A WIFI BASED PREVALENT AUTOMATED BILLING SYSTEM

A PROJECT REPORT

Submitted by

KOWSALYA.A

in partial fulfillment for the award of the degree

of

BACHELOR OF ENGINEERING

IN

COMPUTER SCIENCE AND ENGINEERING

ST.XAVIER'S CATHOLIC COLLEGE OF ENGINEERING

CHUNKANkADAI-629003

ANNA UNIVERSITY

CHENNAI 600 025

APRIL2020

ANNA UNIVERSITY: CHENNAI 600 025

BONAFIDE CERTIFICATE

Certified that this project report titled **"A WIFI BASED PREVALENT AUTOMATED BILLING SYSTEM"** is the bonafide work of **"KOWSALYA.A (962216104060)"** who carried out the project work under my supervision.

SIGNATURE	**SIGNATURE**
Dr. R. P. Anto Kumar, M.E., Ph.D.,	Mrs.Dr.A.Subitha, B.E.,M.Tech.,Ph.D.,
HEAD OF THE DEPARTMENT	**SUPERVISOR**
Professor	Professor
Department of Computer Science	Department of Computer Science
and Engineering	and Engineering
St.Xavier's Catholic College of	St.Xavier's Catholic College of
Engineering, Chunkankadai,	Engineering, Chunkankadai,
Nagercoil-629003	Nagercoil-629003

Submitted for B.E Degree Project Work (CS6811) Viva-Voce examination held at St.Xavier's Catholic College of Engineering on ……………………

Internal Examiner **External Examiner**

ACKNOWLEDGEMENT

ACKNOWLEDGEMENT

First of all I praise the Almighty God for his presence and abundant grace in giving us knowledge, wisdom and strength to take up this project and complete on time. I express my gratitude to my parents who have been the major contribution of inspiration and encouragement to me throughout my career.

I would like to deliver our heartiest gratitude to our correspondent Rev.fr.Dr.M.Maria william, for making facilities for the successful completion of my work.

I am especially indebted to our Head of the Department Dr.R.P.Anto Kumar, M.E., Ph.D. for her keen involvement to do my project work successfully.

I convey my deepest gratitude to my guide, Dr.A.Subitha, B.E., M.tech. , Ph.D., for his motivation and guidance and also all the other staff members of our department for their timely suggestions to complete this report.

I express my gratitude and sincere thanks to our project coordinator MS.A.Bamila virgin Louis, M.E., for giving me innovative ideas and whole hearted encouragement for completing this project successfully.

ABSTRACT

In shopping malls one can see huge rush on weekends public holidays public holidays and during festive season due to the various discounts offers. Customers buy different artefacts and keep them in cart. At billing section, the cashier prepares the final bill by using barcode reader which is a time consuming procedure and consequently elongated lines can be seen at the billing center. So we proposed automated billing systems in shopping malls using android. Customer used android app to buy artefacts in shopping malls by using QR code in artefacts.

TABLE OF CONTENTS

CHAPTER 1

1. INTRODUCTION

1.1 OVERVIEW OF THE PROJECT

In shopping malls, one can see huge rush on weekends, public holidays and during festive seasons due to the various discounts offers. Customers buy different artefacts and keep them in cart. At billing section, the cashier prepares the final bill using barcode reader which is a time-consuming procedure and consequently elongated lines can be seen at the billing counter.

We proposed Wi-Fi based Prevalent Automated Billing System in Shopping Malls using Android. Customer uses android based app to buy artefacts in shopping malls by scanning QR code in are facts.

1.2 Web Design

Responsive web design is a method of developing your website so that it automatically adjusts to the screen size and device used to view the site. This ensures a consistent user experience and allows you to replicate your site's content without reducing it for smaller screens, as you need to do with a mobile site. Every site that ZAG builds is built using responsive design.

A Content Management System (CMS) is a Web application that uses a database (usually MySQL) or other methods to create, edit, and store HTML content in a manageable way. Content is created and edited on the web in an administration portion of the web application (referred to as the Backend). The resulting content is then displayed to the viewers on the regular site (referred to as the Frontend).

1.2.1 Software Development

Software Development is the computation lifecycle procedure involving analysing, designing, developing, implementing, testing, delivering and maintaining a set of programming which can help the human to utilize the computer in a smart way.

We provide application and software development service for both domestic and international customers. Software is varied in size and requirements of the clients.

1.3 Android App Development

There are more than 1.2 Million Android Apps available in Google Play and near to 30 Thousand Apps added every month. With Android developing as market leaders in mobile platform, it becomes need to have your App in Android more than iOS. The hardware which runs Android are also available in vast ranges which makes them a most applicable platform to use for your start up ideas or for your business.

Android is a preferred platform for supreme business houses because of the availability of cheap devices and the easy way to push apps into devices. Many restaurants, retail outlets, factories, warehouses, hospitals etc.. have started using android apps for their internal applications.

CHAPTER 2

LITERATURE SURVEY

Naik, et Al(2016) in their paper a eifi based automated billing system proposed a method to detect the bar code reader through bar code reader.the sensing subsystem senses the artefacts using barcode reader.at the same time location also tracked with the help of network with JavaScript programming. Then the communication subsystem transfer the exact location and value of the detected app through the bar code reader.

Ganapriya,et Al.,(2017)in this paper "a WiFi based automated billing system"proposed a way for an innovative methods to prevent automated billing system.In this method,the advanced sensor is used to track nad updated the artefacts.Using the data's obtained more damaged area can be prioritized and damage control can be reduced.

Yoki,et Al.,(2012) I this paper "automated billing system" proposed a project for the vehicle that can be equipped with sensor to collect data of the small artefacts.

CHAPTER 3

3. SYSTEM SPECIFICATION

The purpose of the software requirement specification is to produce the specification of the task and also establish complete information about the requirement, behaviour and also the other constraint like functional performance and so on. The main aim of the software requirement specification is to completely specify the technical requirements for the software product in a concise and in un ambiguous manner.

3.1 HARDWARE SPECIFICATION

• CPU type	:	Intel i3
• Clock speed	:	2.93 GHz
• Ram size	:	4 GB
• Hard disk capacity	:	500 GB
• Monitor type	:	18.5 Inch color monitor
• Keyboard type	:	Standard Keyboard
• Mouse	:	Standard Mouse

3.2 SOFTWARE SPECIFICATION

• O/S	:	Windows XP above
• Frontend	:	Android
• Backend	:	MySQL
• Middleware	:	PHP

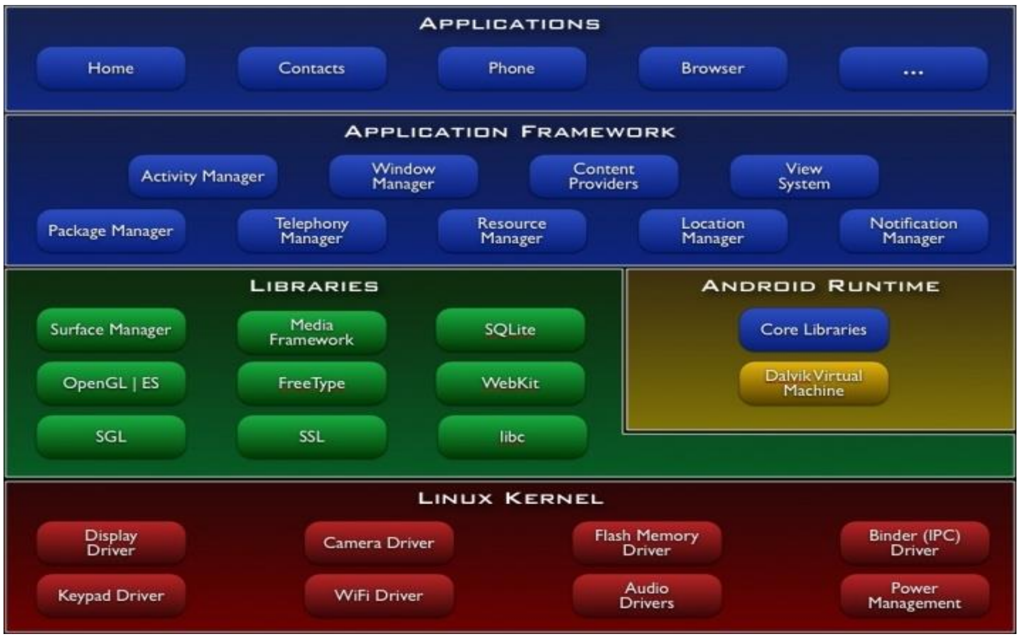
3.3 LANGUAGE DESCRIPTION

ANDROID:

Android is a software stack for mobile devices that includes an operating system, middleware and key applications. Google Inc. purchased the initial developer of the software, Android Inc., in 2005. Android's mobile operating system is based on the Linux kernel. Google and other members of the Open Handset Alliance collaborated on Android's development and release.

The Android SDK provides the tools and APIs necessary to begin developing applications Android platform using the Java programming language. Android has a large community of developers writing applications ("apps") that extend the functionality of the devices. There are currently over 250,000 apps available for Android.

ANDROID ARCHITECTURE



Libraries

Android includes a set of C/C++ libraries used by various components of the Android system. These capabilities are exposed to developers through the Android application framework. Some of the core libraries are listed below:

- **System C library -** a BSD-derived implementation of the standard C system library (libc), tuned for embedded Linux-based devices
- **Media Libraries -** based on PacketVideo'sOpenCORE; the libraries support playback and recording of many popular audio and video formats, as well as static image files, including MPEG4, H.264, MP3, AAC, AMR, JPG, and PNG
- **Surface Manager -** manages access to the display subsystem and seamlessly composites 2D and 3D graphic layers from multiple applications
- **LibWebCore -** a modern web browser engine which powers both the Android browser and an embeddable web view
- **SGL -** the underlying 2D graphics engine
- **3D libraries -** an implementation based on OpenGL ES 1.0 APIs; the libraries use either hardware 3D acceleration (where available) or the included, highly optimized 3D software rasterizer
- **FreeType -** bitmap and vector font rendering

Android Runtime

Android includes a set of core libraries that provides most of the functionality available in the core libraries of the Java programming language. Every Android application runs in its own process, with its own instance of the Dalvik virtual machine. Dalvik has been written so that a device can run multiple VMs efficiently.

The Dalvik VM executes files in the Dalvik Executable (.dex) format which is optimized for minimal memory footprint. The VM is register-based, and runs classes compiled by a Java language compiler that have been transformed into the. dex format by the included "dx" tool.

Linux Kernel

Android relies on Linux version 2.6 for core system services such as security, memory management, process management, network stack, and driver model. The kernel also acts as an abstraction layer between the hardware and the rest of the software stack. The Linux kernel is an operating system kernel used by the Linux family of Unix-likeoperating systems. It is one of the most prominent examples of free and open source software.

The Linux kernel is released under the GNU General Public License version 2 (GPLv2),(plus some firmware images with various licenses), and is developed by contributors worldwide. Day-to-day development takes place on the Linux kernel mailing list.

At Google, the team led by Rubin developed a mobile device platform powered by the Linux kernel. Google marketed the platform to handset makers and carriers on the premise of providing a flexible, upgradable system. Google had lined up a series of hardware component and software partners and signaled to carriers that it was open to various degrees of cooperation on their part.

Speculation about Google's intention to enter the mobile communications market continued to build through December 2006.Reports from the BBC and The Wall Street Journal noted that Google wanted its search and applications on mobile phones and it was working hard to deliver that. Print and online media outlets soon reported rumors that Google was developing a Google-branded handset. Some speculated that as Google was defining technical specifications, it was showing prototypes to cell phone manufacturers and network operators.

Hardware running Android

The main supported platform for Android is the ARM architecture. The Android OS can be used as an operating system for cellphones, netbooks and tablets, including the Dell Streak, Samsung Galaxy Tab, TV and other devices. The first commercially available phone to run the Android operating system was the HTC Dream, released on 22 October 2008. In early 2010 Google collaborated with HTC to launch its flagship Android device, the Nexus One. This was followed later in 2010 with the Samsung-made Nexus S.

The early feedback on developing applications for the Android platform was mixed.Issues cited include bugs, lack of documentation, inadequate QA infrastructure, and no public issue-tracking system. (Google announced an issue tracker on 18 January 2008.) In December 2007, Merge Lab mobile startup founder Adam MacBeth stated, "Functionality is not there, is poorly documented or just doesn't work... It's clearly not ready for prime time."Despite this, Android-targeted applications began to appear the week after the platform was announced. The first publicly available application was the Snake Game the Android Dev. Phone is a SIM-unlocked and hardware-unlocked device that is designed for advanced developers. While developers can use regular consumer devices purchased at retail to test and use their applications, some developers may choose not to use a retail device, preferring an unlocked or no-contract device.

The Android Software Development Kit (SDK) includes a comprehensive set of development tools. These include a debugger, libraries, a handset emulator (based on QEMU), documentation, sample code, and tutorials. The SDK is downloadable on the android developer website. Currently supported development platforms include computers running Linux (any modern desktop Linux distribution), Mac OS X 10.4.9 or later, Windows XP or later. The officially supported integrated development environment (IDE) is Eclipse (currently 3.5 or 3.6) using the Android Development Tools (ADT) Plugin, though developers may use any text editor to edit Java and XML files then use command line tools (Java Development Kit and Apache Ant are required) to create, build and debug Android applications as well as control attached Android devices (e.g., triggering a reboot, installing software package(s) remotely). Android applications are packaged in. apk format and stored under /data/app folder on the Android OS (the folder is accessible to root user only for security reasons). APK package contains. dex files(compiled byte code files called Dalvik executable), resource files, etc.

Android Operation System

Android is an operating system based on Linux with a Java programming interface. It provides tools, e.g. a compiler, debugger and a device emulator as well as its own Java Virtual machine (Dalvik Virtual Machine - DVM). Android is created by the Open Handset Alliance which is led by Google.

Android uses a special virtual machine, e.g. the Dalvik Virtual Machine. Dalvik uses special bytecode. Therefore, you cannot run standard Java bytecode on Android. Android provides a tool "dx" which allows converting Java Class files into "dex" (Dalvik Executable) files. Android applications are packed into an. apk (Android Package) file by the program "aapt" (Android Asset Packaging Tool) to simplify development Google provides the Android Development Tools (ADT) for Eclipse.

PHP

PHP started out as a small open source project that evolved as more and more people found out how useful it was. Rasmus Lerdorf unleashed the first version of PHP way back in 1994.

- PHP is a recursive acronym for "PHP: Hypertext Preprocessor".
- PHP is a server side scripting language that is embedded in HTML. It is used to manage dynamic content, databases, session tracking, even build entire e-commerce sites.
- It is integrated with a number of popular databases, including MySQL, PostgreSQL, Oracle, Sybase, Informix, and Microsoft SQL Server.

- PHP is pleasingly zippy in its execution, especially when compiled as an Apache module on the Unix side. The MySQL server, once started, executes even very complex queries with huge result sets in record-setting time.
- PHP supports a large number of major protocols such as POP3, IMAP, and LDAP. PHP4 added support for Java and distributed object architectures (COM and CORBA), making n-tier development a possibility for the first time.
- PHP is forgiving: PHP language tries to be as forgiving as possible.
- PHP Syntax is C-Like.

Common uses of PHP

PHP performs system functions, i.e. from files on a system it can create, open, read, write, and close them.

- PHP can handle forms, i.e. gather data from files, save data to a file, thru email you can send data, return data to the user.
- You add, delete, modify elements within your database thru PHP.
- Access cookies variables and set cookies.
- Using PHP, you can restrict users to access some pages of your website.
- It can encrypt data.

Characteristics of PHP

Five important characteristics make PHP's practical nature possible:

- ✓ Simplicity
- ✓ Efficiency
- ✓ Security
- ✓ Flexibility
- ✓ Familiarity

PHP Environment Setup

In order to develop and run PHP Web pages three vital components need to be installed on your computer system.

Web Server

PHP will work with virtually all Web Server software, including Microsoft's Internet Information Server (IIS) but then most often used is freely available Apache Server. Download Apache for free here:http://httpd.apache.org/download.cgi

Database

PHP will work with virtually all database software, including Oracle and Sybase but most commonly used is freely available MySQL database. Download MySQL for free here: http://www.mysql.com/downloads/index.html

PHP Parser

In order to process PHP script instructions a parser must be installed to generate HTML output that can be sent to the Web Browser. This tutorial will guide you how to install PHP parser on your computer.

PHP Variables

The main way to store information in the middle of a PHP program is by using a variable.

Here are the most important things to know about variables in PHP.

- All variables in PHP are denoted with a leading dollar sign ($).
- The value of a variable is the value of its most recent assignment.
- Variables are assigned with the = operator, with the variable on the left-hand side and the expression to be evaluated on the right.
- Variables can, but do not need, to be declared before assignment.
- Variables in PHP do not have intrinsic types - a variable does not know in advance whether it will be used to store a number or a string of characters.
- Variables used before they are assigned have default values.
- PHP does a good job of automatically converting types from one to another when necessary.
- PHP variables are Perl-like.

PHP has a total of eight data types which we use to construct our variables:

- **Integers:** are whole numbers, without a decimal point, like 4195.
- **Doubles:** are floating-point numbers, like 3.14159 or 49.1.
- **Booleans:** have only two possible values either true or false.
- **NULL:** is a special type that only has one value: NULL.
- **Strings:** are sequences of characters, like 'PHP supports string operations.'
- **Arrays:** are named and indexed collections of other values.
- **Objects:** are instances of programmer-defined classes, which can package up both other kinds of values and functions that are specific to the class.
- **Resources:** are special variables that hold references to resources external to PHP (such as database connections).

PHP Coding Standard

Every company follows a different coding standard based on their best practices. Coding standard is required because there may be many developers working on different modules so if they will start inventing their own standards then source will become very un-manageable and it will become difficult to maintain that source code in future.

Here are several reasons why to use coding specifications:

- Your peer programmers have to understand the code you produce. A coding standard acts as the blueprint for all the team to decipher the code.
- Simplicity and clarity achieved by consistent coding saves you from common mistakes.
- If you revise your code after some time then it becomes easy to understand that code.
- Its industry standard to follow a particular standard to being more quality in software.

There are few guidelines which can be followed while coding in PHP.

- **Indenting and Line Length** - Use an indent of 4 spaces and don't use any tab because different computers use different setting for tab. It is recommended to keep lines at approximately 75-85 characters long for better code readability.
- **Control Structures** - These include if, for, while, switch, etc. Control statements should have one space between the control keyword and opening parenthesis, to

distinguish them from function calls. You are strongly encouraged to always use curly braces even in situations where they are technically optional.

6.3.2 MySQL

A database is a separate application that stores a collection of data. Each database has one or more distinct APIs for creating, accessing, managing, searching and replicating the data it holds.Other kinds of data stores can be used, such as files on the file system or large hash tables in memory but data fetching and writing would not be so fast and easy with those types of systems. So nowadays, we use relational database management systems (RDBMS) to store and manage huge volume of data. This is called relational database because all the data is stored into different tables and relations are established using primary keys or other keys known as foreign keys.

A Relational DataBase Management System (RDBMS) is a software that:

- Enables you to implement a database with tables, columns and indexes.
- Guarantees the Referential Integrity between rows of various tables.
- Updates the indexes automatically.
- Interprets an SQL query and combines information from various tables.

RDBMS Terminology

Before we proceed to explain MySQL database system, let's revise few definitions related to database.

Database: A database is a collection of tables, with related data.

Table: A table is a matrix with data. A table in a database looks like a simple spreadsheet.

Column: One column (data element) contains data of one and the same kind, for example the column postcode.

Row: A row (= tuple, entry or record) is a group of related data, for example the data of one subscription.

Redundancy: Storing data twice, redundantly to make the system faster.

Primary Key: A primary key is unique. A key value can not occur twice in one table. With a key, you can find at most one row.

Foreign Key: A foreign key is the linking pin between two tables.

Compound Key: A compound key (composite key) is a key that consists of multiple columns, because one column is not sufficiently unique.

Index: An index in a database resembles an index at the back of a book.

Referential Integrity: Referential Integrity makes sure that a foreign key value always points to an existing row.

MySQL Database

MySQL is a fast, easy-to-use RDBMS being used for many small and big businesses. MySQL is developed, marketed, and supported by MySQL AB, which is a Swedish company. MySQL is becoming so popular because of many good reasons:

- MySQL is released under an open-source license. So you have nothing to pay to use it.
- MySQL is a very powerful program in its own right. It handles a large subset of the functionality of the most expensive and powerful database packages.
- MySQL uses a standard form of the well-known SQL data language.
- MySQL works on many operating systems and with many languages including PHP, PERL, C, C++, JAVA, etc.
- MySQL works very quickly and works well even with large data sets.
- MySQL is very friendly to PHP, the most appreciated language for web development.
- MySQL supports large databases, up to 50 million rows or more in a table. The default file size limit for a table is 4GB, but you can increase this (if your operating system can handle it) to a theoretical limit of 8 million terabytes (TB).
- MySQL is customizable. The open-source GPL license allows programmers to modify the MySQL software to fit their own specific environments.

CHAPTER 4

4. SYSTEM ANALYSIS

Systems development can generally be thought of as having two major components: Systems analysis and Systems design. System design is the process of planning a new business system or one to replace or complement an existing system. But before this planning can be done, we must thoroughly understand the old system and determine how computers can best be used to make its operation more effective. System analysis, then, is the process of gathering and interpreting facts, diagnosing problems and using the information to recommend improvements to the system. Here the existing system was studied thoroughly by the collection of relevant data about the system. Then the proposed system was analysed thoroughly by the needs

4.1 EXISTING SYSTEM

The Existing System does not support direct selling process to merchant, So Existing system facing many unwanted problems like broker commission.

Disadvantages

- One of the main difficulties is that consumers have to stand in elongated queues for billing.
- Another problem is, many customers buy household items on a financial plan. At most of the instances, it is at the end of acquiring goods customers recognize the total bill amount. It might be possible that the total bill amount is more than their total budget

4.2 PROPOSED SYSTEM

I proposed Wi-Fi based Prevalent Automated Billing System in Shopping Malls using Android. Customer uses android based app to buy artefacts in shopping malls by scanning QR code in arefacts. Customer can set their own budget for purchase. Customer scan the artefacts using the QR code add to trolley. Application will show the price against number of items in the list and show subtotal of item.

Customer can remove the artefacts using single click. After selecting the artefacts, Customer pay the bill through application and confirm the purchase using confirm button in the application. The selected artefacts list will transfer to centralised system using Wi-Fi. Casher prints the bill and check the artefacts purchased and allow customer to leave. On the other hands, QR code generated each artefacts and data related artefacts such product price, name, measure managed in centralised server.

Advantages

- Time wastage is reduced.
- Minimum number of employee enough.
- Secure shopping system

4.3 FEASIBILITY STUDY

Feasibility study aims to objectively and rationally uncover the strengths and weaknesses of an existing business or proposed venture, opportunities and threats present in the environment, the resources required to carry through, and ultimately the prospects for success. In its simplest terms, the two criteria to judge feasibility are cost required and value to be attained. A well-designed feasibility study should provide a historical background of the business or project, a description of the product or service, accounting statements, details of the operations and management, marketing research and policies, financial data, legal requirements and tax obligations. Generally, feasibility studies precede technical development and project implementation.

A feasibility study evaluates the project's potential for success; therefore, perceived objectivity is an important factor in the credibility of the study for potential investors and lending institutions. It must therefore be conducted with an objective, unbiased approach to provide information upon which decisions can be based.

- Technical Feasibility

- Economic Feasibility
- Operation Feasibility

Technical feasibility

This assessment is based on an outline design of system requirements, to determine whether the company has the technical expertise to handle completion of the project. When writing a feasibility report, the following should be taken to consideration:

- The part of the business being examined
- The human and economic factor
- The possible solutions to the problem

The technical feasibility assessment is focused on gaining an understanding of the present technical resources of the organization and their applicability to the expected needs of the proposed system. It is an evaluation of the hardware and software and how it meets the need of the proposed system.

Economic feasibility

The purpose of the economic feasibility assessment is to determine the positive economic benefits to the organization that the proposed system will provide. It includes quantification and identification of all the benefits expected. This assessment typically involves a cost/ benefits analysis.

Operational feasibility

Operational feasibility is a measure of how well a proposed system solves the problems, and takes advantage of the opportunities identified during scope definition and how it satisfies the requirements identified in the requirements analysis phase of system development.

To ensure success, desired operational outcomes must be imparted during design and development. These include such design-dependent parameters such as reliability, maintainability, supportability, usability, productivity, disposability, sustainability, affordability and others. A system design and development requires appropriate and timely application of engineering and management efforts to meet the previously mentioned parameters.

Input Design

Input design is the link that ties the information system into the world of its users. The input design involves determining the inputs, validating the data, minimizing the data entry and provides a multi-user facility. Inaccurate inputs are the most common cause of errors in data processing. Errors entered by the data entry operators can be controlled by input design. The user-originated inputs are converted to a computer based format in the design. Input data are collected and organized into groups of similar data. Once identified, the appropriate input are selected for processing. All the input data are validated and if any data violates any conditions, it is transferred to the appropriate tables in the database. In this system the details are to be entered at the time of registration. A page is designed for this purpose which is user friendly and easy to use. The design is done such that the users get appropriate messages when exceptions occur.

Output Design

Computer output is the most important and direct source of information to the user. Output design is a very important phase since the output needs to be in an efficient manner. Efficient and intelligible output design improves the system relationship with the user and helps in decision making. Allowing the user to view the sample screen is important because the user is the ultimate judge of the quality of output. The output module of this system is the selected notifications.

CHATER 5

5. SYSTEM DESIGN

The most creative and challenging phase of the system life cycle is the system sign. Design commences, once the logical model of the existing system is available. Design begins by using identified system problem as a basis for developing objective for the new system. System design is the process of defining the architecture, modules, interfaces, and data for a system to satisfy specified requirements. System design could be seen as the application of the systems theory to satisfy specified requirements. System design could be seen as the application of systems theory to product development. Design phase proceeds in two steps, logical design and physical design.

5.1ARCHITECTURE DIAGRAM

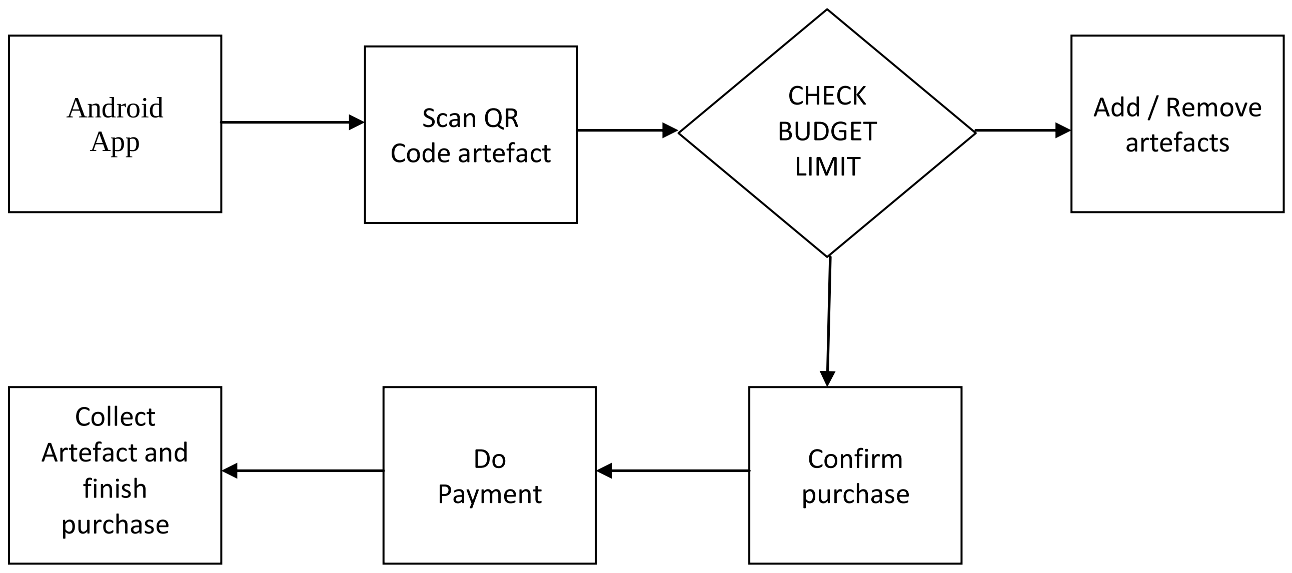


Fig 4.1 Architecture Diagram

CHAPTER 6

6. SYSTEM IMPLEMENTATION

6.1 MODULES DESCRIPTION

This project consists of the following modules:

- Admin Module:
- Android App Module

6.1.1 Product

This module is used to farmer can able to production details such product name and product images, after adding product, farmer may add sub product belongs to their parent product such as sub product name, image, quantity, price and expiry date.

6.1.2 FCM Module

Firebase Cloud Messaging (FCM), formerly known as Google Cloud Messaging (GCM), is a cross-platform cloud solution for messages and notifications for Android, iOS, and web applications, which currently can be used at no cost.

Firebase Cloud Messaging (FCM) is a cross-platform messaging solution that lets you reliably deliver messages at no cost.

Using FCM, you can notify a client app that new email or other data is available to sync. You can send notification messages to drive user re-engagement and retention. For use cases such as instant messaging, a message can transfer a payload of up to 4KB to a client app.

6.1.3 Chat Module

This module is providing communication services between Farmer and Merchant, here two ways to communicate. Instant chat and Selected Chats

6.1.4.4 Search Stock

This module section done by merchant, Merchant can able to search their favourite's products from Single window. If we click any product, he/she get details about selected product. Then they may able to add to CART area.

6.1.5 Cart Module

After adding all products to their CART area,

CHAPTER 7

7. SYSTEM TESTING

The process is performing a variety of tests on a system to explore functionality or to identify problems. System testing is usually required before and after a system is put in place. A series of systematic procedures are referred to while testing is being performed. These procedures tell the tester how the system should perform and where common mistakes may be found. Testers usually try to "break the system" by entering data that may cause the system to malfunction or return incorrect information. For example, a tester may put in a city in a search engine designed to only accept states, to see how the system will respond to the incorrect input.

Usually software is only one element of a larger computer based system. Ultimately, software is interfaced with other software/Hardware system. System Testing is actually a series of different tests whose sole purpose is to exercise the full computer based system.

System testing falls under the black box testing category of software testing. White box testing is the testing of the internal workings or code of a software application. In contrast, black box or system testing is the opposite. System Testing involves the externals working of the software from the user’s perspective.

- In system testing the behavior of whole system/product is tested as defined by the scope of the development project or product.
- System testing is carried out by specialist’s testes or independent testers.

The purpose of testing is to discover errors. Testing is the process of trying to discover every conceivable fault or weakness in a work product. It provides a way to check the functionality of components, sub-assemblies, assemblies and/or a finished product. It is the process of exercising software with the intent of ensuring that the Software system meets its requirements and user expectations and does not fail in an unacceptable manner. There are various types of test. Each test type addresses a specific testing requirement

Goal for Testing

The famous statement by Dijikstra's (in Dahl et al. 1972) is a perfect synthesis of the goal of the testing. If the result delivered by the system are different from the expected ones in just one case, in this unequally shows that the system is incorrect: by contrast, a correct behaviour of the system on a finite number of cases does not guarantee correctness in the general case. For instance. It could have built a program that behaves properly for even integer numbers but not odd number. Clearly, any number of tests will even input values will face to show the error.

Types of Testing

Unit Testing

In computer programming, unit testing is a software testing method by which individual units of source code, sets of one or more computer program modules together with associated control data, usage procedures, and operating procedures, are tested to determine whether they are fit for use. Intuitively, one can view a unit as the smallest testable part of an application. In procedural programming, a unit could be an entire module, but it is more commonly an individual function or procedure. In object-oriented programming, a unit is often an entire interface, such as a class, but could be an individual method. Unit tests are short code fragments created by programmers or occasionally by white box testers during the development process. It forms the basis for component testing.

Ideally, each test case is independent from the others. Substitutes such as method stubs, mock objects, fakes, and test harnesses can be used to assist testing a module in isolation. Unit tests are typically written and run by software developers to ensure that code meets its design and behaves as intended.

Unit testing provides a sort of living documentation of the system. Developers looking to learn what functionality is provided by a unit, and how to use it, can look at the unit tests to gain a basic understanding of the unit's interface (API).Unit test cases embody characteristics that are critical to the success of the unit. These characteristics can indicate appropriate/inappropriate use of a unit as well as negative behaviours that are to be trapped by the unit. A unit test case, in and of itself, documents these critical characteristics, although many software development environments do not rely solely upon code to document the product in development.

When software is developed using a test-driven approach, the combination of writing the unit test to specify the interface plus the refactoring activities performed after the test is passing, may take the place of formal design. Each unit test can be seen as a design element specifying classes, methods, and observable behaviour.

Unit testing involves the design of test cases that validate that the internal program logic is functioning properly, and that program inputs produce valid outputs. All decision branches and internal code flow should be validated. It is the testing of individual software units of the application .it is done after the completion of an individual unit before integration. This is a structural testing, that relies on knowledge of its construction and is invasive.

Black Box Testing

Black-box testing is a method of software testing that examines the functionality of an application without peering into its internal structures or workings. This method of test can be applied to virtually every level of software. It typically comprises most if not all higher level testing, but can also dominate unit testing as well.

Specific knowledge of the application's code/internal structure and programming knowledge in general is not required. The tester is aware of what the software is supposed to do but is not aware of how it does it. For instance, the tester is aware that a particular input returns a certain, invariable output but is not aware of how the software produces the output in the first place.

Test cases are built around specifications and requirements, i.e., what the application is supposed to do. Test cases are generally derived from external descriptions of the software, including specifications, requirements and design parameters. Although the tests used are primarily functional in nature, non-functional tests may also be used.

White Box Testing

White-box testing is a method of testing the application at the level of the source code. These test cases are derived through the use of the design techniques mentioned above: control flow testing, data flow testing, branch testing, path testing, statement coverage and decision coverage as well as modified condition/decision coverage. White-box testing is the use of these techniques as guidelines to create an error free environment by examining any fragile code. These White-box testing techniques are the building blocks of white-box testing, whose essence is the careful testing of the application at the source code level to prevent any hidden errors later on. These different techniques exercise every visible path of the source code to minimize errors and create an error-free environment. The whole point of white box testing is the ability to know which line of the code is being executed and being able to identify what the correct output should be.

1. Unit testing. White box testing is done during unit testing to ensure that the code is working as intended, before any integration happens with previously tested code. White-box testing during unit testing catches any defects early on and aids in any defects that happen later on after the code is integrated with the rest of the application and therefore prevents any type of errors later on,

2. Integration testing. White-box testing at this level is written to test the interactions of each interface with each other. The Unit level testing made sure that each code was tested and working accordingly in an isolated environment and integration examines

the correctness of the behaviour in an open environment through the use of white-box testing for any interactions of interfaces that are known to the programmer.

3. Regression testing. White-box testing during regression testing is the use of recycled white-box test cases at the unit and integration testing levels.

White-box testing's basic procedures involves the tester having a deep level of understanding of the source code being tested. The programmer must have a deep understanding of the application to know what kinds of test cases to create so that every visible path is exercised for testing. Once the source code is understood then the source code can be analyzed for test cases to be created. These are the three basic steps that white-box testing takes in order to create test cases:

1. Input involves different types of requirements, functional specifications, detailed designing of documents, proper source code, security specifications. This is the preparation stage of white-box testing to layout all of the basic information.

2. Processing involves performing risk analysis to guide whole testing process, proper test plan, execute test cases and communicate results.[2] This is the phase of building test cases to make sure they thoroughly test the application the given results are recorded accordingly.

3. Output involves preparing final report that encompasses all of the above preparations and results.

A more modern view is that the dichotomy between white-box testing and black-box testing has blurred and is becoming less relevant. Whereas "white-box" originally meant using the source code, and black-box meant using requirements, tests are now derived from many documents at various levels of abstraction. The real point is that tests are usually designed from an abstract structure such as the input space, a graph, or logical predicates, and the question is what level of abstraction in structure from was derived. That can be the source code, requirements, input space descriptions, or one of dozens of types of design models.

Integration Testing

Integration testing is the phase in software testing in which individual software modules are combined and tested as a group. It occurs after unit testing and before validation testing. Integration testing takes as its input modules that have been unit tested, groups them in larger aggregates, applies tests defined in an integration test plan to those aggregates, and delivers as its output the integrated system ready for system testing.

The purpose of integration testing is to verify functional, performance, and reliability requirements placed on major design items. These "design items", i.e. assemblages (or groups of units), are exercised through their interfaces using black box testing, success and error cases being simulated via appropriate parameter and data inputs. Simulated usage of shared data areas and inter-process communication is tested and individual subsystems are exercised through their input interface. Test cases are constructed to test whether all the components within assemblages interact correctly, for example across procedure calls or process activations, and this is done after testing individual modules, i.e. unit testing. The overall idea is a "building block" approach, in which verified assemblages are added to a verified base which is then used to support the integration testing of further assemblages.

Some different types of integration testing are big bang, top-down, and bottom-up, mixed (sandwich) and risky - hardest. Other Integration Patterns are: Collaboration Integration, Backbone Integration, Layer Integration, Client/Server Integration, Distributed Services Integration and High-frequency Integration.

In this approach, most of the developed modules are coupled together to form a complete software system or major part of the system and then used for integration testing. The Big Bang method is very effective for saving time in the integration testing process. However, if the test cases and their results are not recorded properly, the entire integration process will be more complicated and may prevent the testing team from achieving the goal of integration testing. A type of Big Bang Integration testing is called Usage Model testing. Usage Model Testing can be used in both software and hardware integration testing. The basis behind this type of integration testing is to run user-like workloads in integrated user-like environments. In doing the testing in this manner, the environment is proofed, while the individual components are proofed indirectly through their use. Usage Model testing takes an optimistic approach to testing, because it expects to have few problems with the individual components. The strategy relies heavily on the component developers to do the isolated unit testing for their product.

The goal of the strategy is to avoid redoing the testing done by the developers, and instead flesh-out problems caused by the interaction of the components in the environment. For integration testing, Usage Model testing can be more efficient and provides better test coverage than traditional focused functional integration testing. To be more efficient and accurate, care must be used in defining the user-like workloads for creating realistic scenarios in exercising the environment. This gives confidence that the integrated environment will work as expected for the target customers.Bottom up Testing is an approach to integrated testing where the lowest level components are tested first, then used to facilitate the testing of higher level components. The process is repeated until the component at the top of the hierarchy is tested.

All the bottom or low-level modules, procedures or functions are integrated and then tested. After the integration testing of lower level integrated modules, the next level of modules will be formed and can be used for integration testing. This approach is helpful only when all or most of the modules of the same development level are ready. This method also helps to determine the levels of software developed and makes it easier to report testing progress in the form of a percentage. Top down Testing is an approach to integrated testing where the top integrated modules are tested and the branch of the module is tested step by step until the end of the related module. Sandwich Testing is an approach to combine top down testing with bottom up testing. The main advantage of the Bottom-Up approach is that bugs are more easily found. With Top-Down, it is easier to find a missing branch link.

Software integration testing is the incremental integration testing of two or more integrated software components on a single platform to produce failures caused by interface defects. The task of the integration test is to check that components or software applications, e.g. components in a software system or – one step up – software applications at the company level – interact without error.

Acceptance Testing

Acceptance testing is a test conducted to determine if the requirements of a specification or contract are met. It may involve chemical tests, physical tests, or performance tests. In systems engineering it may involve black-box testing performed on a system (for example: a piece of software, lots of manufactured mechanical parts, or batches of chemical products) prior to its delivery. Acceptance testing is also known as user acceptance testing (UAT), end-user testing, and operational acceptance testing (OAT) or field (acceptance) testing.

The acceptance test suite is run using predefined acceptance test procedures to direct the testers which data to use, the step-by-step processes to follow and the expected result following execution. The actual results are retained for comparison with the expected results. If the actual results match the expected results for each test case, the test case is said to pass. If the quantity of non-passing test cases does not breach the project's predetermined threshold, the test suite is said to pass.

Data validation Testing

Data validation is intended to provide certain well-defined guarantees for fitness, accuracy, and consistency for any of various kinds of user input into an application or automated system.

Definition and design contexts

- As a part of requirements gathering phase in a software engineering or designing a software specification.
- As part of an operations modelling phase in business process modelling.

Deployment contexts

- As part of a user interface.
- As a set of programs or business logic routines in a programming language.
- As a set of stored procedures in a database management system.

For business applications, data validation can be defined through declarative data integrity rules, or procedure-based rules. Data that does not conform to these rules will negatively affect business process execution. Therefore, data validation should start with business process definition and set of business rules within this process. Rules can be collected through the requirements capture exercise. Data type validation is customarily carried out on one or more simple data fields.

The simplest kind of data type validation verifies that the individual characters provided through user input are consistent with the expected characters of one or more known primitive data types; as defined in a programming language or data storage and retrieval mechanism. For example, many database systems allow the specification of the following primitive data types: 1) integer; 2) float (decimal); or 3) string. A validation process involves two distinct steps: (a) Validation Check and (b) Post-Check action. The check step uses one or more computational rules (see section below) to determine if the data is valid. The Post-validation action sends feedback to help enforce validation.

Simple range and constraint validation

Simple range and constraint validation may examine user input for consistency with a minimum/maximum range, or consistency with a test for evaluating a sequence of characters, such as one or more tests against regular expressions.

Code and cross-reference validation

Code and cross-reference validation includes tests for data type validation, combined with one or more operations to verify that the user-supplied data is consistent with one or more external rules, requirements, or validity constraints relevant to a particular organization, context or set of underlying assumptions. These additional validity constraints may involve cross-referencing supplied data with a known look-up table or directory information service such as LDAP.

Structured validation

Structured validation allows for the combination of any of various basic data type validation steps, along with more complex processing. Such complex processing may include the testing of conditional constraints for an entire complex data object or set of process operations within a system.

Test Data and Result

Test data is data which has been specifically identified for use in tests, typically of a computer program. Some data may be used in a confirmatory way, typically to verify that a given set of input to a given function produces some expected result. Other data may be used in order to challenge the ability of the program to respond to unusual, extreme, exceptional, or unexpected input. Test data may be produced in a focused or systematic way (as is typically the case in domain testing), or by using other, less-focused approaches (as is typically the case in high-volume randomized automated tests). Test data may be produced by the tester, or by a program or function that aids the tester. Test data may be recorded for re-use, or used once and then forgotten.

Strategic Approach to Software Testing

The software engineering process can be viewed as a spiral. Initially, system engineering defines the role of the software and leads to software requirement analysis where the information domain, functions, behaviour, performance, constraints and validation criteria for software are established. Moving inward along the spiral, come to design and finally to coding.

A strategy for software testing may also be viewed in the context of the spiral. Unit testing begins at the vertex of the spiral and concentrates on each unit of the software as implemented in source code. Testing progress is used by moving outward along the spiral to integration testing, where the focus is on the design and the construction of the software architecture.

CHAPTER 8

8. CONCLUSION

A Wi-Fi based Prevalent Automated Billing Systems in Shopping Malls using Android is used to purchase the product using android based application. It is reducing wastage of weighting time of the customer in shopping mall. It will keep the track of purchased products against their budge. On the other day, shop required list number of the employee for billing.

I implemented the proposed system and learn the technical thing such android and php programming.

CHAPTER 9

9. FUTURE ENHANCEMENT

- To integrate real payment gateway for payment process
- To implement app for IOS and Windows system.

CHAPTER 10

10. APPENDIX

10.1 SOURCE CODE

Activity_home.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayoutxmlns:android="http://schemas.android.com/apk/res/android"
xmlns:tools="http://schemas.android.com/tools"
android:layout_width="match_parent"
android:layout_height="match_parent"
android:orientation="vertical"
tools:context="shirosoft.farmcalc.HomeActivity">

<android.support.v7.widget.Toolbar
android:id="@+id/toolbar"
android:layout_width="match_parent"
android:layout_height="wrap_content"
android:background="@color/colorPrimary"
/>

<LinearLayout
android:layout_width="match_parent"
android:layout_height="match_parent"
android:orientation="vertical"
android:padding="10dp">

<ScrollView
android:layout_width="match_parent"
android:layout_height="match_parent">

<LinearLayout
android:layout_width="match_parent"
android:layout_height="wrap_content"
android:orientation="vertical">

<TextView
android:layout_width="match_parent"
android:layout_height="40dp"
android:background="@color/colorPrimary"
android:gravity="center"
android:text="PRE FARM PLAN"
android:textColor="#FFFFFF"/>
```

```
<TextView
android:layout_marginTop="10dp"
android:layout_width="match_parent"
android:layout_height="wrap_content"
android:text="Farm Plan activities involve choosing the right variety, developing a cropping calendar,
and preparing the rice and wheat field for planting."
/>

<ImageView
android:layout_marginTop="10dp"
android:layout_width="match_parent"
android:layout_height="230dp"
android:src="@drawable/wsfarm" />

<Button
android:id="@+id/rice"
android:layout_width="match_parent"
android:layout_height="60dp"
android:background="@color/colorPrimary"
android:text="RICE"
android:textColor="#FFFFFF" />

<Button
android:id="@+id/wheat"
android:layout_width="match_parent"
android:layout_height="60dp"
android:layout_marginTop="10dp"
android:background="@color/colorPrimary"
android:text="WHEAT"
android:textColor="#FFFFFF" />

</LinearLayout>

</ScrollView>
</LinearLayout>



</LinearLayout>
```

HomeActivity.java

```
import android.app.Dialog;
import android.content.ActivityNotFoundException;
import android.content.Context;
import android.graphics.Typeface;
import android.graphics.drawable.ColorDrawable;
import android.net.Uri;
import android.support.v7.app.ActionBar;
import android.support.v7.app.AlertDialog;
import android.content.DialogInterface;
import android.content.Intent;
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.support.v7.widget.Toolbar;
import android.text.Spannable;
import android.text.SpannableString;
import android.view.LayoutInflater;
import android.view.Menu;
import android.view.MenuItem;
import android.view.View;
import android.view.Window;
import android.view.WindowManager;
import android.widget.Button;

public class HomeActivityextends AppCompatActivity {

    Button wprocess,rprocess;
private Toolbar toolbar;

    @Override
protected void onCreate(Bundle savedInstanceState) {
super.onCreate(savedInstanceState);
this.getWindow().setFlags(WindowManager.LayoutParams.FLAG_FULLSCREEN,
WindowManager.LayoutParams.FLAG_FULLSCREEN);
setContentView(R.layout.activity_home);
setupToolbar();
        String fontPath = "fonts/Roboto-Bold.ttf";
        Typeface typefaces = Typeface.createFromAsset(getAssets(), fontPath);
rprocess= (Button)findViewById(R.id.rice);
wprocess= (Button)findViewById(R.id.wheat);

rprocess.setTypeface(typefaces);
wprocess.setTypeface(typefaces);

rprocess.setOnClickListener(new View.OnClickListener() {
            @Override
public void onClick(View v) {
startActivity(new Intent(HomeActivity.this,RiceActivity.class));
            }
        });


wprocess.setOnClickListener(new View.OnClickListener() {
            @Override
public void onClick(View v) {
startActivity(new Intent(HomeActivity.this,WheatActivity.class));
            }
        });
    }

private void setupToolbar(){

toolbar = (Toolbar)findViewById(R.id.toolbar);
setSupportActionBar(toolbar);
SpannableString s = new SpannableString("Farm Plan");
s.setSpan(new shirosoft.farmcalc.font.TypefaceSpan(this, "Roboto-Bold.ttf"), 0, s.length(),
Spannable.SPAN_EXCLUSIVE_EXCLUSIVE);
getSupportActionBar().setTitle(s);
    }

    @Override
public void onBackPressed() {
confirmDialog();
return;
    }

    @Override
public booleanonCreateOptionsMenu(Menu menu) {
// Inflate the menu; this adds items to the action bar if it is present.
getMenuInflater().inflate(R.menu.menu_index, menu);
return true;
    }

    @Override
public booleanonOptionsItemSelected(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.
intid = item.getItemId();

//noinspectionSimplifiableIfStatement
if (id == R.id.about)
        {
aboutDialog();
return true;
        }


return super.onOptionsItemSelected(item);
    }

private void aboutDialog()
    {
final Dialog dialog = new Dialog(this);
```

```
dialog.requestWindowFeature(Window.FEATURE_NO_TITLE);
LayoutInflaterinflater = (LayoutInflater)
getSystemService(Context.LAYOUT_INFLATER_SERVICE);
        View view = inflater.inflate(R.layout.about, null, false);
dialog.setCanceledOnTouchOutside(false);
        Window window = dialog.getWindow();
window.setFlags(WindowManager.LayoutParams.FLAG_FULLSCREEN,
WindowManager.LayoutParams.FLAG_FULLSCREEN);
dialog.setContentView(view);
dialog.getWindow().setBackgroundDrawable(new ColorDrawable(0));
dialog.getWindow().getAttributes().windowAnimations= R.style.DialogAnimation;
window.setLayout(ActionBar.LayoutParams.MATCH_PARENT,
ActionBar.LayoutParams.MATCH_PARENT);
dialog.show();
    }

private void confirmDialog() {
AlertDialog.BuilderalertDialog = new AlertDialog.Builder(
HomeActivity.this);
alertDialog.setMessage("Are you sure want to quit?");
alertDialog.setTitle("Farm Plan");

//alertDialog.setIcon(R.drawable.javacoffe);
alertDialog.setPositiveButton("YES",
new DialogInterface.OnClickListener() {
public void onClick(DialogInterface dialog, intwhich) {
                        finish();
                    }
                });
alertDialog.setNegativeButton("NO",
new DialogInterface.OnClickListener() {
public void onClick(DialogInterface dialog, intwhich) {
// Write your code here to invoke NO event
dialog.cancel();
                    }
                });
alertDialog.show();
    }
}
```

RiceActivity.java

```
import android.content.Intent;
import android.graphics.Typeface;
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.support.v7.widget.Toolbar;
import android.text.Spannable;
import android.text.SpannableString;
import android.view.View;
import android.widget.Button;

public class RiceActivityextends AppCompatActivity {
   Button rprocess;
private Toolbar toolbar;
   @Override
protected void onCreate(Bundle savedInstanceState) {
super.onCreate(savedInstanceState);
setContentView(R.layout.activity_rice);

setupToolbar();
      String fontPath = "fonts/Roboto-Bold.ttf";
      Typeface typefaces = Typeface.createFromAsset(getAssets(), fontPath);

rprocess= (Button)findViewById(R.id.rprocess);
rprocess.setTypeface(typefaces);
rprocess.setOnClickListener(new View.OnClickListener() {
         @Override
public void onClick(View v) {
startActivity(new Intent(RiceActivity.this,RiceProcess.class));
         }
      });


   }

private void setupToolbar(){

toolbar = (Toolbar)findViewById(R.id.toolbar);
setSupportActionBar(toolbar);
SpannableString s = new SpannableString("Farm Plan");
s.setSpan(new shirosoft.farmcalc.font.TypefaceSpan(this, "Roboto-Bold.ttf"), 0, s.length(),
Spannable.SPAN_EXCLUSIVE_EXCLUSIVE);
getSupportActionBar().setTitle(s);

   }
}
```

RiceProcess.java

```
import android.app.DatePickerDialog;
import android.app.Dialog;
import android.graphics.Typeface;
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.support.v7.widget.Toolbar;
import android.text.Spannable;
import android.text.SpannableString;
import android.view.View;
import android.widget.Button;
import android.widget.DatePicker;
import android.widget.TextView;
import android.widget.Toast;

import java.text.ParseException;
import java.text.SimpleDateFormat;
import java.util.Calendar;

public class RiceProcessextends AppCompatActivity {

private DatePickerdatePicker;
private Calendar calendar;
    Calendar iDate;
SimpleDateFormatmdformat;
private intyear, month, day;
    Button editDate;
private TextViewday_one, day_two, day_three, day_four, day_five, day_six, day_seven,
day_eight, day_nine,
day_ten, day_eleven, day_twelve, day_thirteen, day_fourteen, day_fifteen,
day_sixteen,day_seventeen,
            day_eighteen,day_nineteen,day_twenty,day_twentyone,day_twentytwo;
    String startDate;

private Toolbar toolbar;
private TextView title;

    @Override
protected void onCreate(Bundle savedInstanceState) {
super.onCreate(savedInstanceState);
setContentView(R.layout.activity_rice_process);
setupToolbar();
        String fontPath = "fonts/Roboto-Bold.ttf";
        Typeface typefaces = Typeface.createFromAsset(getAssets(), fontPath);
        title = (TextView)findViewById(R.id.title);
title.setTypeface(typefaces);
        calendar = Calendar.getInstance();
        year = calendar.get(Calendar.YEAR);

        month = calendar.get(Calendar.MONTH);
        day = calendar.get(Calendar.DAY_OF_MONTH);
dayInitialization();
showDate(year, month + 1, day);
```

```
editDate = (Button) findViewById(R.id.editDate);
editDate.setTypeface(typefaces);
editDate.setOnClickListener(new View.OnClickListener() {
          @Override
public void onClick(View v) {
showDialog(999);
Toast.makeText(getApplicationContext(), "ca", Toast.LENGTH_SHORT)
                    .show();
          }
      });
   }

private void setupToolbar(){

toolbar = (Toolbar)findViewById(R.id.toolbar);
setSupportActionBar(toolbar);
SpannableString s = new SpannableString("Farm Plan");
s.setSpan(new shirosoft.farmcalc.font.TypefaceSpan(this, "Roboto-Bold.ttf"), 0, s.length(),
Spannable.SPAN_EXCLUSIVE_EXCLUSIVE);
getSupportActionBar().setTitle(s);

   }

private void dayInitialization() {
day_one= (TextView) findViewById(R.id.day_one);
day_two= (TextView) findViewById(R.id.day_two);
day_three= (TextView) findViewById(R.id.day_three);
day_four= (TextView) findViewById(R.id.day_four);
day_five= (TextView) findViewById(R.id.day_five);
day_six= (TextView) findViewById(R.id.day_six);
day_seven= (TextView) findViewById(R.id.day_seven);
day_eight= (TextView) findViewById(R.id.day_eight);
day_nine= (TextView) findViewById(R.id.day_nine);
day_ten= (TextView) findViewById(R.id.day_ten);
day_eleven= (TextView) findViewById(R.id.day_eleven);
day_twelve = (TextView) findViewById(R.id.day_tweleve);
day_thirteen = (TextView) findViewById(R.id.day_thirteen);
day_fourteen = (TextView) findViewById(R.id.day_fourteen);
day_fifteen = (TextView) findViewById(R.id.day_fifteen);
day_sixteen = (TextView) findViewById(R.id.day_sixteen);
day_seventeen = (TextView) findViewById(R.id.day_seventeen);
day_eighteen  = (TextView) findViewById(R.id.day_eighteen);
day_nineteen = (TextView) findViewById(R.id.day_nineteen);
day_twenty = (TextView) findViewById(R.id.day_twenty);
day_twentyone = (TextView) findViewById(R.id.day_twentyone);
day_twentytwo = (TextView) findViewById(R.id.day_twentytwo);
   }

   @Override
protected Dialog onCreateDialog(intid) {
```

```
// TODO Auto-generated method stub
if (id == 999) {
return new DatePickerDialog(this, myDateListener, year, month, day);
        }
return null;
    }

private DatePickerDialog.OnDateSetListenermyDateListener = new
DatePickerDialog.OnDateSetListener() {
        @Override
public void onDateSet(DatePicker arg0, intarg1, intarg2, intarg3) {
// TODO Auto-generated method stub
            // arg1 = year
            // arg2 = month
            // arg3 = day
showDate(arg1, arg2, arg3);
        }
    };

private void showDate(intyear, intmonth, intday) {
/*dateView.setText(new StringBuilder().append(day).append("/")
                .append(month).append("/").append(year));*/

        // startDate = String.valueOf(day+"/"+month+"/"+year);
        //day_one.setText(startDate);
iDate = Calendar.getInstance();
mdformat = new SimpleDateFormat("dd / MM / yyyy ");
iDate.set(year, month, day);
        String strDate = mdformat.format(iDate.getTime());
day_one.setText(strDate); //Day-One


iDate.add(Calendar.DATE, 1);
strDate = mdformat.format(iDate.getTime());
try {
iDate.setTime(mdformat.parse(strDate));
        } catch (ParseException e) {
e.printStackTrace();
        }

day_two.setText(strDate); //Day_two

iDate.add(Calendar.DATE, 14);
strDate = mdformat.format(iDate.getTime());
try {
iDate.setTime(mdformat.parse(strDate));
        } catch (ParseException e) {
e.printStackTrace();
        }
```

```
day_three.setText(strDate); //Day_three

iDate.add(Calendar.DATE, 15);
strDate = mdformat.format(iDate.getTime());
try {
iDate.setTime(mdformat.parse(strDate));
        } catch (ParseException e) {
e.printStackTrace();
        }

day_four.setText(strDate); //Day Four

iDate.add(Calendar.DATE, 6);
strDate = mdformat.format(iDate.getTime());
try {
iDate.setTime(mdformat.parse(strDate));
        } catch (ParseException e) {
e.printStackTrace();
        }

day_five.setText(strDate); //Day Five

iDate.add(Calendar.DATE, 4);
strDate = mdformat.format(iDate.getTime());
try {
iDate.setTime(mdformat.parse(strDate));
        } catch (ParseException e) {
e.printStackTrace();
        }

day_six.setText(strDate); //Day Six

iDate.add(Calendar.DATE, 10);
strDate = mdformat.format(iDate.getTime());
try {
iDate.setTime(mdformat.parse(strDate));
        } catch (ParseException e) {
e.printStackTrace();
        }

day_seven.setText(strDate); //Day Seven

iDate.add(Calendar.DATE, 1);
strDate = mdformat.format(iDate.getTime());
try {
iDate.setTime(mdformat.parse(strDate));
        } catch (ParseException e) {
e.printStackTrace();
        }
```

```
day_eight.setText(strDate); //Day Eight

iDate.add(Calendar.DATE, 9);
strDate = mdformat.format(iDate.getTime());
try {
iDate.setTime(mdformat.parse(strDate));
        } catch (ParseException e) {
e.printStackTrace();
        }

day_nine.setText(strDate); //Day Nine

iDate.add(Calendar.DATE, 10);
strDate = mdformat.format(iDate.getTime());
try {
iDate.setTime(mdformat.parse(strDate));
        } catch (ParseException e) {
e.printStackTrace();
        }

day_ten.setText(strDate); //Day Ten

iDate.add(Calendar.DATE, 10);
strDate = mdformat.format(iDate.getTime());
try {
iDate.setTime(mdformat.parse(strDate));
        } catch (ParseException e) {
e.printStackTrace();
        }

day_eleven.setText(strDate); //Day eleven

iDate.add(Calendar.DATE, 1);
strDate = mdformat.format(iDate.getTime());
try {
iDate.setTime(mdformat.parse(strDate));
        } catch (ParseException e) {
e.printStackTrace();
        }

day_twelve.setText(strDate); //Day twelve

iDate.add(Calendar.DATE, 2);
strDate = mdformat.format(iDate.getTime());
try {
iDate.setTime(mdformat.parse(strDate));
        } catch (ParseException e) {
e.printStackTrace();
        }
```

```
day_thirteen.setText(strDate); //Day Thirteen

iDate.add(Calendar.DATE, 5);
strDate = mdformat.format(iDate.getTime());
try {
iDate.setTime(mdformat.parse(strDate));
        } catch (ParseException e) {
e.printStackTrace();
        }

day_fourteen.setText(strDate); //Day Fourteen

iDate.add(Calendar.DATE, 2);
strDate = mdformat.format(iDate.getTime());
try {
iDate.setTime(mdformat.parse(strDate));
        } catch (ParseException e) {
e.printStackTrace();
        }

day_fifteen.setText(strDate); //Day Fifteen

iDate.add(Calendar.DATE, 10);
strDate = mdformat.format(iDate.getTime());
try {
iDate.setTime(mdformat.parse(strDate));
        } catch (ParseException e) {
e.printStackTrace();
        }

day_sixteen.setText(strDate); //Day Sixteen

iDate.add(Calendar.DATE, 10);
strDate = mdformat.format(iDate.getTime());
try {
iDate.setTime(mdformat.parse(strDate));
        } catch (ParseException e) {
e.printStackTrace();
        }

day_seventeen.setText(strDate); //Day Seventeen

iDate.add(Calendar.DATE, 9);
strDate = mdformat.format(iDate.getTime());
try {
iDate.setTime(mdformat.parse(strDate));
        } catch (ParseException e) {
e.printStackTrace();
        }
```

```
day_eighteen.setText(strDate); //Day Eighteen

iDate.add(Calendar.DATE, 1);
strDate = mdformat.format(iDate.getTime());
try {
iDate.setTime(mdformat.parse(strDate));
        } catch (ParseException e) {
e.printStackTrace();
        }

day_nineteen.setText(strDate); //Day Seventeen

iDate.add(Calendar.DATE, 10);
strDate = mdformat.format(iDate.getTime());
try {
iDate.setTime(mdformat.parse(strDate));
        } catch (ParseException e) {
e.printStackTrace();
        }

day_twenty.setText(strDate); //Day Twenty

iDate.add(Calendar.DATE, 9);
strDate = mdformat.format(iDate.getTime());
try {
iDate.setTime(mdformat.parse(strDate));
        } catch (ParseException e) {
e.printStackTrace();
        }

day_twentyone.setText(strDate); //Day Twenty One

iDate.add(Calendar.DATE, 10);
strDate = mdformat.format(iDate.getTime());
try {
iDate.setTime(mdformat.parse(strDate));
        } catch (ParseException e) {
e.printStackTrace();
        }

day_twentytwo.setText(strDate); //Day twenty two



}

}
```

References:

1.Ajit Danti Jyoti Y.Kulkarni and P .S . Hiremath,"a WiFi based automated billing system", International Journal of Modelling and Optimization,vol.2No.6, December 2012.

2.S.Ganapriya,V.B.Padmasree,V.Bagvalakshi,,"Iot Based Automated Billing System",vol.8,may 2017.

3.Gunjan Chugh,Diviya Bansal and Sanjuv sofat"an alalysis of automated billing system ",vol.5, April 2017.