CHAPTER 1

INTRODUCTION

1.1 Introduction:

In the recent times, we have noticed how the mobile technology has shown immense growth. It has made its way into every aspect of our lives. We have seen laptops and personal organizers evolve into all-in-one smart phones, cloud computing and virtual private networks allowing accessing any information you want, wherever you are. Landline phones are beginning to slip into memory, and mobile communications and technology are on the rise, expanding and growing rapidly all over the globe. With the introduction of intelligent and smart phones, the handset industry has changed from low budget handset devices to high performance mobile devices. Today's phone is almost everything - it is fashionable, innovative, appealing, high-performing, durable, stylish and multi-tasking. It can serve many purposes such as email, blogging, browsing internet, games, multimedia, access to social networking sites like YouTube, Google search, Facebook and more. The worldwide demand for smart phones has also fueled the business market [1].

Smart phones show higher application usage than feature phones even at the basic built-in application level. The percentage of people who use their phone for only voice communications has been dropping with more people interested in smart phone and what more it offers. The use of the built-in camera, video capability and Wi-Fi features also add to the popularity of the smart phones.

Mobile phone applications, so-called apps, have experienced dramatic growth globally with people today looking for more information on the move. This is one area of mobile phone technology enhancement that allows developers and programmers to offer users just what they seek under their preferred area of interest. Google's Android is one of the latest and unique innovations, which instantly has taken over the mobile market. It is an open source mobile platform which allows developers from around the world to develop applications for Android supported mobile devices. Android supports to develop a location aware application utilizing Global Positioning System (GPS) and Android's Network Location Provider (NLP) to acquire the user location. Although GPS is most accurate, it only 2

works outdoors; it quickly consumes battery power, and doesn't return the location as quickly as users want [2]. Android's Network Location Provider determines user location using cell tower and Wi-Fi signals, providing location information which works indoors and outdoors, responds faster, and uses less battery power. To obtain the user location in the application, both GPS and the Network Location Provider can be used or just one.

1.2 Objectives:

To develop an Automation System By Android and Web which provides:

- Create Sales Representative, Outlet, SKU
- Take Order From Outlet
- Order Distance from Outlet
- Order Date and Time
- Using the simple and cheaper android based platform
- And use PHP on Web And Mobile API

CHAPTER 2

THEORY

2.1 Android Platform:

Smart phones today have made our life easier with millions of people around the world who use their phones to manage their lives, stay in touch with friends, and use the internet for a lot of things. Users of smart phones have come to take it for granted that they can surf the internet from almost any location and do everything they could do with a computer. When mobile phones were first introduced to the public, they were bulky, expensive, and some even required a base unit that had to be transported along with the phone. Good reception was a major problem and in general, early mobile phones could only be used in certain locations[5] were the signal was particularly strong. As mobile phone technology advanced, the difficult in using them became less of a problem. Today, mobile phone reception has improved greatly due to the use of satellites and wireless services. As mobile phones evolved and became simple to use, the importance of them increased accordingly.

It was only a matter of time before Google started to dominate the smart phone arena, and they have, their Android which is now among the popular smart phone operating system on the market. The Android is a smart phone operating system created by Google. The reason for the popularity of the Android is all of the apps available. One of the great advantages of Android is its capacity to run several applications simultaneously. Android gives users the freedom to listen to music, receive notifications etc. without keeping application open. At the same times users can download their favorite apps from the Android market. Global Positioning System, popular and commonly known as GPS is a space-based global navigation satellite system (GNSS) that provides location and time information in all weather, anywhere on or near the Earth, where there is an unobstructed line of sight to four or more GPS satellites[5]. It is maintained by the United States government and is freely accessible by anyone with a GPS receiver[5]. The U.S. military developed and implemented this satellite network as a military navigation system, but soon opened it up to everybody else.

Thanks to the development of GPS technology, an advanced navigation technology-real time GPS service becomes true. It's been fast and wildly adopted in many fields and is becoming to play a crucial role in our modern life, especially in security and safety field. GPS on smart phones is no longer an emerging trend. It's almost a must-have feature nowadays, and more and more handsets are offering it. With the embedded GPS receiver and a mapping service, one can get real-time position tracking, text- and voice-guided directions, and points of interest. The latest location-based services are also expected to boost demand in the GPS handsets industry. A collaborative effort among telecom carriers, mobile phone makers and GPS[5] chip providers has expanded location-based services (LBS) applications and increased demand for models that support these functions.

2.2 Manifest:

Every Android application must have a single manifest[3] file in a root directory (named AndroidManifest.xml[3]). This file contains essential information about the application which is required by the system to run it. All new components have to be registered there along with permissions needed to perform actions requested by the application. The file is also used to inform the system if a developer wishes to expose the application's data to other apps. It specifies the application's reaction to various system events such as incoming calls or photo capturing. Within the Manifest file developers can specify if they want one of their activities (described below) to become a launch activity. For the purpose of this paper applications which contain a launch activity will be called launch applications. Such applications are added to the Android's menu screen so they can be launched independently from other apps

2.3 Main application components:

All Android applications contain zero or more of the following components:

- Activity – a single piece of user interface (UI) [15]. In a system it runs independently from other elements although users have a smooth experience of different activities appearing on the screen in a sequence. Activities usually contain some graphical elements (customized or taken from a pre-defined set) and constitute a specific action that users can perform like writing an email or selecting contact from a list.

- Service – a separate part of the application without a graphical representation (running in a background). Service itself does not start a separate thread because it runs in context of the activity or the broadcast receiver which has started it. However, it usually should use a working thread to perform some complex computation or lengthy I/O operations.

- Content Provider – a mechanism that allows applications to share data between each other or to simply persistently save some information. It is basically an interface which provides standard methods to access data like query, insert, update, delete. No specific type of data structure is imposed by the system, it can be a single file or an SQLite database[12]. Content provider is uniquely identified by its authority and can contain many types of data objects (many tables in a database). The most common method of accessing an object is to query Content Provider with a specific content uri which has a generic form of: content://<authority>/..<type path>../<id>/

- Broadcast Receiver – a part of the application which responds to actions broadcasted by the system itself (dimming the screen, capturing a new photo) or by other parts of the application (the same one or not). Broadcast Receiver does not have its own graphical representation but it can initiate certain actions to keep users informed about its work.

This could be done for instance by creating a notification in a status bar or updating a desktop widget. As it was mentioned previously Android, is an event-driven system. These events are called Intents. Intent API [16] is a very powerful mechanism that manages interactions between application components across the whole system. Developers do not have to write any additional lines of code in order to integrate their application with other ones as long as they know their set of supported Intents. Moreover, it does not make any difference whether you send an Intent to a separate application or another part of the same one – the mechanism is exactly the same. A good example is the Map application which supports Intents that contain a request to display specific geo location. Developers can simply broadcast this Intent whenever they want to show a map with a specific location. Additionally, should users have installed a different app which masks the functionality of the Map application (by filtering Intents), this new application would be used automatically instead. The Intent mechanism allows developers to easily reuse different system components across the platform. They can focus on really innovative and unique functionality and simply add ('attach') new components like maps or navigation to their applications which makes them even more compelling and useful

2.4 Process handling:

One of the key issues when designing an operating system for mobile devices is memory management. Android is trying to do as much as possible in the background without bothering anybody, but at the same time there is remarkable space for alterations and modifications. Therefore, developers should at least be aware of the most basic aspects of process handling not to create applications that would be ’arrogant’ or even ’hostile’ to the system.

Very often Android has to deal with situations of memory shortage. The system has to reclaim resources in order to allow new processes to run. In such situations Android ranks all active processes[4] according to the following order:

1. Foreground process – the process which hosts a foreground activity or a service that is bound to a foreground activity. It basically means the part of the system that the user is currently interacting with, so at any given time there are only few such processes. These processes are killed only in absolute critical situations as not terminating them will most likely cause lack of responsiveness (or even displaying error messages to the user).

2. Visible process – a process that does not have any active components, but still is visible to the user. A good example is a process which hosts an activity that launched another not-full-screen activity. Visible processes are considered important and will only get killed if the system cannot find enough memory for foreground processes.

3. Service process – a process which basically hosts a service and at the moment it is neither a foreground nor a visible process. Those processes however may be doing some important tasks for the, user like playing music in a background, or downloading some data from the Internet.

4. Background process – a process which handles an activity that is currently not visible. These processes are likely to get killed at any given time so activities should be prepared for that.

5. Empty process – a process without any of the application’s components. The only reason why the system holds these processes alive is for caching purposes.

Developers should constantly be aware of the fact that their processes could get killed and reestablished by the system in the background even if they did not expect such a situation to happen in the first place. To allow a smooth navigation between different screens, Android introduces a number of helping methods to save and restore the state of activities. It is good practice not only to use these methods whenever there is a need to, but also to do it wisely. For instance, no lengthy operations should ever be performed within them. A good example is an activity where users can edit a new text message being interrupted by an incoming call. The screen is captured by the new activity, decreasing the priority of the old one. Therefore, the process which handles the message-editing activity is more likely to get killed. Users would probably expect to be able to continue editing the text of the message after finishing the call. However, this would be impossible if the text had not been persistently saved beforehand.

2.5 Platform Architecture:

Android is an open source, Linux-based software stack created for a wide array of devices and form factors. The following diagram shows the major components of the Android platform[4].

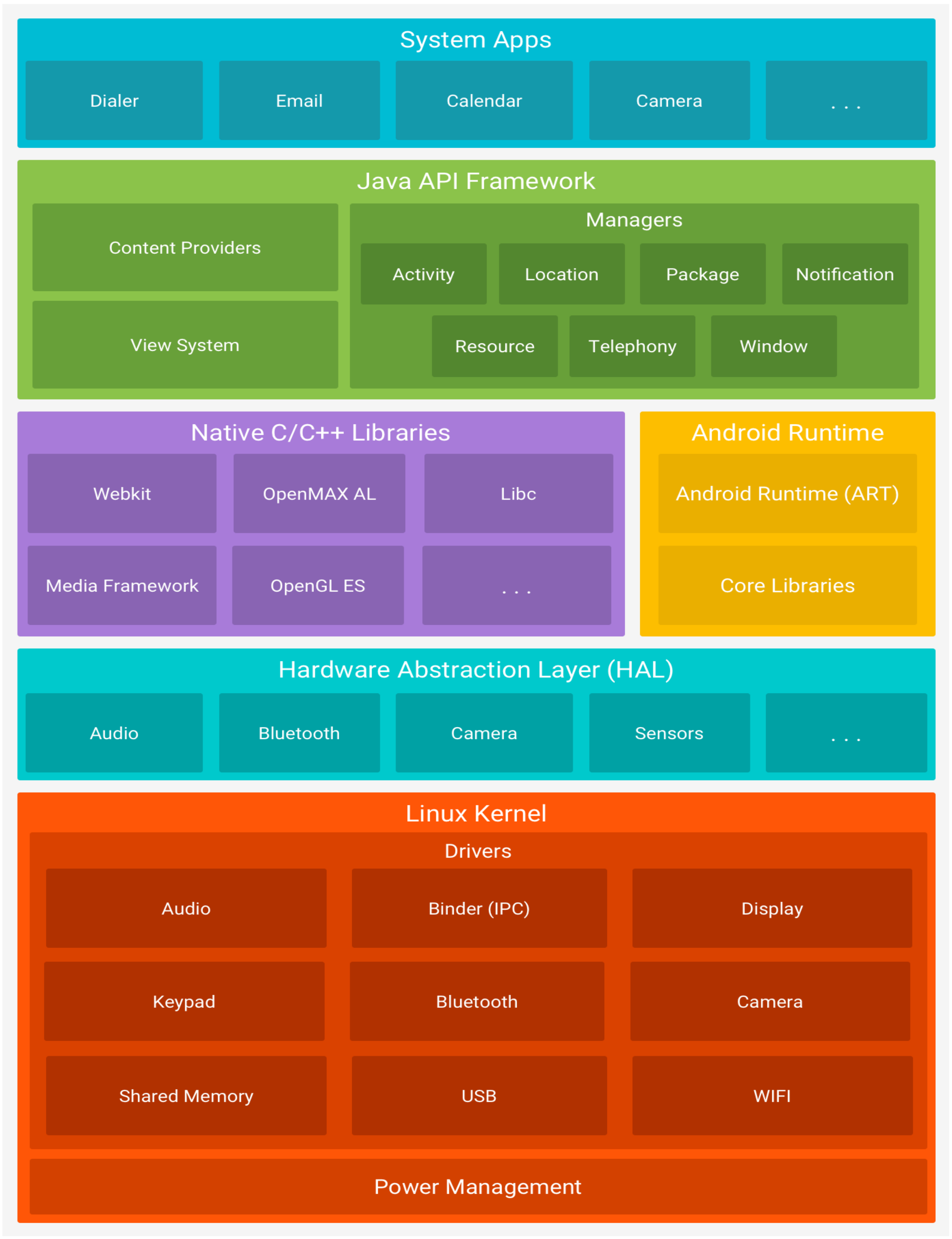


Fig:2.5.1 Platform Architecture

- The Linux Kernel: The foundation of the Android platform is the Linux kernel. For example, the Android Runtime (ART) relies on the Linux kernel for underlying functionalities such as threading and low-level memory management. Using a Linux kernel allows Android to take advantage of key security features and allows device manufacturers to develop hardware drivers for a well-known kernel[4].

- Hardware Abstraction Layer (HAL): The hardware abstraction layer (HAL) provides standard interfaces that expose device hardware capabilities to the higher-level Java API framework. The HAL consists of multiple library modules, each of which implements an interface for a specific type of hardware component, such as the camera or Bluetooth module. When a framework API makes a call to access device hardware, the Android system loads the library module for that hardware component[4].

- Android Runtime: For devices running Android version 5.0 (API level 21) or higher, each app runs in its own process and with its own instance of the Android Runtime (ART). ART is written to run multiple virtual machines on low-memory devices by executing DEX files, a bytecode format designed specially for Android that's optimized for minimal memory footprint. Build toolchains, such as Jack, compile Java sources into DEX bytecode, which can run on the Android platform[4].

- Some of the major features of ART include the following:

    1. Ahead-of-time (AOT) and just-in-time (JIT) compilation
    2. Optimized garbage collection (GC)
    3. Better debugging support, including a dedicated sampling profiler, detailed diagnostic exceptions and crash reporting, and the ability to set watchpoints to monitor specific fields

  Prior to Android version 5.0 (API level 21), Dalvik was the Android runtime. If your app runs well on ART, then it should work on Dalvik as well, but the reverse may not be true.

  Android also includes a set of core runtime libraries that provide most of the functionality of the Java programming language, including some Java 8 language features, that the Java API framework uses.

- Native C/C++ Libraries: Many core Android system components and services, such as ART and HAL, are built from native code that require native libraries written in C and C++. The Android platform provides Java framework APIs to expose the functionality of some of these native libraries to apps. For example, you can access OpenGL ES through the Android framework’s Java OpenGL API to add support for drawing and manipulating 2D and 3D graphics in your app[4].

  If you are developing an app that requires C or C++ code, you can use the Android NDK to access some of these native platform libraries directly from your native code.

- Java API Framework: The entire feature-set of the Android OS is available to you through APIs written in the Java language. These APIs form the building blocks you need to create Android apps by simplifying the reuse of core, modular system components and services, which include the following:

    1. A rich and extensible View System you can use to build an app’s UI, including lists, grids, text boxes, buttons, and even an embeddable web browser
    2. A Resource Manager, providing access to non-code resources such as localized strings, graphics, and layout files
    3. A Notification Manager that enables all apps to display custom alerts in the status bar
    4. An Activity Manager that manages the lifecycle of apps and provides a common navigation back stack
    5. Content Providers that enable apps to access data from other apps, such as the Contacts app, or to share their own data

    Developers have full access to the same framework APIs that Android system apps use.

- System Apps: Android comes with a set of core apps for email, SMS messaging, calendars, internet browsing, contacts, and more. Apps included with the platform have no special status among the apps the user chooses to install. So a third-party app can become the user's default web browser, SMS messenger, or even the default keyboard (some exceptions apply, such as the system's Settings app).

    The system apps function both as apps for users and to provide key capabilities that developers can access from their own app. For example, if your app would like to deliver an SMS message, you don't need to build that functionality yourself—you can instead invoke whichever SMS app is already installed to deliver a message to the recipient you specify.

2.6 Location Manager:

The most enticing of the Android features are the Location Based Services that give the application geographical context through Location Providers (GPS etc.)[5]. Most Android devices allow determining the current geo-location. This can be done via a GPS device, via cell tower triangulation or via Wi-Fi networks for which the geo-location[5] is known.

Android provides the package "android.location"[5] which provides the API to determine the current geo position. The class "LocationManager" provides access to the location service. To call the location manager, the context is set to getSystemService () for location services. The Android device might have several providers available and a "Criteria" object can be used to select the “best” among them. Then a "LocationListener" is registered with the "LocationManager" which receives periodic updates about the geo position. The class "LocationProvider" is the superclass of the different location providerswhich deliver the information about the current location. It is important to register with the location manager to receive the location update which is done using the onResume () method and can be unregistered from location notifications in the onPause () method [9]. Tracking the location can be expensive on the battery and CPU of the device, so this is a good practice to stop doing the work while the application is paused would not require any updates.

The LocationListener which is implemented allows getting certain callbacks. Those callbacks include location, provider and status changes. Besides all this, appropriate permissions need to be added to the AndroidManifest.xml file to notify the Android system

2.7 Data Storage:

The storage system in Android is mainly divided into three types: Shared Preferences, Files and SQLite Databases [10]. SmartSales makes use of SQLite Databases[12] to save the locations and their geo-points. The application as such is stored in Android system uses internal storage of the device. By default, the application once installed is stored in the internal storage of the Android system[4]. This is private to the application and other applications cannot access it. When the user uninstalls the application, these files get removed.

By default all files, databases and preferences is restricted to the application that created them. If the data has to be shared with other applications then the Content Providers is used. Content Providers is not a storage mechanism but provide well defined interface for sharing private data.

Shared preferences store data in terms of key-value pairs and can be accessed by application components working in the same context.

Data can also be stored on local files directly using standard Java I/O classes and methods. These methods only support files in the current application folder; specifying path separators will cause an exception to be thrown.

Android also provides full support to SQLite databases[5]. All databases that are created in the application are accessible by name to any class in the application but none outside. This is implemented by creating a sub-class to SQLiteOpenHelper and overriding the onCreate() method to execute the SQLite command to create the tables in the databases. The methods getWritableDatabase() [11] and getReadableDatabase() [5] are called for write to and read from the database which return SQLiteDatabase [5] object.

The Android SDK includes a sqlite3 database tool which is required to browse the table contents, run SQL commands and perform other SQL functions. Objects that are used to insert new rows in the database are called contentValues [5]. Each contentValue represent a new row. Cursor is an object which is used for extracting over query results. It acts as a pointer to the result set from a database query. Cursor provides several functions navigate through your query results like moveToFirst, moveToNext, getCount, getColumnName etc.

All Android databases are stored in the data/data/<package_name>/databases folder on the device.

In this application, relational database is used to create three tables for storing the various categories of items, the list of items in each category and the tagged locations.

2.8 Sync Adapter:

Synchronizing data between an Android device and web servers can make your application significantly more useful and compelling for your users. For example, transferring data to a web server makes a useful backup, and transferring data from a server makes it available to the user even when the device is offline. In some cases, users may find it easier to enter and edit their data in a web interface and then have that data available on their device, or they may want to collect data over time and then upload it to a central storage area[9].

Although you can design your own system for doing data transfers in your app, you should consider using Android's sync adapter framework. This framework helps manage and automate data transfers, and coordinates synchronization operations across different apps. When you use this framework, you can take advantage of several features that aren't available to data transfer schemes you design yourself:

- Plug-in architecture: Allows you to add data transfer code to the system in the form of callable components.

- Automated execution: Allows you to automate data transfer based on a variety of criteria, including data changes, elapsed time, or time of day. In addition, the system adds transfers that are unable to run to a queue, and runs them when possible[9].

- Automated network checking: The system only runs your data transfer when the device has network connectivity[9].

- Improved battery performance: Allows you to centralize all of your app's data transfer tasks in one place, so that they all run at the same time. Your data transfer is also scheduled in conjunction with data transfers from other apps. These factors reduce the number of times the system has to switch on the network, which reduces battery usage.

- Account management and authentication: If your app requires user credentials or server login, you can optionally integrate account management and authentication into your data transfer.

This class shows you how to create a sync adapter and the bound Service that wraps it, how to provide the other components that help you plug the sync adapter into the framework, and how to run the sync adapter to run in various ways.

2.9 PHP Fundamentals:

PHP powers hundreds of millions of websites worldwide. Undoubtedly it is one of the most popular server side scripting language. If you want to start web programming it is one of the first language you should pick up. In this introductory course in PHP you will get all the fundamental concepts required to start your PHP web programming[14].

We have kept the learning curve fairly simple and anyone with basic knowledge of programming can quickly pick up PHP with the help of this course[14].

In the course you will go through basic web programming concepts and will be able to master PHP programming by the end of it.

2.9.1 PHP Mobile API:

Application program interface (API) is a set of routines, protocols, and tools for building software applications. An API specifies how software components should interact. Additionally, APIs are used when programming graphical user interface (GUI) components. A good API makes it easier to develop a program by providing all the building blocks. A programmer then puts the blocks together[16].

CHAPTER 3

MY PROJECT

3.1 CRUD Operation:

CRUD(Create , Read , Update , Delete) PHP operations , these are some of the basic things of PHP web application , Records are insert , select update and delete using PHP and MySQL, Creating a Simple Insert, Select, Update and Delete using PHP with MySQL Database is easy task. learning a crud operations is the first way to understand this is a simple and easy[14].

3.1.1 Sales Representative, Outlet, SKU:

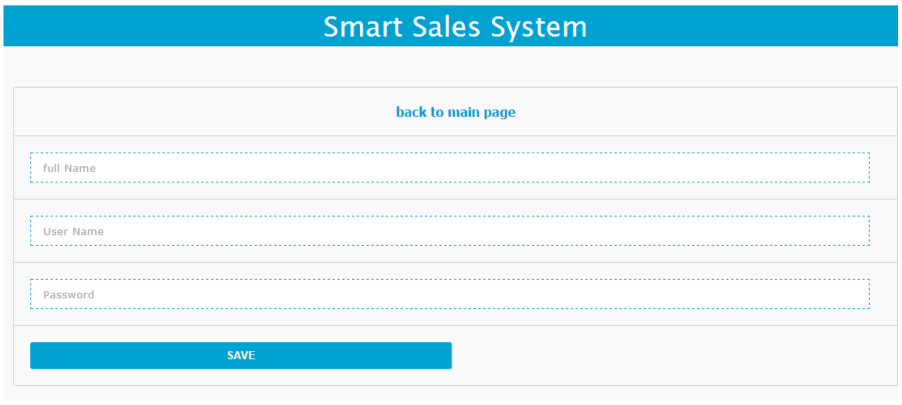
```
CREATE TABLE `tbl_user` (
  `id` int(11) NOT NULL AUTO_INCREMENT,
  `user_name` varchar(255) COLLATE utf8_unicode_ci NOT NULL,
  `password` varchar(255) COLLATE utf8_unicode_ci NOT NULL,
  `full_name` varchar(255) COLLATE utf8_unicode_ci NOT NULL,
  PRIMARY KEY (`id`);

CREATE TABLE `tbl_sku` (
  `id` int(11) NOT NULL AUTO_INCREMENT,
  `price` double NOT NULL,
  `name` varchar(255) COLLATE utf8_unicode_ci NOT NULL,
  `token` varchar(255) COLLATE utf8_unicode_ci NOT NULL,
  PRIMARY KEY (`id`);

CREATE TABLE `tbl_outlet` (
  `id` int(11) NOT NULL AUTO_INCREMENT,
  `sr_id` int(11) NOT NULL,
  `name` varchar(255) COLLATE utf8_unicode_ci NOT NULL,
  `address` varchar(255) COLLATE utf8_unicode_ci NOT NULL,
  `owner` varchar(255) COLLATE utf8_unicode_ci NOT NULL,
  `mobile` varchar(255) COLLATE utf8_unicode_ci NOT NULL,
  `lat` varchar(50) COLLATE utf8_unicode_ci NOT NULL,
  `lon` varchar(50) COLLATE utf8_unicode_ci NOT NULL,
  `verified` tinyint(4) NOT NULL,
  `token` varchar(100) COLLATE utf8_unicode_ci NOT NULL,
  PRIMARY KEY (`id`)  PRIMARY KEY (`id`);
```

dbconfig.php

```
<?php
$host = "localhost";
$user = "";
$password = "";
$database = "";
mysql_connect($host,$user,$password);
mysql_select_db($database);
?>
```

Create : Insert New Records

For providing user interface for the database insert, we have to create a form, containing all the fields the `tbl_user` table has, except id which is auto generated numeric field. And, HTML form view will be as follows.



Smart Sales System

back to main page

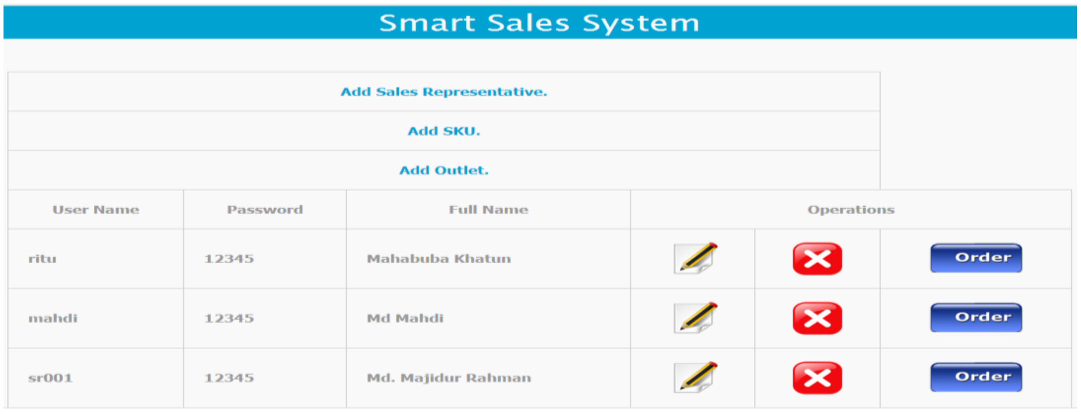
full Name
User Name
Password
SAVE

Fig: 3.1.1 Insert in MySql

```
<?php
include_once 'dbconfig.php';
if(isset($_POST['btn-save']))
{
 // variables for input data
 $full_name = $_POST['full_name'];
 $password = $_POST['password'];
 $user_name = $_POST['user_name'];

 // sql query for inserting data into database
        $sql_query = "INSERT INTO tbl_user(full_name,password,user_name)
VALUES('$full_name','$password','$user_name')";
 mysql_query($sql_query);

}
?>
```

Read : Retrieving Records From Table

After data insert we can retrieve all the records from tbl_user table :
Retrieving Records From tbl_user table
following is the PHP code for selecting records from table.



Smart Sales System

Add Sales Representative.

Add SKU.

Add Outlet.

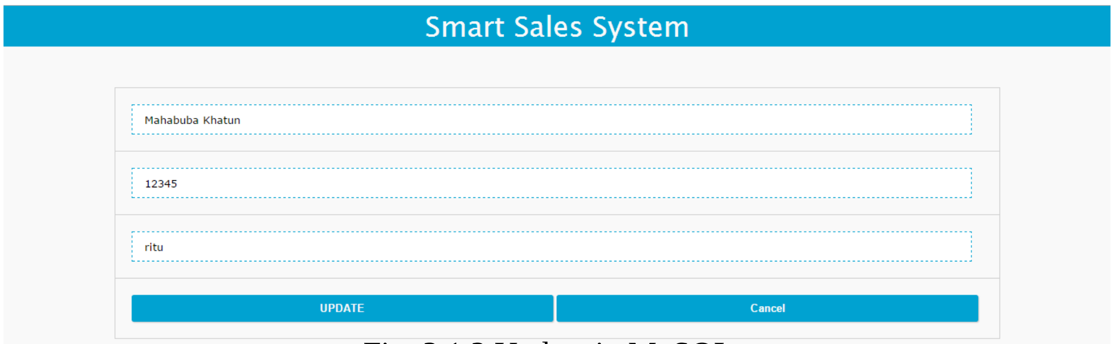
User Name	Password	Full Name	Operations		
ritu	12345	Mahabuba Khatun	✎	✖	Order
mahdi	12345	Md Mahdi	✎	✖	Order
sr001	12345	Md. Majidur Rahman	✎	✖	Order

```
<?php
$sql_query="SELECT * FROM tbl_user";
$result_set=mysql_query($sql_query);
while($row = mysql_fetch_array($result))
{
    ?>
    <!--Add HTML code here to display records.-->
    <?php
}
?>
```

After that, list of database records will be displayed to the browser, as shown below. And, this list contains icons to trigger update and delete operations for each record.

Update : Update MySQL Records.

to complete this operation with each record we will use edit icon that displayed with each record as shown above screenshot on clicking this icon for the specified row the page will jump to the edit_data.php with the specified row and details which are to be edited are filled up in the below form , and it will look like.


Smart Sales System

Mahabuba Khatun

12345

ritu

Fig: 3.1.2 Update in MySQL

after selecting data and Click on Update Button execute UPDATE Query as follow

```
if(isset($_POST['btn-update']))
{
 // variables for input data
 $full_name = $_POST['full_name'];
 $password = $_POST['password'];
 $user_name = $_POST['user_name'];
 // sql query for update data into database
 $sql_query = "UPDATE users SET
full_name='$full_name',password='$password',user_name='$user_name' WHERE
user_id=".$_GET['edit_id'];
        mysql_query($sql_query);
}
```

Delete : Delete Records From MySQL Table.

By clicking on delete icon of any specified user record, the users id will set to the URL QueryString of the page and the QueryString index.php?delete_id=any_value if set the Sql Query DELETE will be executed by if(isset($_GET['delete_id'])) function , and delete statement is as follow :

```
if(isset($_GET['delete_id']))
{
 $sql_query="DELETE FROM tbl_user WHERE id=".$_GET['delete_id'];
 mysql_query($sql_query);
 header("Location: $_SERVER[PHP_SELF]");
}
```

3.2 The Activity Lifecycle

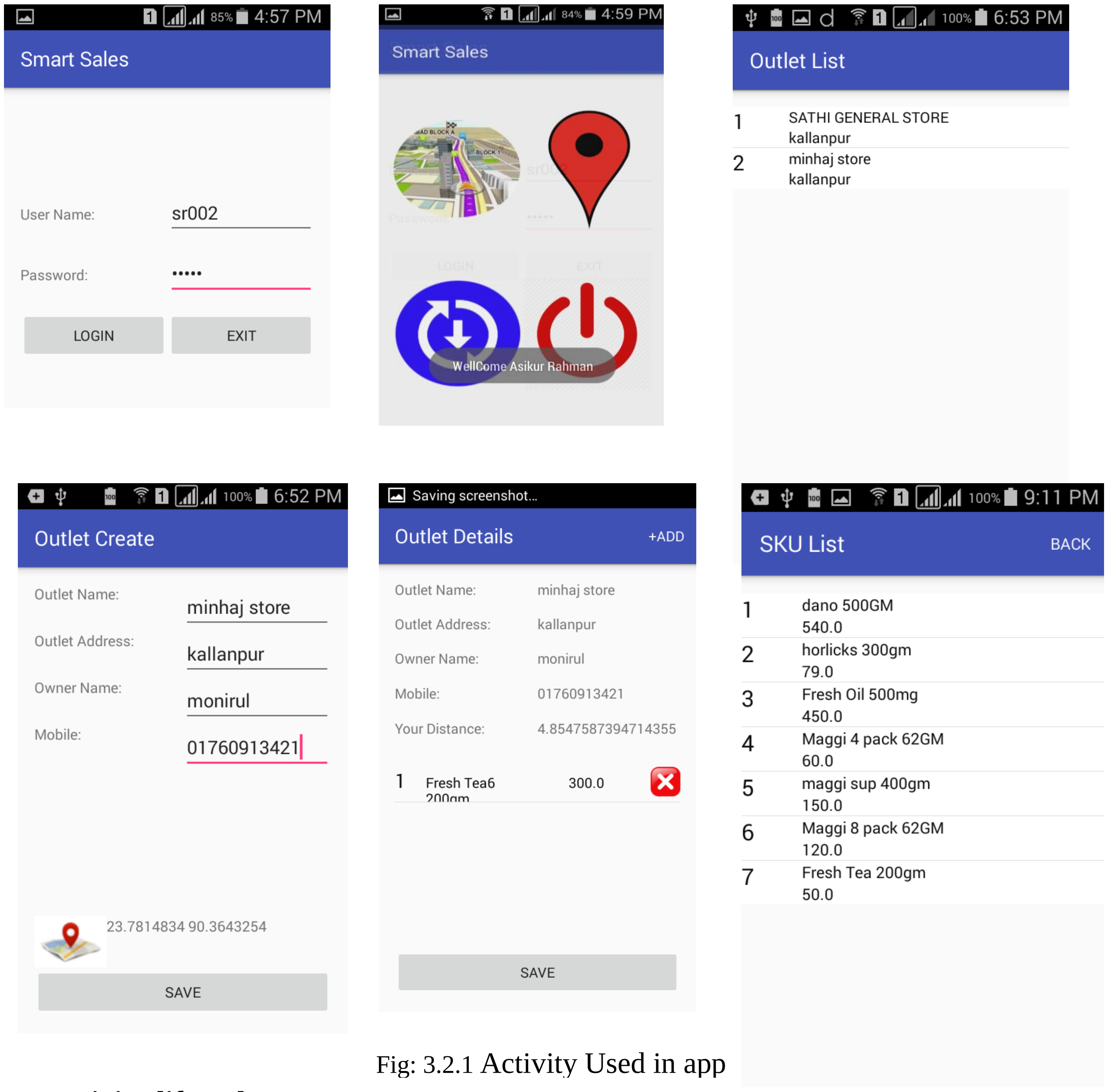
As a user navigates through, out of, and back to your app, the Activity instances in your app transition through different states in their lifecycle. The Activity class provides a number of callbacks that allow the activity to know that a state has changed: that the system is creating, stopping, resuming, or destroying an activity[17].

Within the lifecycle callback methods, you can declare how your activity behaves when the user leaves and re-enters the activity. For example, if you're building a streaming video player, you might pause the video and terminate the network connection when the user switches to another app. When the user returns, you can reconnect to the network and allow the user to resume the video from the same spot. In other words, each callback allows you to perform specific work that's appropriate to a given change of state. Doing the right work at the right time and handling transitions properly make your app more robust and performant[4] . For example, good implementation of the lifecycle callbacks can help ensure that your app avoids:

- Crashing if the user receives a phone call or switches to another app while using your app.
- Consuming valuable system resources when the user is not actively using it.
- Losing the user's progress if they leave your app and return to it at a later time.
- Crashing or losing the user's progress when the screen rotates between landscape and portrait orientation.

This document explains the activity lifecycle in detail[17].. The document begins by describing the lifecycle paradigm. Next, it explains each of the callbacks: what happens internally while they execute, and what you should implement during them. It then briefly introduces the relationship between activity state and a process’s vulnerability to being killed by the system. Last, it discusses several topics related to transitions between activity states.

- LoginActivity
- MainActivity
- MainMenuActivity
- OutletCreateActivity
- OutletDetaileActivity
- OutletListActivity
- SKUListActivity


Smart Sales

User Name: sr002

Password: •••••

LOGIN EXIT

Smart Sales

WellCome Asikur Rahman

Outlet List

1	SATHI GENERAL STORE kallanpur
2	minhaj store kallanpur

Outlet Create

Outlet Name:	minhaj store
Outlet Address:	kallanpur
Owner Name:	monirul
Mobile:	01760913421

23.7814834 90.3643254

SAVE

Outlet Details +ADD

Outlet Name:	minhaj store
Outlet Address:	kallanpur
Owner Name:	monirul
Mobile:	01760913421
Your Distance:	4.8547587394714355

1	Fresh Tea6 200gm	300.0

SAVE

SKU List BACK

1	dano 500GM 540.0
2	horlicks 300gm 79.0
3	Fresh Oil 500mg 450.0
4	Maggi 4 pack 62GM 60.0
5	maggi sup 400gm 150.0
6	Maggi 8 pack 62GM 120.0
7	Fresh Tea 200gm 50.0

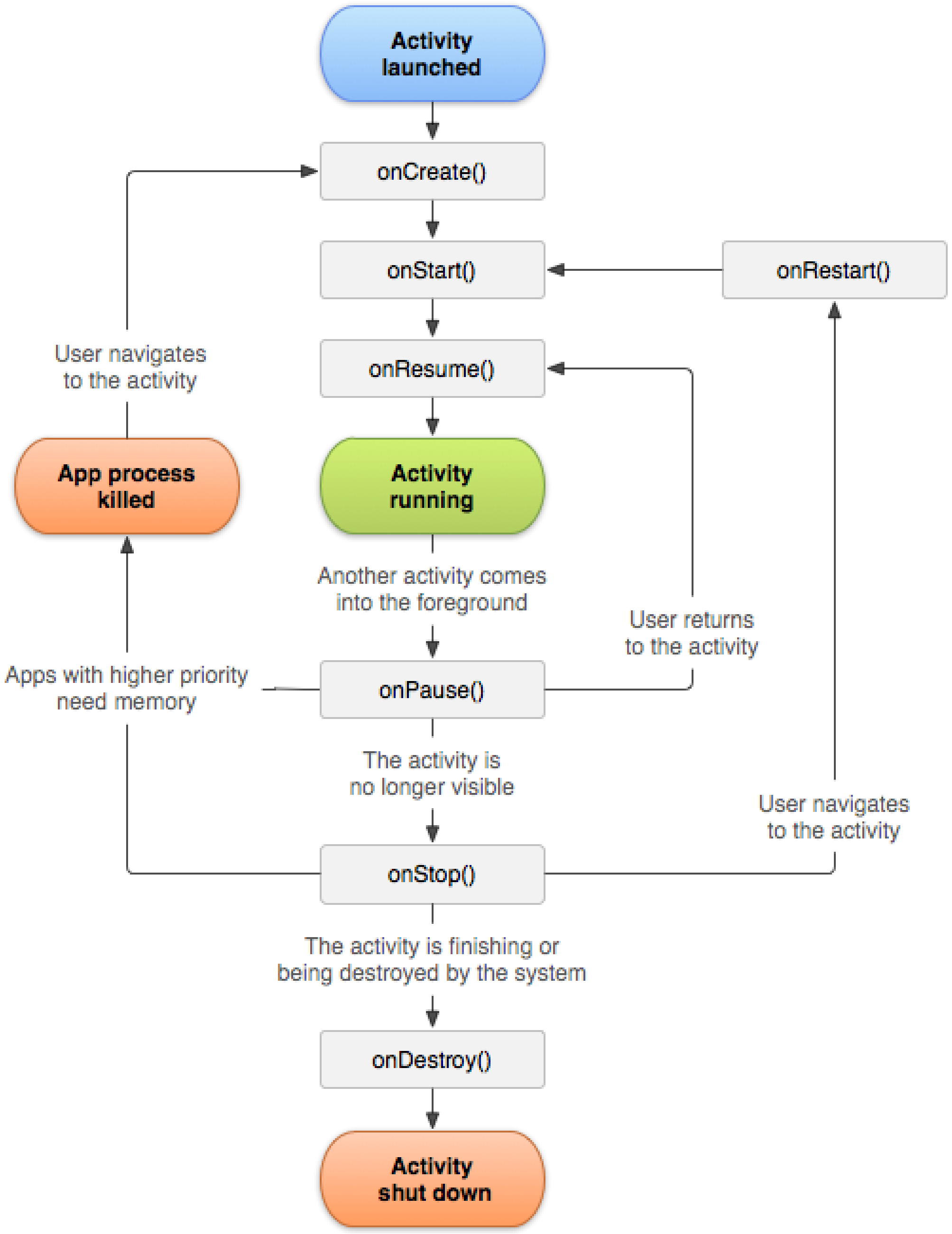
Fig: 3.2.1 Activity Used in app

Activity-lifecycle concepts :

To navigate transitions between stages of the activity lifecycle, the Activity class provides a core set of six callbacks: onCreate(), onStart(), onResume(), onPause(), onStop(), and onDestroy(). The system invokes each of these callbacks as an activity enters a new state[17]..

As the user begins to leave the activity, the system calls methods to dismantle the activity. In some cases, this dismantlement is only partial; the activity still resides in memory (such as when the user switches to another app), and can still come back to the foreground. If the user returns to that activity, the activity resumes from where the user left off. The system's likelihood of killing a given process—along with the activities in it—depends on the state of the activity at the time. Activity state and ejection from memory provides more information on the relationship between state and vulnerability to ejection.

Figure 11 presents a visual representation of this paradigm.



Activity launched
onCreate()
onStart()
onRestart()
User navigates to the activity
onResume()
App process killed
Activity running
Another activity comes into the foreground
User returns to the activity
Apps with higher priority need memory
onPause()
The activity is no longer visible
User navigates to the activity
onStop()
The activity is finishing or being destroyed by the system
onDestroy()
Activity shut down

Fig: 3.2.2 Activity-lifecycle

onCreate():

You must implement this callback, which fires when the system first creates the activity. On activity creation, the activity enters the Created state. In the onCreate() method, you perform basic application startup logic that should happen only once for the entire life of the activity. For example, your implementation of onCreate() might bind data to lists, initialize background threads, and instantiate some class-scope variables. This method receives the parameter savedInstanceState, which is a Bundle object containing the activity's previously saved state. If the activity has never existed before, the value of the Bundle object is null.

onStart():

When the activity enters the Started state, the system invokes this callback. The onStart() call makes the activity visible to the user, as the app prepares for the activity to enter the foreground and become interactive. For example, this method is where the app initializes the code that maintains the UI. It might also register a BroadcastReceiver that monitors changes that are reflected in the UI.

onResume():

When the activity enters the Resumed state, it comes to the foreground, and then the system invokes the onResume() callback. This is the state in which the app interacts with the user. The app stays in this state until something happens to take focus away from the app. Such an event might be, for instance, receiving a phone call, the user's navigating to another activity, or the device screen's turning off.

When an interruptive event occurs, the activity enters the Paused state, and the system invokes the onPause() callback.

onPause():

The system calls this method as the first indication that the user is leaving your activity (though it does not always mean the activity is being destroyed). Use the onPause() method to pause operations such animations and music playback that should not continue while the Activity is in the Paused state, and that you expect to resume shortly. There are several reasons why an activity may enter this state. For example:

- Some event interrupts app execution, as described in the onResume() section. This is the most common case.
- In Android 7.0 (API level 24) or higher, multiple apps run in multi-window mode. Because only one of the apps (windows) has focus at any time, the system pauses all of the other apps.
- A new, semi-transparent activity (such as a dialog) opens. As long as the activity is still partially visible but not in focus, it remains paused.

You can use the onPause() method to release system resources, such as broadcast receivers, handles to sensors (like GPS), or any resources that may affect battery life while your activity is paused and the user does not need them.

onStop():

When your activity is no longer visible to the user, it has entered the Stopped state, and the system invokes the onStop() callback. This may occur, for example, when a newly launched activity covers the entire screen. The system may also call onStop() when the activity has finished running, and is about to be terminated.

In the onStop() method, the app should release almost all resources that aren't needed while the user is not using it. For example, if you registered a BroadcastReceiver in onStart() to listen for changes that might affect your UI, you can unregister the broadcast receiver in onStop(), as the user can no longer see the UI. It is also important that you use onStop() to release resources that might leak memory, because it is possible for the system to kill the process hosting your activity without calling the activity's final onDestroy() callback.

onDestroy():

Called before the activity is destroyed. This is the final call that the activity receives. The system either invokes this callback because the activity is finishing due to someone's calling finish(), or because the system is temporarily destroying the process containing the activity to save space. You can distinguish between these two scenarios with the isFinishing() method. The system may also call this method when an orientation change occurs, and then immediately call onCreate() to recreate the process (and the components that it contains) in the new orientation.

3.3 Android View

We already know about Android layouts. Android Basic views section would include following items:

- Android EditText
- Android TextBox
- Android Button
- Android Radio Button
- Android Checkbox
- Android Toggle Button
- Android Radio Group
- Android Image Button

Let us study these elements in brief.

Button: Button is pushed or pressed or clicked. We already know how to use button widget but let us know what it is. We have attributes related to buttons. Every view should have a unique identifier which identifies the element in the project. Make sure, two elements should not share their unique ID. We implement certain methods and interfaces which listens to user input and act accordingly.

TextView: This view is basically meant for displaying the text of application. For example the label of application is displayed here.

EditText: This is a modification of Text View. It is editable. User can give input dynamically.

Check Box: This widget is a two-state button. It can be either checked or unchecked.

Radio Button: This is also a two-state button. It can be checked or unchecked. Unlike checkbox if it is checked once, cannot be unchecked. It can be checked dynamically.

Radio Group: This houses radio buttons together. Checking any one of the radio buttons in the group unchecks rest of the buttons.

Image Button: This a button with image. User can press or click this button. Image on the button is referenced from the src folder of our project.

3.4 Array Adapter

In Android development, any time we want to show a vertical list of scrollable items we will use a ListView which has data populated using an Adapter. The simplest adapter to use is called an ArrayAdapter because the adapter converts an ArrayList of objects into View items loaded into the ListView container[18].

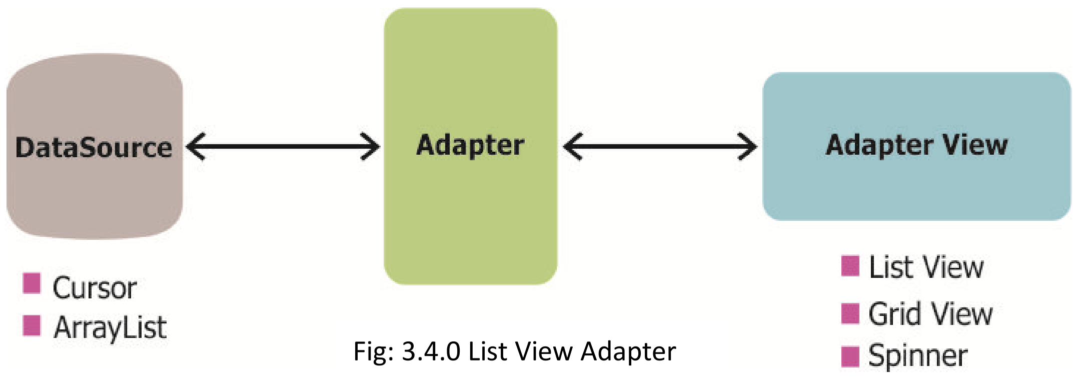


Fig: 3.4.0 List View Adapter

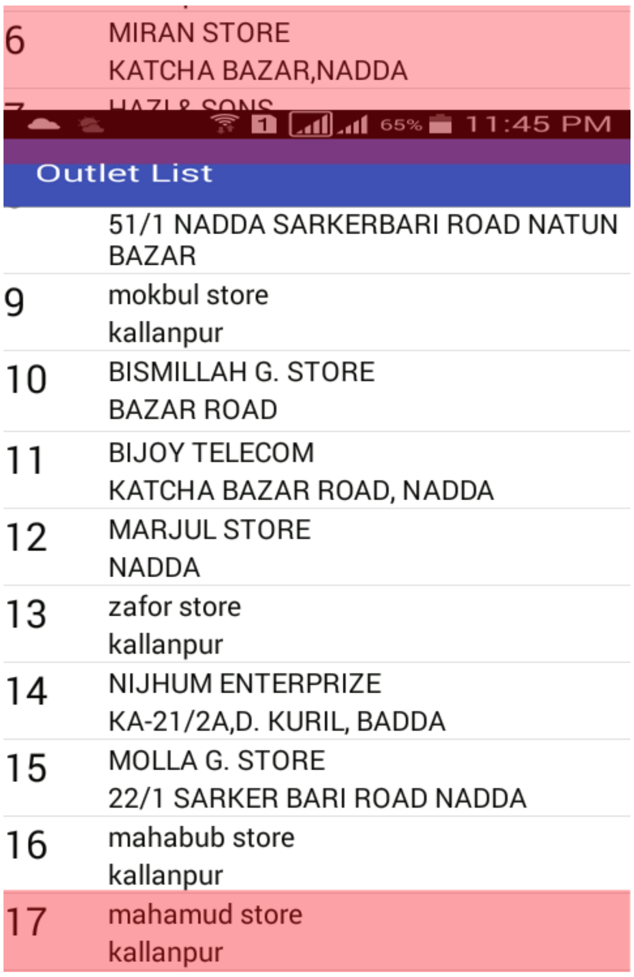
The ArrayAdapter fits in between an ArrayList (data source) and the ListView (visual representation) and configures two aspects:

Which array to use as the data source for the list
How to convert any given item in the array into a corresponding View object
Note as shown above that there are other data sources besides an ArrayAdapter such as the CursorAdapter which instead binds directly to a result set from a Local SQLite Database[12].

Row View Recycling:

When using an adapter and a ListView, we need to make sure to understand how view recycling works.

When your ListView is connected to an adapter, the adapter will instantiate rows until the ListView has fully populated with enough items to fill the full height of the list. At that point, no additional row items are created in memory.

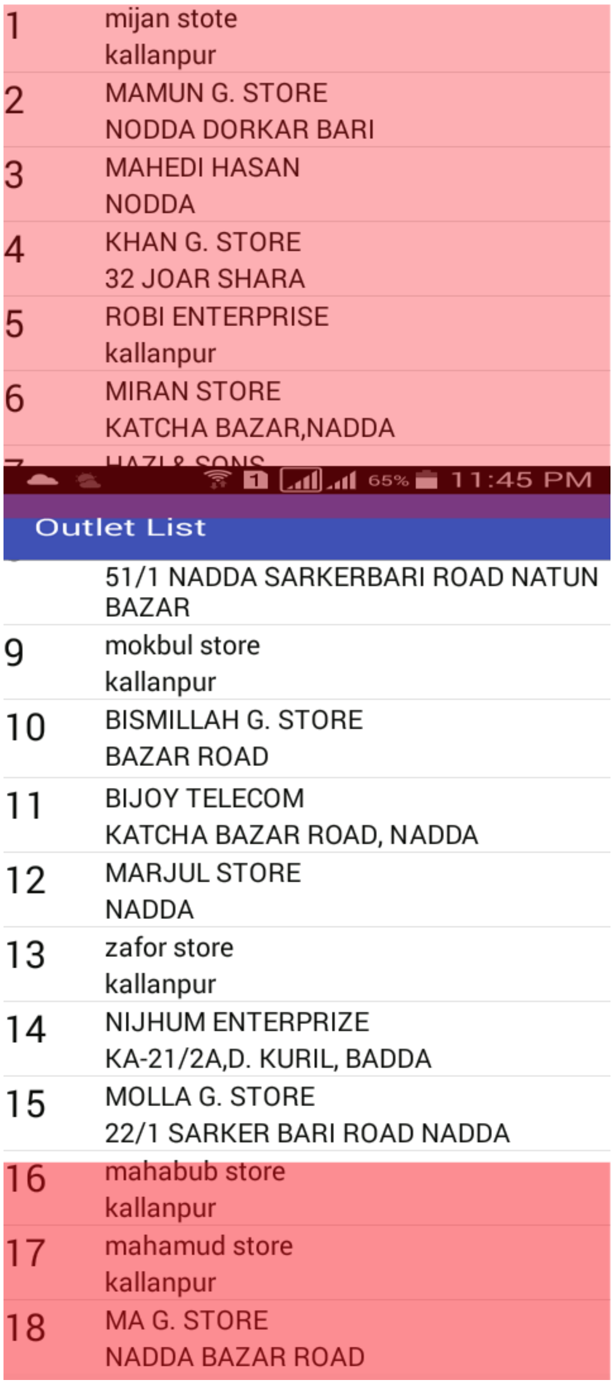


6	MIRAN STORE KATCHA BAZAR,NADDA

11:45 PM

Outlet List

	51/1 NADDA SARKERBARI ROAD NATUN BAZAR
9	mokbul store kallanpur
10	BISMILLAH G. STORE BAZAR ROAD
11	BIJOY TELECOM KATCHA BAZAR ROAD, NADDA
12	MARJUL STORE NADDA
13	zafor store kallanpur
14	NIJHUM ENTERPRIZE KA-21/2A,D. KURIL, BADDA
15	MOLLA G. STORE 22/1 SARKER BARI ROAD NADDA
16	mahabub store kallanpur
17	mahamud store kallanpur



1	mijan stote kallanpur
2	MAMUN G. STORE NODDA DORKAR BARI
3	MAHEDI HASAN NODDA
4	KHAN G. STORE 32 JOAR SHARA
5	ROBI ENTERPRISE kallanpur
6	MIRAN STORE KATCHA BAZAR,NADDA

11:45 PM

Outlet List

	51/1 NADDA SARKERBARI ROAD NATUN BAZAR
9	mokbul store kallanpur
10	BISMILLAH G. STORE BAZAR ROAD
11	BIJOY TELECOM KATCHA BAZAR ROAD, NADDA
12	MARJUL STORE NADDA
13	zafor store kallanpur
14	NIJHUM ENTERPRIZE KA-21/2A,D. KURIL, BADDA
15	MOLLA G. STORE 22/1 SARKER BARI ROAD NADDA
16	mahabub store kallanpur
17	mahamud store kallanpur
18	MA G. STORE NADDA BAZAR ROAD

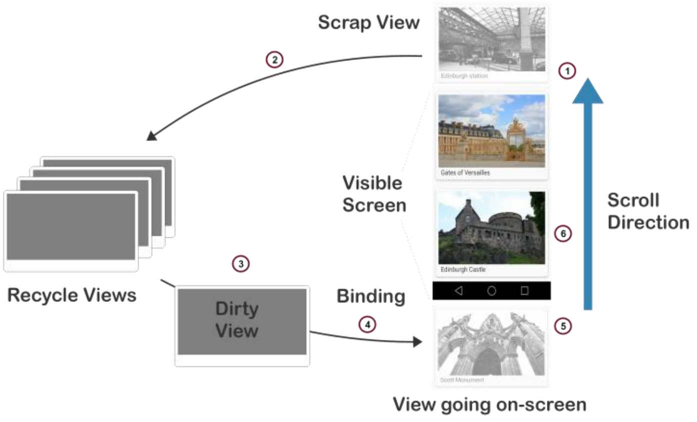


Fig: 3.4.1 List View Scroll view

Instead, as the user scroll through the list, items that leave the screen are kept in memory for later use and then every new row that enters the screen reuses an older row kept around in memory. In this way, even for a list of 1000 items, only ~7 item view rows are ever instantiated or held in memory. Here is a visual overview of recycling:

Refer to this ListView guide for another look at how this works to optimize the performance of your lists. Be sure to check out this Udacity video on view recycling as well. If you wish to evaluate how fast your ListView is rendering, check out the Profiling GPU tool, which provides a graphical way of visualizing the layout performance.

Creating a Custom ArrayAdapter

- OrderedSKUListAdapter
- OutletListAdapter
- SKUListAdapter

Defining the Model:

```
public class Outlet {
   int id;
   int srId;
   String name;
   String address;
   String ownerName;
   String mobile;
   double lat;
   double lon;
   double distance;
   int verified;
   int visited;

   public Outlet() {
   }
}
```

We can create a custom ListView of User objects by subclassing ArrayAdapter to describe how to translate the object into a view within that class and then using it like any other adapter.

Creating the View Template:

Next, we need to create an XML layout that represents the view template for each item in res/layout/outlet_list_item.xml:

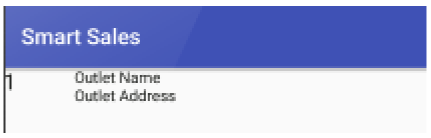


Fig: 3.4.2 Outlet List Item

Defining the Adapter:

Next, we need to define the adapter to describe the process of converting the Java object to a View (in the getView method). The naive approach to this (without any view caching) looks like the following:

```
public class OutletListAdapter extends ArrayAdapter<Outlet> {
   public OutletListAdapter(Context context, List<Outlet> objects) {
      super(context, 0, objects);
   }
```

```
    @Override
    public View getView(int position, View convertView, ViewGroup parent) {
        if (convertView == null) {
            convertView = LayoutInflater.from(getContext()).inflate(R.layout.outlet_list_item,
              parent,false);
        }
        Outlet selectedOutlet = getItem(position);
        TextView txtOutletName = (TextView) convertView.findViewById(R.id.txtOutletName);
        TextView txtOutletAddress = (TextView) convertView.findViewById(R.id.txtOutletAddress);
        TextView txtOutletCount = (TextView) convertView.findViewById(R.id.txtOutletCount);
        LinearLayout lvbg = (LinearLayout) convertView.findViewById(R.id.lvbg);
        txtOutletName.setText(selectedOutlet.getName());
        txtOutletAddress.setText(selectedOutlet.getAddress());
        if (selectedOutlet.getVisited()==1){
            lvbg.setBackgroundColor(Color.GREEN);
        }else {
            lvbg.setBackgroundColor( Color.parseColor("#FFFFFF"));
        }
        txtOutletCount.setText(position + 1 + "");
        txtOutletName.setTag(selectedOutlet);
        return convertView;
    }
}
```

Attaching the Adapter to a ListView:

```
OutletModel  outletModel =new OutletModel(getContentResolver());
OutletListAdapter adapter=new
OutletListAdapter(getApplicationContext(),outletModel.getOutletList().getOutletArray());
ListView lvOutletList = (ListView) findViewById(R.id.lvOutletList);
lvOutletList.setAdapter(adapter);
```

3.5 Business Object

A business object is an entity within a multitiered software application that works in conjunction with the data access and business logic layers to transport data.

Outlet
OutletList
SalesOrder
SalesOrderLine
SalesOrderLineList
SKU
SKUList

3.6 Model Class:

All about the data. What is manipulated, what is stored and how
OutletModel
SalesOrderModel
SKUModel

3.7 MVC Design Pattern

The primary goal of implementing the MVC pattern is that doing so allows you to then later 'pull out' any one of them and slap in a new one without little or no necessary change to the other two.
Model: All about the data. What is manipulated, what is stored and how.

View: All about the UI or presentation. What is displayed and how.

Controller: The event handler. Dictates when the other two are run in response to events.

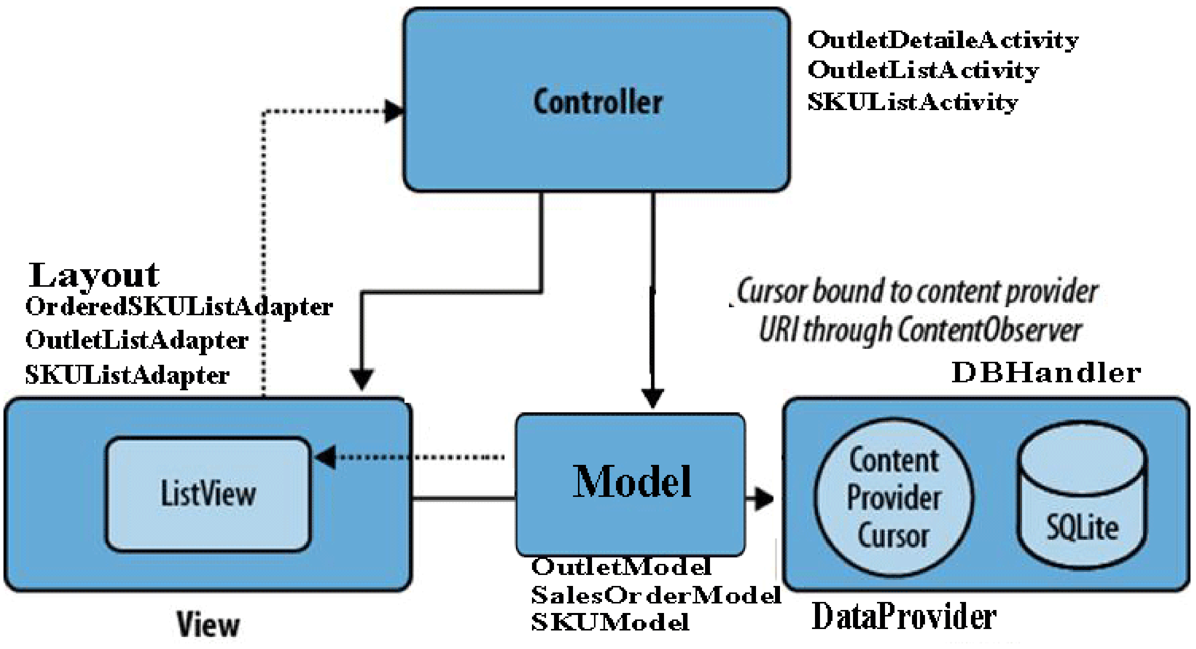


Figure 3.7.1 : MVC Design

In Android, this implementation of MVC has the controller in the form of a class that extends the Activity class. After all, it is this class that initially receives the 'events' that make up the lifecycle of an Android Activity (i.e. onStart(), onCreate(), onSuspend(), onStop(), onResume(), onDestroy). This lifecycle will likely change as Android evolves so it makes sense to represent it as the Controller component of the MVC pattern.

Again, with this MVC implementation, I can pull out any one of the three components and put in essentially a whole new interface (View), or a whole new database (Model), or an new Activity lifecycle (Controller) with no change to the other two. The only necessity is that each component respects the boilerplate template listed below.

In this implementation, the three MVC components are separately represented by three java classes: appView.java, appController.java, appModel.java

When reviewing each class, make note of the member variables, mController, mAppView, and mAppModel, and see how and when they are referenced in each java file. These member variables are 'the hooks' that allow each component to reference each other.

Further, you'll note mAppModel breaks down even further and uses an additional class called dbHelper. This allows you to separate 'what' data is manipulated from 'how' that data is manipulated and stored.

3.8 Sync Adapter:

SyncAdapter is standard and preferred way to sync application data with server, cloud or MBaaS backend. SyncAdapter is the most efficient way to make your application update to date. It will help you to present fresh data when user open application. You can increase user experience by not making wait to user till you call web service and fetch the data. Almost all popular android application use SyncAdapter, you check from Setting -> Account, you will find accounts i.e. google, twitter, facebook, etc.. It means these application use SyncAdapter[9].

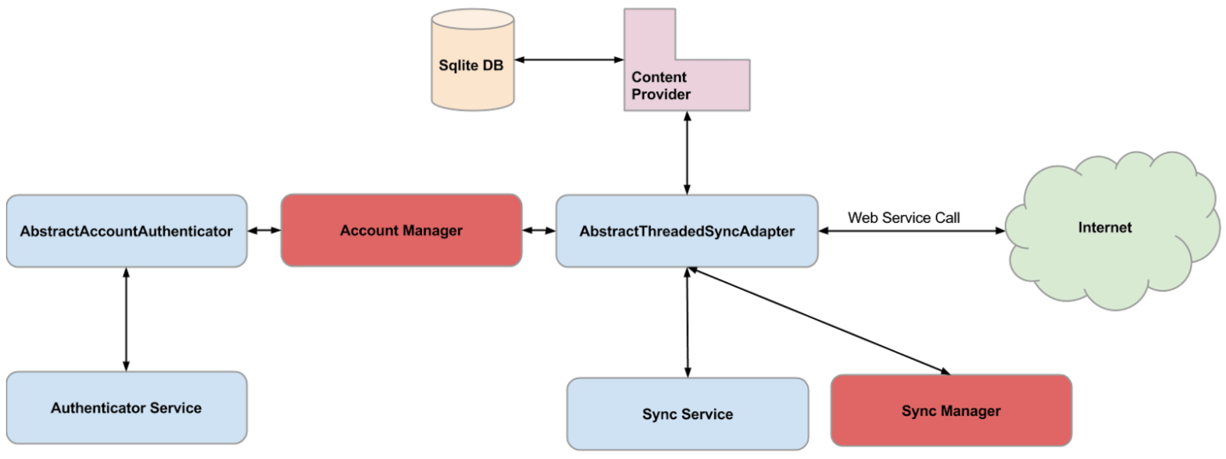


Fig:3.8.1 Android : Sync Adapter

Concepts

Concepts are very important to read before jump to the demo. Many developer don't read API document or understand concept and directly start sample and try to learn from it but I suggest to read concept first before open your IDE and read the code. SyncAdapter has mainly two section 1. Account management 2. Sync Adapter. If your app requires user credentials or server login, you can optionally integrate account management and authentication into your data transfer.

SyncAdapter has following sections.

1. **Sqlite Database:** I guess you all are master of Sqlite database, SyncAdapter will store data in Sqlite using Content Provider. You may choose other options as well.
2. **Content Provider:** Act as bridge between your database and SyncAdapter. To expose your data in Rest like URL pattern.
3. **AbstractAccountAuthenticator:** As developer we need to extend this class and override methods, It is primarily used to manage authentication and account management. To use SyncAdapter you must have custom account. This class is responsible to create account, maintain auth token and various things related to authentication.
4. **Authenticator Service:** This is normal Service, which we are using daily. The only difference is that this service create object of AbstractAccountAuthenticator class and bind.
5. **AbstractThreadedSyncAdapter:** As developer we need to extend this class and override methods. . This is the main piece of SyncAdapter puzzle. It has method onPerformSync, in which we need to write our code.
6. **Sync Service:** This is normal Service. It use to create object of AbstractThreadedSyncAdapter class and bind.

7. **Authenticator.xml:** You need to create this file under res/xml/ folder. This file is required to bind your authenticator component into the sync adapter and account frameworks, you need to provide these framework with metadata that describes the component. You can choose your own file name.
8. **SyncAdapter.xml:** You need to create this file under res/xml/ folder. The metadata specifies the account type you've created for your sync adapter, declares a content provider authority associated with your app.
9. **AndroidManifast.xml:** You must register Sync Service, Authenticator service and few other things in AndroidManifast file in order to work SyncAdapter, This is the final piece of puzzle.

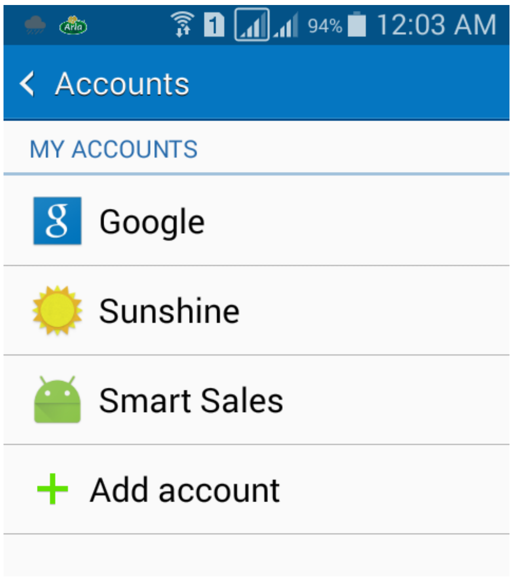


Fig: 3.8.1 Account

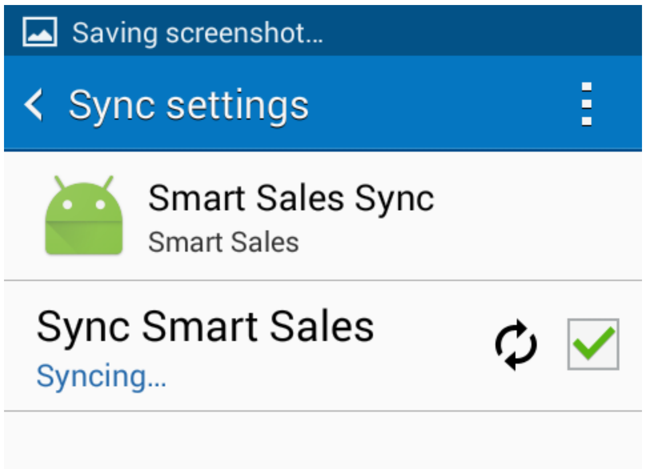
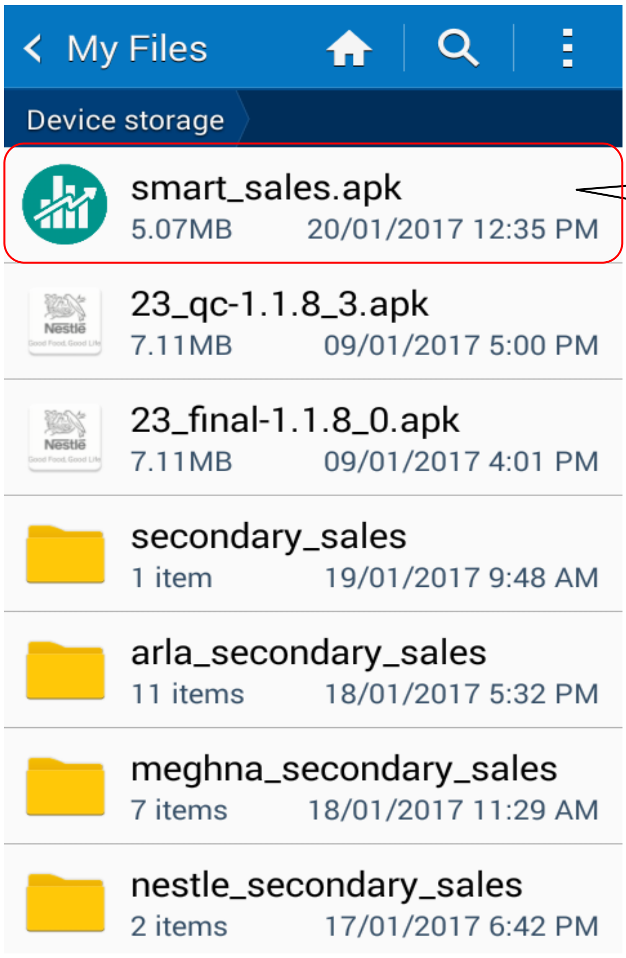


Fig: 3.8.2 Sync Setting

How to Install Mobile Application:


My Files

Device storage

smart_sales.apk 5.07MB 20/01/2017 12:35 PM

23_qc-1.1.8_3.apk 7.11MB 09/01/2017 5:00 PM

23_final-1.1.8_0.apk 7.11MB 09/01/2017 4:01 PM

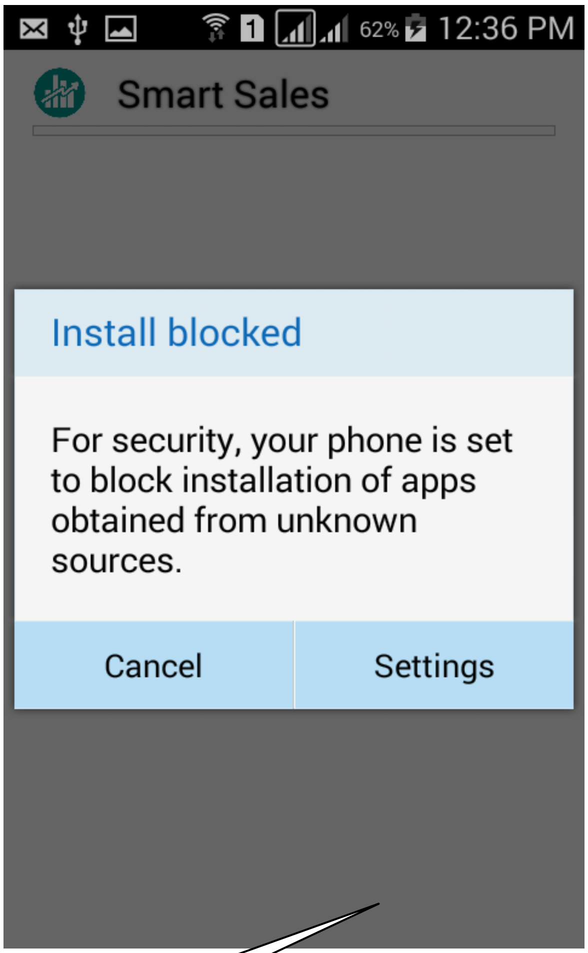
secondary_sales 1 item 19/01/2017 9:48 AM

arla_secondary_sales 11 items 18/01/2017 5:32 PM

meghna_secondary_sales 7 items 18/01/2017 11:29 AM

nestle_secondary_sales 2 items 17/01/2017 6:42 PM

Copy smart_sales.apk Android application file in your Mobile and open file manager It's Look Like This Image and Click on it.


12:36 PM

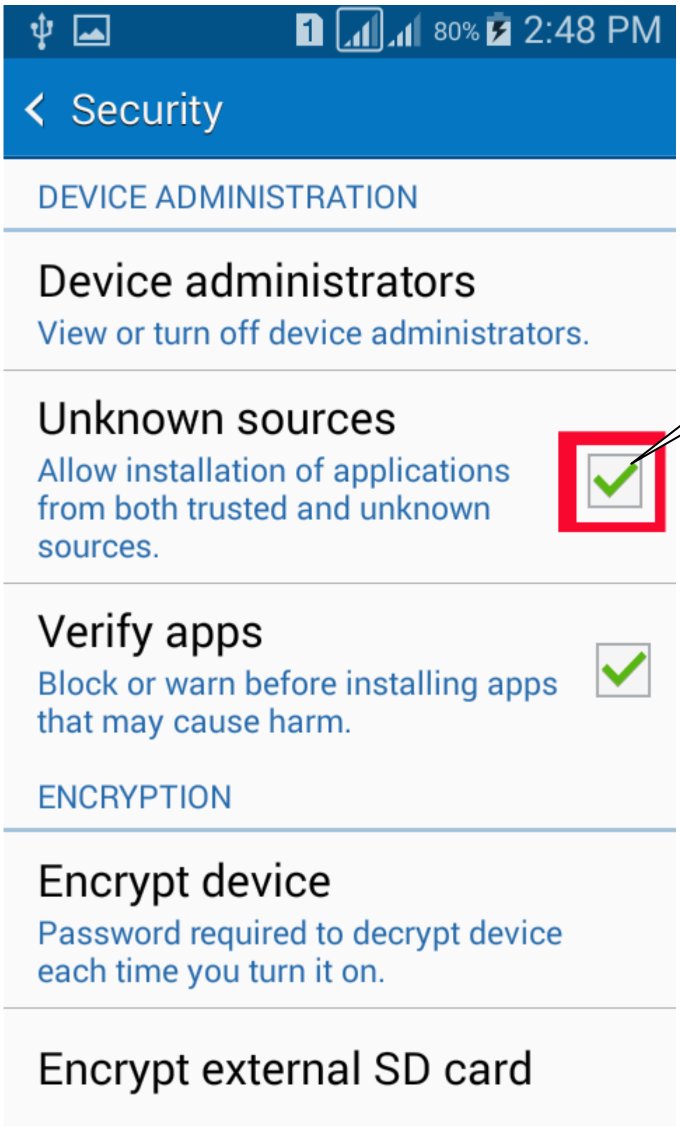
Smart Sales

Install blocked

For security, your phone is set to block installation of apps obtained from unknown sources.

Cancel Settings

Click on Settings.



2:48 PM

Security

DEVICE ADMINISTRATION

Device administrators
View or turn off device administrators.

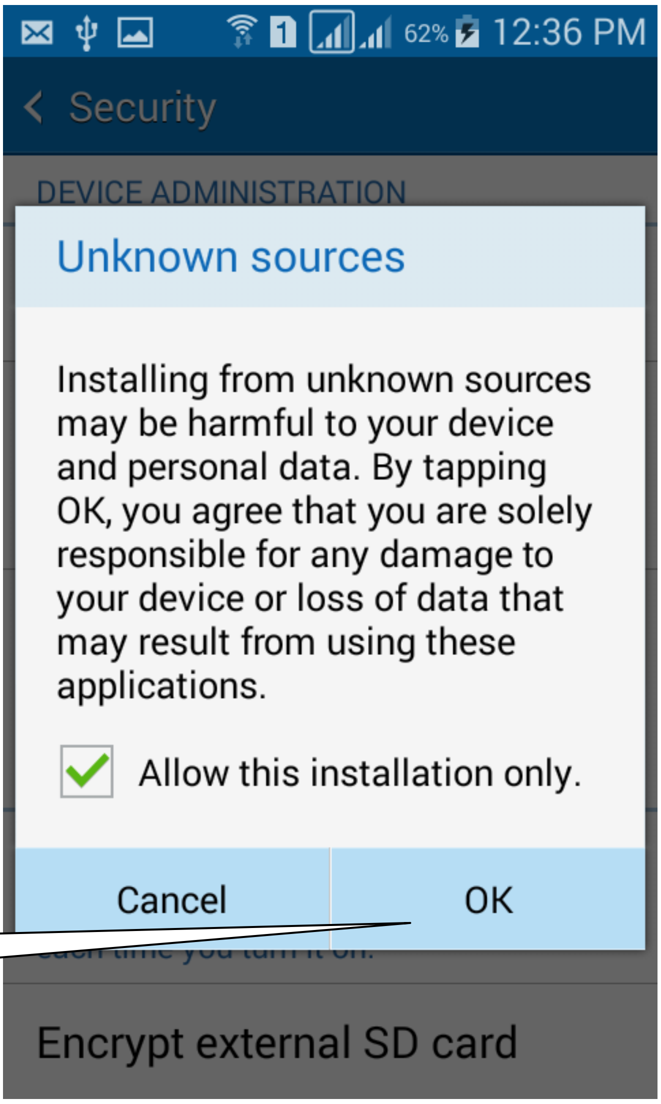
Unknown sources ✓
Allow installation of applications from both trusted and unknown sources.

Verify apps ✓
Block or warn before installing apps that may cause harm.

ENCRYPTION

Encrypt device
Password required to decrypt device each time you turn it on.

Encrypt external SD card



12:36 PM

Security

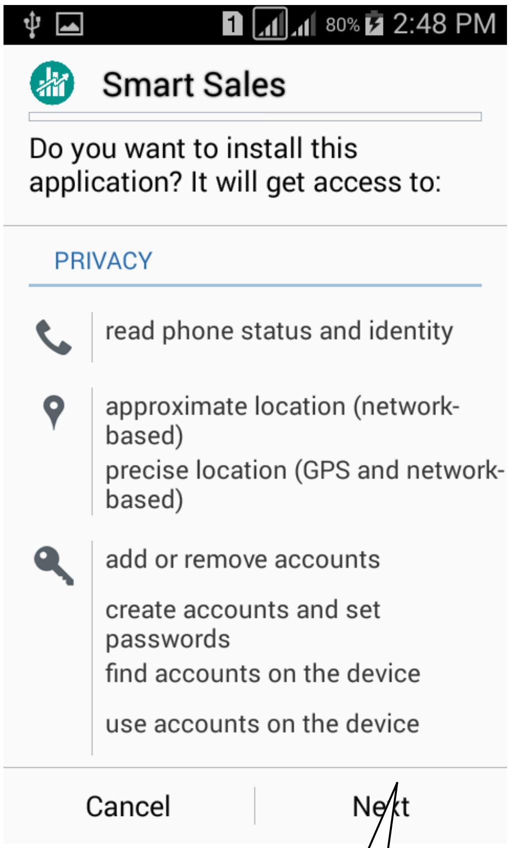
Unknown sources

Installing from unknown sources may be harmful to your device and personal data. By tapping OK, you agree that you are solely responsible for any damage to your device or loss of data that may result from using these applications.

✓ Allow this installation only.

Cancel | OK

Encrypt external SD card


2:48 PM

Smart Sales

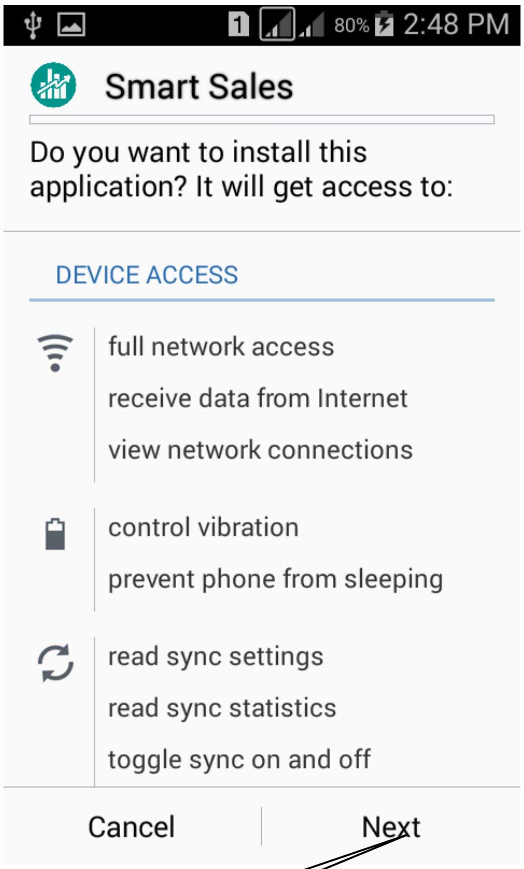
Do you want to install this application? It will get access to:

PRIVACY

read phone status and identity

approximate location (network-based)
precise location (GPS and network-based)

add or remove accounts
create accounts and set passwords
find accounts on the device
use accounts on the device


2:48 PM

Smart Sales

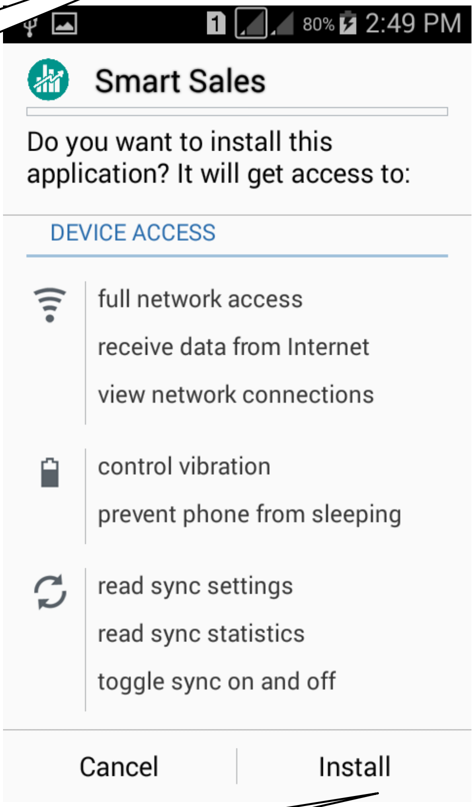
Do you want to install this application? It will get access to:

DEVICE ACCESS

full network access
receive data from Internet
view network connections

control vibration
prevent phone from sleeping

read sync settings
read sync statistics
toggle sync on and off


2:49 PM

Smart Sales

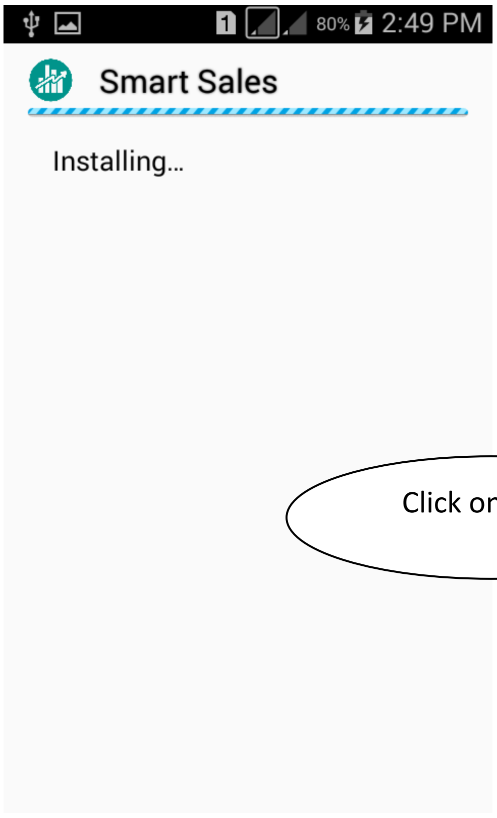
Do you want to install this application? It will get access to:

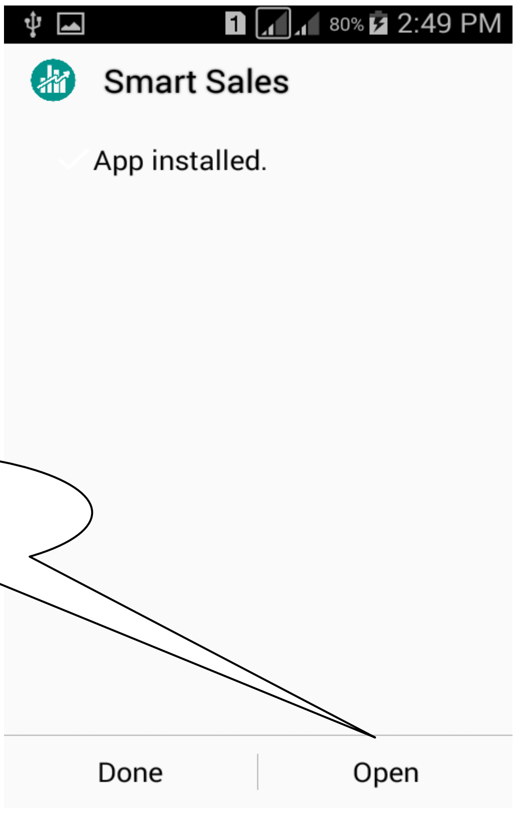
DEVICE ACCESS

full network access
receive data from Internet
view network connections

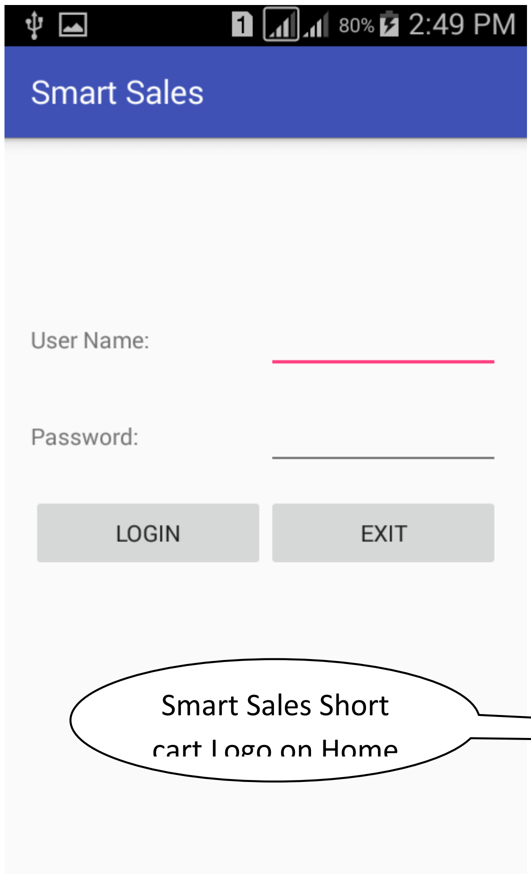
control vibration
prevent phone from sleeping

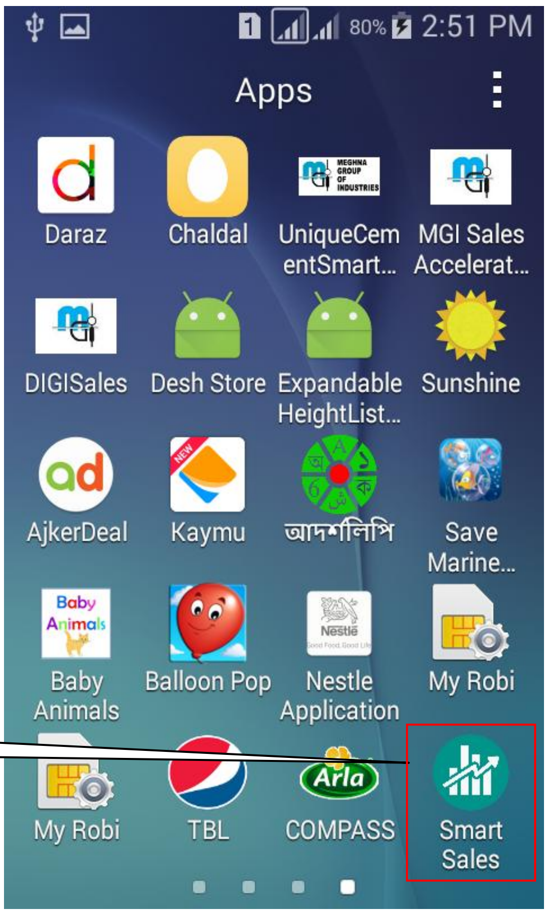
read sync settings
read sync statistics
toggle sync on and off





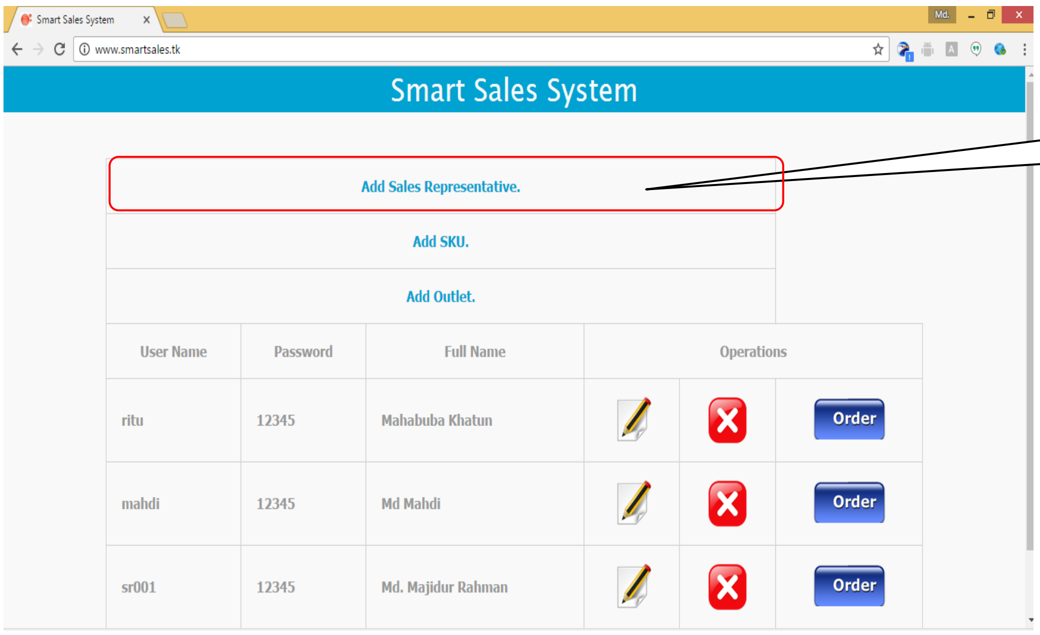
Click on Open





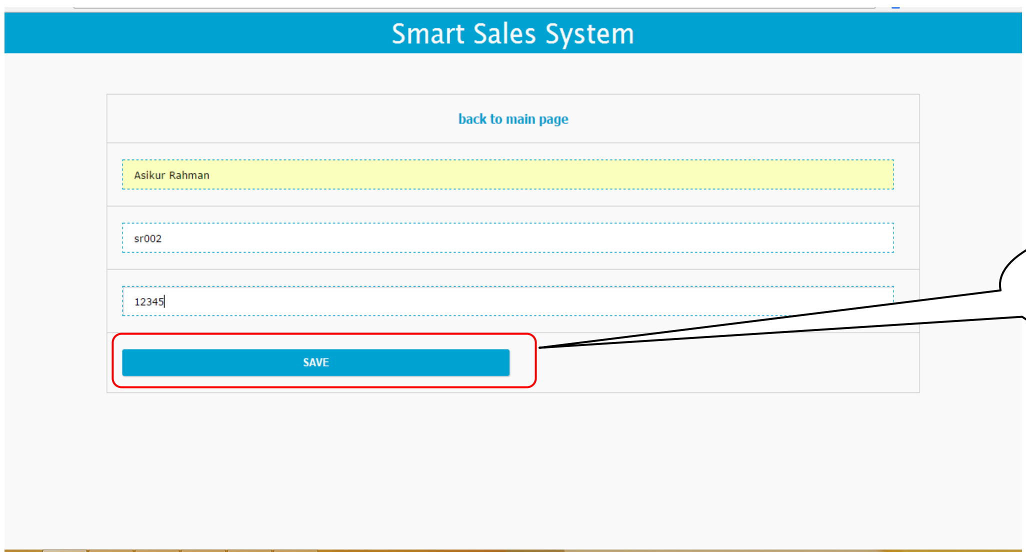
Smart Sales Short cart Logo on Home

Sales Representative CRUD and Login Process:



User Name	Password	Full Name	Operations		
ritu	12345	Mahabuba Khatun			Order
mahdi	12345	Md Mahdi			Order
sr001	12345	Md. Majidur Rahman			Order

Add Sales Representative Here

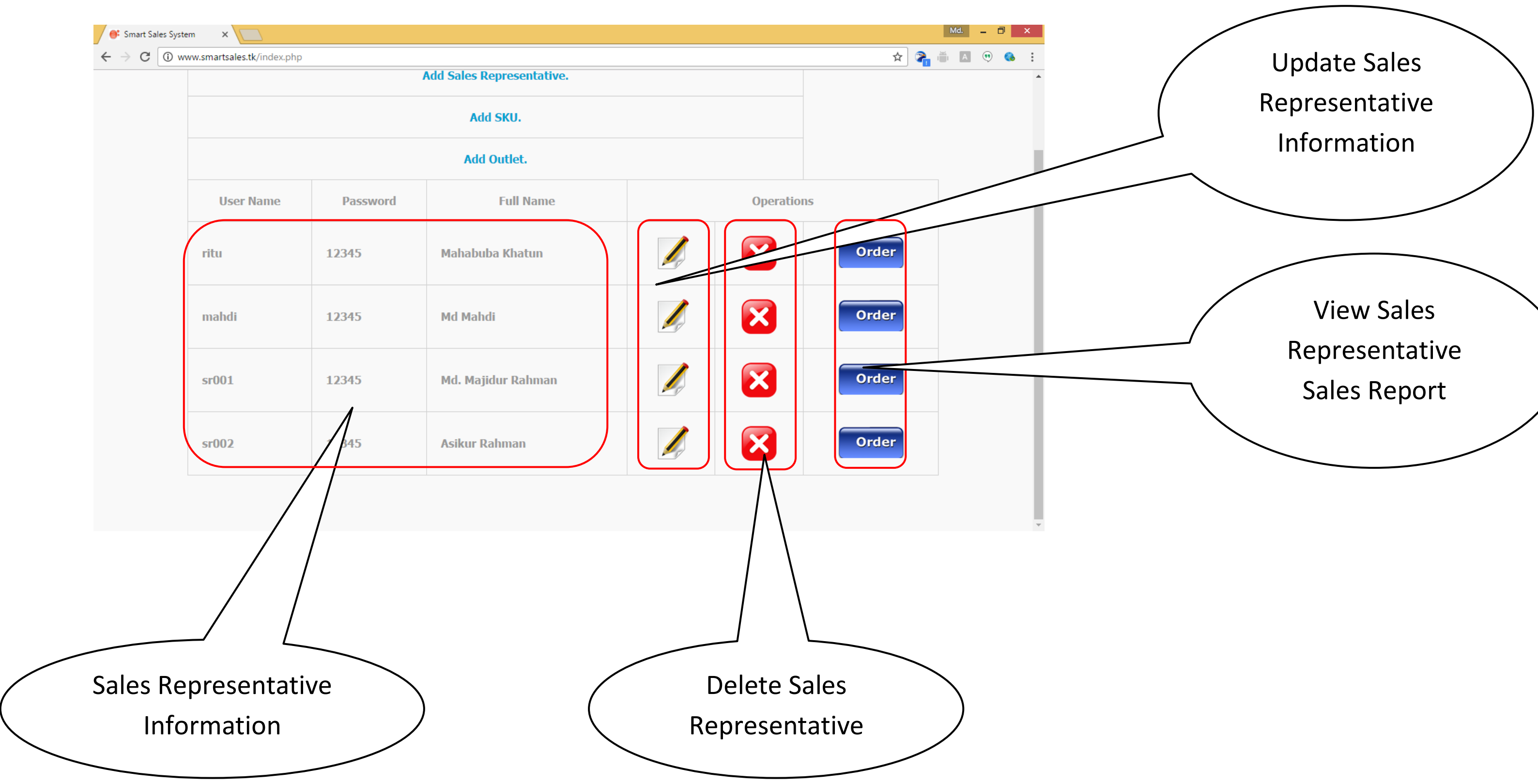


Asikur Rahman

sr002

12345

Save Sales Representative


Smart Sales System

www.smartsales.tk/index.php

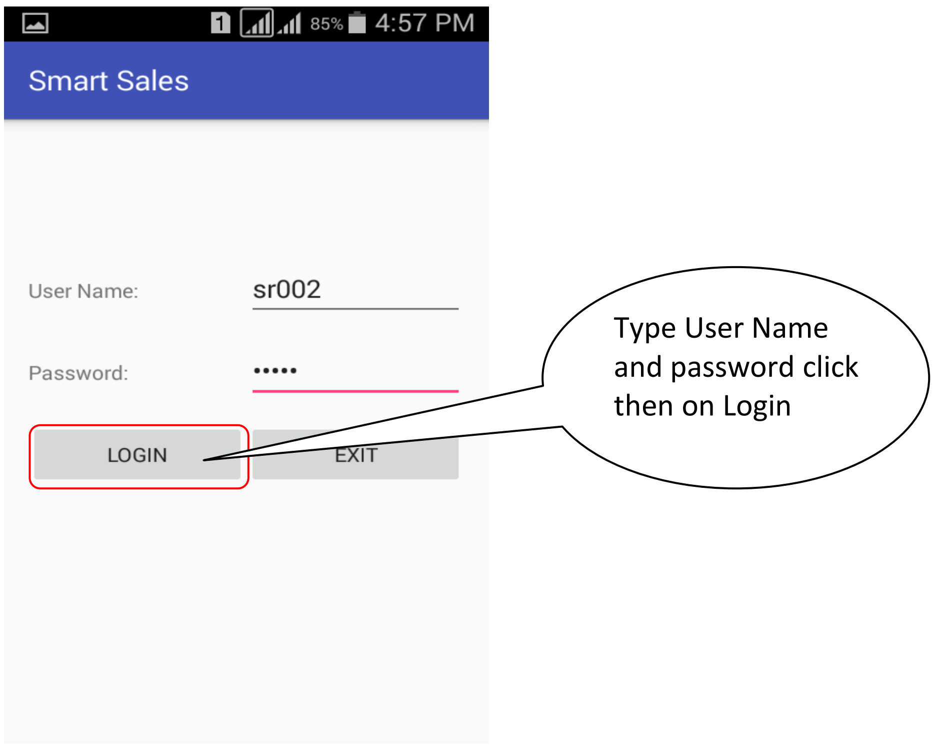
Add Sales Representative.

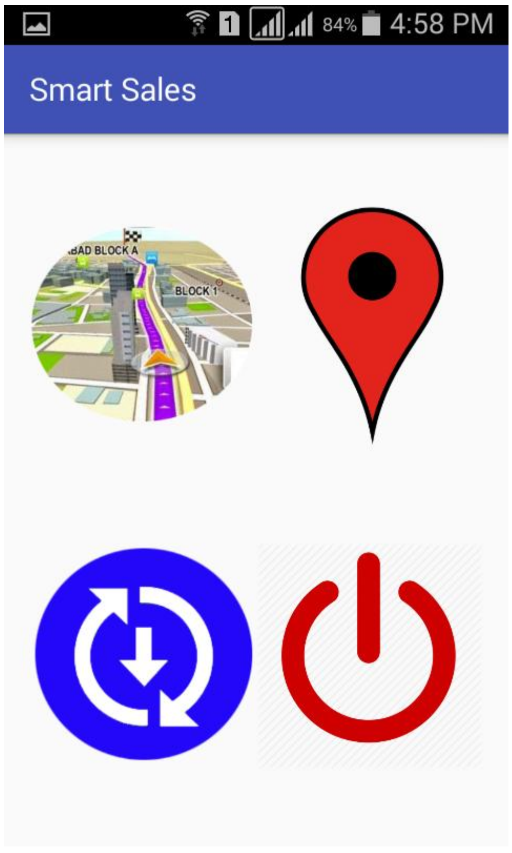
Add SKU.

Add Outlet.

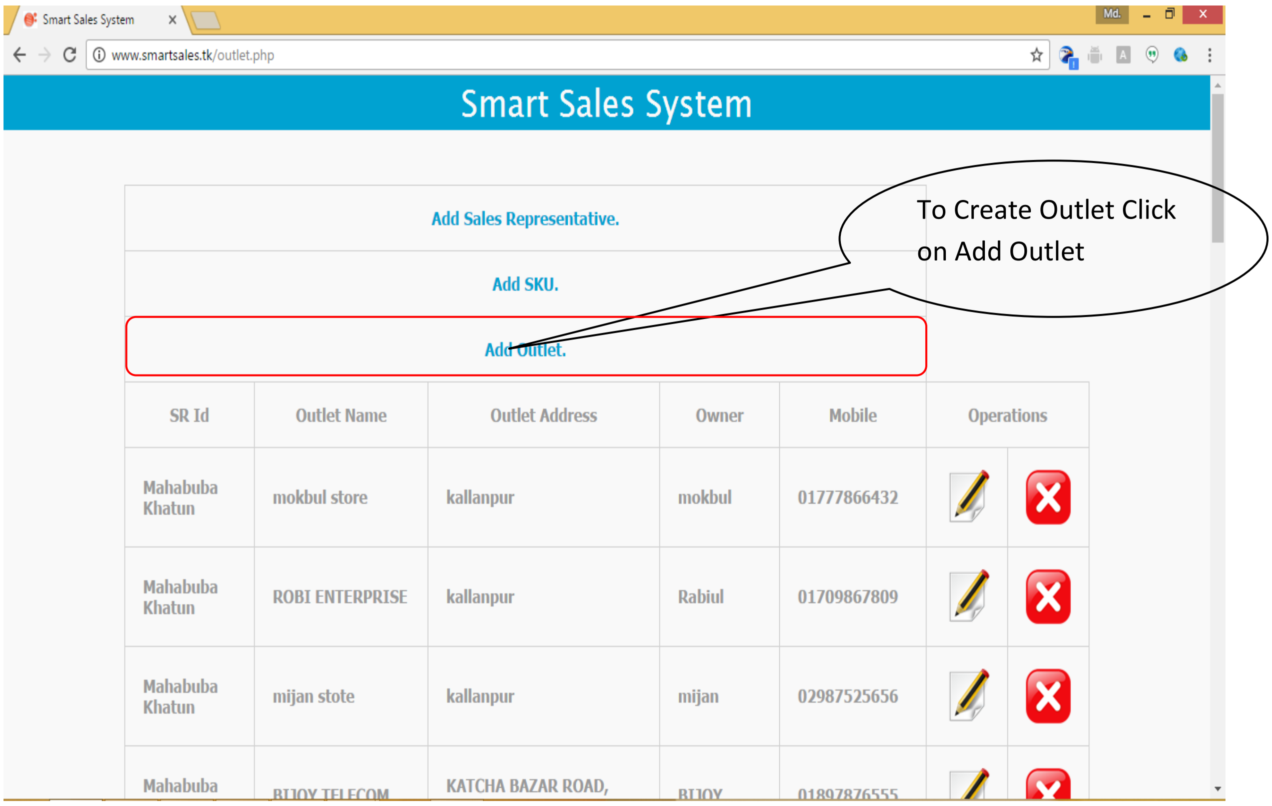
User Name	Password	Full Name	Operations		
ritu	12345	Mahabuba Khatun	✎	✖	Order
mahdi	12345	Md Mahdi	✎	✖	Order
sr001	12345	Md. Majidur Rahman	✎	✖	Order
sr002	1[illegible]345	Asikur Rahman	✎	✖	Order

Update Sales Representative Information
View Sales Representative Sales Report
Sales Representative Information
Delete Sales Representative

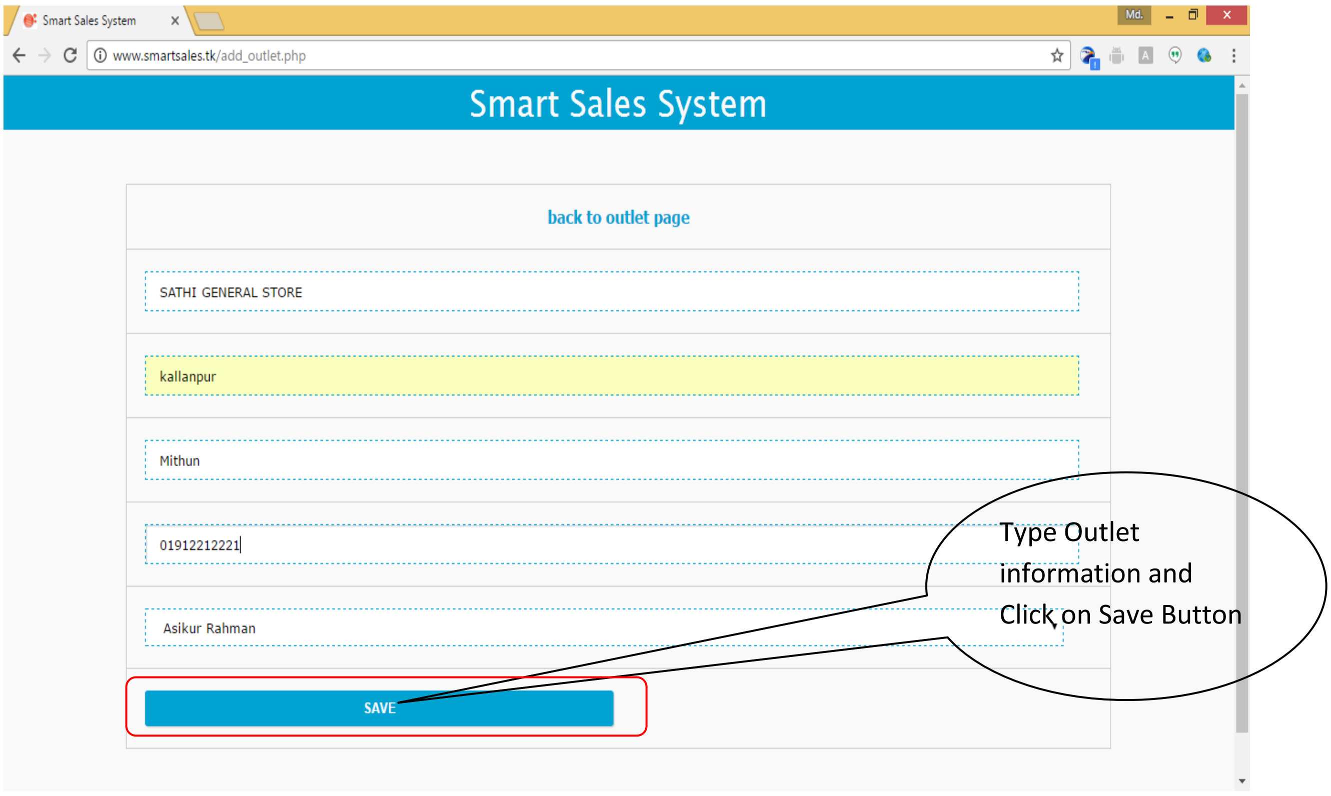




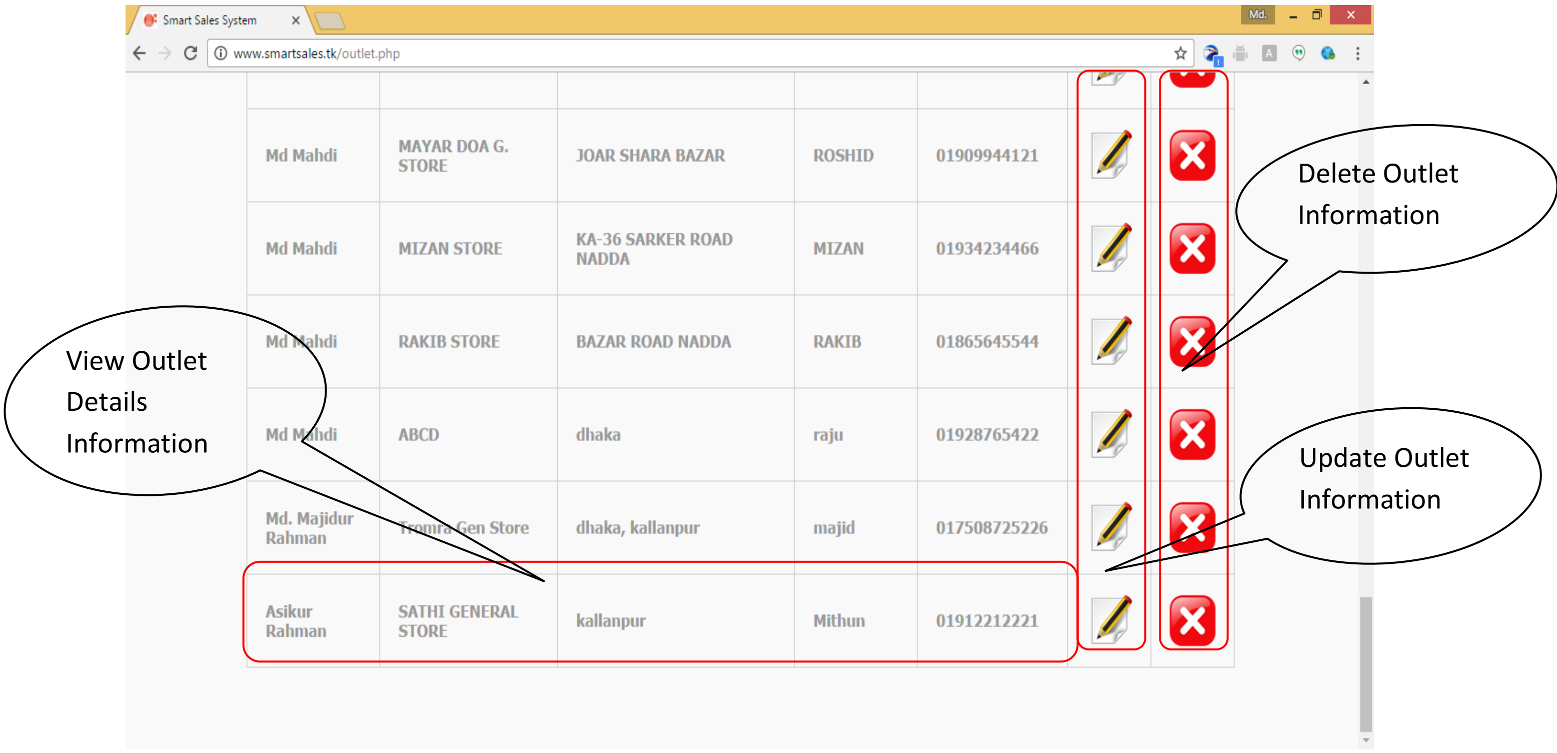
Outlet CRUD and Mobile Application Process:


Smart Sales System
www.smartsales.tk/outlet.php
Smart Sales System
Add Sales Representative.
Add SKU.
Add Outlet.
To Create Outlet Click on Add Outlet

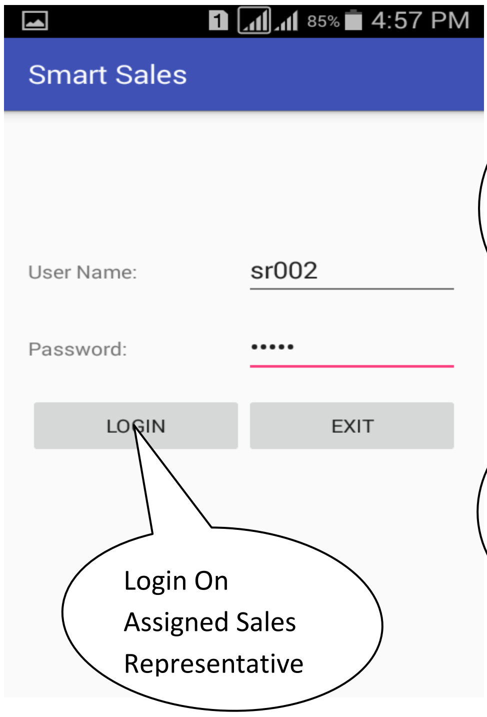
SR Id	Outlet Name	Outlet Address	Owner	Mobile	Operations	
Mahabuba Khatun	mokbul store	kallanpur	mokbul	01777866432		
Mahabuba Khatun	ROBI ENTERPRISE	kallanpur	Rabiul	01709867809		
Mahabuba Khatun	mijan stote	kallanpur	mijan	02987525656		
Mahabuba	BIJOY TELECOM	KATCHA BAZAR ROAD,	BIJOY	01897876555		


Smart Sales System
www.smartsales.tk/add_outlet.php
Smart Sales System
back to outlet page
SAVE
Type Outlet information and Click on Save Button

SATHI GENERAL STORE
kallanpur
Mithun
01912212221
Asikur Rahman

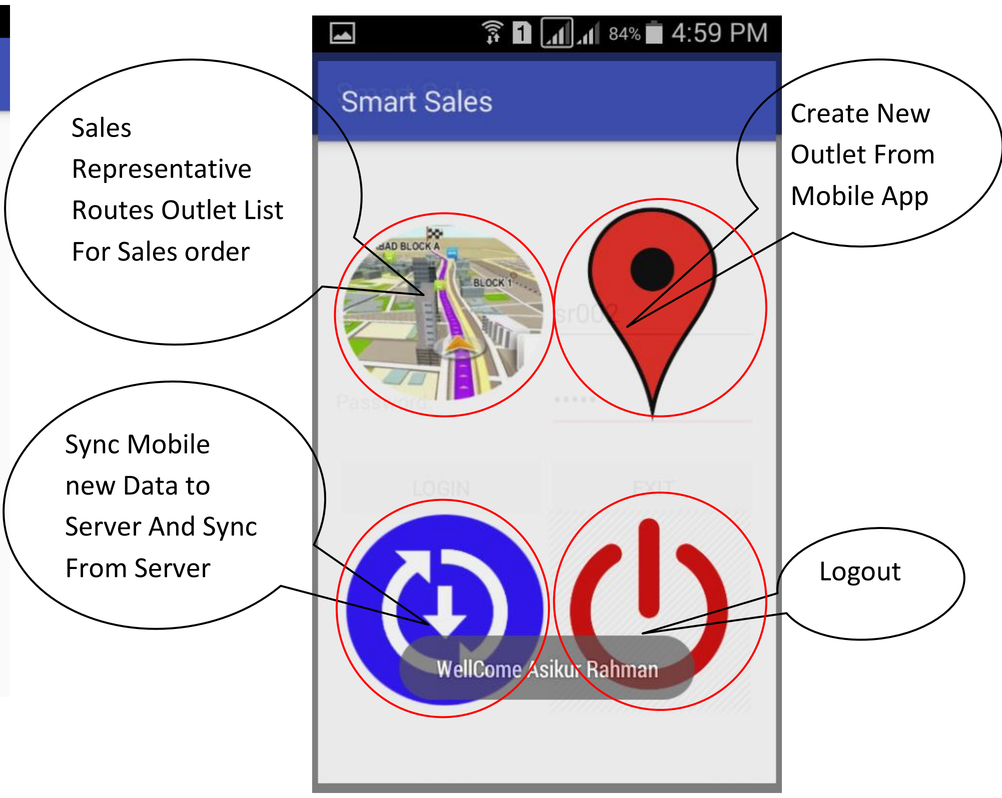

Smart Sales System
www.smartsales.tk/outlet.php
Delete Outlet Information
View Outlet Details Information
Update Outlet Information

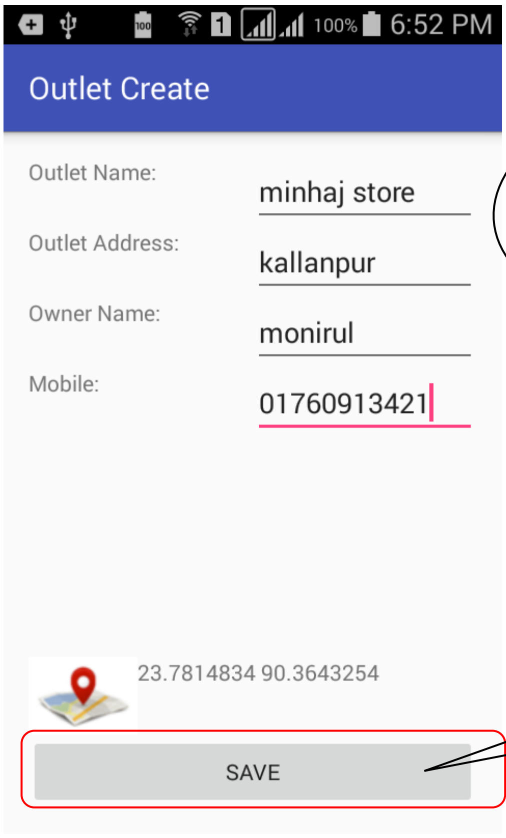
Md Mahdi	MAYAR DOA G. STORE	JOAR SHARA BAZAR	ROSHID	01909944121	✎	✖
Md Mahdi	MIZAN STORE	KA-36 SARKER ROAD NADDA	MIZAN	01934234466	✎	✖
Md Mahdi	RAKIB STORE	BAZAR ROAD NADDA	RAKIB	01865645544	✎	✖
Md Mahdi	ABCD	dhaka	raju	01928765422	✎	✖
Md. Majidur Rahman	Tromra Gen Store	dhaka, kallanpur	majid	017508725226	✎	✖
Asikur Rahman	SATHI GENERAL STORE	kallanpur	Mithun	01912212221	✎	✖



User Name: sr002

Password: •••••


84% 4:59 PM
Smart Sales
Sales Representative Routes Outlet List For Sales order
Create New Outlet From Mobile App
Sync Mobile new Data to Server And Sync From Server
Logout
WellCome Asikur Rahman



6:52 PM

Outlet Create

Outlet Name: minhaj store

Outlet Address: kallanpur

Owner Name: monirul

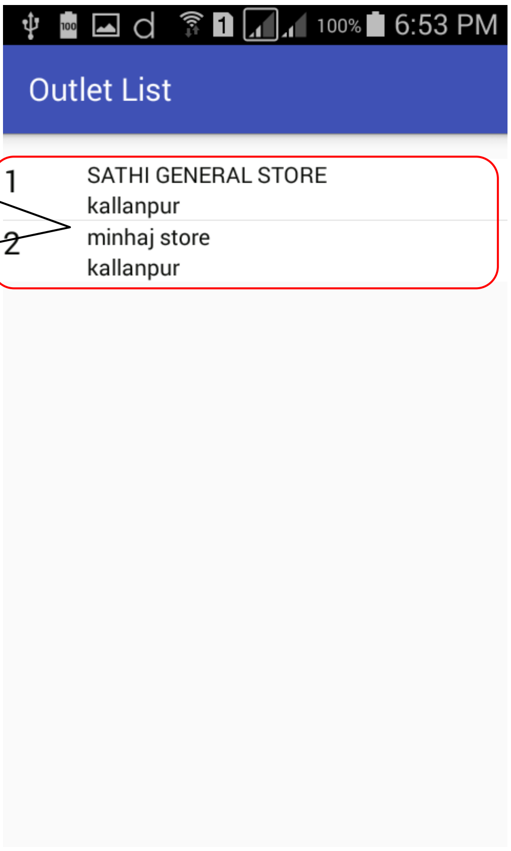
Mobile: 01760913421

23.7814834 90.3643254

SAVE

Save Outlet

View Outlet Created From Server or Mobile and Take order

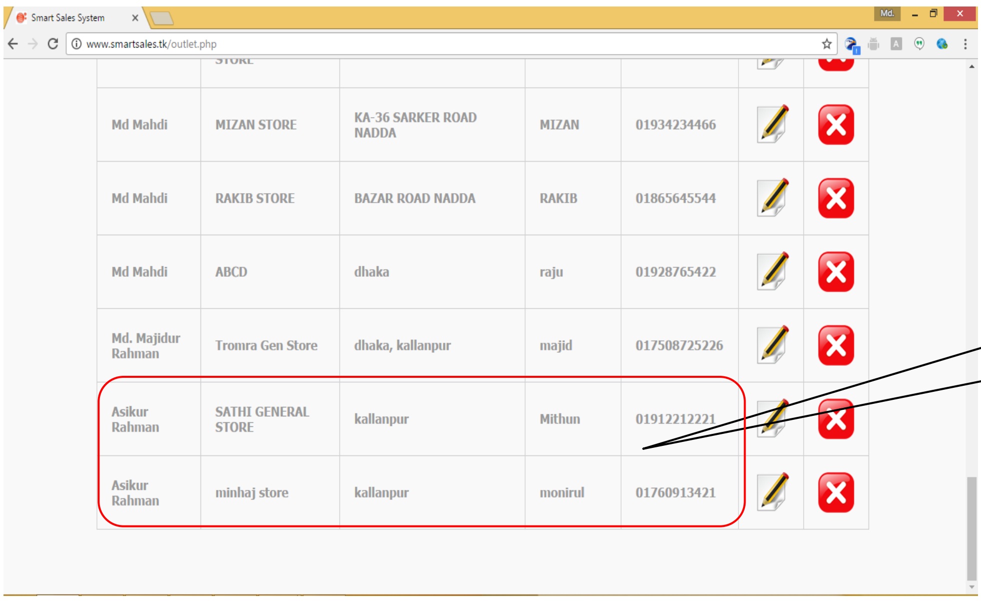


6:53 PM

Outlet List

1 SATHI GENERAL STORE
kallanpur

2 minhaj store
kallanpur



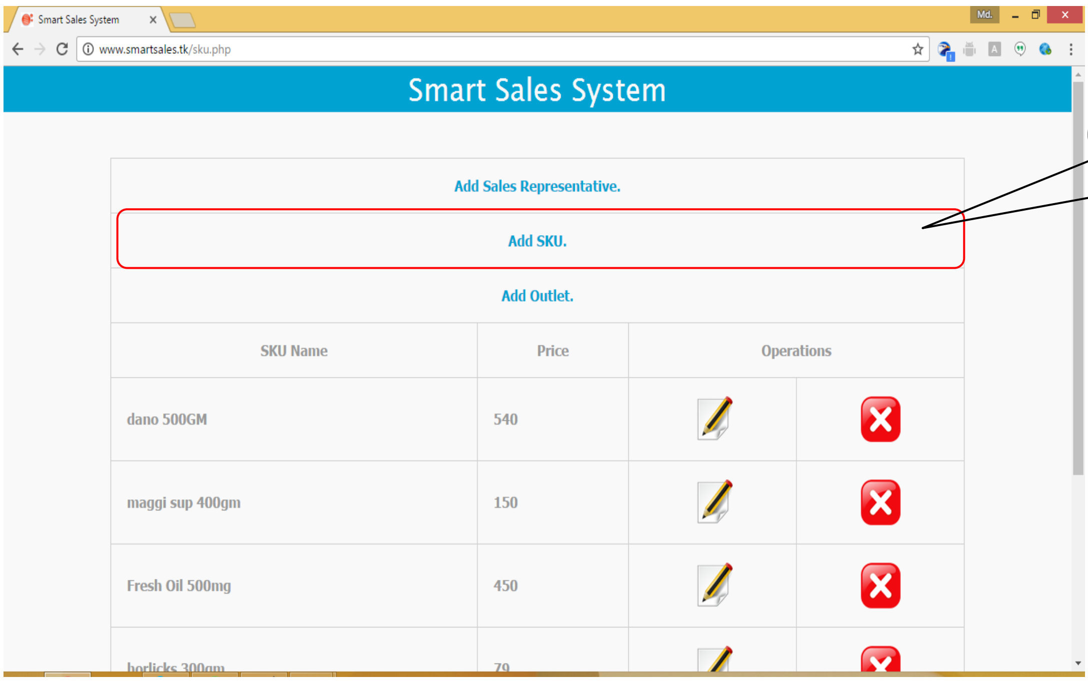
Smart Sales System

www.smartsales.tk/outlet.php

	STORE					
Md Mahdi	MIZAN STORE	KA-36 SARKER ROAD NADDA	MIZAN	01934234466		
Md Mahdi	RAKIB STORE	BAZAR ROAD NADDA	RAKIB	01865645544		
Md Mahdi	ABCD	dhaka	raju	01928765422		
Md. Majidur Rahman	Tromra Gen Store	dhaka, kallanpur	majid	017508725226		
Asikur Rahman	SATHI GENERAL STORE	kallanpur	Mithun	01912212221		
Asikur Rahman	minhaj store	kallanpur	monirul	01760913421		

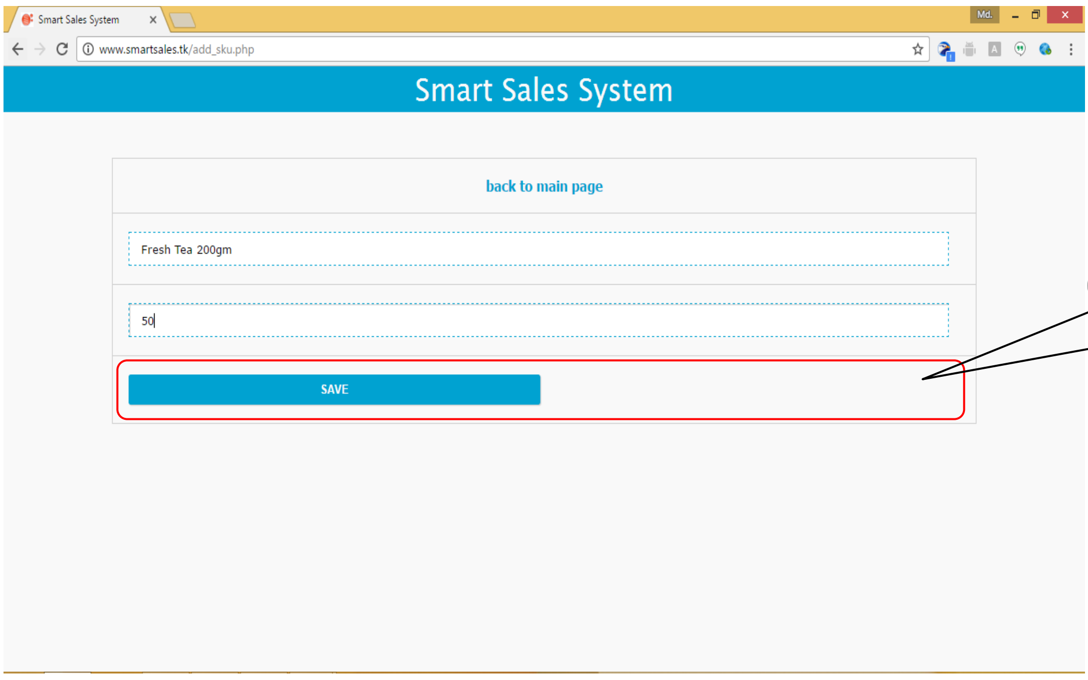
View Outlet Details Information

SKU CRUD and Order Process:

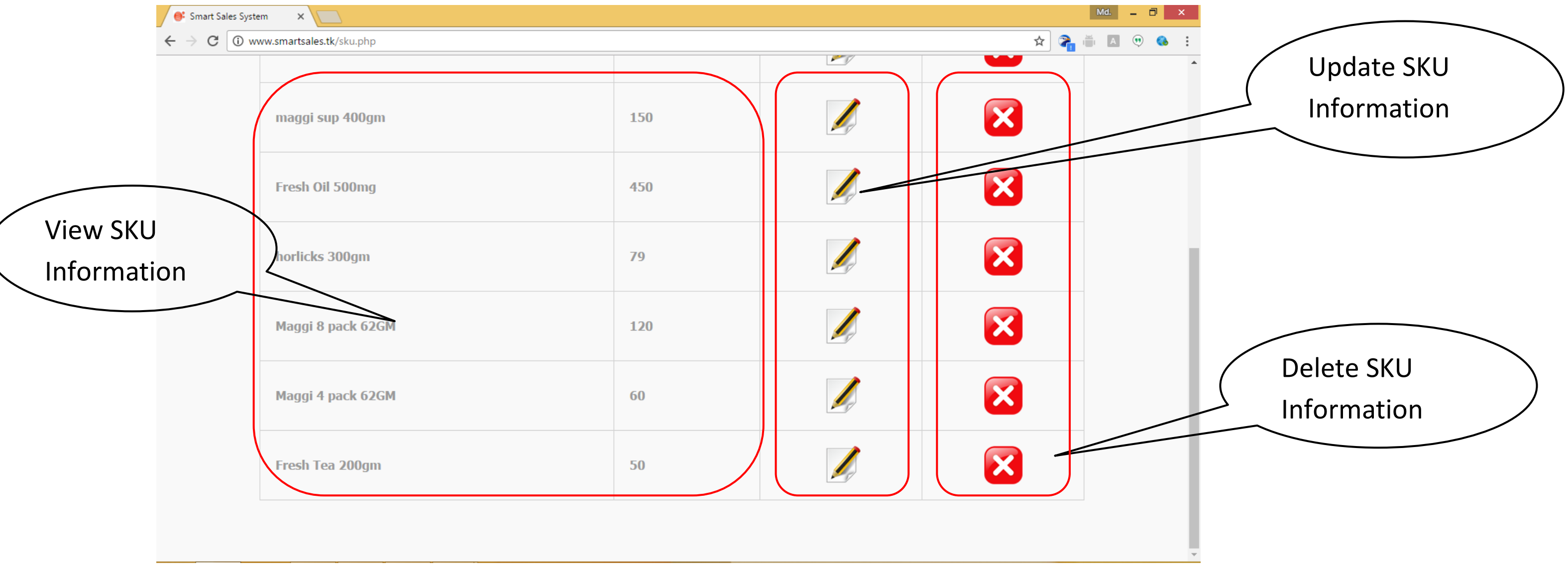


SKU Name	Price	Operations	
dano 500GM	540		
maggi sup 400gm	150		
Fresh Oil 500mg	450		
horlicks 300gm	79		

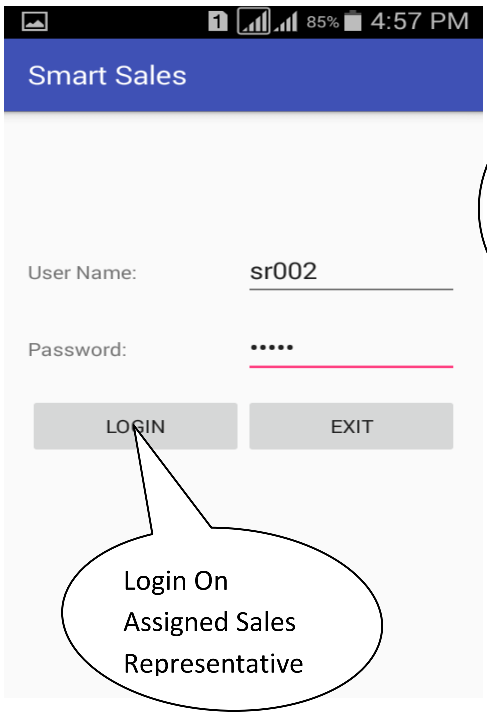
To Create a New Product click on Add SKU



Type SKU Name and Price then Click on Save

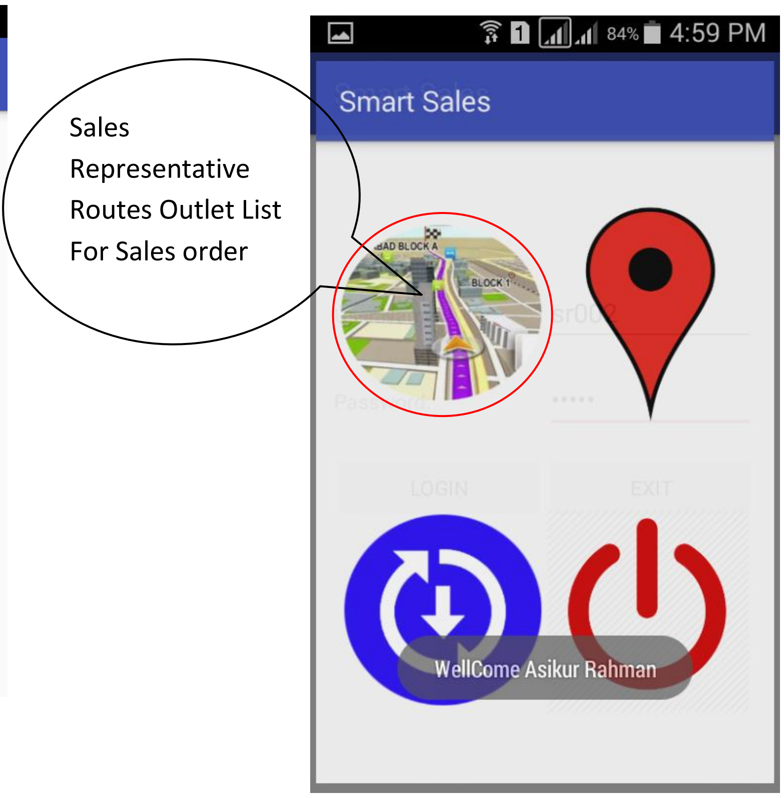


maggi sup 400gm	150	✎	✖
Fresh Oil 500mg	450	✎	✖
horlicks 300gm	79	✎	✖
Maggi 8 pack 62GM	120	✎	✖
Maggi 4 pack 62GM	60	✎	✖
Fresh Tea 200gm	50	✎	✖

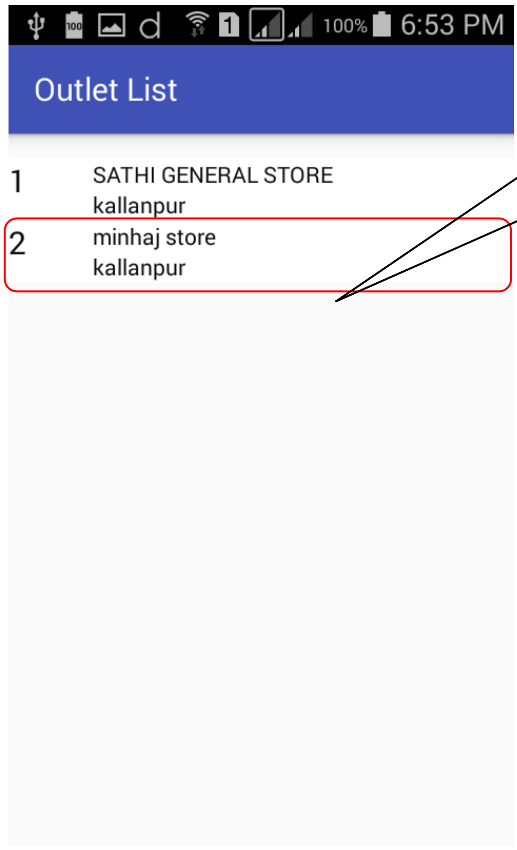


User Name: sr002

Password: •••••



WellCome Asikur Rahman

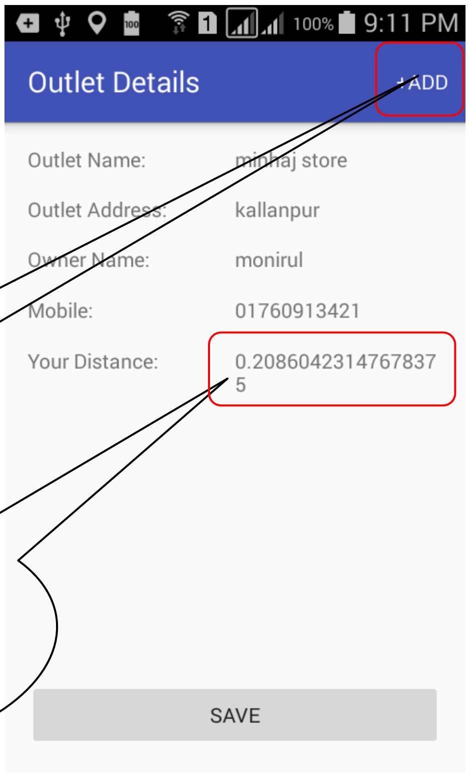

100% 6:53 PM

Outlet List

1 SATHI GENERAL STORE
kallanpur

2 minhaj store
kallanpur

To View Outlet Details and Take order click on it


100% 9:11 PM

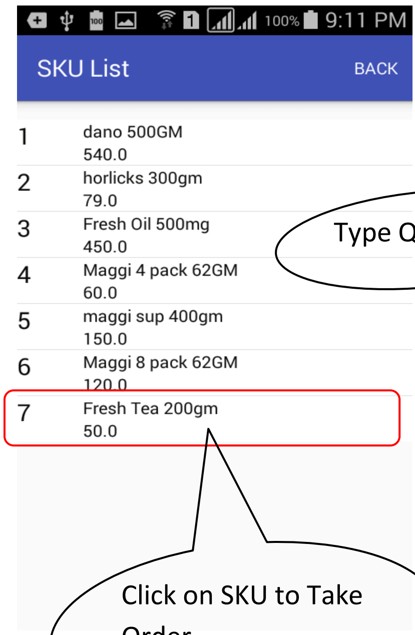
Outlet Details +ADD

Outlet Name:	minhaj store
Outlet Address:	kallanpur
Owner Name:	monirul
Mobile:	01760913421
Your Distance:	0.20860423147678375

SAVE

To Take Order Click +add

Sales Reprehensive Distance form Outlet


100% 9:11 PM

SKU List BACK

1 dano 500GM
540.0

2 horlicks 300gm
79.0

3 Fresh Oil 500mg
450.0

4 Maggi 4 pack 62GM
60.0

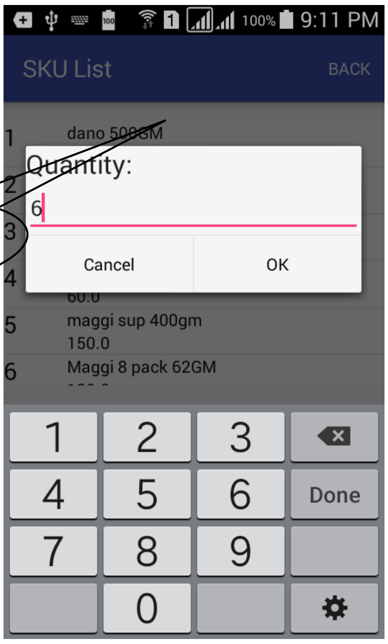
5 maggi sup 400gm
150.0

6 Maggi 8 pack 62GM
120.0

7 Fresh Tea 200gm
50.0

Click on SKU to Take Order

Type Quantity


100% 9:11 PM

SKU List BACK

1 dano 500GM

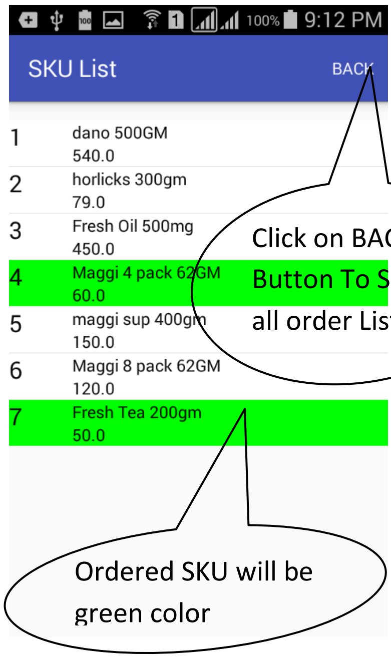
Quantity:

6

Cancel OK

5 maggi sup 400gm
150.0

6 Maggi 8 pack 62GM


100% 9:12 PM

SKU List BACK

1 dano 500GM
540.0

2 horlicks 300gm
79.0

3 Fresh Oil 500mg
450.0

4 Maggi 4 pack 62GM
60.0

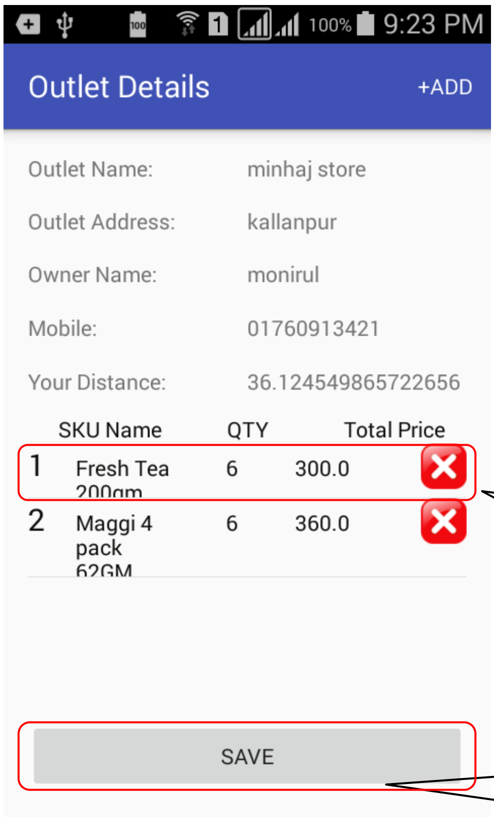
5 maggi sup 400gm
150.0

6 Maggi 8 pack 62GM
120.0

7 Fresh Tea 200gm
50.0

Click on BACK Button To See all order List

Ordered SKU will be green color



100% 9:23 PM

Outlet Details +ADD

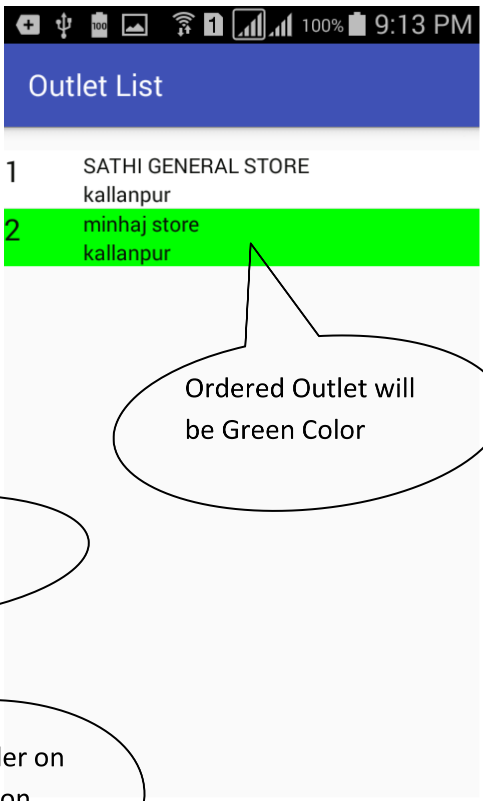
Outlet Name:	minhaj store
Outlet Address:	kallanpur
Owner Name:	monirul
Mobile:	01760913421
Your Distance:	36.124549865722656

	SKU Name	QTY	Total Price
1	Fresh Tea 200gm	6	300.0
2	Maggi 4 pack 62GM	6	360.0

SAVE

To Cancel order click on Delete

To Send Order on Server Click on Save

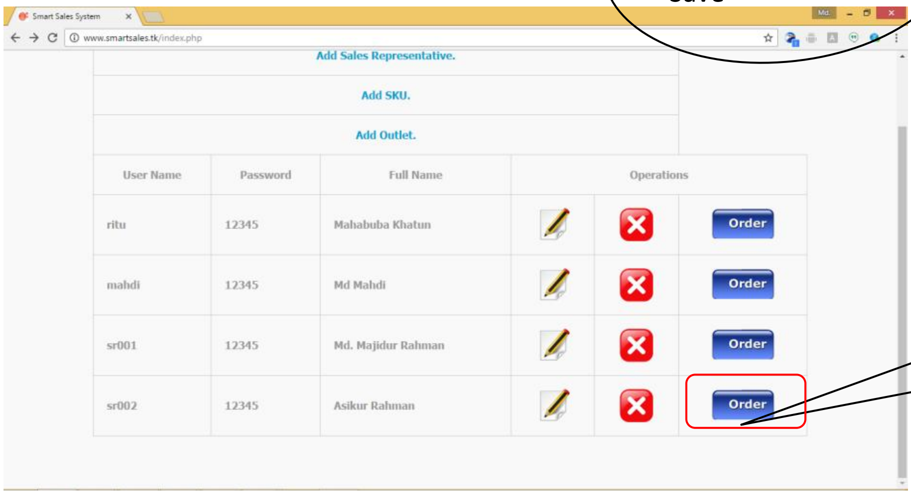


100% 9:13 PM

Outlet List

1	SATHI GENERAL STORE kallanpur
2	minhaj store kallanpur

Ordered Outlet will be Green Color



Smart Sales System

www.smartsales.tk/index.php

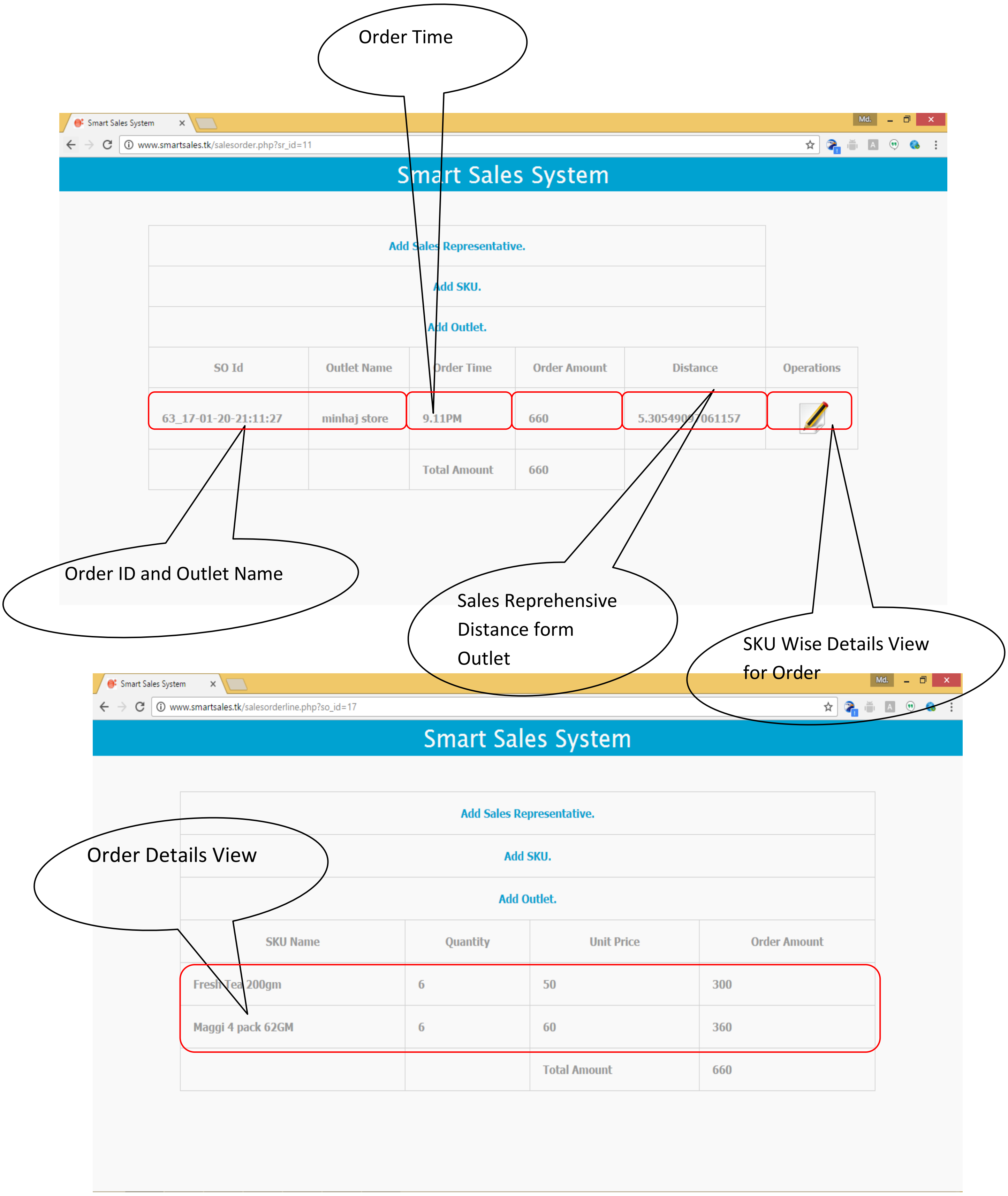
Add Sales Representative.

Add SKU.

Add Outlet.

User Name	Password	Full Name	Operations		
ritu	12345	Mahabuba Khatun			Order
mahdi	12345	Md Mahdi			Order
sr001	12345	Md. Majidur Rahman			Order
sr002	12345	Asikur Rahman			Order

To See Sales Representative wise Order Click on Order button


Order Time
Order ID and Outlet Name
Sales Reprehensive Distance form Outlet
SKU Wise Details View for Order
Order Details View

Smart Sales System

www.smartsales.tk/salesorder.php?sr_id=11

Smart Sales System

Add Sales Representative.

Add SKU.

Add Outlet.

SO Id	Outlet Name	Order Time	Order Amount	Distance	Operations
63_17-01-20-21:11:27	minhaj store	9.11PM	660	5.305490970061157	✎
		Total Amount	660		

Smart Sales System

www.smartsales.tk/salesorderline.php?so_id=17

Smart Sales System

Add Sales Representative.

Add SKU.

Add Outlet.

SKU Name	Quantity	Unit Price	Order Amount
Fresh Tea 200gm	6	50	300
Maggi 4 pack 62GM	6	60	360
		Total Amount	660

Conclusion:

The motivation of this thesis project is to build an Android based application And PHP for providing a simple. By using Android SDK and Android Studio IDE as the development environment the application is built keeping in mind about the design standards and maintainability of the code.

The main aim of my project work we have achieved that is Making a Automation System for Sales Representative Observe Sales Representative Activities.

Reference:

[1] http://www.developer.android.com/

[2] https://www.android.com/

[3] https://developer.android.com/guide/topics/manifest/manifest-intro.html

[4] https://developer.android.com/guide/platform/index.html

[5] https://developer.android.com/reference/android/location/LocationManager.html

[6] https://developer.android.com/training/volley/index.html

[7] http://stackoverflow.com/

[8] https://developer.android.com/guide/components/intents-common.html

[9] https://developer.android.com/training/sync-adapters/index.html

[10] https://udacity.com/

[11] https://developer.android.com/reference/android/content/ContentProvider.html

[12] https://www.sqlite.org/

[13] https://developer.android.com/reference/android/accounts/AccountManager.html

[14] http://www.higherpass.com/php/Tutorials/Php-Fundamentals/

[15] https://developer.android.com/design/patterns/index.html

[16] https://www.leaseweb.com/labs/2015/10/creating-a-simple-rest-api-in-php/

[17] https://developer.android.com/guide/components/activities/activity-lifecycle.html

[18] https://developer.android.com/reference/android/widget/ArrayAdapter.html