems analysis, systems architecture and systems engineering.

### 5.1 Different designing techniques

#### a) Structured Design:

Structured design is a conceptualization of problem into several well-organized elements of solution. It is basically concerned with the solution design. Benefit of structured design is, it gives better understanding of how the problem is being solved. Structured design also makes it simpler for designer to concentrate on the problem more accurately.

Structured design is mostly based on 'divide and conquer' strategy where a problem is broken into several small problems and each small problem is individually solved until the whole problem is solved.

#### b) Function Oriented Design

In function-oriented design, the system is comprised of many smaller sub-systems known as functions. These functions are capable of performing significant task in the system. The system is considered as top view of all functions.

Function oriented design inherits some properties of structured design where divide and conquer methodology is used

#### c) Object Oriented Design

Object oriented design works around the entities and their characteristics instead of functions involved in the software system. This design strategies focuses on entities and its characteristics. The whole concept of software solution revolves around the engaged entities.

## 6. Optimization of Code

### 6.1 Optimizing own code

Optimization of the code is directly related to size of our application.

**1) Remove unused Resource** – Resource like image can take much size in our apk.so we should remove unused resources from res folder.

**Ex.** – There are two ways to remove unused resource from apk.

**CTRL + ALT + SHIFT + i** – Type unused resources and Enter which will give list of resources (drawables,strings etc.) which are not used in our app currently. We can delete them.

**Shrink unused resources from gradle file.**

```
android {
    buildTypes {
        release {
            minifyEnabled true
            shrinkResources true
        }
    }
}
```

Which will remove unused resource from your release apk.

**2 ) Don't use Images for all Density** – Android supports different density with different devices like ldpi,mdpi,hdpi,xxhdpi,xxhdpi. Use only those density images which you think large number of users are going to use. It is already recommended that all devices should use at least xxhdpi images.

**3 ) Avoid Frame by Frame Animation** – Frame by frame animation can enlarge for application, which is having multiple images for different density. Try to achieve such things with GIF images.

**4 ) Reuse Images** – We can use same resources like we need an image with mirror effect.

So we can use same image with rotation.

**EX.**

```
<?xml version="1.0" encoding="utf-8"?>
<rotate xmlns:android="http://schemas.android.com/apk/res/android"
    android:drawable="@drawable/ic_arrow_expand"
    android:fromDegrees="180"
    android:pivotX="50%"
    android:pivotY="50%"
    android:toDegrees="180" />
```

**5 ) Remove unused Code** – Nowadays we use so many third party libraries to achieve some features easily in our app which is nothing wrong to do, but there are some unused code and classes that takes place in application.

Android provides facility of Proguard to remove such code from your application.

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

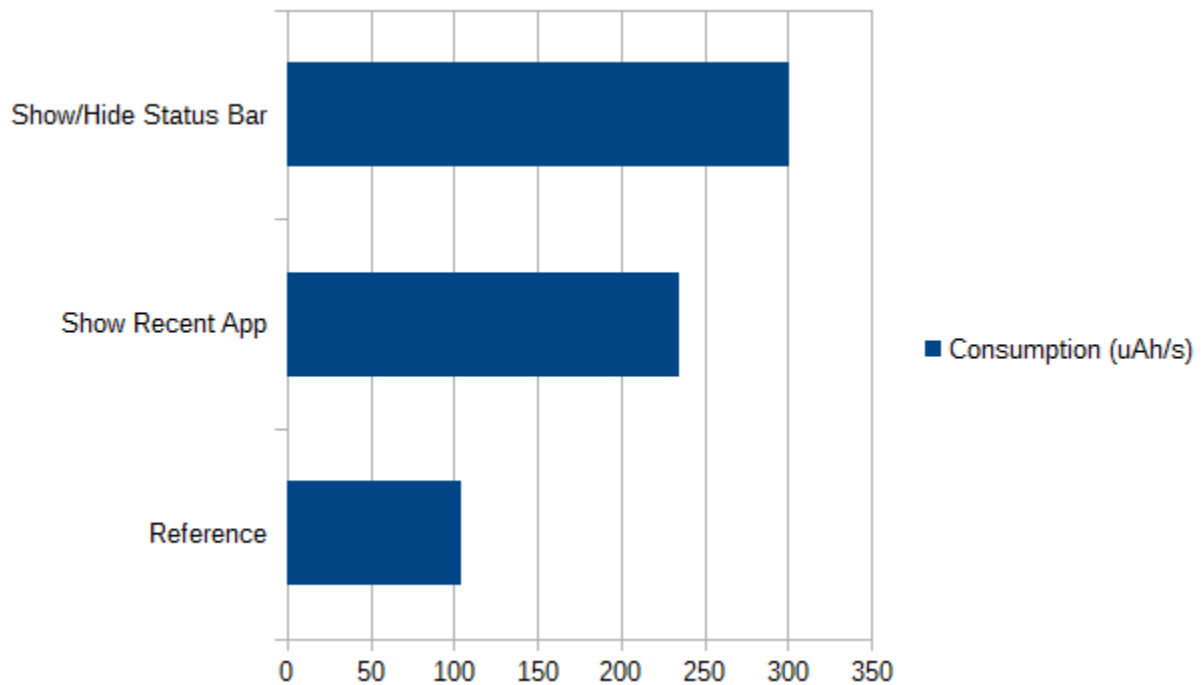
## 6.2 Code efficiency

### Step 1 : Let's get to work

**First step in any optimization approach: measure.** If you want to improve things, you need to assess their status. So we have **to measure the consumption of resources** (CPU, RAM, data) and **consumption of energy** caused by using this app. While we're at it, please note that the energy is theoretically expressed in Wh (Wattt.hour), but in a mobile context, it's usually spoken of in Ah (Ampère.hour) which is a convenient way to refer to the battery's capacity. Typically, the battery of your recent device has a capacity of 3,000 Ah.

This is where the **Android developer** is often left barehanded. Luckily enough, we work for a customer of [GREENSPECTOR](#), so we have access to our favourite tool. The workshop is therefore composed of: - a [GREENSPECTOR](#) server, - a developer's workstation (with Android Studio, adb, UIAutomator...), - a smartphone: in our case, it's a Nexus running Android 7.0, on which we install the [GREENSPECTOR](#) energy probe dedicated to this model.

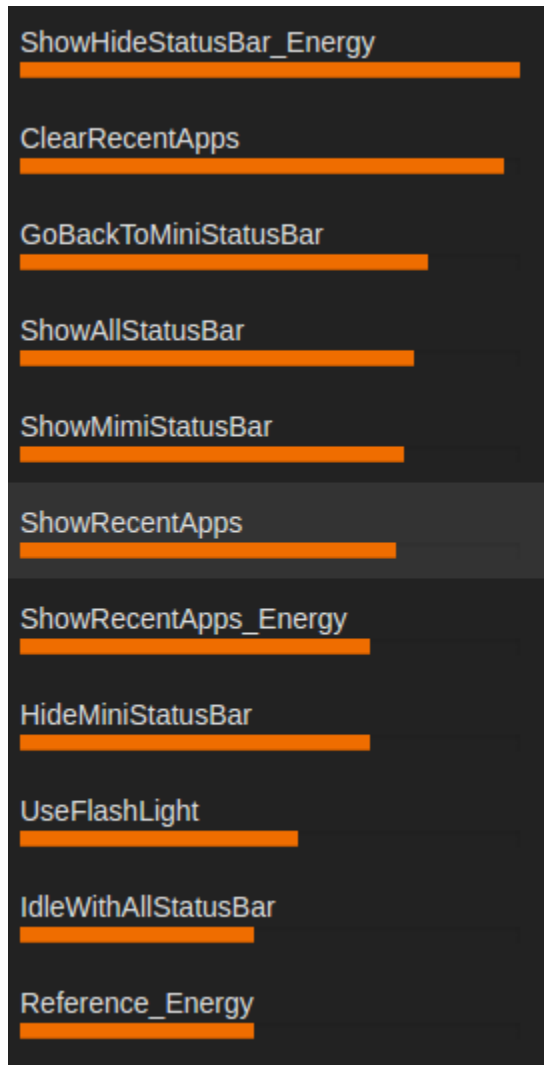
Hence, we can launch test runs for each of the functional cases that we've selected. Toss in some UIAutomator to automate simple test cases, and calls to the "Meter Android" API from [GREENSPECTOR](#). Oh! And don't forget, prior to running the functional test cases, to run a "blank" (or "idle") test case to establish the reference consumption for our platform. First results come in soon:



The **ShowStatusBar test** case consumes 3 times more than the reference (idle) case, and ShowRecentApp consume 2.4 times more. That's a lot!... or not? This is the question we are asked every time. Here, you may intuitively say that it's a bit heavy, given that it's only for displaying icons and notifications. In any case, these first measures enable us:

- **To compare the functional cases with respect to the consumption of the platform when idle** (which is much more relevant than to try and assess absolute figures);
- **To compare the functional cases between them** (which makes it possible to prioritize the rest of the work, more on this later).

Please note that I'm not giving away all the details here, in order to keep this piece readable for most readers. But you'll be able to find them in a document available for download on our site. For example, here's a screenshot of [GREENSPECTOR](#) with all the measured functional cases, and their relative impacts:



## Step 2 : Look for the big rocks

**An optimization work is not an exhaustive and planned approach.** We are in a constant search of balance between the hoped-for gains, and the workload that would be needed to obtain these gains. Hence, we use this good old  $80/20$  rule, or what we like to call “looking for the big rocks”: if your road is blocked by a rock, you don’t have to mind the sand in your shoes yet.

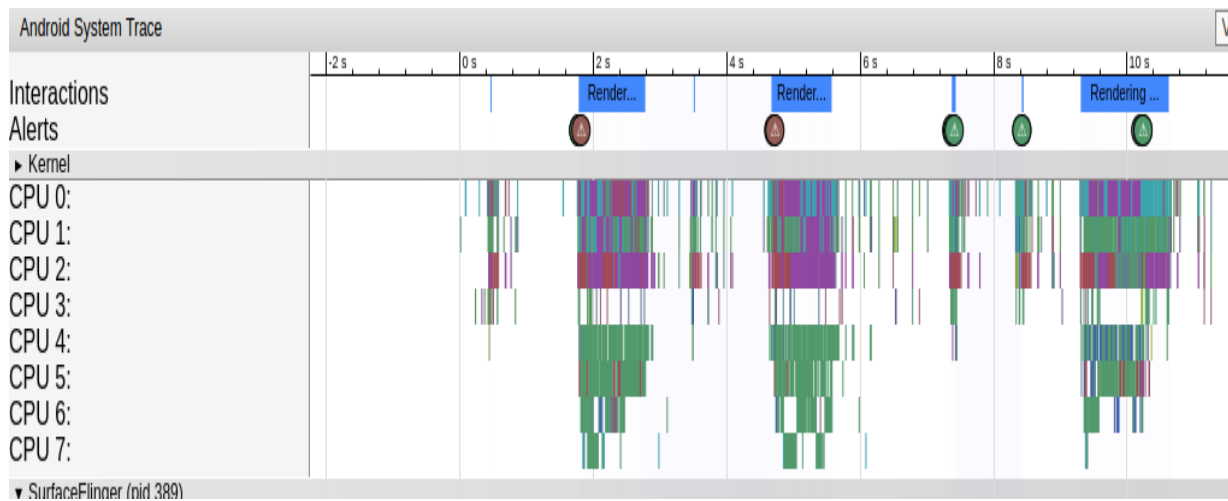
In our case, the search for the big rocks has already begun: thanks to the measures carried out, we are able to target the most consuming test cases.

When we get the experts toolbox out of the van (**Android Systrace**, **Android Traceview**, **HierarchyViewer**...) we already know where and how to use them. These tools being very accurate on narrow points and their understanding being rather arduous, the foremost step saved us a lot of time. Here's a sample TraceView result, filtered on SystemUI

Method	Excluded Time (ns)	Included Time (ns)	Calls
com.android.systemui.qs.TouchAnimator\$FloatKeyframeSet.interpolate	7007	177020	1326
com.android.systemui.qs.TouchAnimator.setPosition	5205	223740	349
com.android.systemui.qs.TouchAnimator\$KeyframeSet.setValue	4416	181553	1326
com.android.systemui.statusbar.stack.NotificationStackScrollLayout.getNotGoneChildCount	1910	3316	100
com.android.systemui.qs.QSTile\$State.copyTo	1683	11725	29
com.android.systemui.qs.QSContainer.setQsExpansion	968	266442	32
com.android.systemui.statusbar.stack.NotificationStackScrollLayout.getFirstChildNotGone	907	1584	78
com.android.systemui.BatteryMeterDrawable.<init>	837	15886	4
com.android.systemui.statusbar.phone.QuickStatusBarHeader.updateVisibilities	701	75304	29
com.android.systemui.qs.tiles.CellularTile.handleUpdateState	664	66869	12
com.android.systemui.statusbar.stack.NotificationStackScrollLayout.getLayoutMinHeight	629	2714	78

methods:

Sample **Android Systrace** trace, where we see that “redraws” happen way too often:



These tools, and a peek at the source code, allow us to understand that **the consumption of resources** is mainly caused by 2 behaviours: - Screen refreshing, which is triggered each time an event occurs, - Movement tracking and animations in response to these movements, which also trigger screen refreshes.

However, not all events need to trigger a refresh of our StatusBar: here's a good first optimization lead to follow. By the way, you may ask how such a behaviour can take place in some code produced by **Google**, which generally applies good developing practices?... We may call this a kind of rebound effect. We've been told to avoid pooling programming and to switch to event programming as a better practice. All right, but it's not magical: you have to control the events. Perhaps some kind of orchestrator in the app could help manage all these events? Otherwise event programming may become the new pooling programming.

Well, some investigations later, our audit led to **an actions plan with 7 main points**: - Simplify the elements layout - Do not refresh the screen for all events, be selective in triggers - Reduce the number of method calls in MotionEvent - Check the impact of the BatteryMeterDrawable object, which seems to over-consume cheerfully - Optimize the screen refresh method - Follow-up with investigating a potential bug - Apply the good practices of Android eco-design in order to refine the source code. We see that with this "big stones" approach, the optimization of the code by applying static rules only comes in the last position in the list. This is an important point, on which we often insist: **a static audit of the source code is good, but if you want to be effective (dare I say "efficient"?), you have to integrate it in a broader approach.**

### **Step 3 : back to the workshed (some refactoring)**

Once the action plan is established, it's usually up to the project team to implement it. We remain in support to guide if needed. In this case, we took part in the refactoring, in a joint task force with the in-house developers.

### **Step 4 : the moment of truth**

This title is misleading. In fact, **there are several moments of truth.**



If we want to know which points of the action plan have really worked, if we want to be able to go back if they have not, then we have to apply them one by one and, above all, check each time that they have had a positive impact.

So we take one point, we correct, we measure. Then we go to point 2, we correct, we measure. Of course, if you're using Git, you may want to work on branches in order to handle this at best. In [GREENSPECTOR](#), we declare a "v2" of the **SystemUI application**. Then we launch a small campaign of measures on the same smartphone. The final step is simply to go to the "Evolution" tab, to compare the consumptions of the functional test cases between v1 and v2:

ShowHideStatusBar_Energy				
Platform CPU	16.71 %	15.51 %	-1.20 %	-7.2 %
Platform Discharge	59.36 mAh	55.55 mAh	-3.81 mAh	-6.41 %
Platform Discharge per second	301.96 µAh/s	273.88 µAh/s	-28.08 µAh/s	-9.3 %
Process CPU	0 %	0 %	0 %	0 %
Process Data	0 B	0 B	0 B	0 %
Network Packets	0	0	0	0 %
Process Memory	132.84 MB	130.49 MB	-2.35 MB	-1.77 %

On the **ShowStatusBar case**, we notice a battery discharge rate (Platform Discharge per second) **9% smaller than before!** On the other functional test cases, we also notice lesser consumptions. But above all, given the very short time of this operation (2 weeks in all), we were only able to apply points 2 and 3 of the action plan! There are **still significant improvements areas to explore**, scented bugs to be confirmed, etcetera.

## 7. System Testing and Implementation

### 7.1 Software Testing

**Software testing** is a process, to evaluate the functionality of a software application with an intent to find whether the developed software met the specified requirements or not and to identify the defects to ensure that the product is defect free in order to produce the quality product.

### 7.2 Software Quality Assurance

Testing is one of the key stages of program development. A high quality end product is not possible without sound quality assurance. All development companies carefully structure their approach to testing when they are establishing a robust and careful production process. The integration of testing every phase of production — from project analysis and planning to final delivery — is particularly important.

The experience of the testing and QA department at Ypsilon IT Solutions is unique in Indore. Our testing process helps improve product quality, reduce costs, build effective relationships within the project team and with the client, and ensure the timely delivery of the end product. Our team has over 10 years' QA experience, including testing mobile platforms and products. Our processes harness the unique experience and global best practice built by our team in the best companies around the world.

### 7.3 Scope of Testing

Everyone has their own style of testing. Some testers just focus on what they see from their eyes and the rest are passionate about everything that works behind the scenes of any mobile application.

If you are an Android/Android Tester, I would suggest you to at least get yourself familiar with some common limitations/ basic functionalities of Android or Android as it always

adds value to our style of testing. I know things are difficult to understand without citing examples.

**Given below are few Examples:**

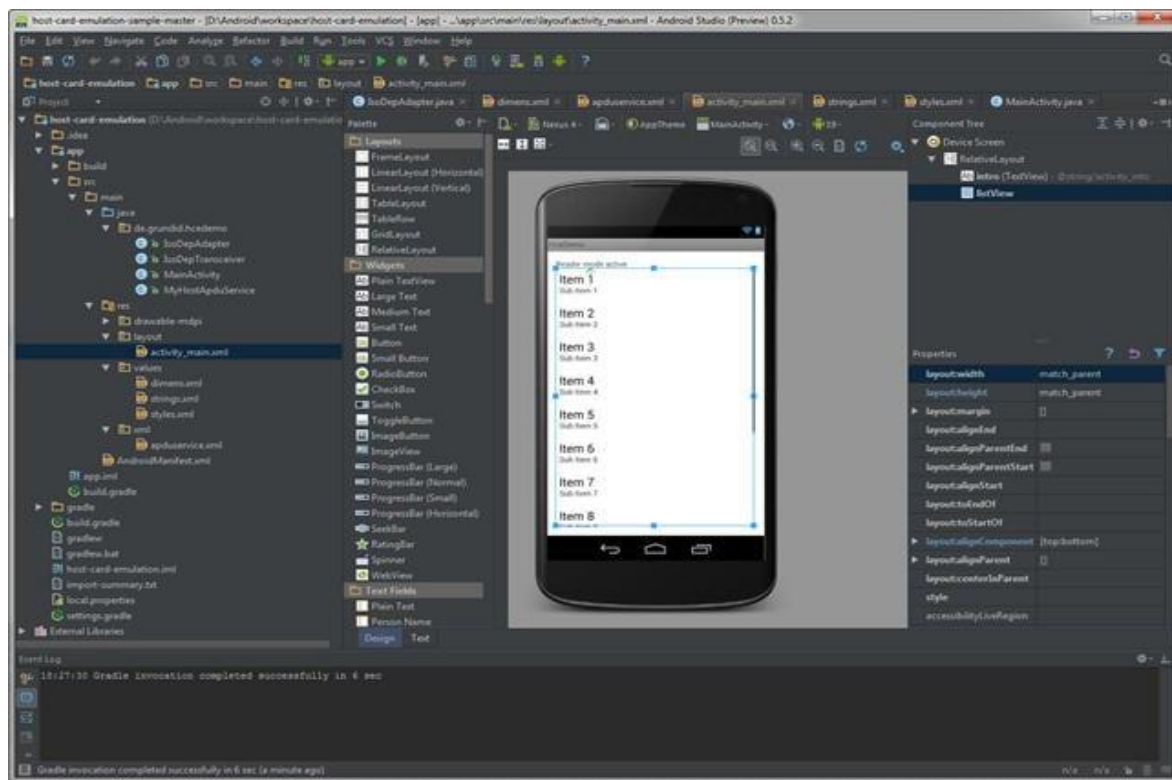
- We cannot change the permissions like camera, storage etc. on app level in Android devices which are below 5.0.1 version.
- For Android below 9.0 version, call kit was not there. Just to brief you in simple words, call kit is used by a calling app and displays full-screen view when a user is getting a call from the calling apps such as WhatsApp, Skype etc. Whereas for Android versions below 9.0 we see those calls as a notification banner.
- Many of you might have come across problems in Paytm where your app is not redirecting you to the payment page of the bank in case you want to add money to your wallet. We think the above is an issue with our bank or Paytm server but it is just that our `AndroidSystemWebView` is not updated. Little knowledge about programming is always helpful for you and to share with your team.
- In simple words, whenever an app is opening any web page in it, then `AndroidSystemWebView` should be updated.

## **7.4 Type of Testing**

Here is the list of the best Android application testing tools:

- a. Experitest
- b. Robotium
- c. MonkeyRunner
- d. Ranorex
- e. Appium
- f. UI Automator

## Robotium Android Testing Tool



Robotium is one the first and frequently utilized automated testing tools for software supported on Android.

Robotium is a free Android UI testing tool. It is suitable for tests automation for different Android versions and sub-versions. Software developers often describe it as Selenium for Android. Tests created by Robotium are written in Java. In fact, Robotium is a library for unit tests.

But it takes much time and efforts to create tests by means of Robotium, as one must work with the program source code in order to automate tests. The tool is also unsuitable for interaction with system software; it cannot lock and unlock a smartphone or a tablet. There is no Record and Play function in Robotium, and it does not provide screenshots.

### 7.4.1 White Box Testing

**WHITE BOX TESTING** (also known as Clear Box Testing, Open Box Testing, Glass Box Testing, Transparent Box Testing, Code-Based Testing or Structural Testing) is a software testing method in which the internal structure/design/implementation of the item being tested is known to the tester. The tester chooses inputs to exercise paths through the code and determines the appropriate outputs. Programming know-how and the implementation knowledge is essential. White box testing is testing beyond the user interface and into the nitty-gritty of a system.

This method is named so because the software program, in the eyes of the tester, is like a white/transparent box; inside which one clearly sees.

#### Definition by ISTQB

- **white-box testing:** Testing based on an analysis of the internal structure of the component or system.
- **white-box test design technique:** Procedure to derive and/or select test cases based on an analysis of the internal structure of a component or system.

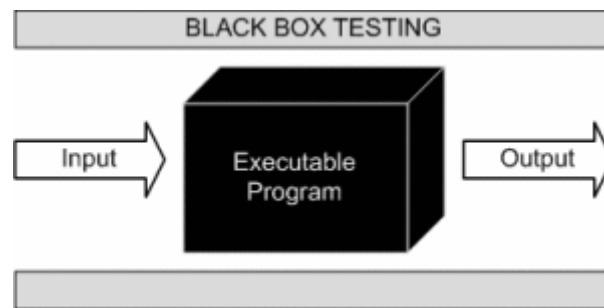
#### **Example**

A tester, usually a developer as well, studies the implementation code of a certain field on a webpage, determines all legal (valid and invalid) AND illegal inputs and verifies the outputs against the expected outcomes, which is also determined by studying the implementation code.

White Box Testing is like the work of a mechanic who examines the engine to see why the car is not moving.

### 7.4.2 Black Box Testing

**BLACK BOX TESTING**, also known as Behavioral Testing, is a software testing method in which the internal structure/design/implementation of the item being tested is not known to the tester. These tests can be functional or non-functional, though usually functional.



This method is named so because the software program, in the eyes of the tester, is like a black box; inside which one cannot see. This method attempts to find errors in the following categories:

- Incorrect or missing functions
- Interface errors
- Errors in data structures or external database access
- Behavior or performance errors
- Initialization and termination errors

#### Definition by ISTQB

- **black box testing:** Testing, either functional or non-functional, without reference to the internal structure of the component or system.
- **black box test design technique:** Procedure to derive and/or select test cases based on an analysis of the specification, either functional or non-functional, of a component or system without reference to its internal structure.

## Example

A tester, without knowledge of the internal structures of a website, tests the web pages by using a browser; providing inputs (clicks, keystrokes) and verifying the outputs against the expected outcome.

### 7.4.3 Regression Testing

Regression Testing is defined as a type of software testing to confirm that a recent program or code change has not adversely affected existing features.

Regression Testing is nothing but a full or partial selection of already executed test cases which are re-executed to ensure existing functionalities work fine.

This testing is done to make sure that new code changes should not have side effects on the existing functionalities. It ensures that the old code still works once the new code changes are done.

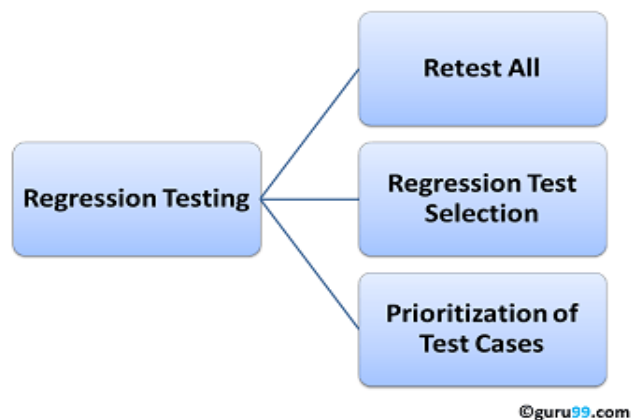
### Need of Regression Testing

Regression Testing is required when there is a

- Change in requirements and code is modified according to the requirement
- New feature is added to the software
- Defect fixing
- Performance issue fix

## How to do Regression Testing

Software maintenance is an activity which includes enhancements, error corrections, optimization and deletion of existing features. These modifications may cause the system to work incorrectly. Therefore, Regression Testing becomes necessary. Regression Testing can be carried out using the following techniques:



## 7.5 Test Strategy

Agile test strategy supports DevOps and continuous testing. And continuous testing is important to improving product quality.

In Agile development, testing needs to happen early and often. So, instead of waiting for development to be finished before testing begins, testing happens continuously as features are added.

Tests are prioritized just like user stories. Testers aim to get through as many tests as they can in an iteration. Adding automated testing tools can help testers get through more of the testing backlog.



QA is everyone's responsibility in Agile. So, Agile testers and developers need to work closely together. Communication and collaboration are key. Agile development is often driven by tests. Developers use Agile testing methods like TDD (test-driven development) to write the test first. Then they write the code that will be verified by the test. And developers and Agile testers should collaborate before user stories (e.g., requirements) are set.

## 7.6 Test Planning

Testing is conducted to detect issues related to:

- memory consumption
- power utilization
- network connectivity
- operating in the background
- switching between applications
- memory leakage.

### Interrupt Testing

As far as mobile devices have a huge range of functions, the work of the application may be interrupted by various reasons, e.g., an upcoming call, message, other apps notifications, mail, low memory warning, inserting a cable, etc.

### Usability Testing

Usability testing is applied to check whether the application is easy to use and understand from the user's point of view.

### Installation and Launch testing

During installation testing, an engineer checks whether there are any issues during the installation, uninstallation, and updating of the application. Once the application has

been installed, an engineer checks launching process. The application must be loaded quickly and correctly. Closing the application should not require much time as well.

### **Functional Testing**

All the functions and features of the application are tested to verify whether they operate according to the specification.

### **Security testing**

Security testing is conducted to find the application vulnerabilities and prevent private data losses.

### **Regression testing**

Regression testing is a re-execution of tests that had been done before the code changes. Its purpose is to verify whether a new functionality has affected the existing one.

### **Pass/Fail Conditions**

All the conditions when tests pass or fail are defined and described.

### **Test Report**

Test Report helps to summarize testing activity in a formal way. It should contain:

- application name and overview
- testing hardware and software environment
- the number of tests cases executed/passed/failed. For each issue that has been encountered, the following information is provided:
  - bug description
  - bug status (open, fixed, etc.)
  - bug location
  - steps to reproduce an issue.

## 8. Tools and Technology Used

### Hardware Requirements

Device	: Android Smart Phone
RAM	: 512MB or Above
Chipset	: Qualcomm MSM8226 Snapdragon 400 or Above
Internal Space	: 1GB or Above
Processor	: Quad - core or above

### Software Requirements

Operating System	: Android OS
Database	: MySQL 5.0
IDE/Tools	: Android Studio /Net Beans /Java Development Kit 8.0
Web Server	: XAMPP Server / Apache Tomcat

## **Back End and Front End**

### **8.1 Front End:**

- Android Studio
- Net Beans

### **8.2 Back End:**

- XAMPP Server
- Apache Tomcat
- MySQL Server

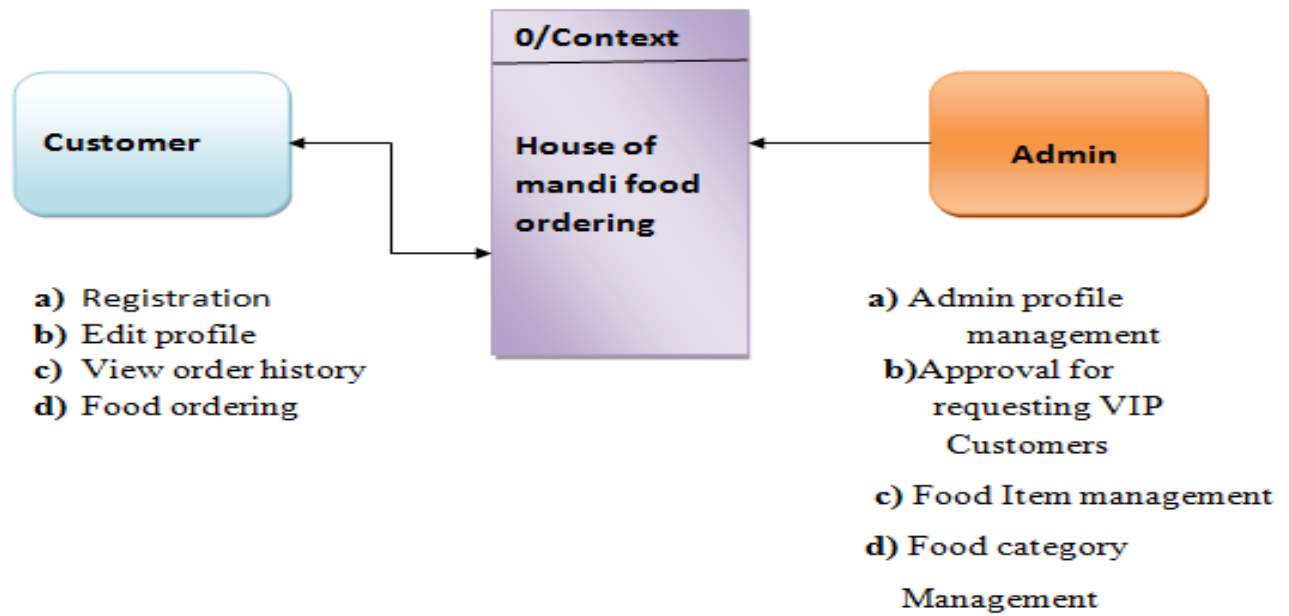
## 9. Data Flow Diagram:

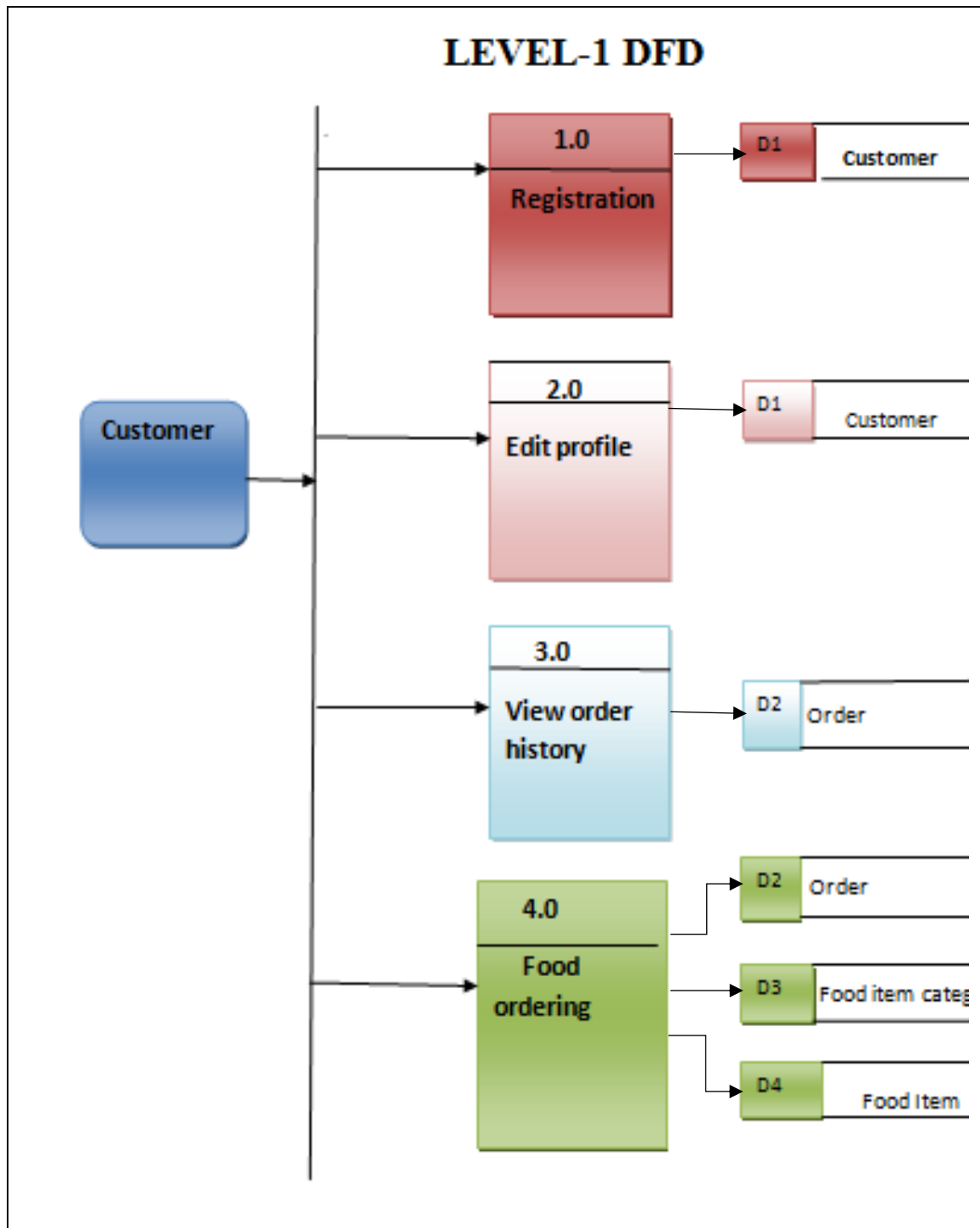
Data Flow Diagramming is a means of representing a system at any level of detail with a graphic network of symbols showing data flows, data stores, data processes, and data sources/destination. Smart Draw contains all the needed data flow diagram symbols and easy-to-use templates that help you get started. Stamp shapes to your drawing area and connect them easily with keyboard shortcuts or intuitive commands located on the Smart Panel to the left of your drawing area. You can even nest different levels of data flow diagrams by using Smart Draw's hyperlink function. A data flow diagram (DFD) is a significant modeling technique for analyzing and constructing information processes. DFD literally means an illustration that explains the course or movement of information in a process. DFD illustrates this flow of information in a process based on the inputs and outputs. A DFD can be referred to as a Process Model.

A designer usually draws a context-level DFD showing the relationship between the entities inside and outside of a system as one single step. This basic DFD can be then disintegrated to a lower level diagram demonstrating smaller steps exhibiting details of the system that is being modeled.

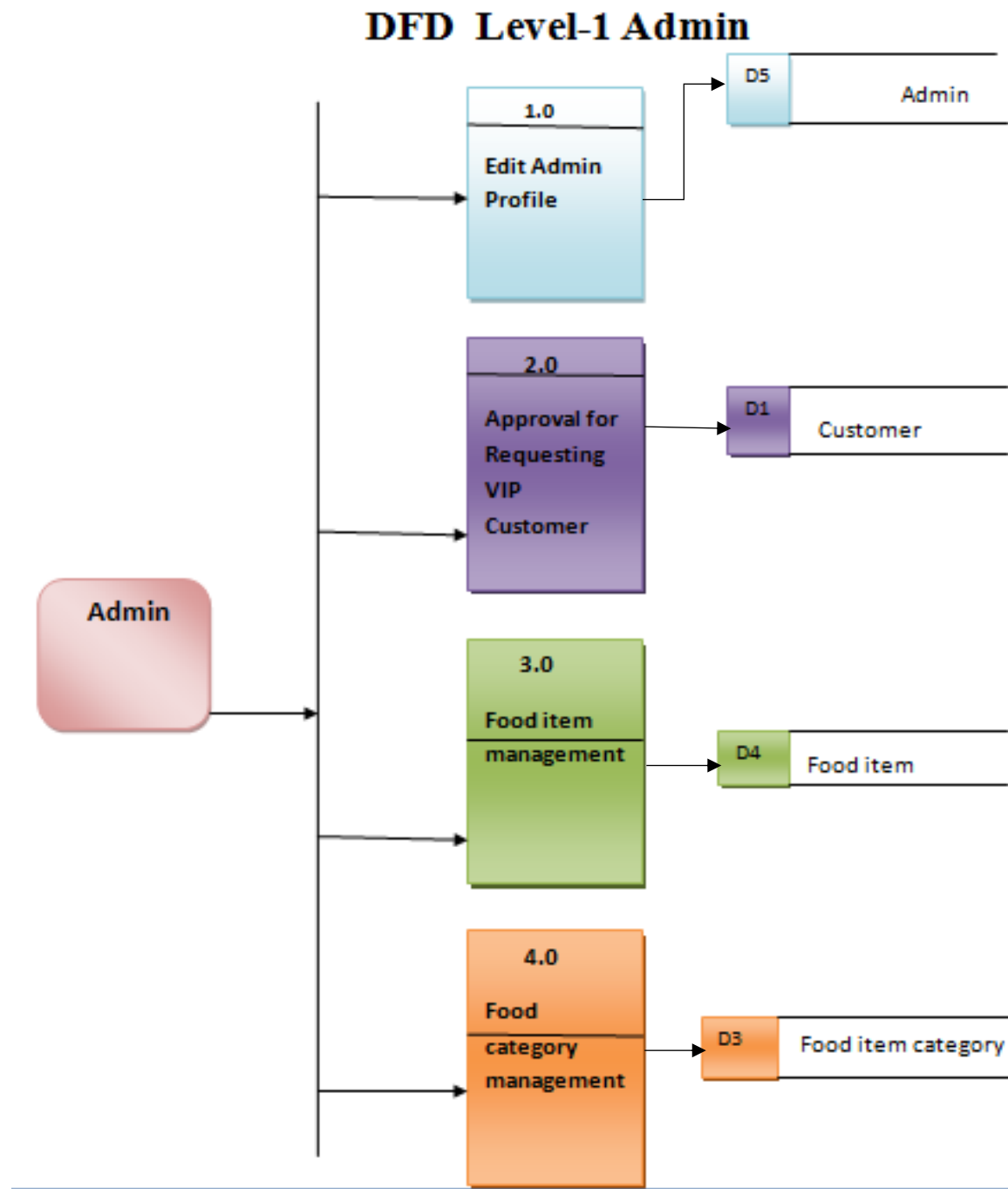
## Context -level -DFD:

## CONTEXT LEVEL DFD

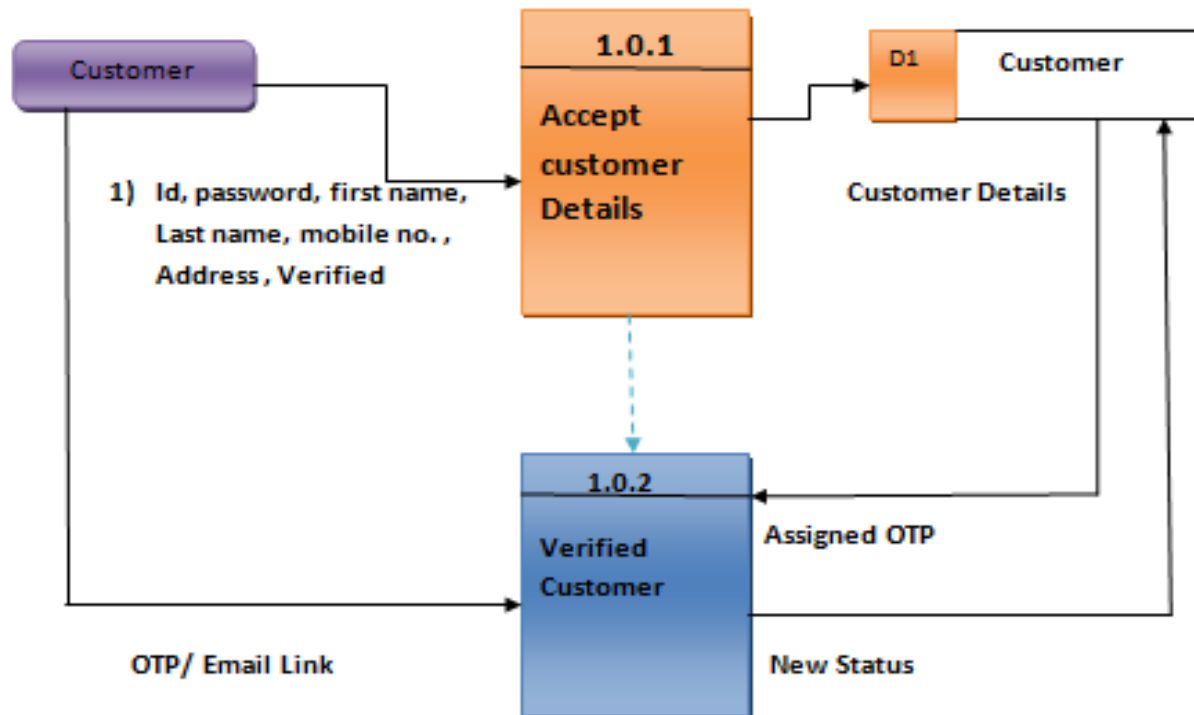


**First level DFD (User Management):**

## First level DFD (Admin Management):

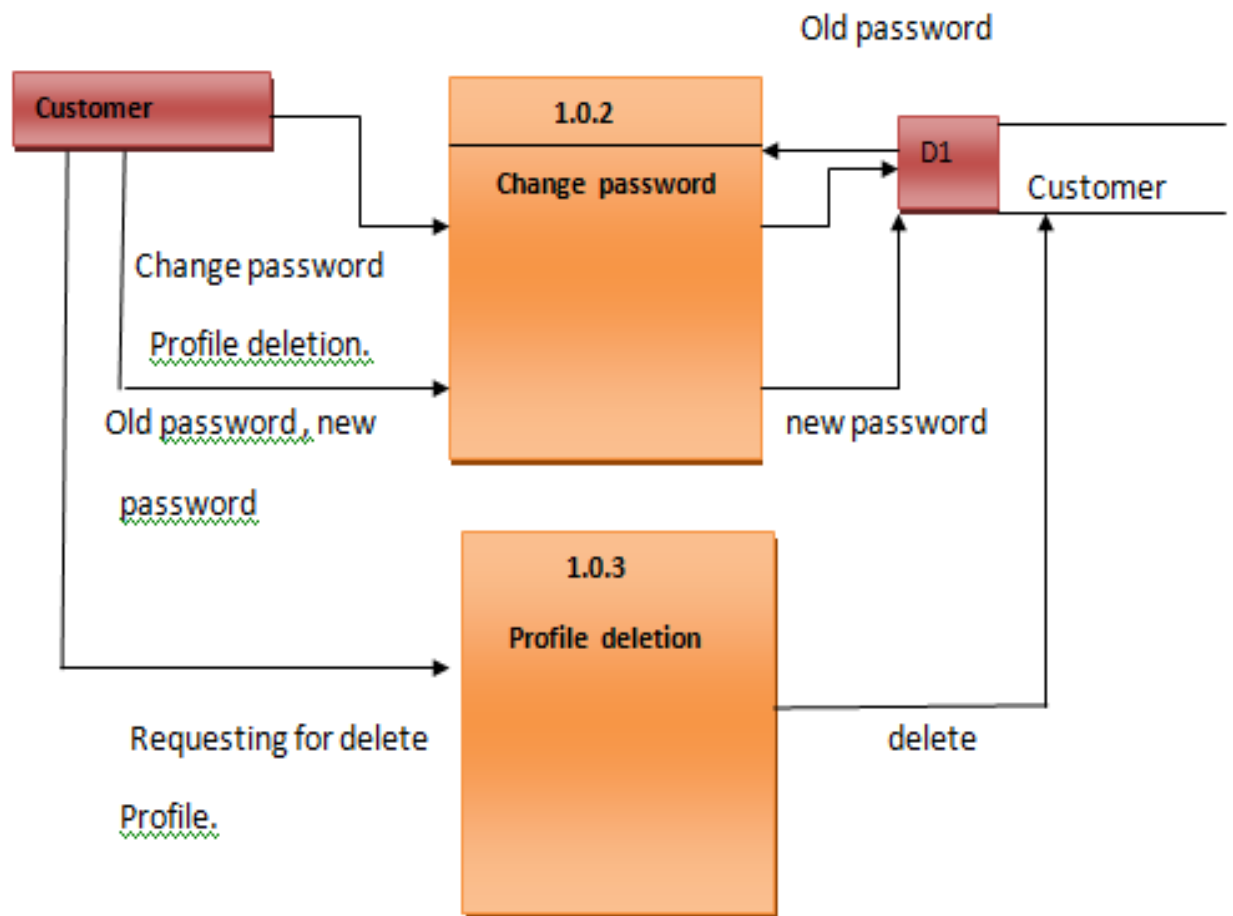




**Second level DFD (Customer Management):****1>Customer registration**

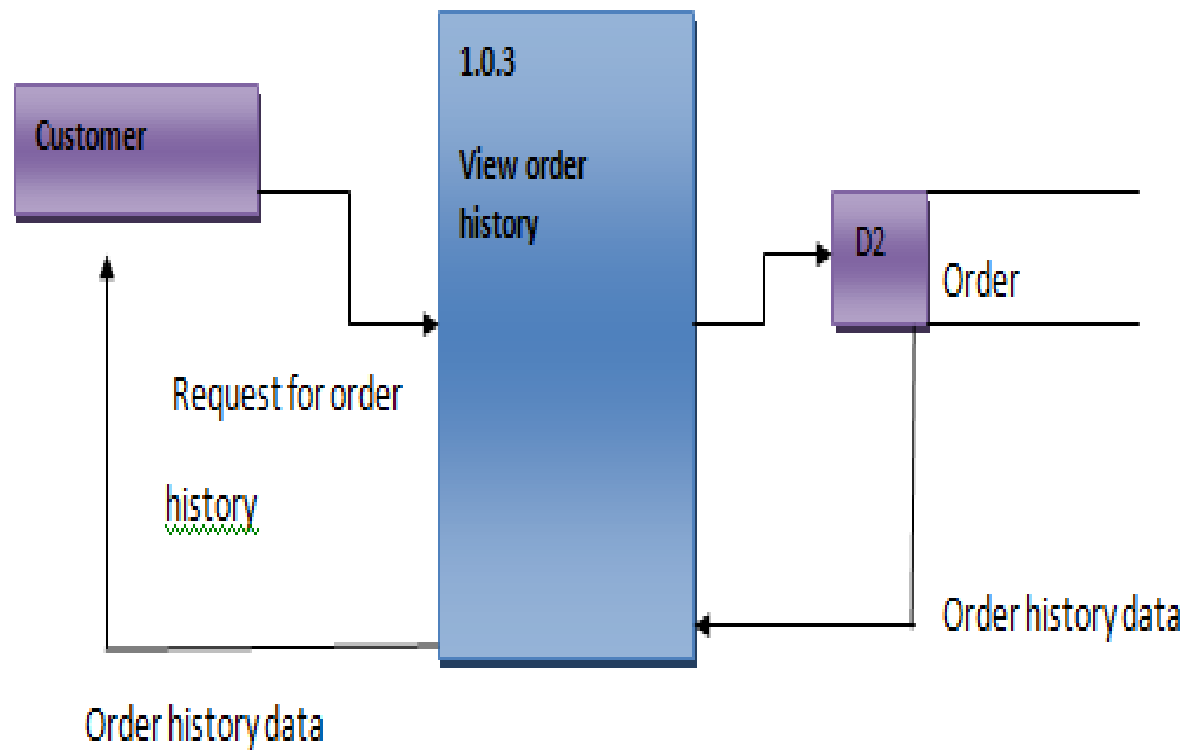
## Second level DFD (Customer Edit Profile):

## 2&gt; Edit profile:-



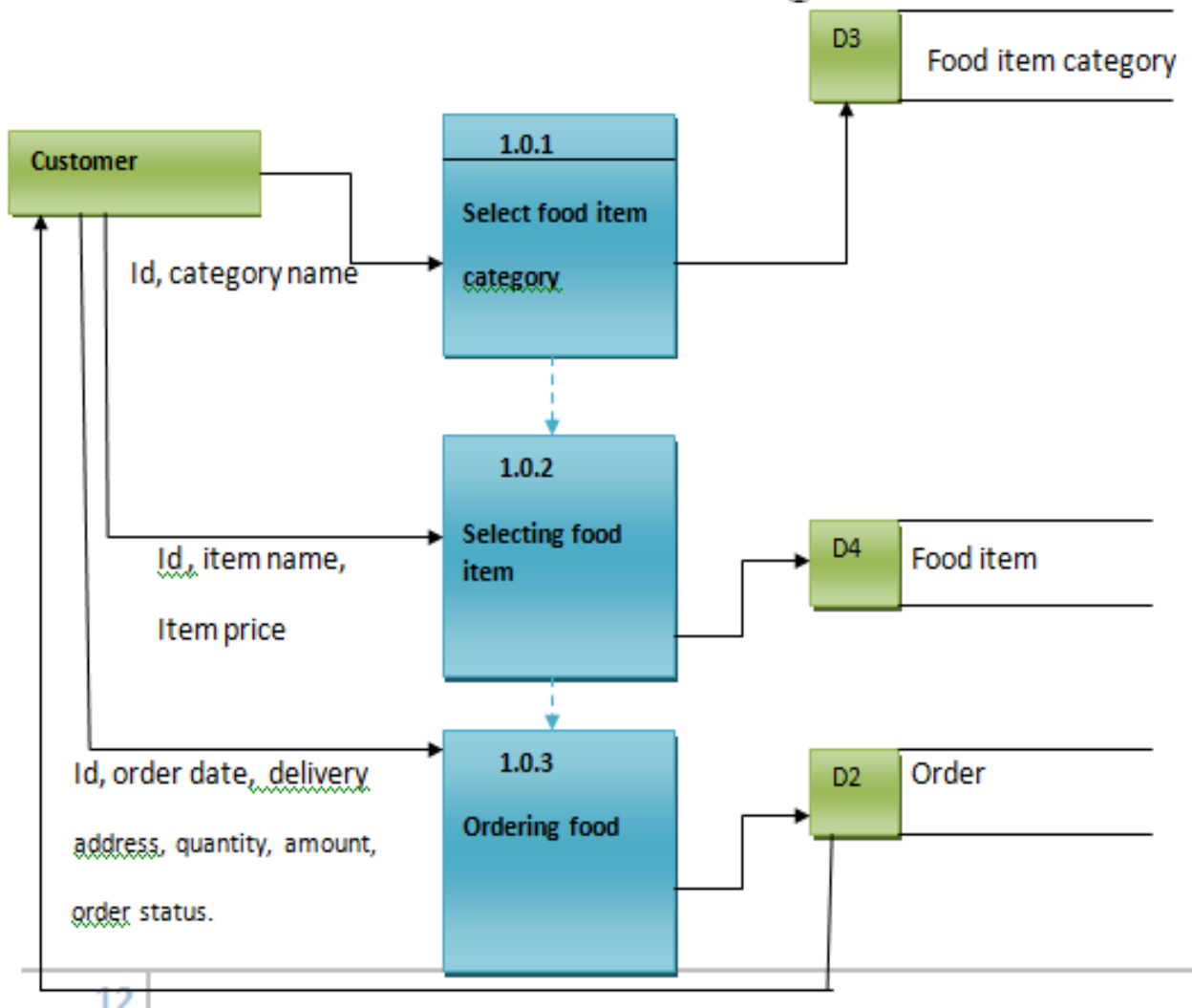
Second level DFD (View Order History):

### 3>view order history request



Second level DFD(Customer Food Ordering):

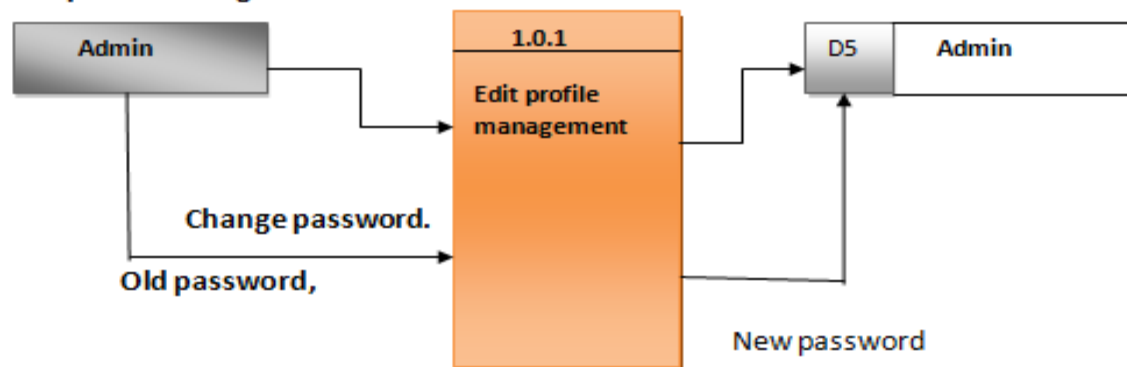
## 4>food ordering



Second level DFD (Admin Management):

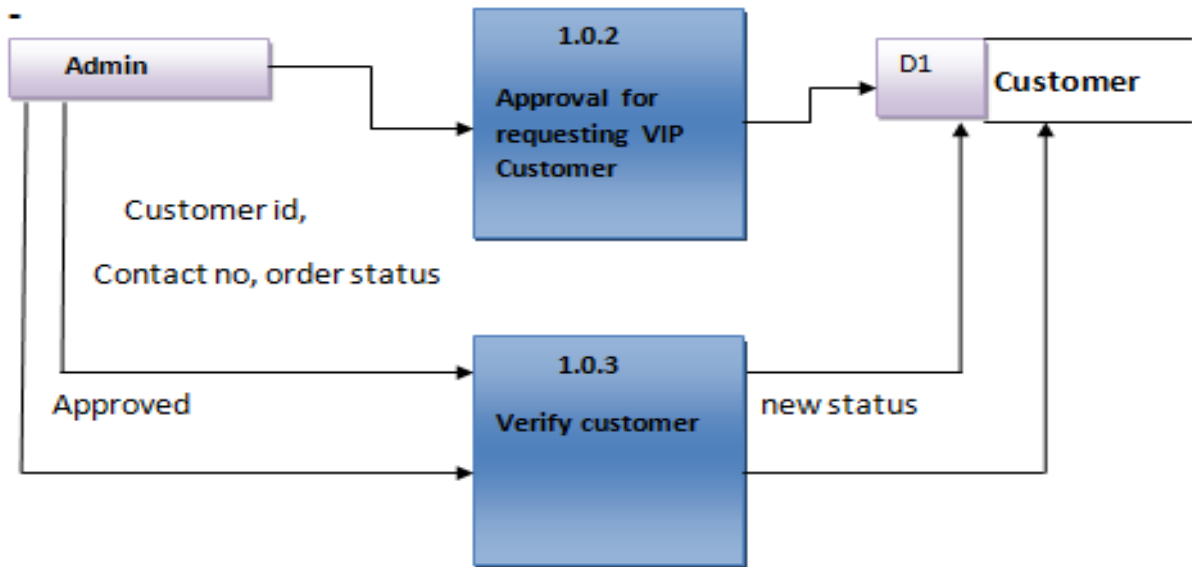
## Admin level -2 DFD

1>Edit profile management:-



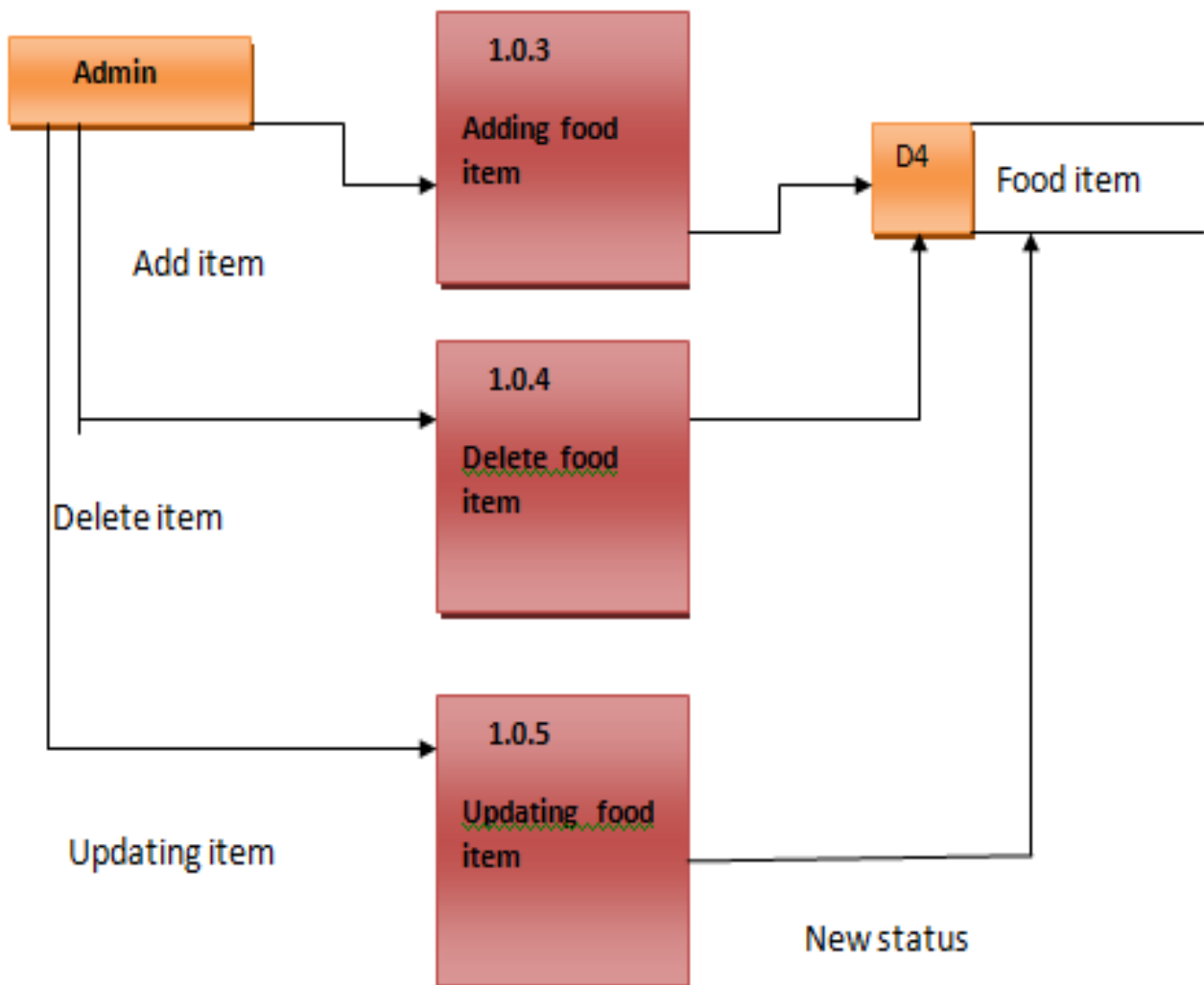
Second level DFD (Approved Requesting VIP Customer):

## 2> Approval requesting VIP customer



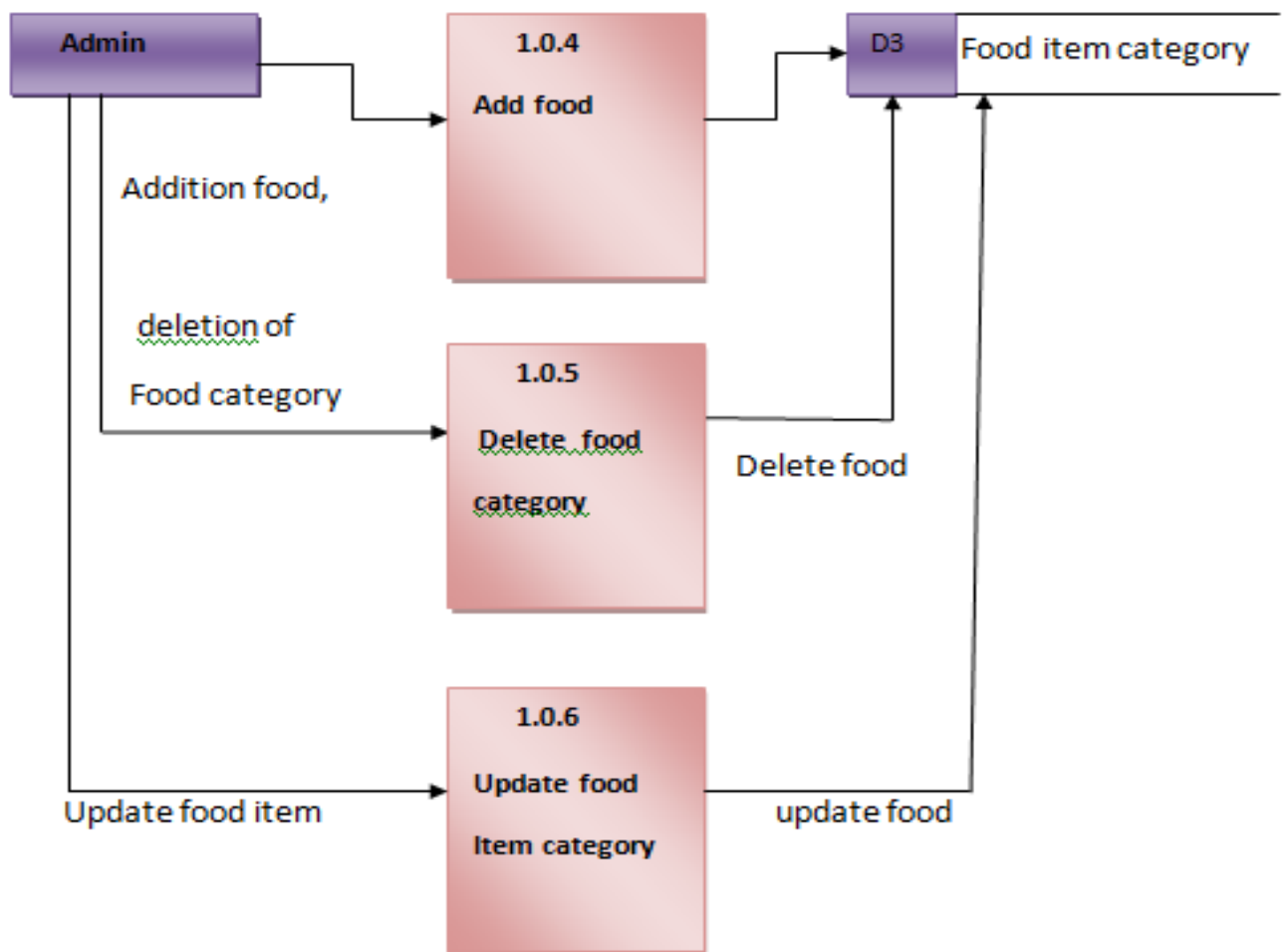
Second level DFD (Food Item Management):

### 3>Admin food item management:-



Second level DFD (Food Category Management):

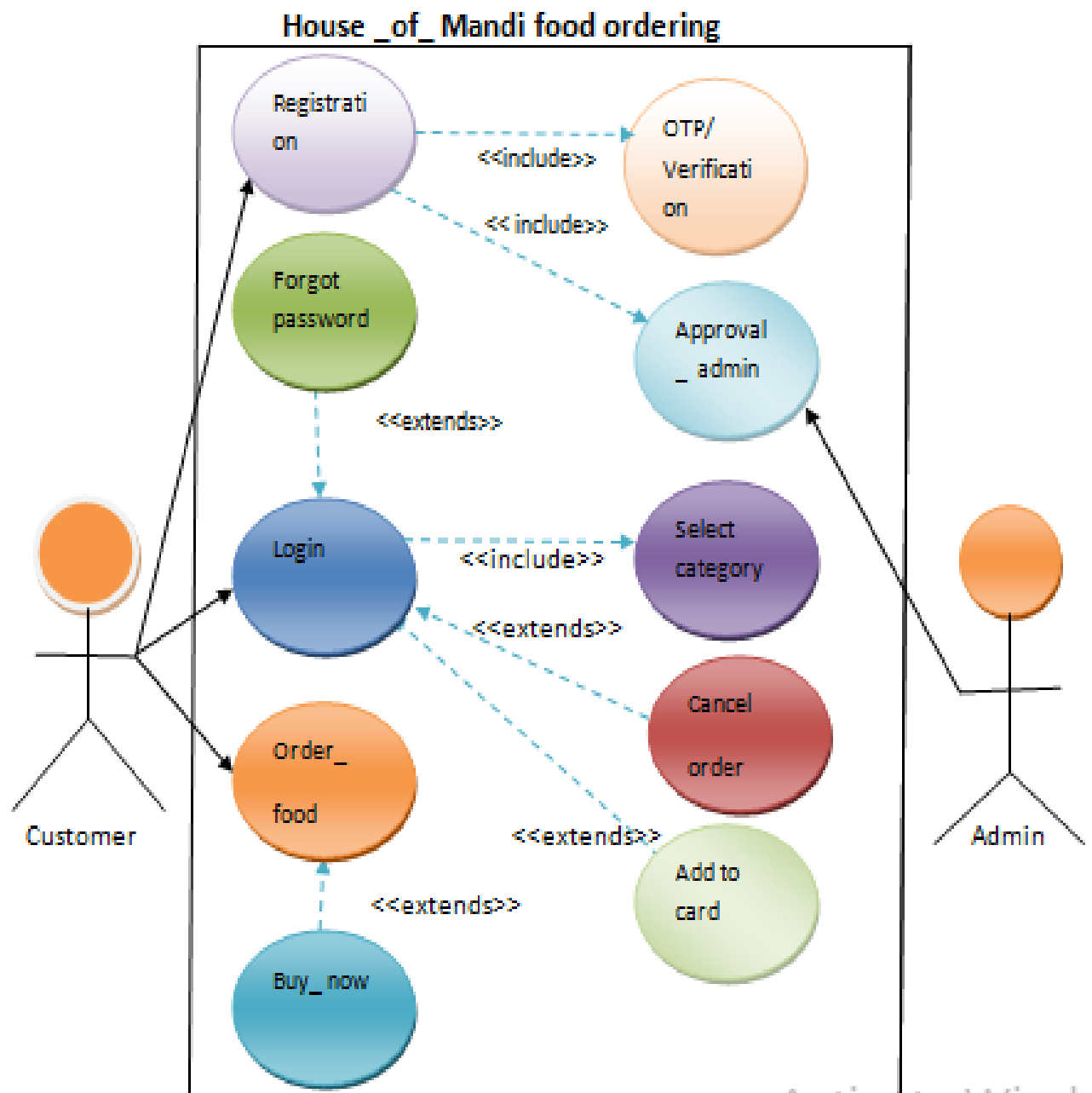
#### 4>Admin food category management:-



Use-case Diagram (Customer):



## Use case diagram



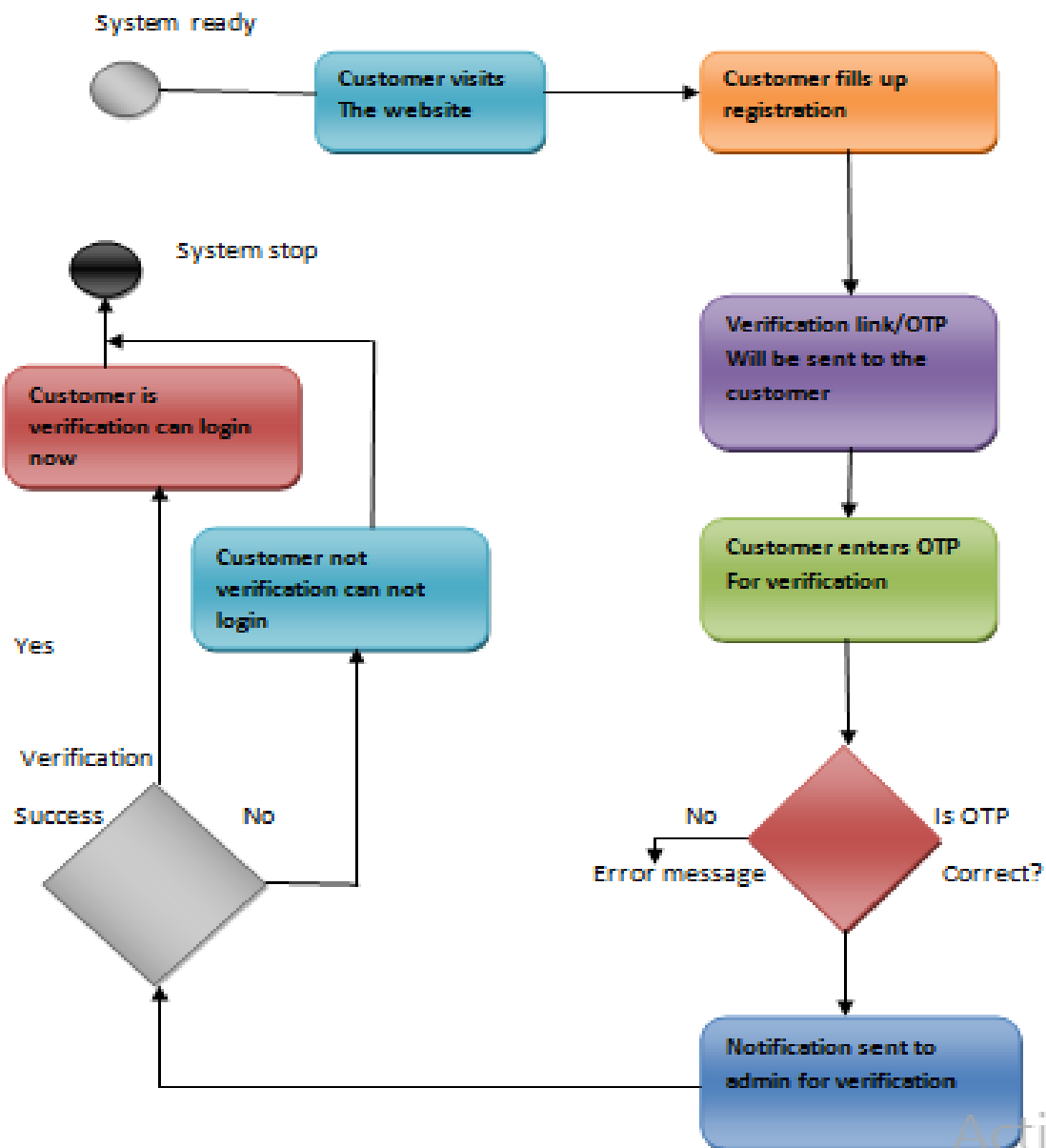
## Use-case Diagram (Admin):

## Use case diagram admin

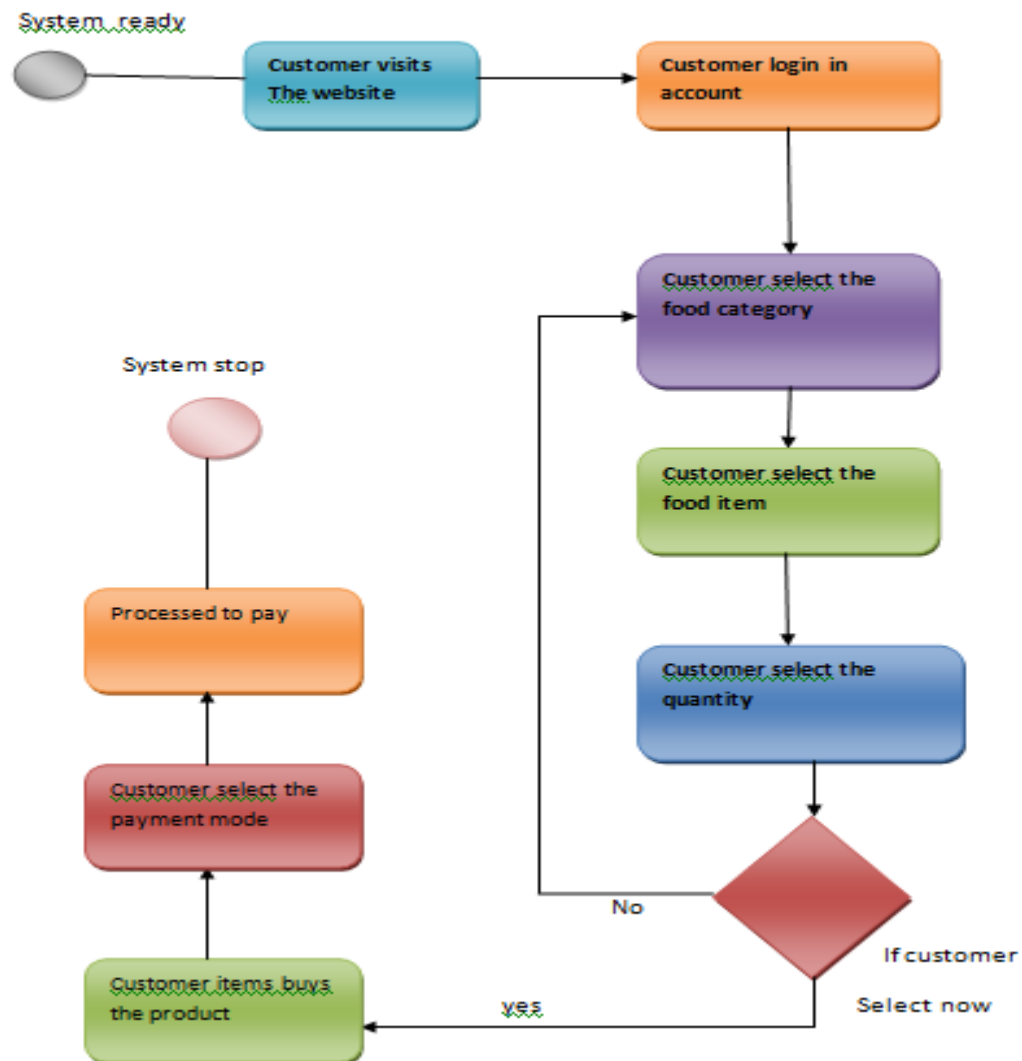


## Activity diagram (flow chart)

## Activity diagram(flow chart)



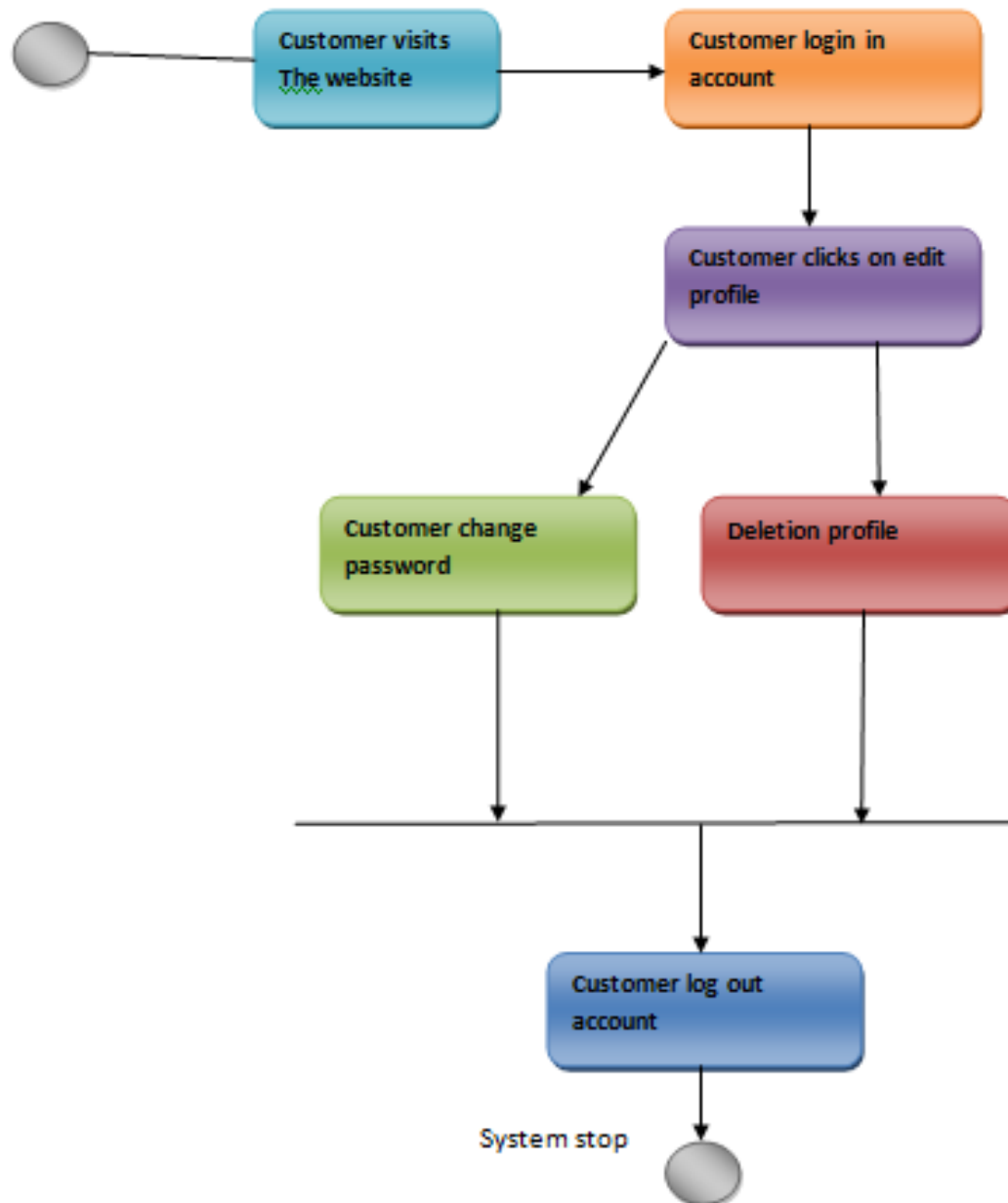
## Order Food:

**2> order food**

## Edit Profile:

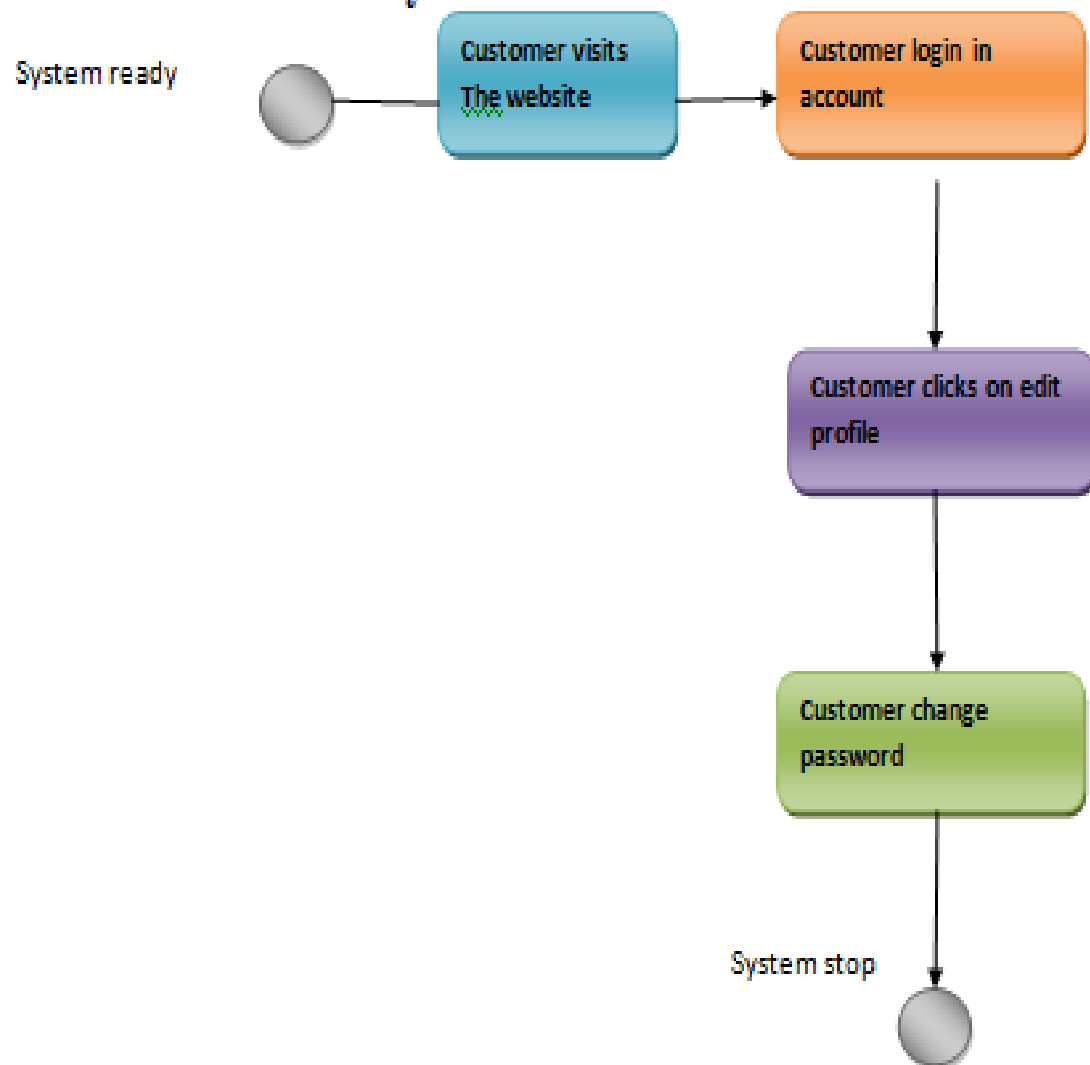
## 3&gt; Edit profile

System ready



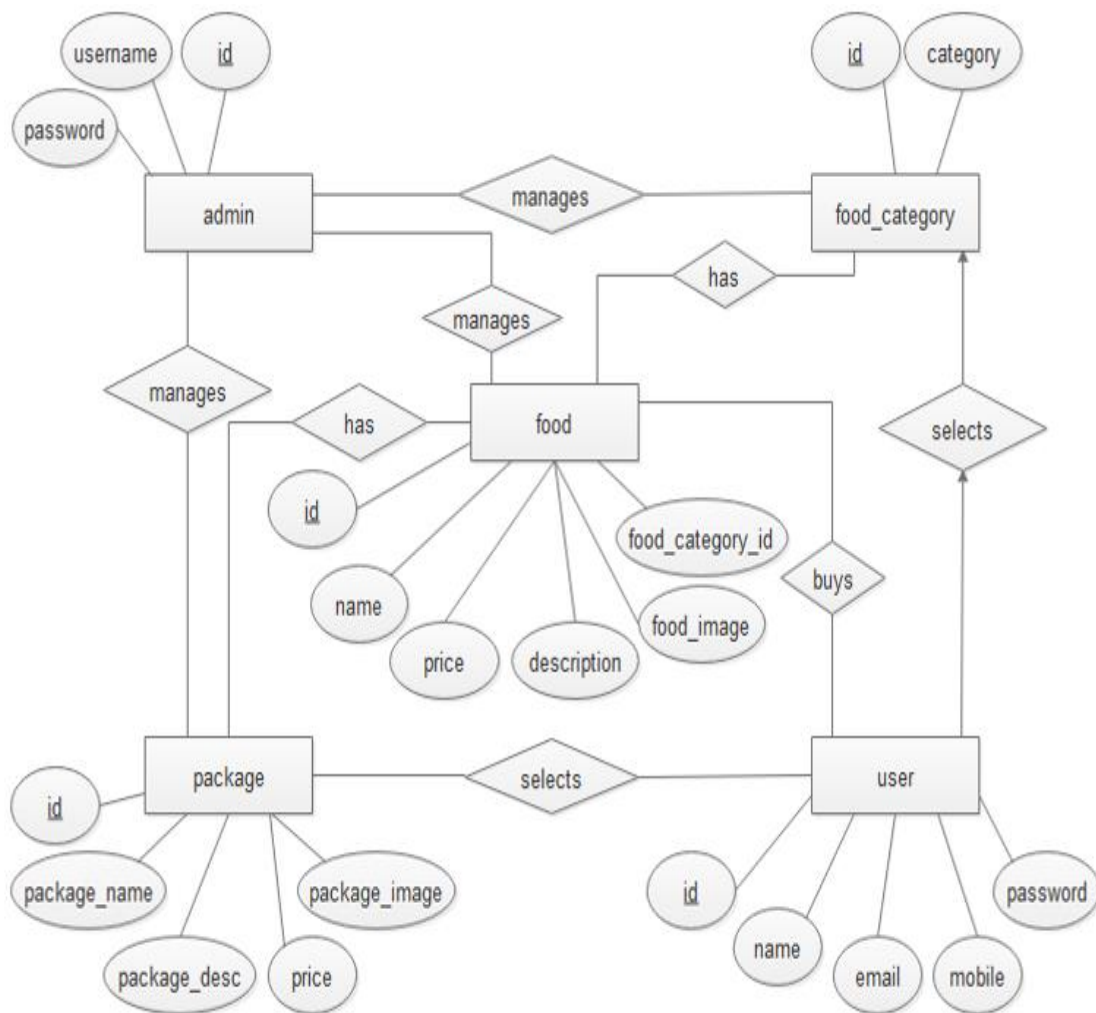
View order food:

#### 4>view order history

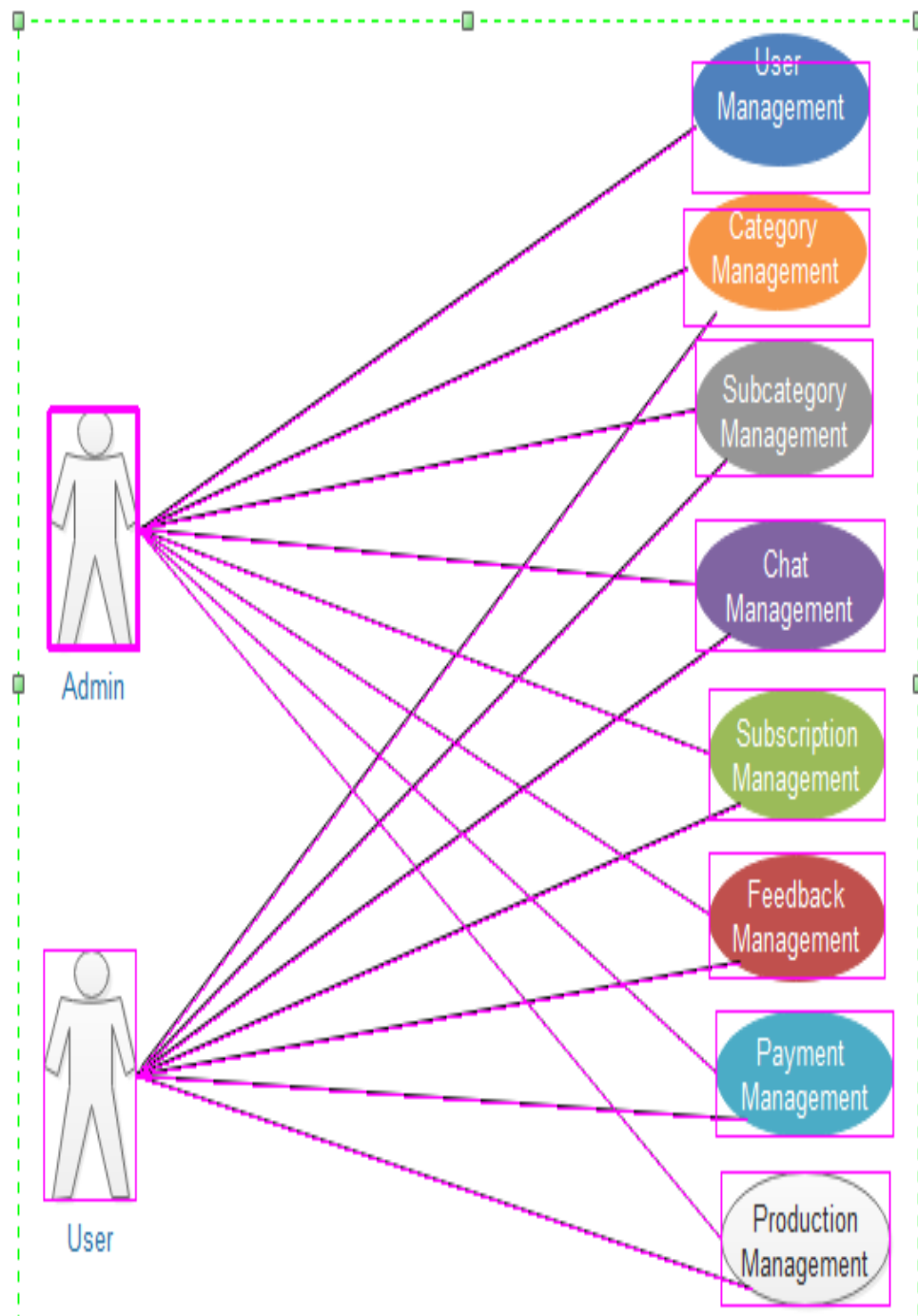


## 10. E-R Diagram:

An entity relationship diagram (ERD) shows the relationships of entity sets stored in a database. An entity in this context is an object, a component of data. An entity set is a collection of similar entities. These entities can have attributes that define its properties. By defining the entities, their attributes, and showing the relationships between them, an ER diagram illustrates the logical structure of databases. ER diagrams are used to sketch out the design of a database.



### 3.5 UML Diagrams





## 11. Tables Description

Database Normalization is a technique of organizing the data in the database. Normalization is a systematic approach of decomposing tables to eliminate data redundancy (repetition) and undesirable characteristics like Insertion, Update and Deletion Anomalies. Normalization is a process of organizing the data in database to avoid data redundancy, insertion anomaly, update anomaly & deletion anomaly.

### DATA DICTIONARY:

#### 1) Admin table:

Column	Column Type	Constraints
*Id	Varchar(40)	Unique (Primary Key)
Password	Varchar(20)	Unique

#### 2) Customer table:

Column	Column Type	Constraints
*Id	Varchar(40)	Unique (Primary Key)
Password	Varchar(20)	UNIQUE , NOT_NULL
FirstName	Varchar(40)	Not Null
last Name	Varchar(40)	Not Null
mobileNumber	Varchar(10)	UNIQUE , NOT_NULL
Verified	Boolean	
approved	Boolean	

**3) Food Item Category table:**

Column	Column Type	Constraints
*Id	Varchar(40)	AUTO_INCREMENT(Primary Key)
categoryName	Varchar(40)	UNIQUE , NOT_NULL

**4) FoodItem table:**

Column	Column Type	Constraints
*Id	Varchar(40)	AUTO_INCREMENT(Primary Key)
itemName	Varchar(40)	UNIQUE , NOT_NULL
itemPrice	int(4)	NOT_NULL
img_path	varchar(200)	UNIQUE , NOT_NULL
categoryID	Int(10)	UNIQUE , NOT_NULL(Foreign key references id Fooditem category table)

**5) Order table:**

Column	Column Type	Constraints
*Id	Varchar(40)	AUTO_INCREMENT(Primary Key)
orderDate	timestamp	
deliveryAddress	varchar(200)	NOT_NULL
Quantity	Int	NOT_NULL
Amount	Int	NOT_NULL
orderDesc	varchar(200)	NOT_NULL
orderStatus	varchar(20)	NOT_NULL
paymentMode	varchar(40)	NOT_NULL
customerID	varchar(40)	

**Sample codes:-****(Android)****SplashActivity.java**:-

```

package com.example.justeat;

import android.content.Intent;
import android.os.Handler;
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;

public class SplashActivity extends AppCompatActivity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_splash);

        new Handler().postDelayed(new Runnable() {

            @Override
            public void run() {
                // This method will be executed once the timer is over
                Intent i = new Intent(SplashActivity.this, PhasingActivity.class);
                startActivity(i);
                finish();
            }
        }, 3000);
    }
}

```

**PhasingActivity.java**

```

package com.example.justeat;

import android.content.Intent;
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;

public class PhasingActivity extends AppCompatActivity
{
    Button btnSignUp,btnSignIn;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_phasing);

        btnSignIn = findViewById(R.id.btnSignIn);
        btnSignIn.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                Intent it = new Intent(PhasingActivity.this,MainActivity.class);
                startActivity(it);
            }
        });

        btnSignUp = findViewById(R.id.btnSignUp);
        btnSignUp.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                Intent it = new Intent(PhasingActivity.this,SignupActivity.class);
                startActivity(it);
            }
        });
    }
}

```

**SignupActivity.java**

```

package com.example.justeat;

import android.app.Dialog;
import android.content.Intent;
import android.graphics.Typeface;
import android.support.v7.app.AppCompatActivity;

import android.os.Bundle;
import android.view.View;
import android.widget.Button;
import android.widget.EditText;

import android.widget.Toast;
import android.app.ProgressDialog;

import com.android.volley.DefaultRetryPolicy;
import com.android.volley.Request;
import com.android.volley.Response;
import com.android.volley.VolleyError;
import com.android.volley.toolbox.JsonObjectRequest;
import com.android.volley.toolbox.StringRequest;
import com.google.gson.Gson;
import org.json.JSONObject;
import java.util.HashMap;

public class SignupActivity extends AppCompatActivity {
    EditText fname,lname,passwordd,mobphone,mail,usraddr,cnfpasswordd;
    Button signup;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_signup);

        fname =(EditText) findViewById(R.id.edFname);
        lname =(EditText) findViewById(R.id.edLname);
        usraddr = (EditText) findViewById(R.id.usraddr);
        passwordd = (EditText) findViewById(R.id.passwrdd);
        cnfpasswordd = (EditText) findViewById(R.id.confirm_passwrdd);
        mail = (EditText) findViewById(R.id.mail);
        mobphone = (EditText) findViewById(R.id.mobphone);
        signup = (Button) findViewById(R.id.sup);
    }
}

```

```

Typeface custom_font =
Typeface.createFromAsset(getAssets(),"fonts/cantarell_regular.ttf");
signup.setTypeface(custom_font);
fname.setTypeface(custom_font);
lname.setTypeface(custom_font);
mail.setTypeface(custom_font);
mobphone.setTypeface(custom_font);
passwordd.setTypeface(custom_font);
cnfpasswordd.setTypeface(custom_font);
usraddr.setTypeface(custom_font);

signup.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        final ProgressDialog pd = new ProgressDialog(SignupActivity.this);
        pd.setTitle("CONNECTING TO SERVER");
        pd.setMessage("Registering User Details");
        pd.setProgressStyle(ProgressDialog.STYLE_SPINNER);
        pd.show();

        String myurl =ServerAddress.MYSERVER+"/RegisterCustomer.jsp";

        String tag_string_req = "myrequest_2";
        Gson g = new Gson();
        HashMap<String,String>hmap=
            new HashMap<>();
        hmap.put("firstName",fname.getText().toString());
        hmap.put("lastName",lname.getText().toString());
        hmap.put("mobileNumber",mobphone.getText().toString());
        hmap.put("address",usraddr.getText().toString());
        hmap.put("PASS",passwordd.getText().toString());
        hmap.put("ID",mail.getText().toString());

        try {
            JSONObject job = new JSONObject(g.toJson(hmap).trim());
//            Toast.makeText(SignupActivity.this,
//            "json is"+job.toString(),Toast.LENGTH_LONG).show();
            JSONObjectRequest request =
                new JSONObjectRequest(Request.Method.POST,
                    myurl, job, new Response.Listener<JSONObject>() {
                        @Override
                        public void onResponse(final JSONObject response) {
                            try {
                                pd.dismiss();
                                String str =

```

```

        response.getString("cid");
    if (str.equals("failure")) {

        Toast.makeText(SignupActivity.this, "Customer Not
Registered !!!",
            Toast.LENGTH_SHORT).show();
    }
    else {
        Toast.makeText(SignupActivity.this,
            "Please enter OTP sent on your email-id !!!",
            Toast.LENGTH_SHORT).show();

        final Dialog d = new Dialog(SignupActivity.this);

        d setContentView(R.layout.dialogue_confirm);

        Button btn = d.findViewById(R.id.btnSubmit);

        final EditText edtOTP = d.findViewById(R.id.editTextOtp);

        btn.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {

                try {
                    String custid = response.getString("cid");

                    String otp = edtOTP.getText().toString();

                    StringRequest srq = new
StringRequest(Request.Method.GET,
ServerAddress.MYSERVER + "/VerifyCustomer.jsp?ID="+custid+"&genpass="+otp,
                    new Response.Listener<String>() {
                        @Override
                        public void onResponse(String response) {

                            try {
                                JSONObject obj = new
JSONObject(response.trim());

                                String msg = obj.getString("msg");

                                if (msg.equals("verified"))
                                {
                                    Toast.makeText(SignupActivity.this,

```

```

                                "Verification Successful",
Toast.LENGTH_SHORT).show();

                                d.dismiss();
                                Intent in = new Intent(SignupActivity.this,
MainActivity.class);

                                startActivity(in);
                                finish();
                                }
                                else
                                {
                                    Toast.makeText(SignupActivity.this,
"Invalid OTP entered !!!", Toast.LENGTH_SHORT).show();
                                }
                                }
                                catch (Exception ex)
                                {
                                    ex.printStackTrace();
                                }
                                }
                                },
                                new Response.ErrorListener() {
                                    @Override
                                    public void onErrorResponse(VolleyError error)
{

                                        d.dismiss();
                                        Toast.makeText(SignupActivity.this
                                        , "Volley Error "+error.getMessage(),
                                        Toast.LENGTH_SHORT).show();
                                    }
                                });

                                srq.setRetryPolicy(new DefaultRetryPolicy(
                                0,
                                DefaultRetryPolicy.DEFAULT_MAX_RETRIES,
                                DefaultRetryPolicy.DEFAULT_BACKOFF_MULT));

                                ApplicationController.getInstance().addToRequestQueue(srq,"OTP request");

                                }
                                catch (Exception ex)
                                {
                                    {
                                        ex.printStackTrace();
                                    }
                                }
                                }

```



```

        });

        d.show();
    }
} catch (Exception ex) {
    ex.printStackTrace();
}

}

},

new Response.ErrorListener() {
    @Override
    public void onErrorResponse(VolleyError error) {
        pd.dismiss();
        Toast.makeText(SignupActivity.this
            , "Volley Error "+error.getMessage(),
            Toast.LENGTH_SHORT).show();
    }
});

request.setRetryPolicy(new DefaultRetryPolicy(
    0,
    DefaultRetryPolicy.DEFAULT_MAX_RETRIES,
    DefaultRetryPolicy.DEFAULT_BACKOFF_MULT));

AppController.getInstance().addToRequestQueue(request,tag_string_req);

}
catch (Exception ex){
    ex.printStackTrace();
}
}
});

}
}

```

### MainActivity.java

```
package com.example.justeat;

import android.app.ProgressDialog;
import android.content.Intent;
import android.graphics.Paint;
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;
import android.widget.EditText;
import android.widget.RadioButton;
import android.widget.RadioGroup;
import android.widget.TextView;
import android.widget.Toast;

import com.android.volley.Request;
import com.android.volley.Response;
import com.android.volley.VolleyError;
import com.android.volley.toolbox.JsonObjectRequest;
import com.google.gson.Gson;

import org.json.JSONObject;

import java.util.HashMap;

public class MainActivity extends AppCompatActivity {

    Button btnLogin;
    TextView btnSignup, frgtpass;
    EditText usr,pswd;
    RadioGroup rgp;
```

String **choice**;

@Override

```

protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);

    rgp = findViewById(R.id.rgp);
    usr = findViewById(R.id.edtName);
    pswd = findViewById(R.id.edtPass);
    btnLogin = findViewById(R.id.btnLogin);
    frgtpass = findViewById(R.id.forgot_password);
    frgtpass.setPaintFlags(frgtpass.getPaintFlags() | Paint.UNDERLINE_TEXT_FLAG);
    frgtpass.setText("Forgot Password");

    rgp.setOnCheckedChangeListener(new RadioGroup.OnCheckedChangeListener() {
        @Override
        public void onCheckedChanged(RadioGroup group, int checkedId) {

            switch(checkedId)
            {
                case R.id.radAdmin : choice = "admin"; break;

                case R.id.radCustomer : choice = "customer"; break;
            }
        }
    });

    frgtpass.setOnClickListener(new View.OnClickListener() {
        @Override

```

```

public void onClick(View v) {
    Intent registerIntent = new Intent(MainActivity.this, PasswordActivity.class);
    startActivity(registerIntent);
}
});

```

```

btnLogin.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {

        final ProgressDialog pd =
            new ProgressDialog(MainActivity.this);

        pd.setTitle("CONNECTING TO SERVER");
        pd.setMessage("Verifying User Details");
        pd.setProgressStyle(ProgressDialog.STYLE_SPINNER);
        pd.show();

        String myurl = "";

        if(choice.equals("admin"))
            myurl = ServerAddress.MYSERVER+"/AdminLogin.jsp";
        else
            myurl = ServerAddress.MYSERVER+"/CustomerLogin.jsp";

        String tag_string_req = "myrequest_1";

        Gson g = new Gson();
    }
});

```

```

HashMap<String,String> hmap =
    new HashMap<>();

hmap.put("ID",usr.getText().toString());
hmap.put("PASS",pswd.getText().toString());

try{
    JSONObject job =
        new JSONObject(g.toJson(hmap).trim());

    JsonRequest request =
        new JsonRequest(Request.Method.POST,
            myurl, job, new Response.Listener<JSONObject>() {
                @Override
                public void onResponse(JSONObject response) {

                    try{
                        pd.dismiss();
                        String str =
                            response.getString("cid");

                        if (str.equals("SUCCESS") && choice.equals("admin")) {
                            Toast.makeText(MainActivity.this,
                                "LOGIN SUCCESSFUL !!!",
                                Toast.LENGTH_SHORT).show();

                            Intent in = new Intent(MainActivity.this,
                                AdminHomeActivity.class);
                            startActivity(in);

```

```

        finish();
    }
    else if(str.equals("failure"))
    {
        Toast.makeText(MainActivity.this,
            "LOGIN UNSUCCESSFUL !!!",
Toast.LENGTH_SHORT).show();
    }
    else if(!str.equals("SUCCESS") && choice.equals("customer")){

        Toast.makeText(MainActivity.this,
            "LOGIN SUCCESSFUL !!!",
Toast.LENGTH_SHORT).show();

        Intent in = new Intent(MainActivity.this,
            CustomerHomeActivity.class);
        startActivity(in);
        finish();
    }
}
catch (Exception ex)
{
    ex.printStackTrace();
}

},

new Response.ErrorListener() {
    @Override
    public void onErrorResponse(VolleyError error) {

```

```

        pd.dismiss();
        Toast.makeText(MainActivity.this,
            "Volley Error "+error.getMessage(),
            Toast.LENGTH_SHORT).show();

    }

});

AppController.getInstance().
    addToRequestQueue(request,tag_string_req);

}
catch (Exception ex)
{
    ex.printStackTrace();
}

}

});

}

}

```

**AdminHomeActivity.java**

```
package com.example.justeat;

import android.app.Dialog;
import android.app.ProgressDialog;
import android.os.Bundle;
import android.support.v4.app.Fragment;
import android.support.v4.app.FragmentManager;
import android.support.v4.app.FragmentTransaction;
import android.view.View;
import android.support.v4.view.GravityCompat;
import android.support.v7.app.ActionBarDrawerToggle;
import android.view.MenuItem;
import android.support.design.widget.NavigationView;
import android.support.v4.widget.DrawerLayout;
import android.support.design.widget.FloatingActionButton;

import android.support.v7.app.AppCompatActivity;
import android.support.v7.widget.Toolbar;
import android.view.Menu;
import android.widget.Button;
import android.widget.EditText;
import android.widget.Toast;

import com.android.volley.DefaultRetryPolicy;
import com.android.volley.Request;
import com.android.volley.Response;
import com.android.volley.VolleyError;
import com.android.volley.toolbox.StringRequest;
```



```

import org.json.JSONObject;

public class AdminHomeActivity extends AppCompatActivity
    implements NavigationView.OnNavigationItemSelectedListener {

    Fragment[] fragments;
    FragmentManager mgr;
    FragmentTransaction ft;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_admin_home);

        fragments = new Fragment[8];

        fragments[0] = new ApproveCustFragment();
        fragments[1] = new AddFoodItemFragment();
        fragments[2] = new ShowFoodItemFragment();
        fragments[3] = new ShowDeleteCustomerFragment();
        fragments[4] = new ShowDeleteFoodCategoryFragment();

        mgr = getSupportFragmentManager();

        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, "Replace with your own action",
Snackbar.LENGTH_LONG)
//              .setAction("Action", null).show();
//      }
//  });

DrawerLayout drawer = findViewById(R.id.drawer_layout);
NavigationView navigationView = findViewById(R.id.nav_view);
ActionBarDrawerToggle toggle = new ActionBarDrawerToggle(
    this, drawer, toolbar, R.string.navigation_drawer_open,
R.string.navigation_drawer_close);
drawer.addDrawerListener(toggle);
toggle.syncState();
navigationView.setNavigationItemSelectedListener(this);
}

@Override
public void onBackPressed() {
    DrawerLayout drawer = findViewById(R.id.drawer_layout);
    if (drawer.isDrawerOpen(GravityCompat.START)) {
        drawer.closeDrawer(GravityCompat.START);
    } else {
        super.onBackPressed();
    }
}

@Override
public boolean onCreateOptionsMenu(Menu menu) {
    // Inflate the menu; this adds items to the action bar if it is present.
    getMenuInflater().inflate(R.menu.admin_home, menu);

```

```

    return true;
}

@Override
public boolean onOptionsItemSelected(MenuItem item) {
    // Handle action bar item clicks here. The action bar will
    // automatically handle clicks on the Home/Up button, so long
    // as you specify a parent activity in AndroidManifest.xml.
    int id = item.getItemId();

    //noinspection SimplifiableIfStatement
    if (id == R.id.action_settings) {
        return true;
    }

    return super.onOptionsItemSelected(item);
}

@SuppressWarnings("StatementWithEmptyBody")
@Override
public boolean onNavigationItemSelected(MenuItem item) {
    // Handle navigation view item clicks here.
    int id = item.getItemId();

    if (id == R.id.approve_customer) {

        ft = mgr.beginTransaction();

        ft.replace(R.id.approve_cust_linear, fragments[0]);
    }
}

```

```

ft.detach(fragments[0]);
ft.attach(fragments[0]);

ft.commit();
} else if (id == R.id.add_food_category) {

    final Dialog d = new Dialog(AdminHomeActivity.this);
    d setContentView(R.layout.add_food_category);

    final EditText edt = d.findViewById(R.id.edtFoodCat);
    Button btn = d.findViewById(R.id.btnAddCatg);

    btn.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View v) {

            final ProgressDialog pd = new ProgressDialog(AdminHomeActivity.this);
            pd.setTitle("CONNECTING TO SERVER");
            pd.setMessage("Uploading Data");
            pd.setProgressStyle(ProgressDialog.STYLE_SPINNER);
            pd.show();

            StringRequest req1 = new StringRequest(Request.Method.GET,
ServerAddress.MYSERVER +
"/AddFoodItemCategory.jsp?foodcat="+edt.getText().toString(),
            new Response.Listener<String>() {
                @Override
                public void onResponse(String response) {

                    pd.dismiss();

```

```

    try
    {
        JSONObject obj = new JSONObject(response.trim());

        if(obj.getString("msg").equals("success"))
        {
            d.dismiss();

            Toast.makeText(AdminHomeActivity.this,
                "Food Category Added Successfully !!!",
                Toast.LENGTH_SHORT).show();
        }
        else
        {
            d.dismiss();

            Toast.makeText(AdminHomeActivity.this,
                "Food Category Not Added !!!",
                Toast.LENGTH_SHORT).show();
        }
    }
    catch (Exception ex)
    {
        ex.printStackTrace();
    }
},

new Response.ErrorListener() {
    @Override
    public void onErrorResponse(VolleyError error) {
        pd.dismiss();
    }
}

```

```

        Toast.makeText(AdminHomeActivity.this, "Volley Error :
"+error.getMessage(), Toast.LENGTH_SHORT).show();
    }
    });

    req1.setRetryPolicy(new
DefaultRetryPolicy(0,DefaultRetryPolicy.DEFAULT_MAX_RETRIES,DefaultRetryPolicy.DE
FAULT_BACKOFF_MULT));

AppController.getInstance().addToRequestQueue(req1,"Add_Food_Category_Request
");
    }
    });

    d.show();

} else if (id == R.id.add_food_item) {

    ft = mgr.beginTransaction();

    ft.replace(R.id.approve_cust_linear,fragments[1]);

    ft.detach(fragments[1]);
    ft.attach(fragments[1]);

    ft.commit();
} else if (id == R.id.view_food_item) {

    ft = mgr.beginTransaction();

```

```

        ft.replace(R.id.approve_cust_linear,fragments[2]);

        ft.detach(fragments[2]);
        ft.attach(fragments[2]);

        ft.commit();
    } else if (id == R.id.show_customer) {

        ft = mgr.beginTransaction();

        ft.replace(R.id.approve_cust_linear,fragments[3]);

        ft.detach(fragments[3]);
        ft.attach(fragments[3]);

        ft.commit();
    }
    else if(id == R.id.view_food_category)
    {
        ft = mgr.beginTransaction();
        ft.replace(R.id.approve_cust_linear,fragments[4]);
        ft.detach(fragments[4]);
        ft.attach(fragments[4]);
        ft.commit();
    }
    DrawerLayout drawer = findViewById(R.id.drawer_layout);
    drawer.closeDrawer(GravityCompat.START);
    return true;
}
}

```

**ApproveCustFragment.java**

```
package com.example.justeat;

import android.app.ProgressDialog;
import android.os.Bundle;
import android.support.annotation.NonNull;
import android.support.annotation.Nullable;
import android.support.v4.app.Fragment;
import android.support.v7.widget.LinearLayoutManager;
import android.support.v7.widget.RecyclerView;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.Toast;

import com.android.volley.DefaultRetryPolicy;
import com.android.volley.Request;
import com.android.volley.Response;
import com.android.volley.VolleyError;
import com.android.volley.toolbox.StringRequest;
import com.dto.Customer;

import org.json.JSONArray;
import org.json.JSONObject;

import java.util.ArrayList;

public class ApproveCustFragment extends Fragment {
```



```

RecyclerView rcv;
RecyclerView.Adapter adapter;
RecyclerView.LayoutManager mgr;
ArrayList<Customer> clist;
String url =
    ServerAddress.MYSERVER+"/GetAllUnApprovedCustomers.jsp";

View v;

@Nullable
@Override
public View onCreateView(@NonNull LayoutInflater inflater, @Nullable ViewGroup
container, @Nullable Bundle savedInstanceState) {

    v = inflater.inflate(R.layout.approve_cust_fragment,container,false);

    return v;
}
@Override
public void onResume() {
    super.onResume();
    loadData();
}
public void loadData()
{
    rcv = v.findViewById(R.id.recycle_approve_cust);
    clist = new ArrayList<>();

    mgr = new LinearLayoutManager(getActivity());
    rcv.setLayoutManager(mgr);

```

```

final ProgressDialog pd =
    new ProgressDialog(getActivity());
pd.setTitle("CONNECTING TO SERVER");
pd.setMessage("Downloading Customers");
pd.setProgressStyle(ProgressDialog.STYLE_SPINNER);
pd.show();

StringRequest srq = new StringRequest(Request.Method.GET, url,
    new Response.Listener<String>() {
        @Override
        public void onResponse(String response) {
            if(response.trim().equals("no"))
            {
                pd.dismiss();
                Toast.makeText(getActivity(), "No records to display",
Toast.LENGTH_SHORT).show();
            }
            else {
                try {
                    pd.dismiss();
                    JSONArray arr = new JSONArray(response.trim());

                    for (int i = 0; i < arr.length(); i++) {
                        JSONObject obj = arr.getJSONObject(i);

                        Customer cust = new Customer();
                        cust.setId(obj.getString("id"));
                        cust.setFirstName(obj.getString("firstName"));
                        cust.setLastName(obj.getString("lastName"));
                        cust.setMobileNumber(obj.getString("mobileNumber"));
                        clist.add(cust);
                    }
                }
            }
        }
    }
);

```

```

    }

    adapter = new ApproveCustomerAdapter(clist, getActivity());
    rcv.setAdapter(adapter);
} catch (Exception ex) {
    ex.printStackTrace();
}
}
}
},
new Response.ErrorListener() {
    @Override
    public void onErrorResponse(VolleyError error) {

        Toast.makeText(getActivity(),
            "Connection error : "+error.getMessage(),
            Toast.LENGTH_SHORT).show();
        pd.dismiss();
    }
});

srq.setRetryPolicy(new DefaultRetryPolicy(
    0,
    DefaultRetryPolicy.DEFAULT_MAX_RETRIES,
    DefaultRetryPolicy.DEFAULT_BACKOFF_MULT));

AppController.getInstance().addToRequestQueue(srq,"approve_reject_request");
}
}

```

**ApproveCustomerAdapter.java**

```
package com.example.justeat;

import android.content.Context;
import android.support.annotation.NonNull;
import android.support.v7.widget.RecyclerView;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.Button;
import android.widget.TextView;
import android.widget.Toast;

import java.util.ArrayList;

import com.android.volley.DefaultRetryPolicy;
import com.android.volley.Request;
import com.android.volley.Response;
import com.android.volley.VolleyError;
import com.android.volley.toolbox.StringRequest;
import com.dto.Customer;

import org.json.JSONObject;

public class ApproveCustomerAdapter extends
RecyclerView.Adapter<ApproveCustomerAdapter.MyViewHolder>
{
    MyViewHolder holder;
    ArrayList<Customer> clist;
```

Context **context**;

```
public ApproveCustomerAdapter(ArrayList<Customer> clist, Context context) {
    this.clist = clist;
    this.context = context;
}
```

@NonNull

@Override

```
public MyViewHolder onCreateViewHolder(@NonNull ViewGroup viewGroup, int i) {
```

View v =

```
LayoutInflater.from(context).inflate(R.layout.cust_approve_reject,viewGroup,false);
```

```
holder = new MyViewHolder(v);
```

```
return holder;
```

```
}
```

@Override

```
public void onBindViewHolder(@NonNull MyViewHolder myViewHolder, int i) {
```

```
final int position = i;
```

```
myViewHolder.txtID.setText(clist.get(i).getId());
```

```
myViewHolder.txtFname.setText(clist.get(i).getFirstName());
```

```

myViewHolder.txtLname.setText(clist.get(i).getLastName());

myViewHolder.txtMobile.setText(clist.get(i).getMobileNumber());

myViewHolder.btnApprove.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {

        String custid = clist.get(position).getId();

        StringRequest req1 = new StringRequest(Request.Method.GET,
ServerAddress.MYSERVER + "/ApproveCustomer.jsp?custid=" + custid,
        new Response.Listener<String>() {
            @Override
            public void onResponse(String response) {

                try
                {
                    JSONObject obj = new JSONObject(response.trim());

                    if(obj.getString("msg").equals("success")) {
                        Toast.makeText(context, "The Customer has been verified
successfully !!!", Toast.LENGTH_SHORT).show();
                        clist.remove(position);
                        notifyDataSetChanged();
                    }
                    else
                        Toast.makeText(context, "The Customer has not been verified
successfully !!!", Toast.LENGTH_SHORT).show();

```

```

        }
        catch (Exception ex)
        {
            ex.printStackTrace();
        }
    }
},
new Response.ErrorListener() {
    @Override
    public void onErrorResponse(VolleyError error) {

        Toast.makeText(context, "Volley error while approving customer
"+error.getMessage(), Toast.LENGTH_SHORT).show();
    }
});
req1.setRetryPolicy(new DefaultRetryPolicy(
    0,
    DefaultRetryPolicy.DEFAULT_MAX_RETRIES,
    DefaultRetryPolicy.DEFAULT_BACKOFF_MULT));

AppController.getInstance().addToRequestQueue(req1,"approve_customer_request");
}
});
myViewHolder.btnReject.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        String custid = clist.get(position).getId();
        StringRequest req1 = new StringRequest(Request.Method.GET,
ServerAddress.MYSERVER + "/RejectCustomer.jsp?custid=" + custid,
        new Response.Listener<String>() {

```

```

@Override
public void onResponse(String response) {
    try
    {
        JSONObject obj = new JSONObject(response.trim());
        if(obj.getString("msg").equals("success")) {
            Toast.makeText(context, "The Customer has been rejected
successfully !!!", Toast.LENGTH_SHORT).show();

            clist.remove(position);
            notifyDataSetChanged();
        }
        else
            Toast.makeText(context, "The Customer has not been rejected
successfully !!!", Toast.LENGTH_SHORT).show();
    }
    catch (Exception ex)
    {
        ex.printStackTrace();
    }
},
new Response.ErrorListener() {
    @Override
    public void onErrorResponse(VolleyError error) {

        Toast.makeText(context, "Volley error while rejecting customer
"+error.getMessage(), Toast.LENGTH_SHORT).show();
    }
});

req1.setRetryPolicy(new DefaultRetryPolicy(

```



```

        0,
        DefaultRetryPolicy.DEFAULT_MAX_RETRIES,
        DefaultRetryPolicy.DEFAULT_BACKOFF_MULT));
AppController.getInstance().addToRequestQueue(req1,"approve_customer_request");
    }
    });
}
@Override
public int getItemCount() {
    return clist.size();
}
class MyViewHolder extends RecyclerView.ViewHolder
{
    TextView txtID,txtMobile,txtFname,txtLname;
    Button btnApprove,btnReject;
    public MyViewHolder(View v)
    {
        super(v);
        txtID = v.findViewById(R.id.txtCustID);
        txtMobile = v.findViewById(R.id.txtCustMobile);
        txtFname = v.findViewById(R.id.txtCustFname);
        txtLname = v.findViewById(R.id.txtCustLname);
        btnApprove = v.findViewById(R.id.btnApprove);
        btnReject = v.findViewById(R.id.btnReject);
    }
}
}

```

**AddFooditemFragment.java**

```
package com.example.justeat;

import android.app.ProgressDialog;
import android.content.Intent;
import android.graphics.Bitmap;
import android.os.AsyncTask;
import android.os.Bundle;
import android.provider.MediaStore;
import android.support.annotation.NonNull;
import android.support.annotation.Nullable;
import android.support.v4.app.Fragment;
import android.util.Log;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.AdapterView;
import android.widget.Button;
import android.widget.EditText;
import android.widget.ImageView;
import android.widget.Spinner;
import android.widget.Toast;

import com.android.volley.Request;
import com.android.volley.Response;
import com.android.volley.VolleyError;
import com.android.volley.toolbox.StringRequest;
import com.dto.FoodItemCategory;
```

```

import org.json.JSONArray;
import org.json.JSONObject;

import java.io.File;
import java.io.FileOutputStream;
import java.net.URL;
import java.util.ArrayList;

import static android.app.Activity.RESULT_OK;
import static android.content.Context.MODE_PRIVATE;

public class AddFoodItemFragment extends Fragment {

    View v;
    ImageView iv;
    EditText edtName,edtDesc,edtPrice,edtPersonCount;
    Spinner spCategory;
    Button btnAddItem;
    ArrayList<FoodItemCategory> catlist;
    ArrayAdapter<FoodItemCategory> cadapter;
    String imageName="";
    File f;

    @Nullable
    @Override
    public View onCreateView(@NonNull LayoutInflater inflater, @Nullable ViewGroup
container, @Nullable Bundle savedInstanceState) {

        v = inflater.inflate(R.layout.add_food_item,container,false);

```

```
return v;
}
```

```
@Override
```

```
public void onResume() {
    super.onResume();
    loadData();
}
```

```
public void loadData()
{
    iv = v.findViewById(R.id.imgFood);
    edtName = v.findViewById(R.id.edtItemName);
    edtDesc = v.findViewById(R.id.edtFoodDesc);
    edtPrice = v.findViewById(R.id.edtItemPrice);
    edtPersonCount = v.findViewById(R.id.edtPrsnCnt);
    btnAddItem = v.findViewById(R.id.btnAddItem);
    spCategory = v.findViewById(R.id.spFoodCategory);
    //btnAddItem.setEnabled(false);

    long millis = System.currentTimeMillis();

    imageName="img"+millis+".png";

    catlist = new ArrayList<>();

    cadapter = new ArrayAdapter<>(getActivity(),
        android.R.layout.simple_list_item_1,catlist);

    spCategory.setAdapter(cadapter);
```

```
cadapter.setDropDownViewResource(
    android.R.layout.select_dialog_singlechoice);
```

```
final ProgressDialog pd =
    new ProgressDialog(getActivity());
```

```
pd.setTitle("CONNECTING TO SERVER");
pd.setMessage("Downloading Data From Server");
pd.setProgressStyle(ProgressDialog.STYLE_SPINNER);
pd.show();
```

```
StringRequest srq = new StringRequest(Request.Method.GET,
ServerAddress.MYSERVER + "/GetAllFoodCategories.jsp",
    new Response.Listener<String>() {
        @Override
        public void onResponse(String response) {

            pd.dismiss();

            try
            {
                JSONArray arr = new JSONArray(response.trim());

                for(int i = 0 ; i < arr.length() ; i++)
                {
                    JSONObject obj = arr.getJSONObject(i);

                    FoodItemCategory fcat = new FoodItemCategory();
```

```

        fcat.setId(obj.getInt("id"));
        fcat.setCategoryName(obj.getString("categoryName"));
        catlist.add(fcat);
    }

    cadapter.notifyDataSetChanged();
}
catch(Exception ex)
{
    ex.printStackTrace();
}
},
new Response.ErrorListener() {
    @Override
    public void onErrorResponse(VolleyError error) {

        Toast.makeText(getActivity(),
            "Could not load data , Volley error !!!",
            Toast.LENGTH_SHORT).show();
        pd.dismiss();
    }
});

AppController.getInstance().addToRequestQueue(srq,"Get All Categories");

iv.setOnClickListener(new View.OnClickListener() {
    @Override

```

```

public void onClick(View v) {
    Intent in =
        new Intent(MediaStore.ACTION_IMAGE_CAPTURE);
    startActivityForResult(in,123);
}
});

```

```

btnAddItem.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {

        if(edtName.getText().toString().equals(""))
            || edtDesc.getText().toString().equals("") ||
edtPersonCount.getText().toString().equals("")
            || edtPrice.getText().toString().equals(""))
        {
            Toast.makeText(getActivity(),
                "Plz fill up all the details !!!",
                Toast.LENGTH_SHORT).show();
        }
        else {
            String filePath = f.getAbsolutePath();
            MyTask tsk = new MyTask();
            tsk.execute(filePath);
        }
    }
});
}

```

```

@Override

```

```

public void onActivityResult(int requestCode, int resultCode, Intent data) {

    if(requestCode == 123 && data != null &&
        resultCode == RESULT_OK)
    {
        Bitmap bmp=null;

        try {

            bmp = (Bitmap)data.getExtras().get("data");

        } catch (Exception e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        }

        iv.setImageBitmap(bmp);

        File appPackageDir= getActivity().getDir("user_img",MODE_PRIVATE);

        f=new File(appPackageDir, imageName);

        try{
            FileOutputStream fout=
                new FileOutputStream(f);
            //bmp.compress(format, quality, stream);
            boolean result=bmp.compress(Bitmap.CompressFormat.PNG,
                100, fout);
            if(result==true)
                Toast.makeText(getActivity(),

```



```

        "image selected", Toast.LENGTH_LONG).show();

        btnAddItem.setEnabled(true);

        fout.close();

    }catch(Exception ex)
    {
        Log.e("error;", ex.toString());
    }
}
}

```

```

class MyTask extends AsyncTask<String,Void,String>
{
    ProgressDialog pd =
        new ProgressDialog(getActivity());

    @Override
    protected void onPreExecute() {
        pd.setTitle("CONNECTING TO SERVER");
        pd.setMessage("Uploading Food Item");
        pd.setProgressStyle(ProgressDialog.STYLE_SPINNER);
        pd.show();
    }

    @Override
    protected void onPostExecute(String s) {

```

```

super.onPostExecute(s);

pd.dismiss();

if(s.trim().equals("success")) {
    Toast.makeText(getActivity(),
        "Food Item added successfully !!!"
        ,Toast.LENGTH_SHORT).show();
}

else
{
    Toast.makeText(getActivity(),
        "Food Item Upload failed !!!"
        ,Toast.LENGTH_SHORT).show();
}
}

@Override

protected String doInBackground(String... strings)
{
    String filePath=strings[0];
    String str="";
    try{
        URL u=new URL(ServerAddress.MYSERVER +
            "/AddFoodItem.jsp");
        MultipartUtility mpu=
            new MultipartUtility(u);

        mpu.addFilePart("file",
            new File(filePath));
    }
}

```

```

        mpu.addFormField("fooditemname",
            edtName.getText().toString());

        mpu.addFormField("fooddesc",
            edtDesc.getText().toString());

        mpu.addFormField("fooditemprice",
            edtPrice.getText().toString());

        mpu.addFormField("personcount",
            edtPersonCount.getText().toString());

        mpu.addFormField("categoryname",
            spCategory.getSelectedItem().toString());

        byte []result=mpu.finish();

        str=new String(result);

    }catch(Exception ex)
    {
        str=ex.toString();
    }

    return str;
}
}
}

```

**ShowFoodItemFragment.java**

```
package com.example.justeat;

import android.app.ProgressDialog;
import android.os.Bundle;
import android.support.annotation.NonNull;
import android.support.annotation.Nullable;
import android.support.v4.app.Fragment;
import android.support.v7.widget.LinearLayoutManager;
import android.support.v7.widget.RecyclerView;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.Button;
import android.widget.ImageView;
import android.widget.TextView;
import android.widget.Toast;

import com.android.volley.DefaultRetryPolicy;
import com.android.volley.Request;
import com.android.volley.Response;
import com.android.volley.VolleyError;
import com.android.volley.toolbox.StringRequest;
import com.dto.Customer;
import com.dto.FoodItem;

import org.json.JSONArray;
import org.json.JSONObject;
```

```

import java.util.ArrayList;

public class ShowFoodItemFragment extends Fragment {
    View v;
    // TextView tvFoodName,tvFoodCat,tvFoodPrice;
    // Button btnFoodUpdate,btnFoodDelete;
    // ImageView ivFoodItem;
    RecyclerView rcv;
    RecyclerView.Adapter adapter;
    RecyclerView.LayoutManager mgr;
    ArrayList<FoodItem> flist;
    @Nullable
    @Override
    public View onCreateView(@NonNull LayoutInflater inflater, @Nullable ViewGroup
container, @Nullable Bundle savedInstanceState) {
        v = inflater.inflate(R.layout.show_food_item_fragment,container,false);
        return v;
    }
    @Override
    public void onResume() {
        super.onResume();
        loadData();
    }
    public void loadData()
    {
        // tvFoodName = v.findViewById(R.id.txtFoodName);
        // tvFoodCat = v.findViewById(R.id.txtFoodCat);
        // tvFoodPrice = v.findViewById(R.id.txtFoodPrice);
        // btnFoodDelete = v.findViewById(R.id.btnFoodDelete);
        // btnFoodUpdate = v.findViewById(R.id.btnFoodUpdate);
    }
}

```

```

//    ivFoodItem = v.findViewById(R.id.imgFoodItem);

    rcv = v.findViewById(R.id.show_food_recycle);

    flist = new ArrayList<>();

    mgr = new LinearLayoutManager(getActivity());

    rcv.setLayoutManager(mgr);

    final ProgressDialog pd =
        new ProgressDialog(getActivity());

    pd.setTitle("CONNECTING TO SERVER");

    pd.setMessage("Downloading Food Items");

    pd.setProgressStyle(ProgressDialog.STYLE_SPINNER);

    pd.show();

    StringRequest srq = new StringRequest(Request.Method.GET,
ServerAddress.MYSERVER + "/GetAllFoodItems.jsp",
        new Response.Listener<String>() {
            @Override

            public void onResponse(String response) {
                if(response.trim().equals("no"))
                {
                    pd.dismiss();

                    Toast.makeText(getActivity(), "No food Items to display",
Toast.LENGTH_SHORT).show();
                }

                else {
                    try {
                        pd.dismiss();

                        JSONArray arr = new JSONArray(response.trim());

                        for (int i = 0; i < arr.length(); i++) {
                            JSONObject obj = arr.getJSONObject(i);

                            FoodItem f = new FoodItem();

                            f.setld(obj.getInt("id"));

```

```

        f.setItemName(obj.getString("itemName"));
        f.setItemPrice(obj.getInt("itemPrice"));
        f.setImg_path(obj.getString("img_path"));
        f.setCategoryID(obj.getInt("categoryID"));
        flist.add(f);
    }

    adapter = new ShowFoodItemAdapter(flist, getActivity());

    rcv.setAdapter(adapter);
} catch (Exception ex) {
    ex.printStackTrace();
}
}
}
},
new Response.ErrorListener() {
    @Override
    public void onErrorResponse(VolleyError error) {

        Toast.makeText(getActivity(), "Volley Error : "+error.getMessage(),
Toast.LENGTH_SHORT).show();

    }

});

srq.setRetryPolicy(new DefaultRetryPolicy(
    0,
    DefaultRetryPolicy.DEFAULT_MAX_RETRIES,
    DefaultRetryPolicy.DEFAULT_BACKOFF_MULT));
AppController.getInstance().addToRequestQueue(srq,"Get_Food_Items");
}
}

```

**ShowDeleteFoodCategoryAdapter.java**

```
package com.example.justeat;

import android.app.ProgressDialog;
import android.content.Context;
import android.support.annotation.NonNull;
import android.support.v7.widget.RecyclerView;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.Button;
import android.widget.TextView;
import android.widget.Toast;

import com.android.volley.DefaultRetryPolicy;
import com.android.volley.Request;
import com.android.volley.Response;
import com.android.volley.VolleyError;
import com.android.volley.toolbox.StringRequest;
import com.dto.Customer;
import com.dto.FoodItemCategory;

import org.json.JSONObject;

import java.util.ArrayList;

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



```

MyViewHolder holder;
ArrayList<FoodItemCategory> catlist;
Context context;

public ShowDeleteFoodCategoryAdapter(ArrayList<FoodItemCategory> catlist,
Context context) {
    this.catlist = catlist;
    this.context = context;
}

@NonNull
@Override
public MyViewHolder onCreateViewHolder(@NonNull ViewGroup viewGroup, int i) {

    View v =
LayoutInflater.from(context).inflate(R.layout.food_category_card,viewGroup,false);

    holder = new MyViewHolder(v);

    return holder;
}

@Override
public void onBindViewHolder(@NonNull MyViewHolder myViewHolder, int i) {

    final int position = i;

    myViewHolder.txtFoodCat.setText(catlist.get(i).getCategoryName());

```

```

myViewHolder.btnFoodCatDelete.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {

        int catid = catlist.get(position).getId();

        final ProgressDialog pd = new ProgressDialog(context);

        pd.setTitle("CONNECTING TO SERVER");
        pd.setMessage("Deleting Food Category");
        pd.setProgressStyle(ProgressDialog.STYLE_SPINNER);
        pd.show();

        StringRequest req1 = new StringRequest(Request.Method.GET,
ServerAddress.MYSERVER + "/DeleteFoodCategory.jsp?catid=" + catid,
        new Response.Listener<String>() {
            @Override
            public void onResponse(String response) {

                try
                {
                    pd.dismiss();

                    JSONObject obj = new JSONObject(response.trim());

                    if(obj.getString("msg").equals("success")) {
                        Toast.makeText(context, "The Food Category been deleted
successfully !!!", Toast.LENGTH_SHORT).show();
                        catlist.remove(position);

```

```

        notifyDataSetChanged();
    }
    else
        Toast.makeText(context, "The Food Category has not been
deleted successfully !!!", Toast.LENGTH_SHORT).show();
    }
    catch (Exception ex)
    {
        ex.printStackTrace();
    }
}
},
new Response.ErrorListener() {
    @Override
    public void onErrorResponse(VolleyError error) {

        pd.dismiss();
        Toast.makeText(context, "Volley error while deleting Food Category
"+error.getMessage(), Toast.LENGTH_SHORT).show();
    }
});

req1.setRetryPolicy(new DefaultRetryPolicy(
    0,
    DefaultRetryPolicy.DEFAULT_MAX_RETRIES,
    DefaultRetryPolicy.DEFAULT_BACKOFF_MULT));

AppController.getInstance().addToRequestQueue(req1,"delete_food_category_request
");

```

```
    }  
    });  
}  
  
@Override  
public int getItemCount() {  
    return catlist.size();  
}  
  
class MyViewHolder extends RecyclerView.ViewHolder  
{  
    TextView txtFoodCat;  
    Button btnFoodCatDelete;  
  
    public MyViewHolder(View v)  
    {  
        super(v);  
        txtFoodCat = v.findViewById(R.id.txtFoodCat);  
        btnFoodCatDelete = v.findViewById(R.id.btnFoodCatDelete);  
    }  
}  
  
}
```

**ApproveCustFragment.java**

```
package com.example.justeat;

import android.app.ProgressDialog;
import android.os.Bundle;
import android.support.annotation.NonNull;
import android.support.annotation.Nullable;
import android.support.v4.app.Fragment;
import android.support.v7.widget.LinearLayoutManager;
import android.support.v7.widget.RecyclerView;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.Toast;

import com.android.volley.DefaultRetryPolicy;
import com.android.volley.Request;
import com.android.volley.Response;
import com.android.volley.VolleyError;
import com.android.volley.toolbox.StringRequest;
import com.dto.Customer;

import org.json.JSONArray;
import org.json.JSONObject;

import java.util.ArrayList;

public class ApproveCustFragment extends Fragment {
```

```

RecyclerView rcv;
RecyclerView.Adapter adapter;
RecyclerView.LayoutManager mgr;
ArrayList<Customer> clist;
String url =
    ServerAddress.MYSERVER+"/GetAllUnApprovedCustomers.jsp";

View v;

@Nullable
@Override
public View onCreateView(@NonNull LayoutInflater inflater, @Nullable ViewGroup
container, @Nullable Bundle savedInstanceState) {

    v = inflater.inflate(R.layout.approve_cust_fragment,container,false);

    return v;
}

@Override
public void onResume() {
    super.onResume();
    loadData();
}

public void loadData()
{
    rcv = v.findViewById(R.id.recycle_approve_cust);

    clist = new ArrayList<>();

```

```

mgr = new LinearLayoutManager(getActivity());

rcv.setLayoutManager(mgr);

final ProgressDialog pd =
    new ProgressDialog(getActivity());

pd.setTitle("CONNECTING TO SERVER");
pd.setMessage("Downloading Customers");
pd.setProgressStyle(ProgressDialog.STYLE_SPINNER);
pd.show();

StringRequest srq = new StringRequest(Request.Method.GET, url,
    new Response.Listener<String>() {
        @Override
        public void onResponse(String response) {
            if(response.trim().equals("no"))
            {
                pd.dismiss();
                Toast.makeText(getActivity(), "No records to display",
Toast.LENGTH_SHORT).show();
            }
            else {
                try {

                    pd.dismiss();
                    JSONArray arr = new JSONArray(response.trim());

                    for (int i = 0; i < arr.length(); i++) {

```

```

        JSONObject obj = arr.getJSONObject(i);

        Customer cust = new Customer();
        cust.setId(obj.getString("id"));
        cust.setFirstName(obj.getString("firstName"));
        cust.setLastName(obj.getString("lastName"));
        cust.setMobileNumber(obj.getString("mobileNumber"));

        clist.add(cust);
    }

    adapter = new ApproveCustomerAdapter(clist, getActivity());

    rcv.setAdapter(adapter);

    } catch (Exception ex) {
        ex.printStackTrace();
    }
}
},
new Response.ErrorListener() {
    @Override
    public void onErrorResponse(VolleyError error) {

        Toast.makeText(getActivity(),
            "Connection error : "+error.getMessage(),
            Toast.LENGTH_SHORT).show();

        pd.dismiss();
    }
}

```



```
    }  
    });  
  
    srq.setRetryPolicy(new DefaultRetryPolicy(  
        0,  
        DefaultRetryPolicy.DEFAULT_MAX_RETRIES,  
        DefaultRetryPolicy.DEFAULT_BACKOFF_MULT));  
  
    ApplicationController.getInstance().addToRequestQueue(srq,"approve_reject_request");  
    }  
}
```

**XML Code****activity\_splash.xml**

```

<?xml version="1.0" encoding="utf-8"?>
<android.support.constraint.ConstraintLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:background="@color/colorPub"
    tools:context="com.example.justeat.SplashActivity">
    <ImageView
        android:id="@+id/imageView"
        android:layout_width="169dp"
        android:layout_height="173dp"
        android:background="@drawable/index"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintHorizontal_bias="0.497"
        app:layout_constraintLeft_toLeftOf="parent"
        app:layout_constraintRight_toRightOf="parent"
        app:layout_constraintTop_toTopOf="parent"
        app:layout_constraintVertical_bias="0.388" />
    <ProgressBar android:layout_width="wrap_content"
        android:layout_height="wrap_content" android:indeterminate="true"
        app:layout_constraintLeft_toLeftOf="parent"
        app:layout_constraintRight_toRightOf="parent"
        android:layout_marginTop="8dp"
        app:layout_constraintTop_toBottomOf="@id/imageView" />

</android.support.constraint.ConstraintLayout>

```

**activity\_phasing.xml**

```

<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".PhasingActivity"
    android:background="@drawable/logo">
    <LinearLayout
        android:orientation="vertical"
        android:layout_centerInParent="true"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content">
        <ImageView
            android:layout_width="212dp"
            android:layout_height="108dp"
            android:src="@drawable/cooltext1" />
    </LinearLayout>
    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:orientation="horizontal"
        android:layout_alignParentBottom="true"
        android:weightSum="2">
        <Button
            android:layout_width="0dp"
            android:layout_height="wrap_content"
            android:layout_margin="8dp"

```

```
        android:layout_weight="1"
        android:id="@+id/btnSignUp"
        android:textColor="@color/colorWhite"
        android:background="@color/colorPrimary"
        android:text="Sign Up" />
    <Button
        android:layout_width="0dp"
        android:layout_height="wrap_content"
        android:layout_weight="1"
        android:text="Sign In"
        android:id="@+id/btnSignIn"
        android:layout_margin="8dp"
        android:background="@color/colorAccent"
        android:textColor="@android:color/white"/>
</LinearLayout>
</RelativeLayout>
```

**activity\_main.xml**

```

<?xml version="1.0" encoding="utf-8"?>
<android.support.design.widget.CoordinatorLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:fitsSystemWindows="true"
    tools:context=".MainActivity">
    <LinearLayout
        android:layout_width="fill_parent"
        android:layout_height="fill_parent"
        android:background="@color/colorPrimaryDark"
        android:gravity="center"
        android:orientation="vertical"
        android:padding="@dimen/activity_horizontal_margin">
        <ImageView
            android:layout_width="@dimen/logo_w_h"
            android:layout_height="@dimen/logo_w_h"
            android:layout_gravity="center_horizontal"
            android:layout_marginBottom="30dp"
            app:srcCompat="@drawable/index" />

        <RadioGroup
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_alignParentStart="true"
            android:layout_alignParentLeft="true"

```

```

        android:layout_alignParentTop="true"
        android:layout_marginTop="5dp"
        android:orientation="horizontal"
        android:id="@+id/rgp">
        <RadioButton
            android:id="@+id/radAdmin"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:text="ADMIN LOGIN"
            android:textColor="@color/colorWhite"
        />
        <RadioButton
            android:id="@+id/radCustomer"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:text="CUSTOMER LOGIN"
            android:textColor="@color/colorWhite"/>
    </RadioGroup>
    <android.support.design.widget.TextInputLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content">
        <EditText
            android:id="@+id/edtName"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:hint="User_ID"
            android:textColorHint="@color/colorWhite"
            android:inputType="textEmailAddress"
            android:maxLines="1"
            android:singleLine="true"

```

```

        android:drawableLeft="@drawable/user24"
        android:ems="10"
        android:paddingLeft="2dp"
        android:text=""
        tools:layout_editor_absoluteX="0dp"
        tools:layout_editor_absoluteY="36dp"
        android:textColor="@android:color/white" />
</android.support.design.widget.TextInputLayout>
<android.support.design.widget.TextInputLayout
    android:layout_width="match_parent"
    android:layout_height="wrap_content">
    <EditText
        android:id="@+id/edtPass"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:focusableInTouchMode="true"
        android:hint="Password"
        android:textColorHint="@color/colorWhite"
        android:imeOptions="actionUnspecified"
        android:inputType="textPassword"
        android:drawableLeft="@drawable/lock24"
        android:ems="10"
        android:maxLines="1"
        android:singleLine="true"
        android:textColor="@android:color/white" />
</android.support.design.widget.TextInputLayout>
<Button
    android:id="@+id/btnLogin"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"

```

```

        android:layout_marginTop="20dip"
        android:background="@drawable/mybutton"
        android:text="LOGIN"
        android:textAllCaps="false"
        android:textColor="@color/white"
        android:textSize="15dp" />
<TextView
    android:id="@+id/forgot_password"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:layout_marginTop="20dip"
    android:background="@null"
    android:text="forgot password"
    android:textAllCaps="false"
    android:paddingTop="10dp"
    android:textStyle="bold"
    android:textColor="@color/colorAccent" />
<!-- Link to Login Screen -->
</LinearLayout>
<ProgressBar
    android:id="@+id/progressBar"
    android:layout_width="30dp"
    android:layout_height="30dp"
    android:layout_gravity="center|bottom"
    android:layout_marginBottom="20dp"
    android:visibility="gone" />
</android.support.design.widget.CoordinatorLayout>

```



**activity\_customer\_home.xml**

```

<?xml version="1.0" encoding="utf-8"?>
<android.support.v4.widget.DrawerLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:id="@+id/drawer_layout"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:fitsSystemWindows="true"
    tools:openDrawer="start"
    android:background="@drawable/logo">

    <include
        layout="@layout/app_bar_customer_home"
        android:layout_width="match_parent"
        android:layout_height="match_parent" />

    <android.support.design.widget.NavigationView
        android:id="@+id/nav_view"
        android:layout_width="wrap_content"
        android:layout_height="match_parent"
        android:layout_gravity="start"
        android:fitsSystemWindows="true"
        app:headerLayout="@layout/nav_header_customer_home"
        app:menu="@menu/activity_customer_home_drawer" />

</android.support.v4.widget.DrawerLayout>

```

**add food category.xml**

```

<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    android:orientation="vertical"
    android:layout_width="match_parent"
    android:layout_height="wrap_content">

    <EditText
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:textAppearance="@style/TextAppearance.AppCompat.Large"
        android:id="@+id/edtFoodCat"
        android:hint="Enter Food Category"/>

    <Button
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:id="@+id/btnAddCatg"
        android:textColor="@color/colorWhite"
        android:text="Add Category"
        android:layout_weight="1"
        android:background="@drawable/mybutton"
    />
</LinearLayout>

```

**add food item.xml**

```

<?xml version="1.0" encoding="utf-8"?>
<ScrollView android:layout_width="match_parent"
    android:layout_height="match_parent"
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:background="@color/colorActive">
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    android:orientation="vertical"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:background="@color/colorActive">
    <ImageView    android:layout_width="match_parent"
        android:layout_height="130dp"
        android:src="@drawable/qmenu"
        android:id="@+id/imgFood"/>
    <Spinner android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:id="@+id/spFoodCategory"
        android:spinnerMode="dialog"
        android:prompt="@string/prompt1"/>
    <EditText android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:textAppearance="@style/TextAppearance.AppCompat.Large"
        android:id="@+id/edtItemName"
        android:hint="Food Item Name"
        android:textColorHint="@color/colorWhite"
        android:textColor="@color/colorWhite"/>
    <EditText android:layout_width="match_parent"

```

```

        android:layout_height="wrap_content"
        android:textAppearance="@style/TextAppearance.AppCompat.Large"
        android:id="@+id/edtFoodDesc"
        android:hint="Food Description"
        android:textColorHint="@color/colorWhite"
        android:textColor="@color/colorWhite"
        android:lines="5"/>
    <EditText android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:textAppearance="@style/TextAppearance.AppCompat.Large"
        android:id="@+id/edtItemPrice"
        android:hint="Food Item Price"
        android:textColorHint="@color/colorWhite"
        android:textColor="@color/colorWhite"/>
    <EditText android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:textAppearance="@style/TextAppearance.AppCompat.Large"
        android:id="@+id/edtPrsnCnt"
        android:textColorHint="@color/colorWhite"
        android:hint="Person Count"
        android:textColor="@color/colorWhite"/>
    <Button android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:id="@+id/btnAddItem"
        android:text="Add Item Category"
        android:textColor="@color/colorWhite"
        android:textStyle="bold"
        android:background="@drawable/mybutton"/>
</LinearLayout>
</ScrollView>

```

**activity\_update\_food\_item.xml**

```

<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    android:background="@color/colorActive" >
    <EditText
        android:id="@+id/edtItemName"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_weight="1"
        android:hint="Food Item Name"
        android:textColorHint="@color/colorWhite"
        android:textAlignment="center"
        android:textAppearance="@style/TextAppearance.AppCompat.Large"
        android:textColor="@color/colorWhite" />
    <EditText
        android:id="@+id/edtItemDesc"
        android:layout_width="match_parent"
        android:textColorHint="@color/colorWhite"
        android:layout_height="wrap_content"
        android:layout_weight="1"
        android:hint="Food Item Description"
        android:textAlignment="center"
        android:textAppearance="@style/TextAppearance.AppCompat.Large"
        android:textColor="@color/colorWhite" />

```

```

<EditText
    android:id="@+id/edtfoodItemPrice"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:textColorHint="@color/colorWhite"
    android:layout_weight="1"
    android:hint="Food Item Price"
    android:textAlignment="center"
    android:textAppearance="@style/TextAppearance.AppCompat.Large"
    android:textColor="@color/colorWhite" />

```

```

<EditText
    android:id="@+id/edtPerOff"
    android:layout_width="match_parent"
    android:textColorHint="@color/colorWhite"
    android:layout_height="wrap_content"
    android:layout_weight="1"
    android:hint="Percent Off"
    android:textAlignment="center"
    android:textAppearance="@style/TextAppearance.AppCompat.Large"
    android:textColor="@color/colorWhite" />

```

```

<EditText
    android:id="@+id/edtItemStatus"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:textColorHint="@color/colorWhite"
    android:layout_weight="1"
    android:hint="Food Item Status"
    android:textAlignment="center"
    android:textAppearance="@style/TextAppearance.AppCompat.Large"
    android:textColor="@color/colorWhite" />

```

```

<EditText
    android:id="@+id/edtprsnCnt"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_weight="1"
    android:textColorHint="@color/colorWhite"
    android:hint="Person Count"
    android:textAlignment="center"
    android:textAppearance="@style/TextAppearance.AppCompat.Large"
    android:textColor="@color/colorWhite" />

```

```

<Button
    android:id="@+id/btnUpdateItem"
    android:layout_width="272dp"
    android:layout_height="wrap_content"
    android:layout_marginLeft="65dp"
    android:layout_weight="1"
    android:background="@drawable/mybutton"
    android:gravity="center"
    android:padding="15dp"
    android:text="Update Food Item"
    android:textColor="#fff"
    android:textStyle="bold" />

```

```

</LinearLayout>

```

## 14. Forms

phpMyAdmin

Recent Favorites

New

classification\_project

houseofmandidb

New

admin

customer

fooditem

fooditemcategory

itemorder

itemorderdetail

Server: Local Databases » Database: houseofmandidb

Structure SQL Search Query Export Import Operations Privileges Routines Events

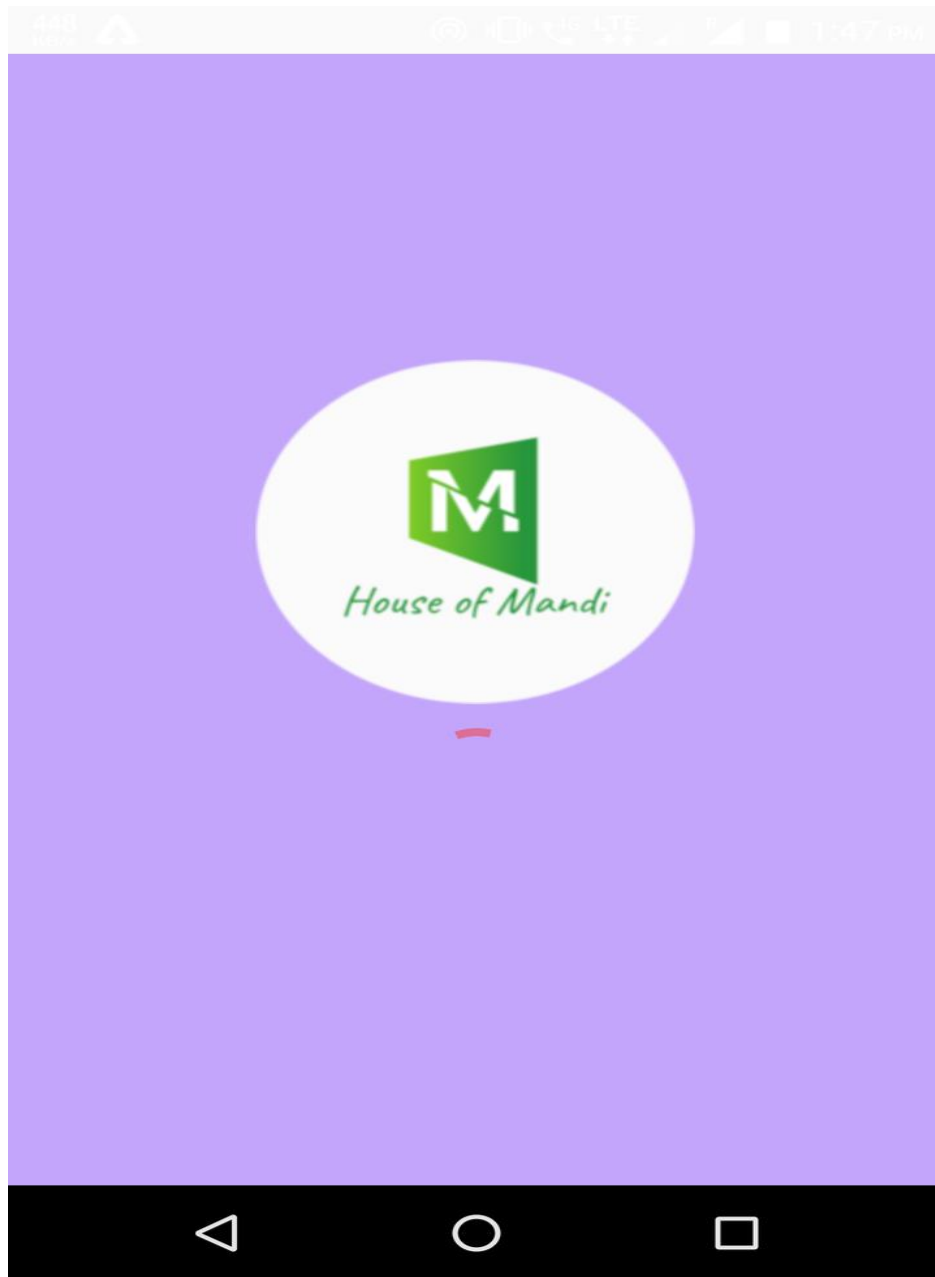
Table	Action	Rows	Type	Collation	Size	Overhead
admin	★ Browse Structure Search Insert Empty Drop	0	InnoDB	latin1_swedish_ci	16 KiB	-
customer	★ Browse Structure Search Insert Empty Drop	0	InnoDB	latin1_swedish_ci	16 KiB	-
fooditem	★ Browse Structure Search Insert Empty Drop	0	InnoDB	latin1_swedish_ci	32 KiB	-
fooditemcategory	★ Browse Structure Search Insert Empty Drop	0	InnoDB	latin1_swedish_ci	16 KiB	-
itemorder	★ Browse Structure Search Insert Empty Drop	0	InnoDB	latin1_swedish_ci	32 KiB	-
itemorderdetail	★ Browse Structure Search Insert Empty Drop	0	InnoDB	latin1_swedish_ci	48 KiB	-
6 tables	Sum	0	MyISAM	latin1_swedish_ci	160 KiB	0 B

↑ ☐ Check all With selected: ▼



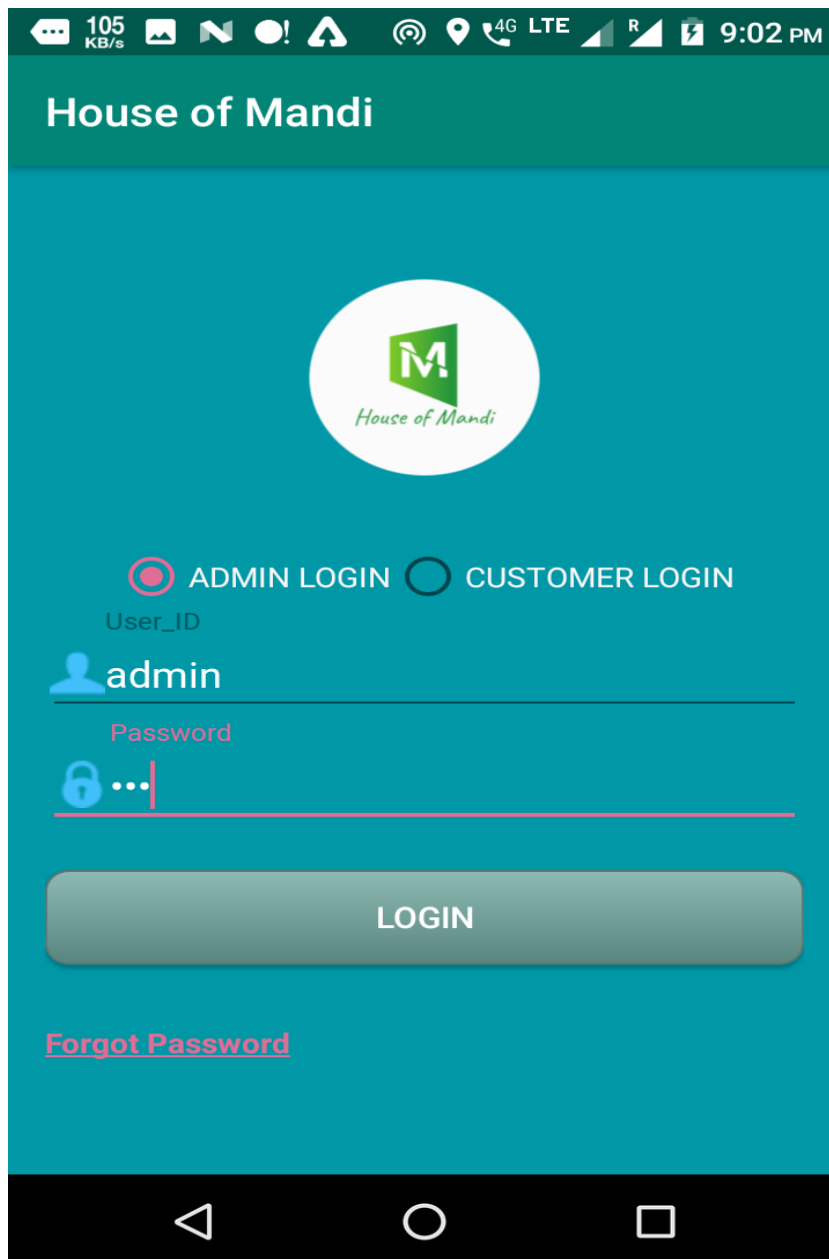
## SCREENSHOTS

### a. Splash Screen:



**b. Home-Page:**



**c. Login-Page: Admin**

The screenshot displays the 'House of Mandi' admin login interface on a mobile device. At the top, a green header bar contains the app's name. Below this, a circular logo with a green 'M' and the text 'House of Mandi' is centered. The login options are 'ADMIN LOGIN' (selected with a red radio button) and 'CUSTOMER LOGIN' (unselected). The 'User\_ID' field is populated with 'admin'. The 'Password' field is masked with three dots. A large, rounded 'LOGIN' button is positioned below the fields. At the bottom, a red link for 'Forgot Password' is visible. The status bar at the very top shows network and battery information, and the Android navigation bar is at the bottom.

House of Mandi

☒ ADMIN LOGIN ☐ CUSTOMER LOGIN

User\_ID

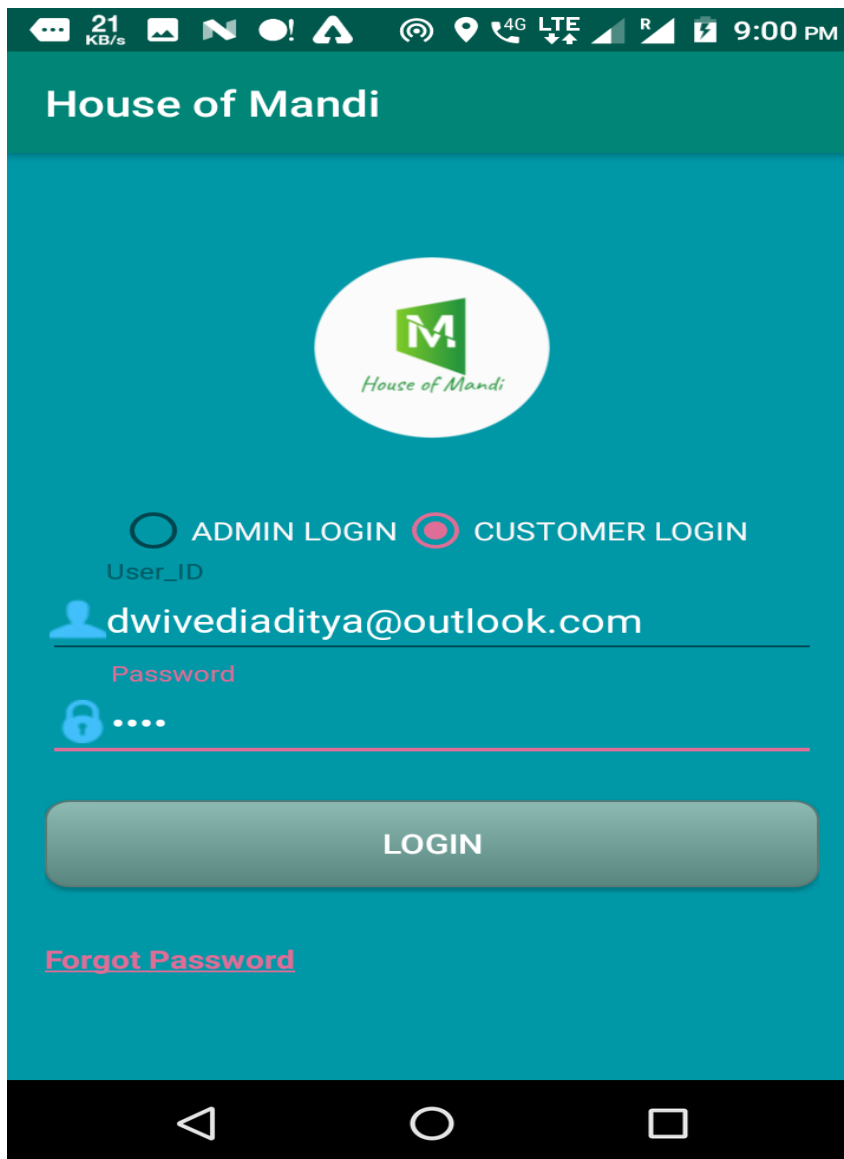
admin

Password

...

LOGIN

[Forgot Password](#)


**d. Login Page: Customer**

The screenshot displays the 'House of Mandi' login interface on a mobile device. At the top, a dark green header bar contains the app's name. Below this, a teal background features the 'House of Mandi' logo, which consists of a green square with a white 'M' and the text 'House of Mandi' underneath. Two radio buttons are present: 'ADMIN LOGIN' (unselected) and 'CUSTOMER LOGIN' (selected, indicated by a red dot). Below the radio buttons, there are two input fields. The first is labeled 'User\_ID' and contains the email 'dwivediaditya@outlook.com'. The second is labeled 'Password' and shows four dots for masked input. A large, rounded 'LOGIN' button is positioned below the password field. At the bottom left, there is a link labeled 'Forgot Password'. The entire interface is framed by a black status bar at the top showing system icons and a time of 9:00 PM, and a black navigation bar at the bottom with standard Android icons.


House of Mandi

☐ ADMIN LOGIN ☒ CUSTOMER LOGIN

User\_ID

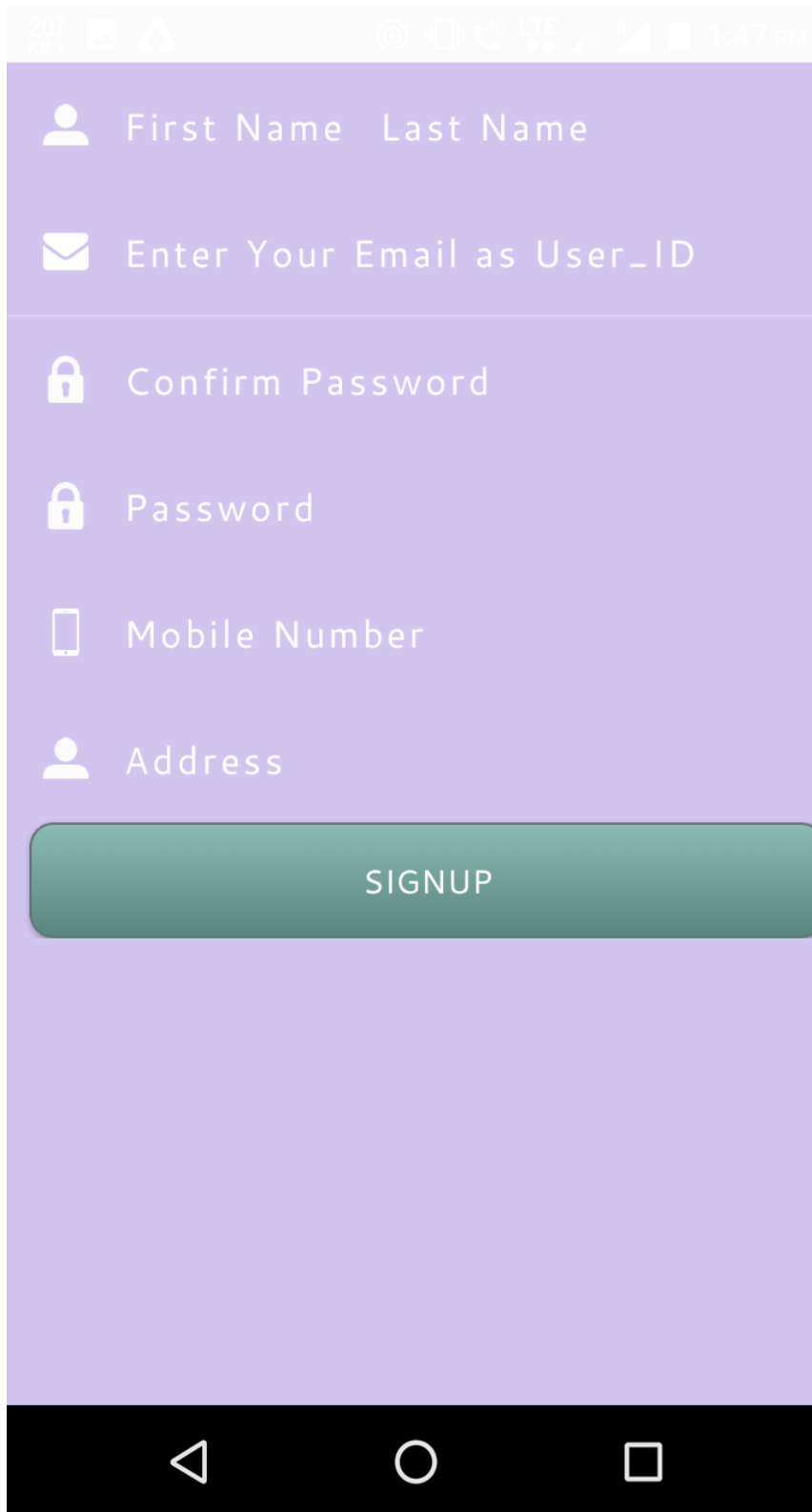
 dwivediaditya@outlook.com

Password

 ....

LOGIN

[Forgot Password](#)

**e. Customer Registration-Page:**

A mobile application interface for customer registration. The background is a solid light purple color. At the top, there is a status bar with icons for signal strength, LTE, battery, and time (1:47 PM). Below the status bar, the registration form consists of several input fields, each with a white icon on the left and a light purple placeholder text on the right. The fields are: 1. First Name (person icon), Last Name (person icon). 2. Enter Your Email as User\_ID (envelope icon). 3. Confirm Password (lock icon). 4. Password (lock icon). 5. Mobile Number (mobile phone icon). 6. Address (person icon). Below the input fields is a large, rounded rectangular button with a green-to-blue gradient and the text 'SIGNUP' in white capital letters. At the bottom of the screen is a black navigation bar with three white icons: a back arrow, a circle, and a square.

207 KB/s

First Name Last Name

Enter Your Email as User\_ID

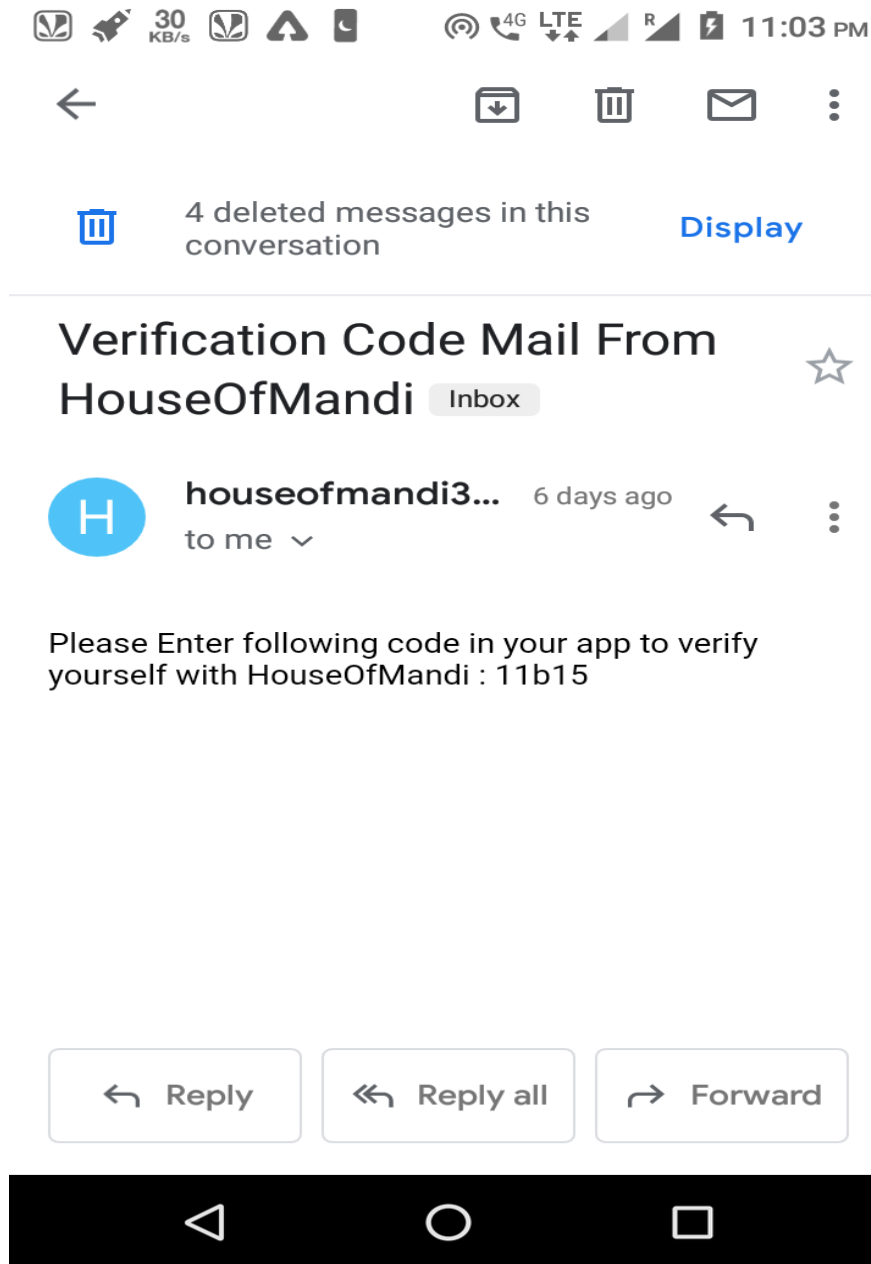
Confirm Password

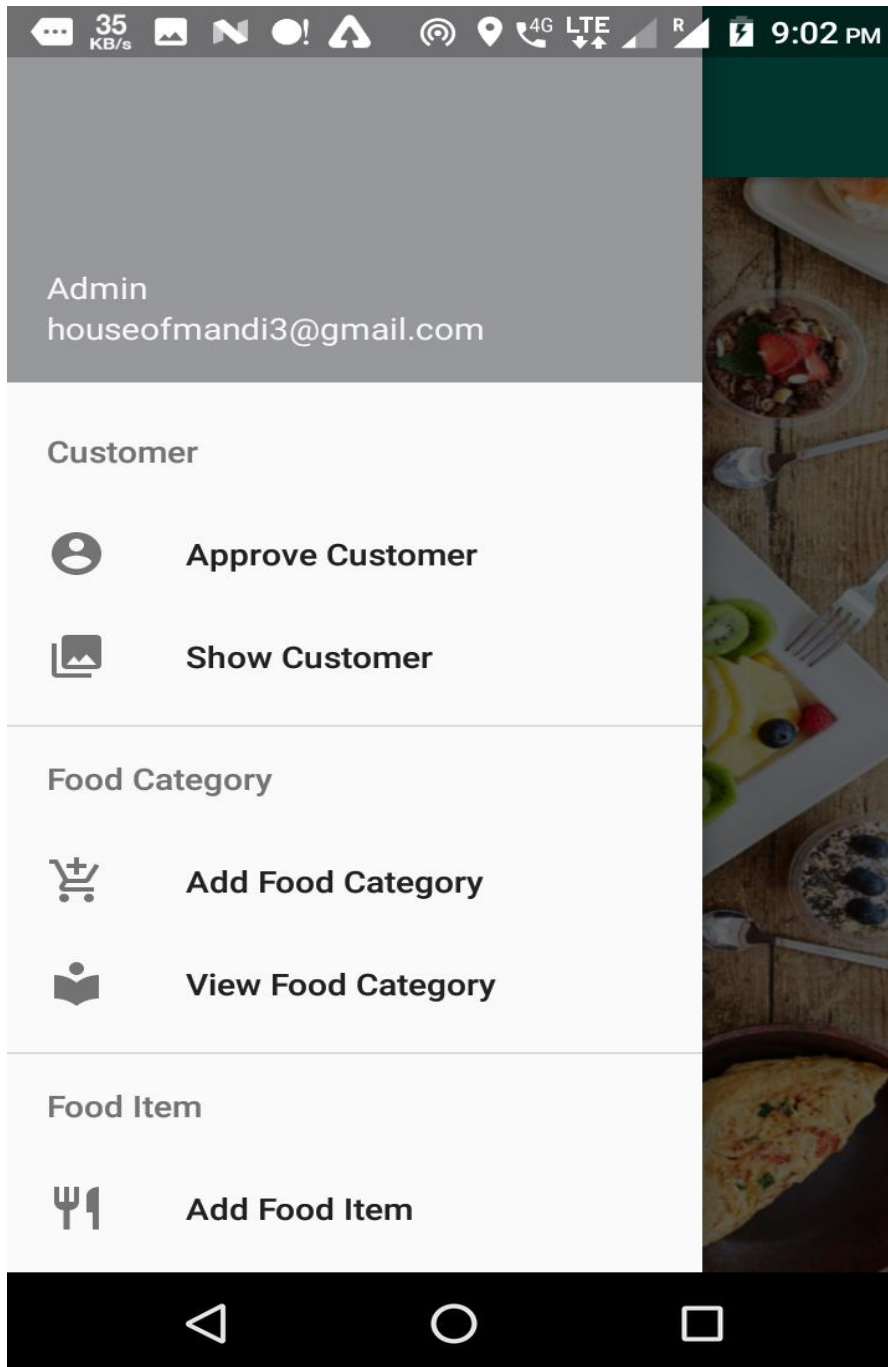
Password

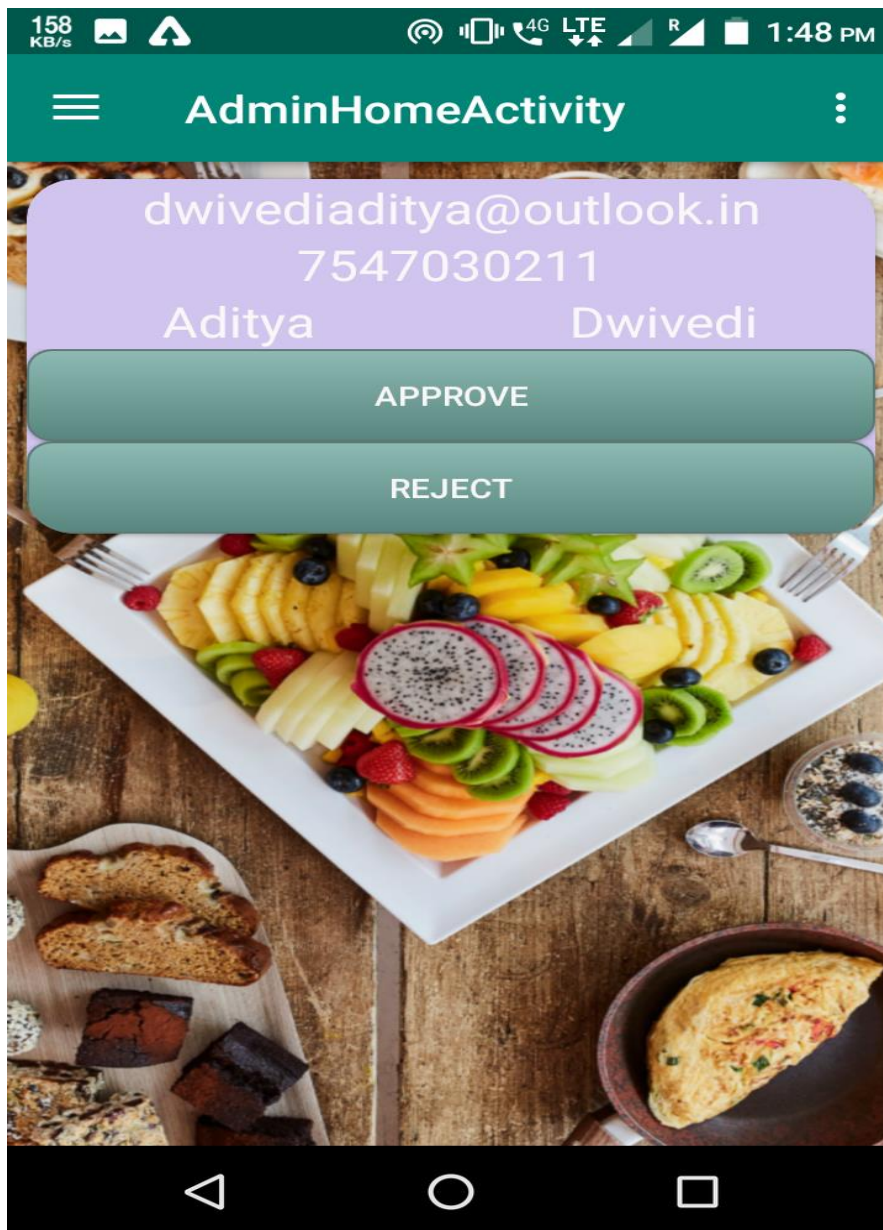
Mobile Number

Address

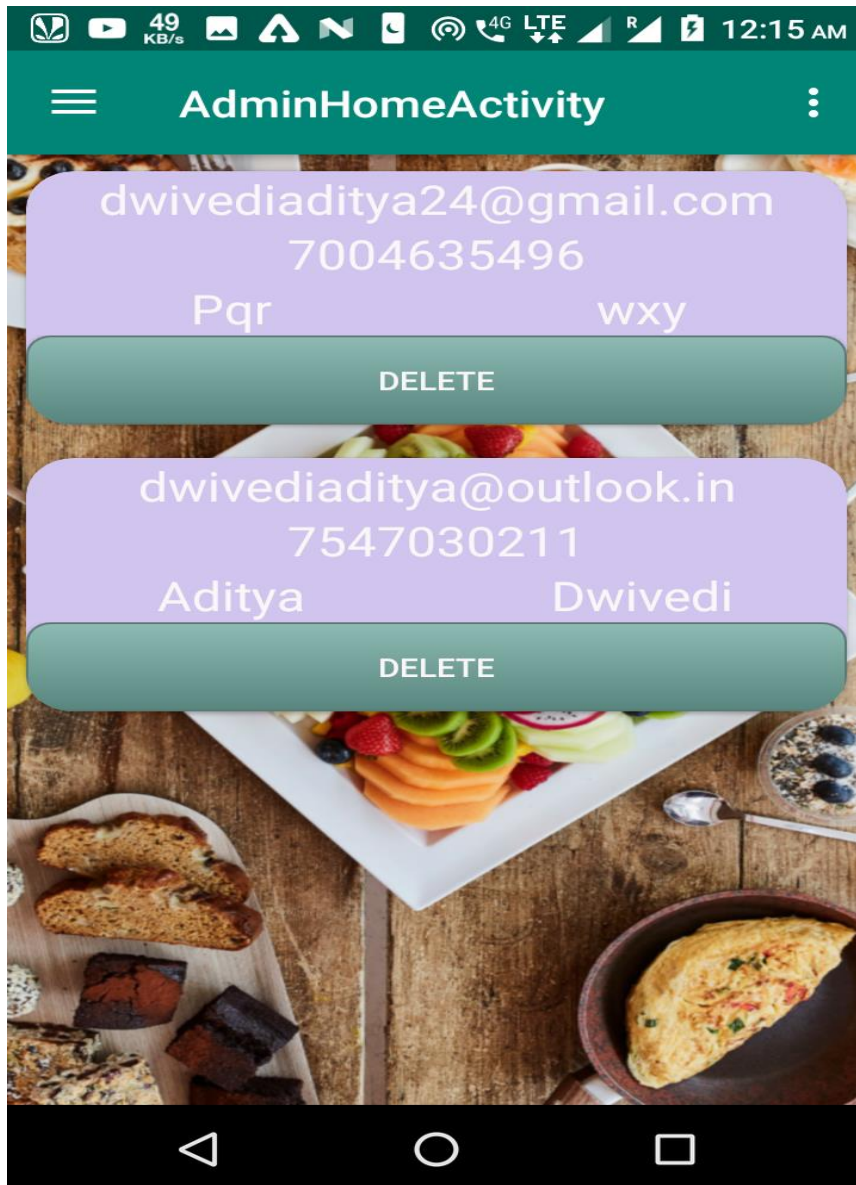
SIGNUP

**f. Verification via E-mail:**

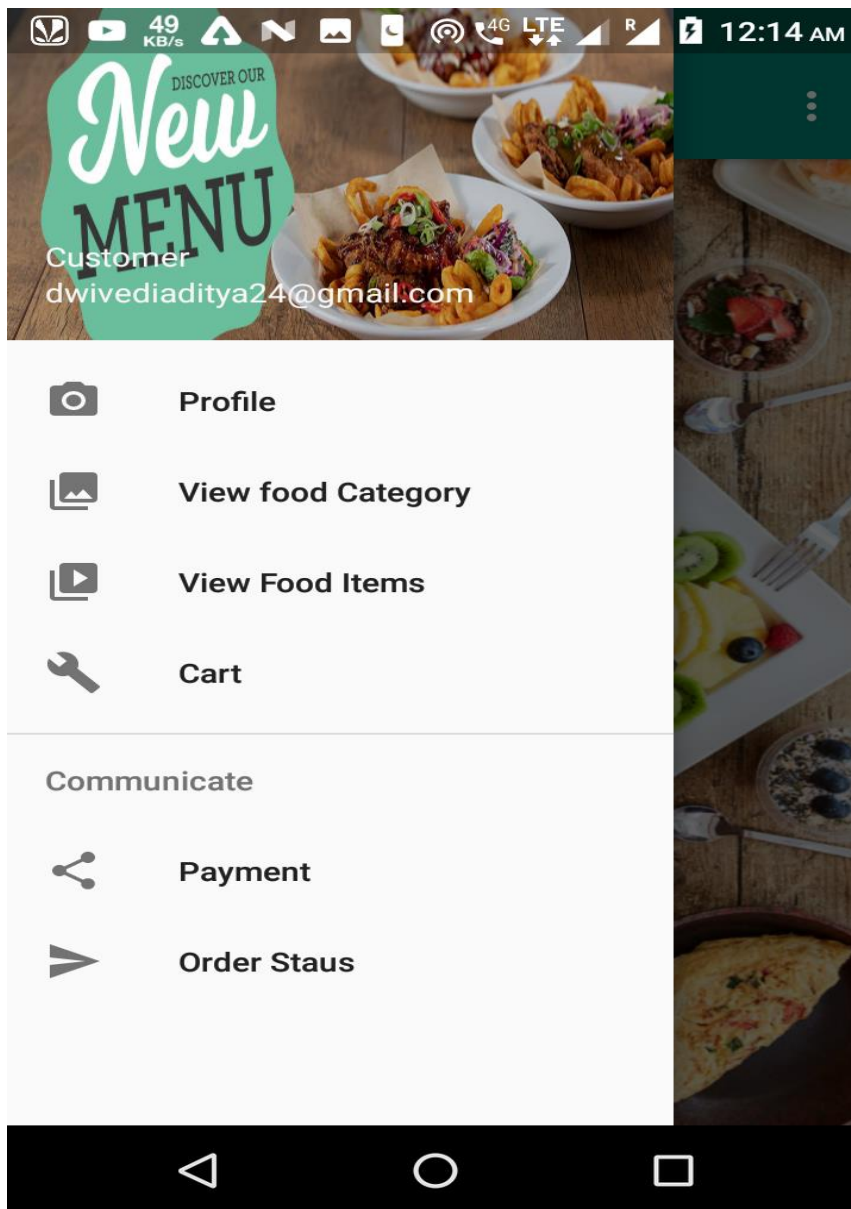
**g. Admin-Home:**

**h. Approve Customer:**

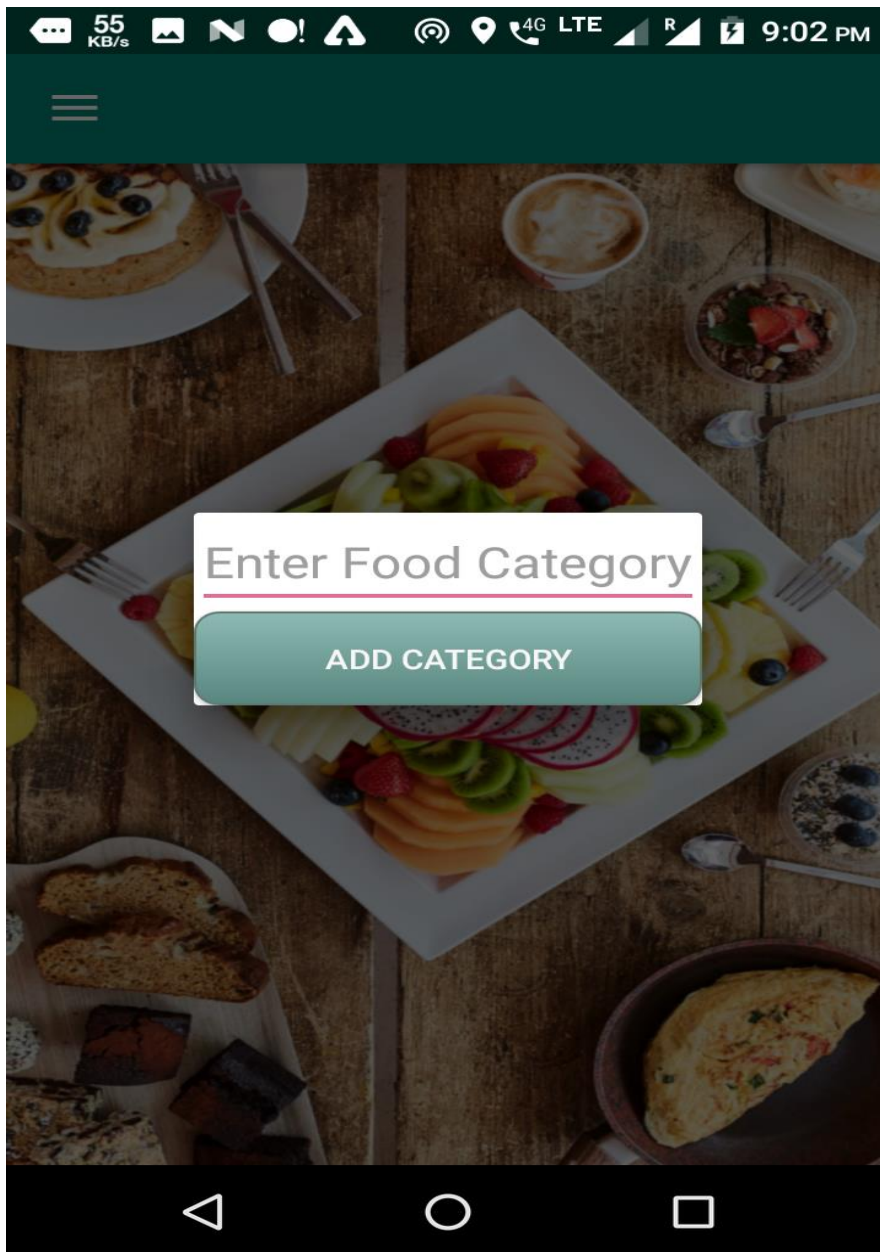


**i. View and Delete Customer:**

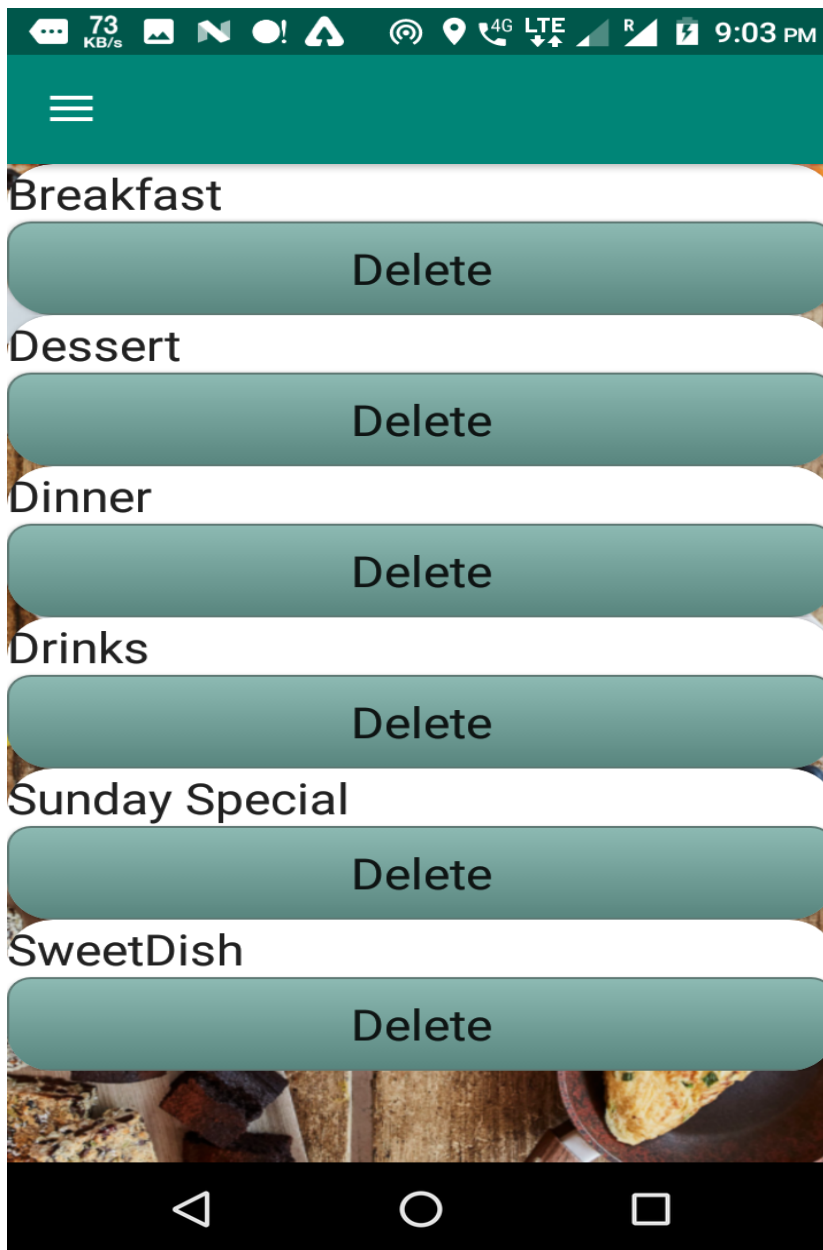
j. **Customer-Panel:**

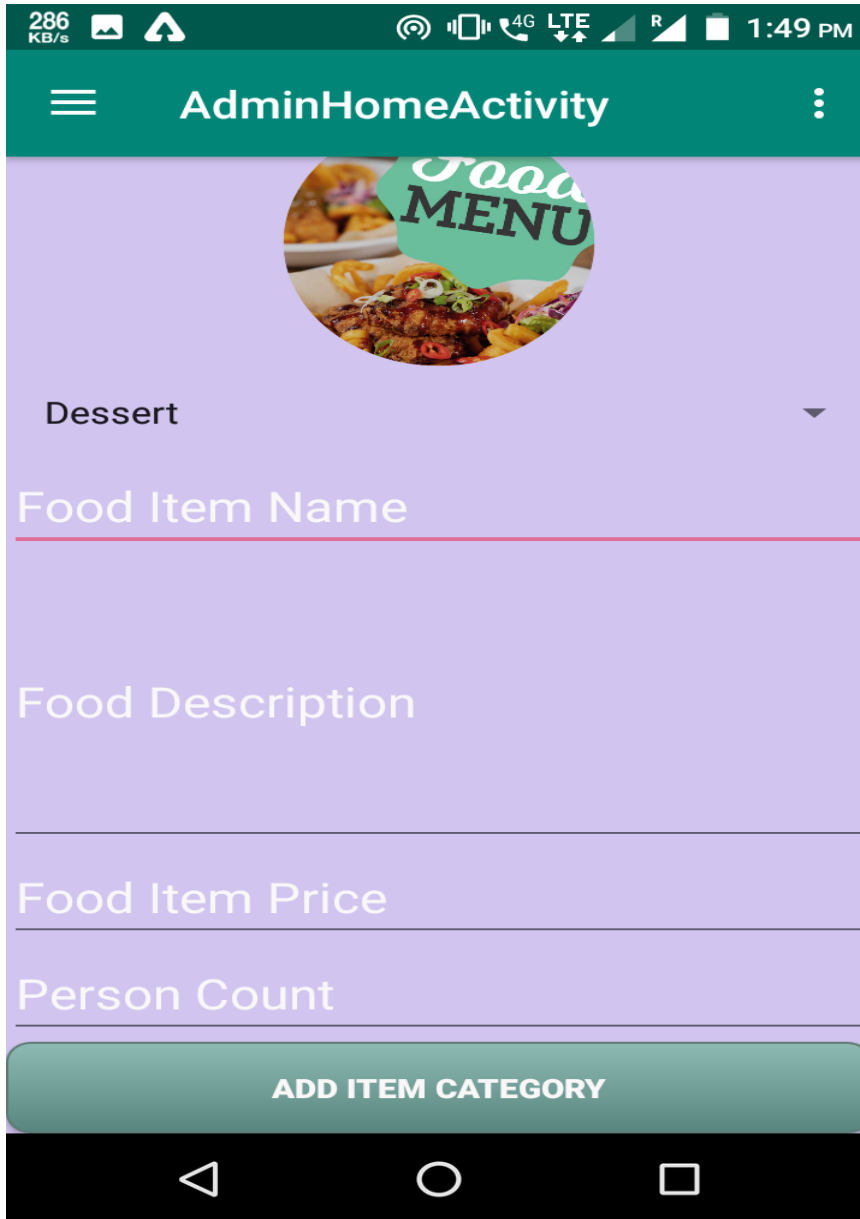


**k. Add Food Category:**



## 1. View Food Category:



**m. Add Food Item:**

The screenshot shows the 'AdminHomeActivity' screen of an Android application. At the top, there is a green header bar with a hamburger menu icon on the left, the text 'AdminHomeActivity' in the center, and a three-dot menu icon on the right. Below the header, there is a circular image of a food dish with a green overlay that says 'Food MENU'. Underneath the image, the word 'Dessert' is displayed with a downward-pointing triangle to its right. Below this, there are four text input fields with the following labels: 'Food Item Name', 'Food Description', 'Food Item Price', and 'Person Count'. At the bottom of the form is a green button with the text 'ADD ITEM CATEGORY'. The entire form is set against a light purple background. The top status bar shows various icons including signal strength, 4G LTE, and battery level, along with the time '1:49 PM'.

286 KB/s

AdminHomeActivity

Food MENU

Dessert ▼

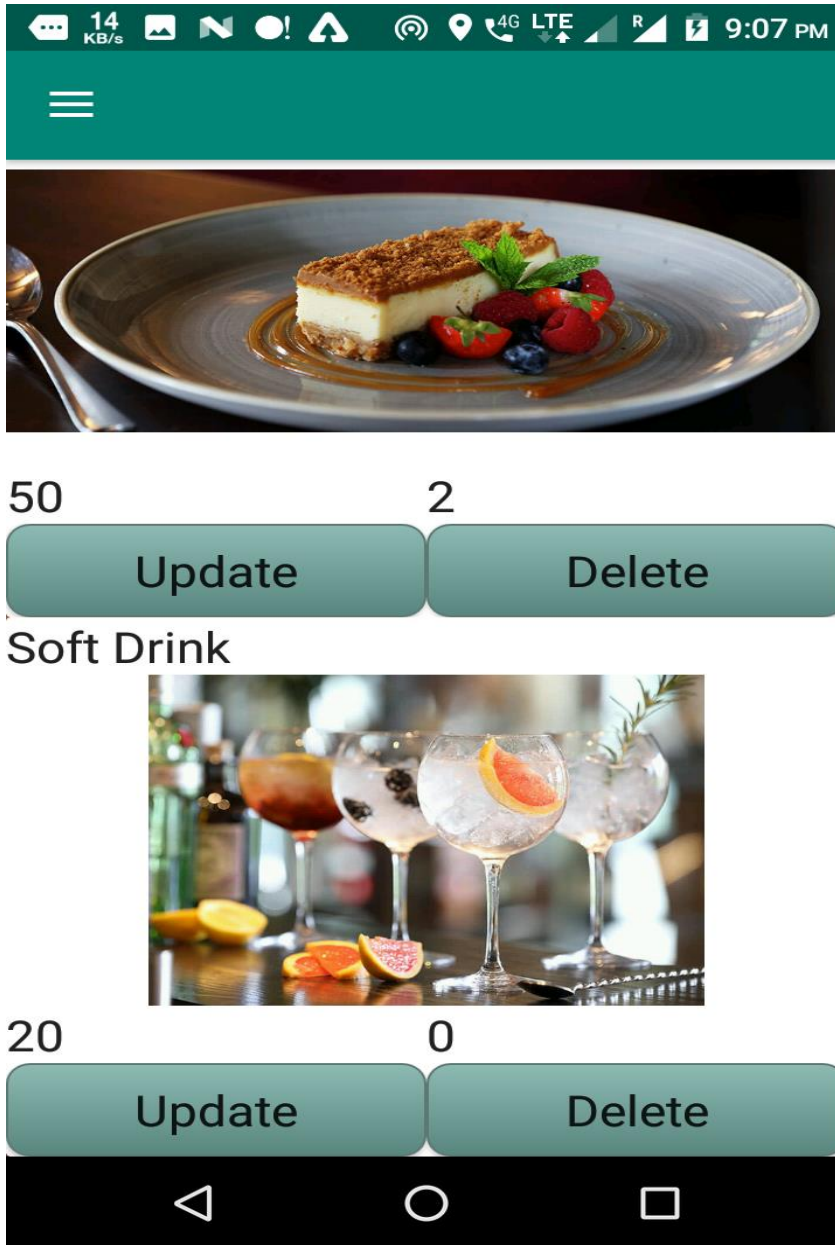
Food Item Name

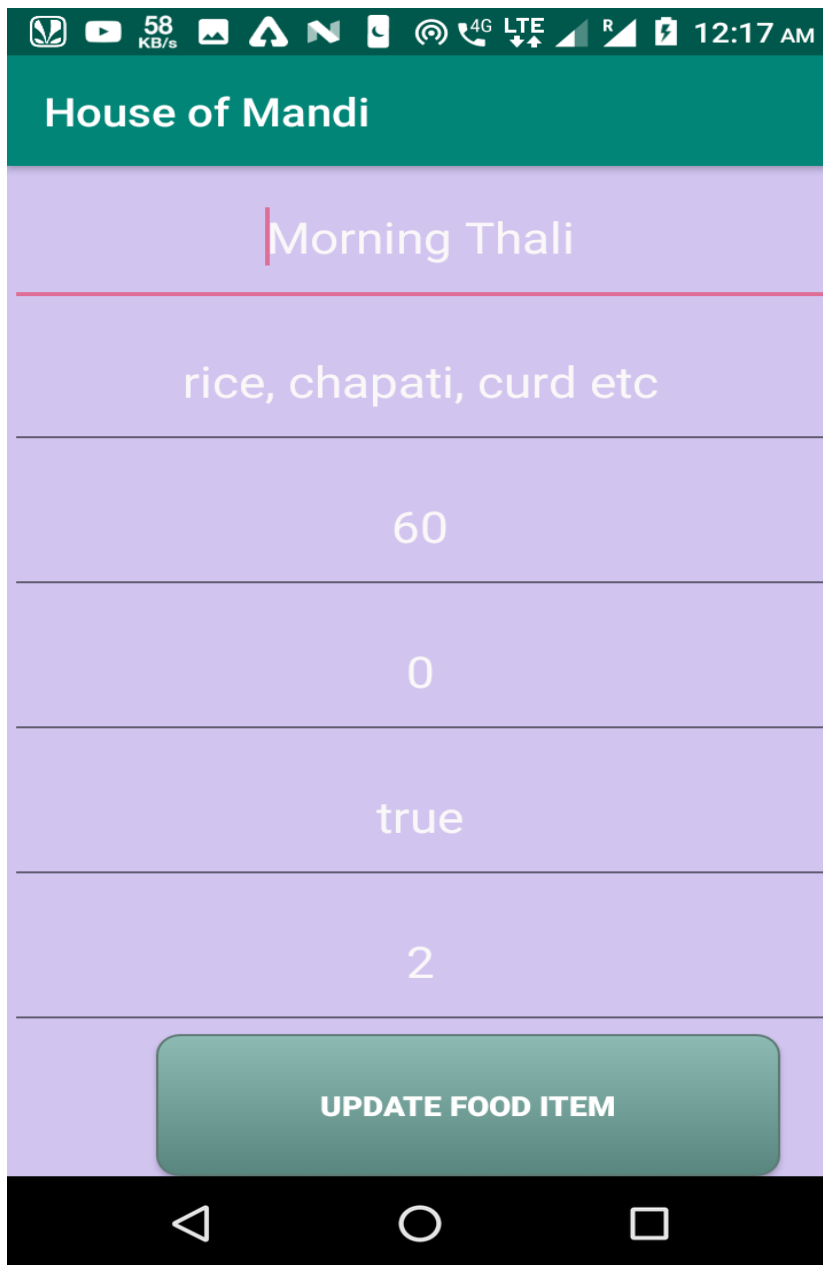
Food Description

Food Item Price

Person Count

ADD ITEM CATEGORY

**n. View Food item:**

**o. Update Food item:**

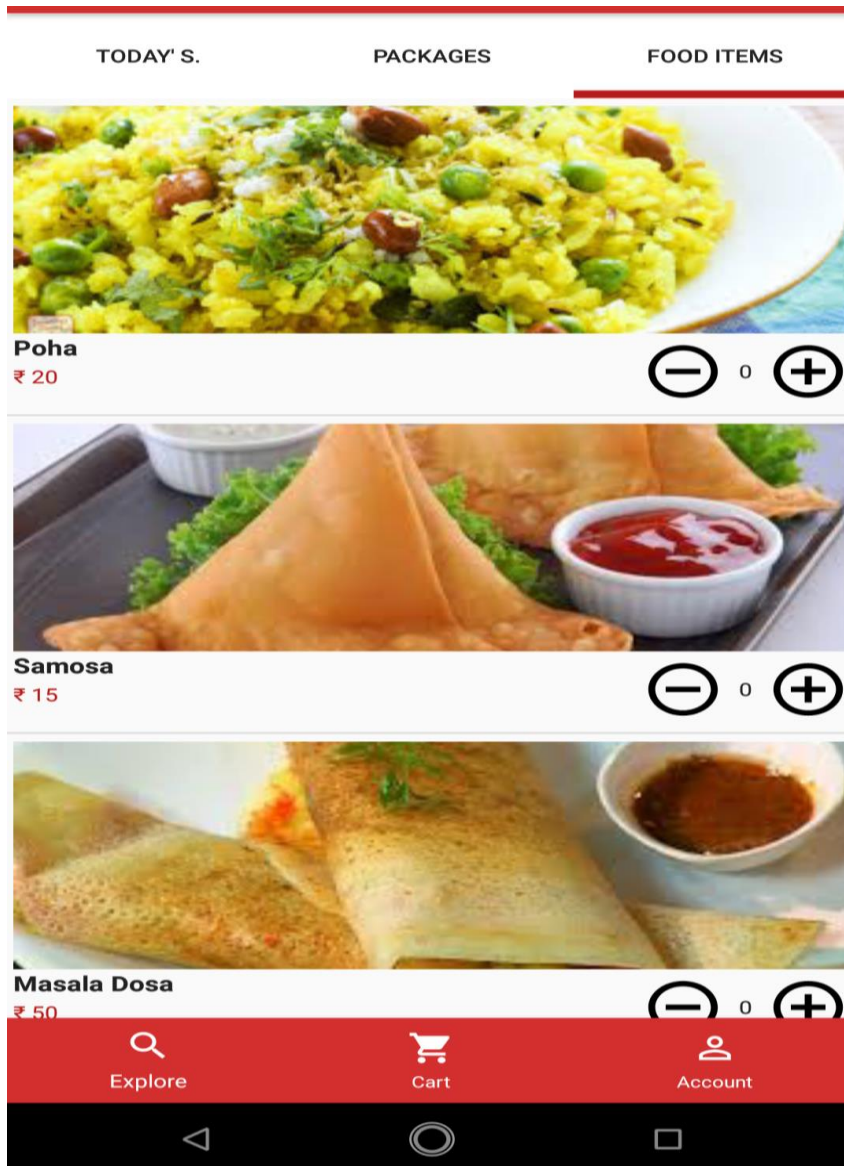
The screenshot shows a mobile application interface for 'House of Mandi'. At the top is a green header with the app name. Below it is a form with a light purple background. The form contains several input fields: a text field with 'Morning Thali', a text field with 'rice, chapati, curd etc', a numeric field with '60', a numeric field with '0', a boolean field with 'true', and a numeric field with '2'. At the bottom of the form is a green button labeled 'UPDATE FOOD ITEM'. The top of the screen shows a status bar with various icons and the time '12:17 AM'. The bottom of the screen shows the Android navigation bar.

Field	Value
Food Item Name	Morning Thali
Description	rice, chapati, curd etc
Price	60
Quantity	0
Available	true
Category	2

**UPDATE FOOD ITEM**



p. View Food and Add to cart:





## 14. Limitations & Future Enhancements

### a. Limitation

- Only admin approval customer can order for food .Every person cannot order food only prime customer can order
- Payment make only by the cash on delivery.
- Customer register by email or phone number. Admin approval request to login

### b. Future Enhancement

- Any user can create ,view edit and maintain information from mobile app or dashboard
- Any user can make order and live delivery tracking for customer order.
- Payment are available on Paytm, Net Banking, Debit Card and Cash on pay.

## 15. Conclusion

House of mandi project developed by using android, java as front end and jsp, MySQL server for database in back end to computerize the process for check in and out system. The project covers only the basic required to sum up, developing a information system on “House Of Mandi” for was a matter of essence.

## 16. Bibliography

1. Android mother site @ [www.android.com](http://www.android.com).
2. Android Developers @ [developer.android.com](http://developer.android.com).
3. Android API Documentation  
@ <http://developer.android.com/reference/packages.html>.
4. Professional Android 4 Application Development by Reto Meier
5. Programming Android Java Programming for the New Generation of Mobile  
Devices by Zigurd Mennieks
6. <https://stackoverflow.com/questions/5724758/android-stackoverflow-error>
7. <https://stackoverflow.com/users/878126/android-developer>
8. <https://www.androidhive.info/2016/01/android-working/>
9. <https://tips.androidhive.info/>