CHAPTER 1

INTRODUCTION

Introduction to project

This project is based on the Android in which we can store and retrieve all the information regarding the Farmer's Market application based on farmer and customer what all we require.

Database is a structured format. So if we store in the database, we can retrieve that particular information by giving a command directly. There is no process for installing database for mobile-phone applications. Whenever the database is needed, then only the database is created through coding. In this project there is a feasibility to change and delete the data which is not required.

The application we are developing is about "**Krushi Snehi Android Application"**. It is an android application used to find out the Information about the farmer and customer based android application in this application we have two modules like farmer and customer application. In farmer application farmer can add the his/her product according to application input parameter and formar can manipulate the product data and also farmer can view the customer order of products. In user application user can view the product then place the order of product and also customer can view the order details. Main module of farmer and customer application is Equipment module in this module farmer or user can view the equipment as per requirement and take the equipment as in rent. All this information is available internet connection is required. All we have to do is to store the information in database.

1.1 Purpose of the project

The main objective of "**Krushi Snehi Android Application**" is to provide very flexible environment to the mobile users. Nowadays each and every person having android mobile and it is easy to carry so that it is gaining the more popularity compared to the other electronic gadgets. In this competitive world we have different types of mobiles and in these mobiles also we are having different types of applications.

Generally, we won't have any application regarding any information about smart based menu card for hotel to order to application. We have to browse for everything. But this application has that flexibility to have the information and we even have the flexibility to change and delete the information if we feel that it is not helpful anymore.

Chapter 2

LITERATURE SURVEY

1. E-KRUSHI MITRA

Abstract- "E-Krushi Mitra" application gives an idea about how to choose the suitable crop by analyzing the soil quality. Based on the quality of soil, it chooses the suitable crop. This application focuses on helping farmers. Its purpose to give profits to farmers the basic idea of this application is online auction. It allows farmer to sell their crops at best price. According to observations of Information and Communication Technologies (ICT) mobile plays vital role in daily life of farmers. The farmers, always depends on weather, for their solutions near cultivation of superior crops in today's modern agricultural world. The traditional methods used by the farmers, uniquely in India, are very slow and fickle. The large amount of yield get loss in the field due to the bacterial attacks and lack of information resources. Annually, such loss exceeds 40% in total. So, there are various ways in which a farmer can utilize application entitled "Krushi Mitra", to assist them for relatively better cultivation and merchandise. The main awareness of this work is focused on farmers as it addresses the key problems of getting the market updates of different products, weather updates and also provides multiple language support. This will effectively help farmers to sell their product in global market and earn remarkable profit. Hence, this framework, which in effect, puts power into a farmer's hand. The experimental setup uses tools like Android SDK. In this research an Android based mobile devices are used for testing.

It gives the idea about how to choose the appropriate crop by analyzing the soil quality. Based on the quality of soil, it chooses the appropriate crop. it also gives the big factor the weather forecast and the general pattern of weather for that region [2]. The mention about the green revolution and increases in an education many farmers moving toward the technology such as mobile application and computer for their crop needs for shopping farming product online [5]. This application focuses on helping farmers. Its purpose to give profits to farmers the basic idea of this application is online auction. it allows farmer to sell their crops at best price. It gives the whole information regarding to crops, Weather status and also user can get the expert advice in Marathi and in English languages. Krushi-Mitra application can be used as smart system which will be more sophisticatedly working for benefit of the user. A user can be made aware about current weather statistics and new information regarding to crops, seeds, fertilizer. just on single click of a button. Farmer can even consult with experts if needed. This application can be very much helpful even if one could

not read the information on the device by native language support provided in it [6]. It is Android based mobile application designed to meet the needs of the Indian farmers providing all the facilities to the farmers related to their agricultural activities such as crop specific data, market prices, weather & news pertaining to farming. This application for agriculture enables the farmer to calculate profitability based on where the grain markets are currently trading and to see how higher or lower grain markets are presently. They would be able to get the current market prices depending upon the commodities. It should carry grain and livestock prices from major Indian agricultural market. [7]

2.Krushi-Mitra: Expert System for Farmers

In today's rapid changing world of internet, we focused on the people especially from the rural areas. The main Aim behind this is that the people in rural areas are far away from internet technology.so, in order to get all the information about agriculture collectively, we have developed a Website and an Application which will help the farmer in many ways. In our system, we have made such an interface which can be accessed by semi-illiterate people. Also; there are options of Marathi as well as English language. So that if any farmer is English illiterate he can get the information in Marathi through one more advantage of Krushi-Mitra Website is that for registered user there is FAQ's facility and also he can type his Queries, which will have resolved in specific duration. Inspite of this, if any farmer faces difficulty in accessing the our Krushi-Mitra Website he can use Krushi-Mitra Application which contains iconic based interface as well as information in speech format i.e. audio clip. Also, if he has some other queries, he can directly contact to expert calling. The rest of the paper is organized as follows. The proposed approach is elaborated in Section III. Section IV provides the system implementation details. The user testing results are discussed in Section V. Finally, Section VI concludes the paper with the future directions of the research. Thus, we not only provide information but also farmers will be motivated to access the information.

3.Kisan Mitra: A Farming Assistance Application using Ionic Framework

Abstract: Rural subsistence farmer across India experience difficulties in gathering relevant and up-to-date agriculture information. This research aims to develop a mobile application for broadcasting agriculture information to rural subsistence farmers. is no complete system that advises the farmers about what crops to grow and in which region prediction of suitable soil w. t to the crop type seed availability as well as an interface that connects the farmer to the supplier and vice versa. Also, this system aims to provide information regarding finance and various government policies. Through this important technology, they directly keep in touch with market suppliers and sell their products at reasonable rates. Kisan Mitra helps the farmers working with the motive of greater profitability by direct communication between farmers to supplier playing a vital role in the enhancement of farmer's business of agriculture. The analysis is done for several districts of the state of Maharashtra, India.

Index Terms - Farming, application, marketing, schemes, loans.

Chapter 3

SYSTEM ANALYSIS

3.1 Study of the System

This involves the investigation of the existing system, which includes a vast level of interviews with the user and the concerned staff in sufficient depth. This also includes the collection and study of detailed information and literature regarding the complete existing procedure.

The detailed initial study properly documented and the failing and problems are noted separately. The system is properly designed and proper outline of the proposed computerized system is prepared. The proposed design is brought against all the known facts and further proposals are made. Various resources including the software, hardware and manpower requirements are decided and are mentioned in the report.

3.2 Input and Output Representation

Actual user community participation and their requirements analysis are key to success of any new information system. To carry out this work, identification of users who will actually use the system is foremost. Users at every level were given opportunity to define their goals, objectives and their respective information needs. In addition to this exercise a critical through investigation of present reports and query generated, were carried out to define any other additional requirements that can be useful to the others.

The findings of users and other related exercise to access particular user needs are summarized below concisely:

- The System must provide a graphical user interface.
- Redundancy must be reduced at the maximum level.
- Discrepancies should be avoided.
- Security systems must be provided.
- Input design is a part of overall system design. The main objective during the input design is as given below:
- To produce a cost-effective method of input.
- To achieve the highest possible level of accuracy.
- To ensure that the input is acceptable and understood by the user.

3.2.1 INPUT STAGES

The main input stages can be listed as below:

- Data recording
- Data transcription
- Data conversion
- Data verification
- Data control
- Data transmission
- Data validation
- Data correction

3.2.2 INPUT TYPES:

It is necessary to determine the various types of inputs. Inputs can be categorized as follows:

- External inputs, which are prime inputs for the system.
- Internal inputs, which are user communications with the system.
- Operational, which is computer department's communication to the system.
- Interactive, which are inputs entered during a dialogue.

3.2.3 INPUT MEDIA:

At this stage choice has to be made about the input media. To conclude about the input media consideration has to be given to:

- Type of input
- Flexibility of format
- Speed
- Accuracy
- Verification methods
- Rejection rates
- Ease of correction
- Storage and handling requirements
- Security
- Easy to use
- Portability

Keeping in view the above description of the input types and input media, it can be said that most of the inputs are of the form of internal and interactive. As input data is to be the directly keyed in by the user, the keyboard can be considered to be the most suitable input device.

3.2.4 OUTPUT DESIGN:

In general, are:

- External Outputs whose destination is outside the organization.
- Internal Outputs whose destination is within organization and they are the User's main interface with the computer.
- Outputs from computer systems are required primarily to communicate the results of processing to users. They are also used to provide a permanent copy of the results for later consultation.

The various types of outputs are:

- Operational outputs whose use is purely within the computer department.
- Interface outputs, which involve the user in communicating directly with the system.

3.2.5 OUTPUT DEFINITION

The outputs should be defined in terms of the following points:

- Type of the output
- Content of the output
- Format of the output
- Frequency of the output
- Volume of the output
- Sequence of the output

It is not always desirable to print or display data as it is held on a computer. It should be decided as which form of the output is the most suitable.

3.2.6 OUTPUT MEDIA:

In the next stage it is to be decided that which medium is the most appropriate for the output. The main considerations when deciding about the output media are:

- The suitability for the device to the particular application.
- The need for a hard copy.
- The software and hardware available.

Keeping in view the above description the project is to have outputs mainly coming under the category of internal outputs. The outputs were needed to be generated as a hard copy and as well as queries to be viewed on the screen. Keeping in view these outputs, the format for the output is taken from the outputs, which are currently being obtained after manual processing. The standard printer is to be used as output media for hard copies.

Chapter 4

SYSTEM DESIGN

4.1 Introduction

Once software requirements have been analyzed and specified, software design as the first of 3 technical activities – design, code generation and test-that are required to build and software. Stages of Design of a Project are:

- Conceptual Design
- Database Design

4.1.2 Conceptual Design:

The conceptual structure of a Database is called schema. Schema shows the kind of data that exists in a database and how these are logically related to each other. A schema can be regarded as a blueprint that portrays, both, kind of data used in building a database and logical relationship, and must correctly represent their inter relationships. Schema is frequently depicted pictorially with Data Flow Diagram (DFD) etc.,

4.1.2 Database Design:

A Database is a stored collection of interrelated data, organized on the basis of relationship in the data rather than the convenience of storage structures. It enables sharing of data among various users as and when required. Database Management System is software that provides more flexibility in the storage and retrieval of data and productive information.

4.1.3 Relational Database:

A database, in which data is stored in tables, allowing relationship among tables and more efficient non-redundant data storage.

4.2 DFD (Data Flow Diagrams)

Data Flow Diagrams are a graphical tool used to describe and analyze the movement of data through a system. DFD's are used to capture the essential feature of both existing real system and future physical implementation of the system.

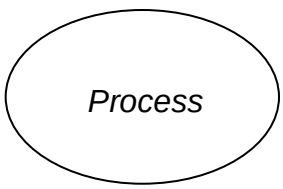
The DFD is a graphical technique that depicts the information flow and the transforms that are applies as data move from input to the output. The DFD is also known as Bubble Chart or Data Flow Graphs or Context diagram.

The data flow diagram may be used to present a system or software at any level of abstraction. A fundamental system model or a context model represents the entire software elements as a single bubble with input and output data indicated by incoming and outgoing arrows respectively.

Data flow diagrams are constructed from four basic building blocks:

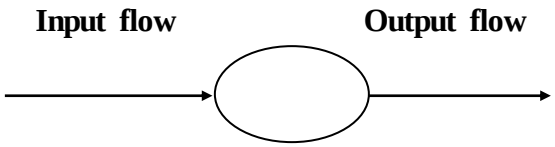
- Processes
- Data flow
- Store
- Terminator

➢ **The Process**



- Also called bubble, function, and transformation.
- Shows part of the system that transforms inputs to outputs.
- Represented graphically as a circle.
- Named with single word, phrase, or sentence.

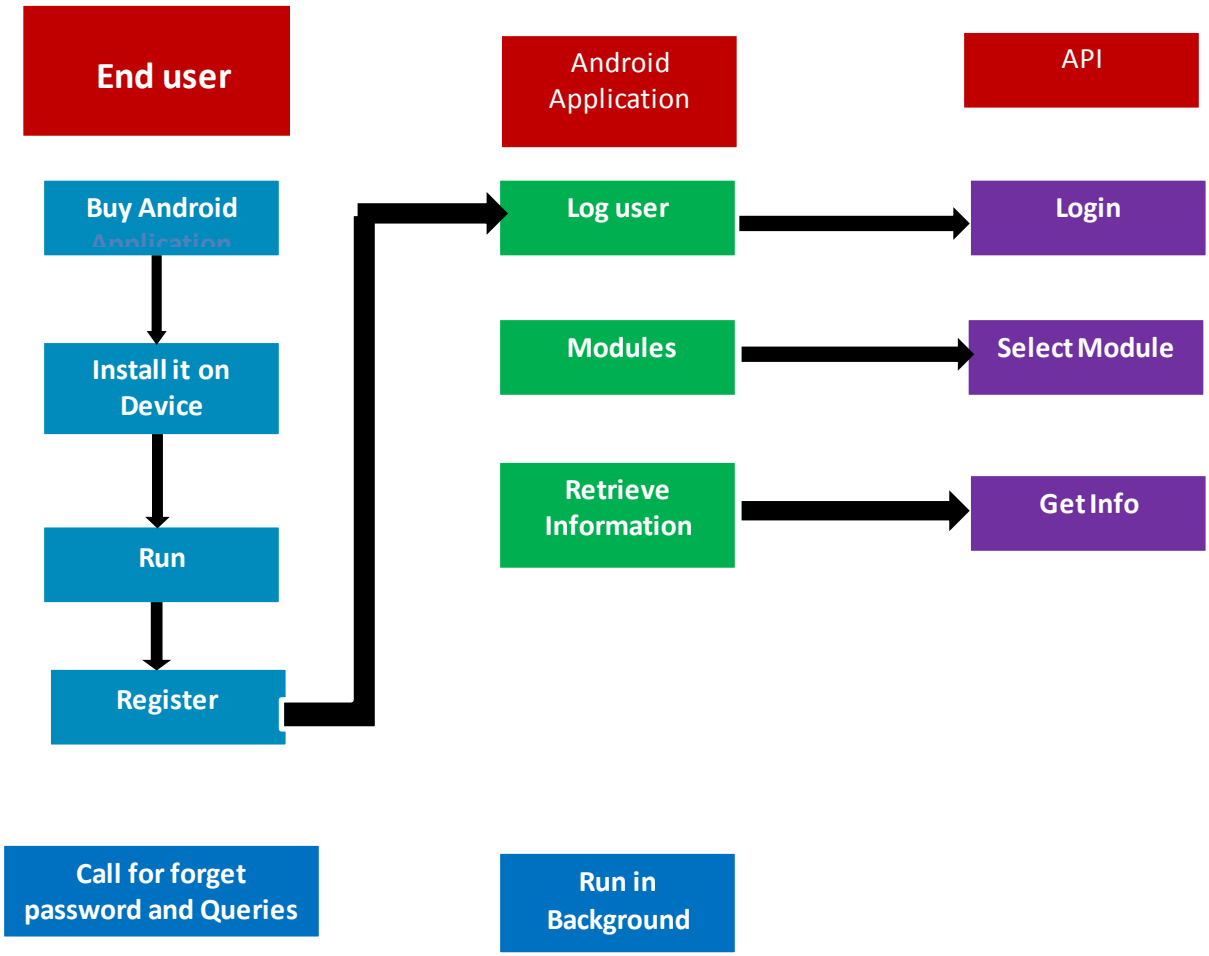
➢ **Data flow**



➢ **Two data packers**

- Represented graphically by an arrow into/out of a process.
- Describes movements of information in the system "data in motion".
- A flow shows direction.
- Double-headed arrow stands for dialogue-convenient packaging of two data packers.
- Data flows can diverge or converge in a DFD.

Data Flow Diagram (Level 0)



Data Flow Diagram (Level 1)

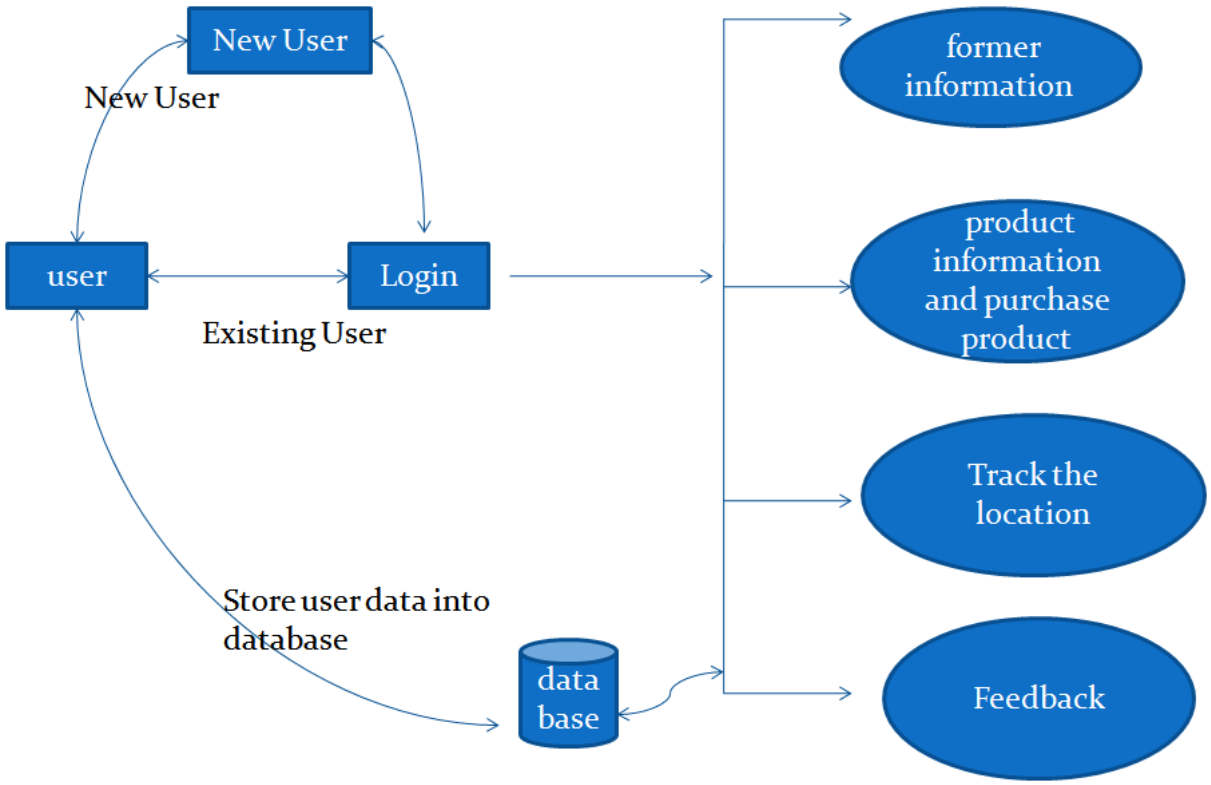


Fig 4.2.2: This shows the separation of all external modules, relationship between those modules and the application.

Data Flow Diagram (Level 2)

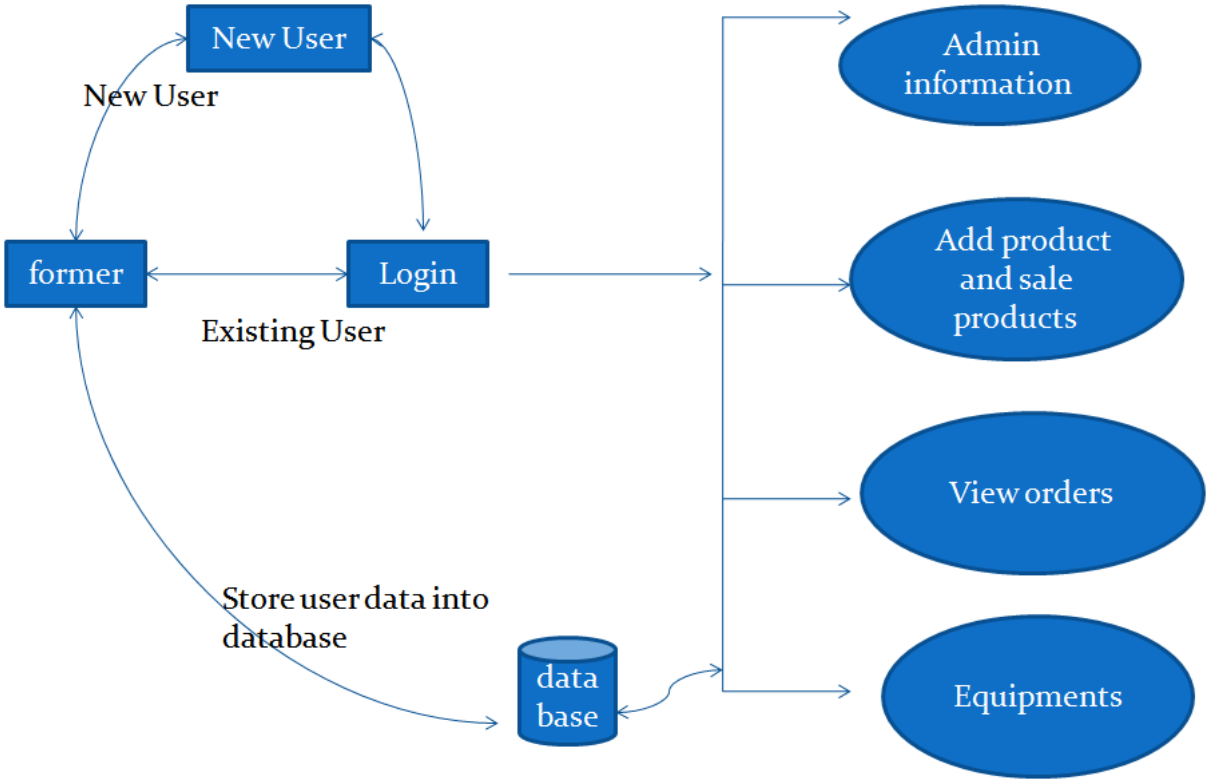


Fig 4.2.3: This differentiates the modules frontend and backend

ER-Diagram:

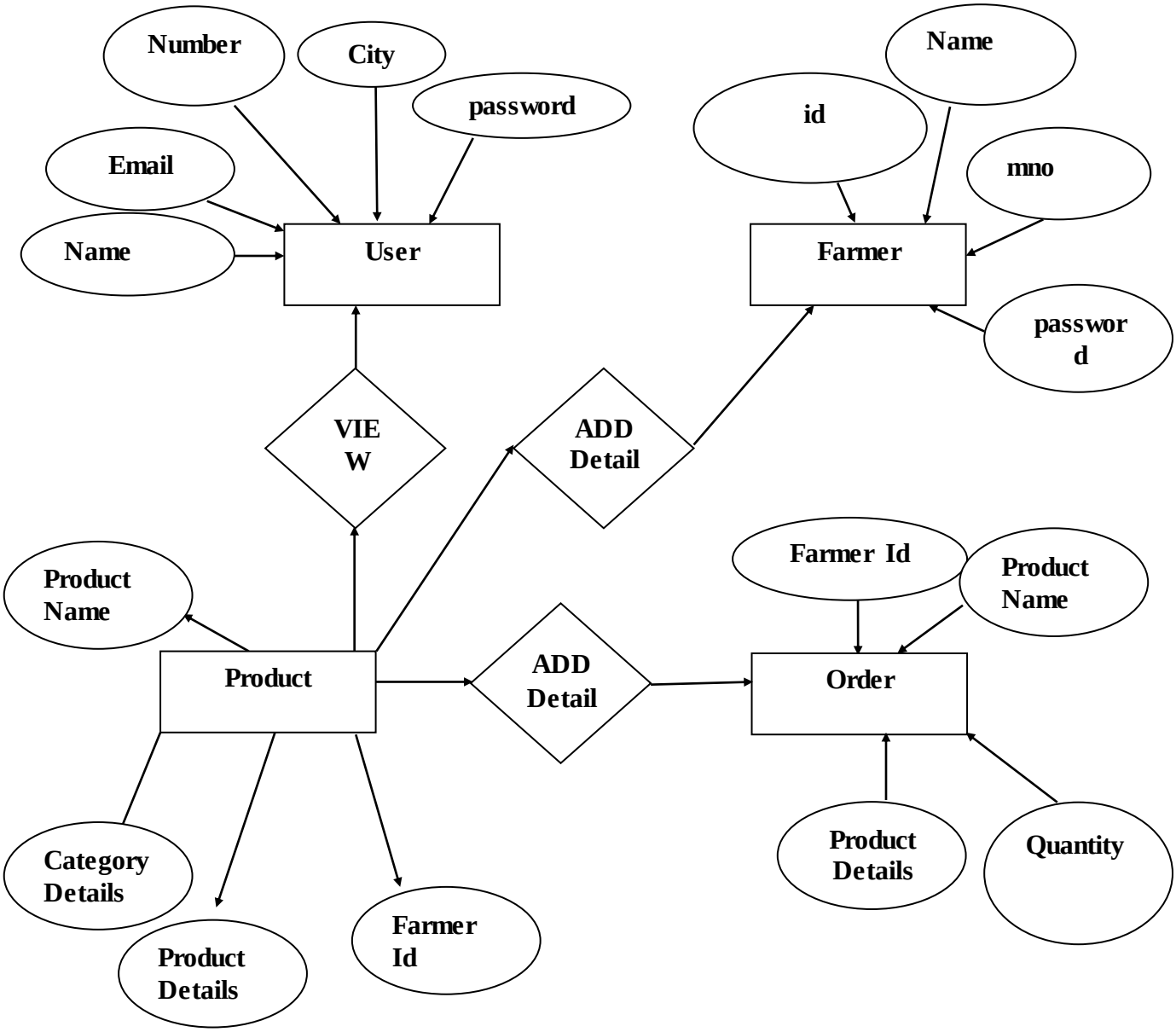


Fig 4.2.4: This illustrates the relationships between entities in a database.

UML (Unified Modeling Language):

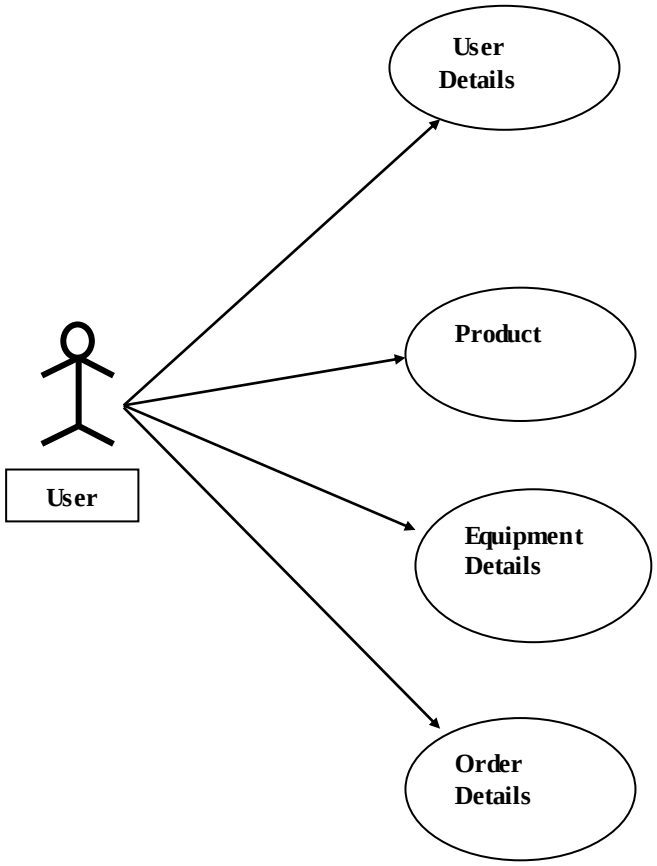


Fig 4.2.5: Standard view to visualize the design of a system.

4.3 Normalization

A Database is a collection of interrelated data stored with a minimum of redundancy to serve many applications. The database design is used to group data into a number of tables and minimizes the artificiality embedded in using separate files. The tables are organized to:

- Reduced duplication of data.
- Simplify functions like adding, deleting, modifying data etc.,
- Retrieving data
- Clarity and ease of use
- More information at low cost

4.3.1 Normalization

Normalization is built around the concept of normal forms. A relation is said to be in a particular normal form if it satisfies a certain specified set of constraints on the kind of functional dependencies that could be associated with the relation. The normal forms are used to ensure that various types of anomalies and inconsistencies are not introduced into the database.

4.3.2 First Normal Form:

A relation R is in first normal form if and only if all underlying domains contained atomic values only.

4.3.3 Second Normal Form:

A relation R is said to be in second normal form if and only if it is in first normal form and every non-key attribute is fully dependent on the primary key.

4.3.4 Third Normal Form:

A relation R is said to be in third normal form if and only if it is in second normal form and every non key attribute is non transitively depend on the primary key.

Chapter 5

IMPLEMANTATION

5.1 Process Modules

This project contains the following five modules:

❖ Farmer Application

- Create Account
- Login
- Add Product Information
- Manipulation of Products
- View Orders
- View the Govt Schema
- Take Equipments Rent
- Feedback

❖ User Application

- Register
- Login
- View product Information
- Purchase products
- To know product information as per city
- Feedback

❖ **Farmer Association Application**

- Register
- Login
- Add agriculture based equipments

5.2 SDLC Methodology:

This document plays a vital role in the development of life cycle (SDLC) as it describes the complete requirement of the system. It means for use by developers and will be the basic during testing phase. Any changes made to the requirements in the future will have to go through formal change approval process. The waterfall model is a sequential design process, often used in software development processes, in which progress is seen as flowing steadily downwards (like a waterfall) through the phases of Conception, Initiation, Analysis, Design, Construction, Testing, Production/Implementation, and Maintenance. In a strict Waterfall model, after each phase is finished, it proceeds to the next one. Reviews may occur before moving to the next phase which allows for the possibility of changes (which may involve a formal change control process). Reviews may also be employed to ensure that the phase is indeed complete; the phase completion criteria are often referred to as a "gate" that the project must pass through to move to the next phase. Waterfall discourages revisiting and revising any prior phase once it's complete. This "inflexibility" in a pure Waterfall model has been a source of criticism by supporters of other more "flexible" models.

The first formal description of the waterfall model is often cited as a 1970 article by Winston W. Royce, although Royce did not use the term "waterfall" in this article. Royce presented this model as an example of a flawed, non-working model. This, in fact, is how the term is generally used in writing about software development—to describe a critical view of a commonly used software development practice.

The waterfall development model originates in the manufacturing and construction industries; highly structured physical environments in which after-the-fact changes are prohibitively costly, if not impossible. Since no formal software development methodologies existed at the time, this hardware-oriented model was simply adapted for software development.

The waterfall model shows a process, where developers are to follow these phases in order:

- Requirements specification (Requirements analysis)
- Software design
- Implementation and Integration
- Testing (or Validation)
- Deployment (or Installation)
- Maintenance

Requirements – defines needed information, function, behavior, performance and interfaces.

Design – data structures, software architecture, interface representations, algorithmic details.

Implementation – source code, database, user documentation, testing.

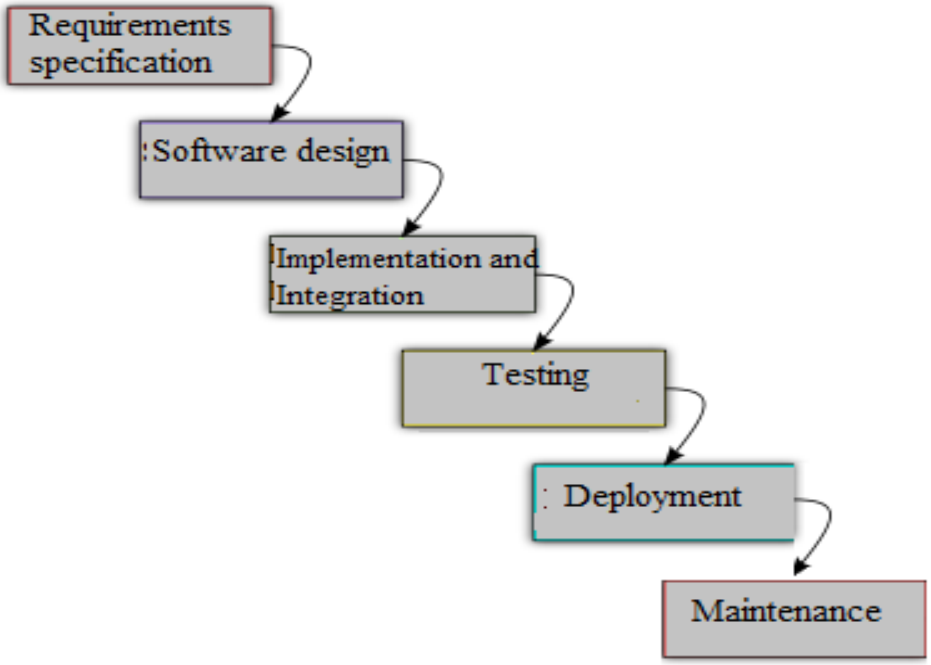


Fig 5.2.1: The following diagram shows how a Water model acts like:

Advantages:

- Easy to understand, easy to use.
- Provides structure to in experienced End User.
- Sets requirements stability.
- Good for management control
- Works well when quality is more important than cost or schedule.
- instantly accessible to users via a browser across a range of devices
- A mobile app is much more dynamic in terms of pure flexibility to update content.
- Application are accessible on all devices

Chapter 6

REQUIREMENT SPECIFICATION

6.1 Functional Requirement

This specification is used to specify the requirements for the initial implementation of the system and update the system in future. The software requirement specification bridges the gap between client/user and the system developer. This is the document that describes the user needs accurately.

6.2 Performance Requirement

This document will provide general description of the project product perspective, and overview of requirement, general constraint and user view of the product while using. In additional will also provide the specific requirement and functional needs for this project such as interface, functional and performance requirements. The purpose of this software requirement specification is to properly document the requirement of the user necessary in order to build this application.

6.3 Software Requirement

- Operating System : Android
- Language : Java
- Minimum SDK version : ANDROID SDK2.3
- Front End : XML
- Back End : PHP, JSON
- Documentation : MS-Office

6.4 Hardware Requirements

CPU type : Intel Pentium 4

Clock speed : 3.0 GHz

Ram size : 512 MB

Hard disk capacity : 40 GB

Monitor type : 15 Inch color monitor

Keyboard type : internet keyboard

Chapter 7

OVERVIEW OF TECHNOLOGIES

7.1 Android System Architecture

In android operating system, there are four layers. Android [1] has its own libraries; it is helpful for developing and designing any application of android platform. These libraries are written in C/C++. Linux kernel is the 1st layer which is written in C. Linux also helps to wrap the application. The unveiling of the Android platform on 5 November 2007 was announced with the founding of the Open Handset Alliance, a consortium of 34 hardware, software and telecom companies devoted to advancing open standards for mobile devices. When released in 2008, most of the Android platform will be made available under the Apache free-software and open source license. Currently Android represents 31.2 percent of the U.S Smartphone market. Android has a large community of developers writing application programs. There are currently over 150,000 apps available for Android. Android Market is the online app store run by Google, though apps can also be downloaded from third party sites.

7.1.1 Features of Android

Application Framework enabling reuse and replacement of components.

- Dalvik Virtual Machine optimized for mobile devices.
- Integrated browser based on the open source Web Kit engine.
- SQLite for structured data storage.
- Media support for common audio, video and still image formats (MPEG4, H.264, MP3, AAC, AMR, JPG, PNG, GIF).
- GSM Telephony (hardware dependent).
- Bluetooth, EDGE, 3G, and Wi-Fi (hardware dependent)

7.1.2 Android Architecture

The following diagram shows the major components of the Android operating system. Each section is described in more detail below:

- Application
- Application Framework
- Libraries
- Android Runtime

7.1.3 Structure

Android is a widely anticipated open source operating system for mobile devices that provides a base operating system, an application middleware layer, a Java software development kit (SDK), and a collection of system applications. Android mobile application development is based on Java language codes, as it allows developers to write codes in the Java language.

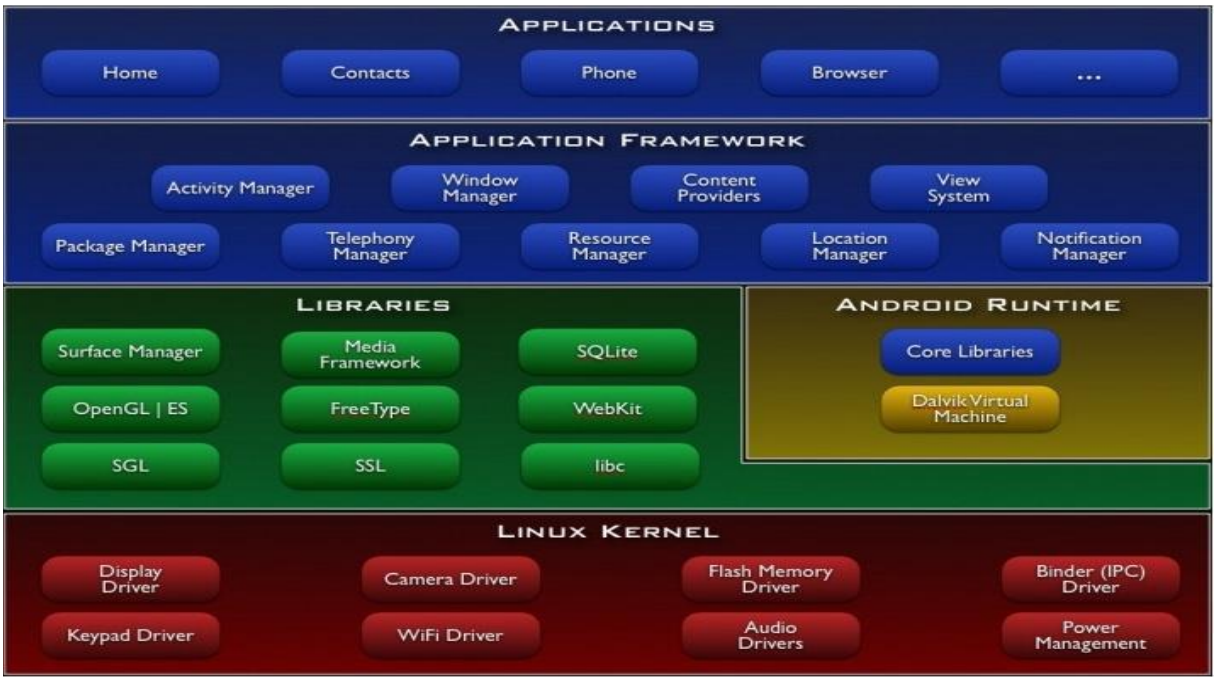


Fig 7.1.1: Android Technological Overview

Android will ship with a set of core applications including an email client, SMS program, calendar, maps, browser, contacts and others. All applications are written using the Java programming language.

7.2 Introduction to Java

It is a product of Sun Microsystems. Java is C++ for the rest. It uses C++ style syntax but has been designed to be much, much easier to use. The core language is quite small and is extended by myriad packages of useful routines. In fact, most of the time programming in Java involves finding the right routine in the right package to do the job.

Other possible advantages of Java include:

- Object-oriented
- Platform independence
- Excellent networking capabilities
- No pointers, so it is easier to program and debug than C++
- Lots of support available in books and on the web
- Wide variety of Application Programmer Interfaces (APIs)
- Implementations are available free of charge for many popular platforms.

7.2.1 The Java Platform

A platform is the hardware or software environment in which a program runs. The Java platform differs from most other platforms in that it's a software-only platform that runs on top of other, hardware-based platforms. Most other platforms are described as a combination of hardware and operating system. The Java platform has two components:

- The Java Virtual Machine (Java VM).
- The Java Application Programming Interface (Java API).

You've already been introduced to the Java VM. It's the base for the Java platform and is ported onto various hardware-based platforms. The Java API is a large collection of ready-made software components that provide many useful capabilities, such as graphical user interface (GUI) widgets.

The Java API is grouped into libraries (packages) of related components. The following figure depicts a Java program, such as an application or applet, that's running on the Java platform. As the figure 7.2.1 shows, the Java API and Virtual Machine insulates the Java program from hardware dependencies.

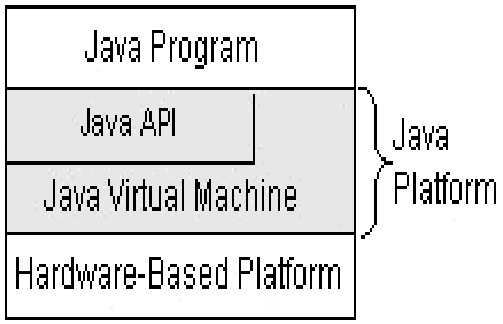


Fig 7.2.1: The Java Platform

As a platform-independent environment, Java can be a bit slower than native code. However, smart compilers, well-tuned interpreters, and just-in-time byte code compilers can bring Java's performance close to that of native code without threatening portability.

7.3 Introduction to Android

7.3.1 Android SDK 2.3

Android is an operating system based on Linux with a Java programming interface. The Android Software Development Kit (Android SDK) provides all necessary tools to develop Android applications. This includes a compiler, debugger and a device emulator, as well as its own virtual machine to run Android programs. Android is primarily developed by Google. Android allows background processing, provides a rich user interface library, supports 2-D and 3-D graphics using the OpenGL libraries, access to the file system and provides an embedded SQLite database. Android application consists of different components and can re-use components of other applications.

This leads to the concept of a *task* in Android; an application can re-use other Android components to archive a task. For example, you can trigger from your application another application which has itself registered with the Android system to handle photos. In this other application you select a photo and return to your application to use the selected photo.

7.3.2 Basic Android User Interface components

The following gives a short overview of the most important user interface components in Android.

- **Activity**

An Activity represents the visual representation of an Android application. Activities use Views and Fragments to create the user interface and to interact with the user. An Android application can have several Activities.

- **Views and ViewGroups**

- Views are user interface widgets, e.g. buttons or text fields. The base class for all Views is the android.view.View class. Views have attributes which can be used to configure their appearance and behavior.

- A ViewGroup is responsible for arranging other Views. View Groups is also called layout managers. The base class for these layout managers is the android.view.ViewGroup class which extends the View class.

- ViewGroups can be nestled to create complex layouts. You should not nestle ViewGroups too deeply as this has a negative impact on the performance.

7.3.3 Other Android components

Android has several more components which can be used in our Android application.

- **Intents**

Intents are asynchronous messages which allow the application to request functionality from other components of the Android system, e.g. from Services or Activities. An application can call a component directly (explicit Intent) or ask the Android system to evaluate registered components based on the Intentdata (implicit Intents). For example, the application could implement sharing of data via Intent and all components which allow sharing of data would be available for the user to select. Applications register themselves to Intent via an IntentFilter. Intents allow combining loosely coupled components to perform certain tasks.

- **Services**

Services perform background tasks without providing a user interface. They can notify the user via the notification framework in Android.

- **ContentProvider**

A ContentProvider provides a structured interface to application data, via ContentProvider your application can share data with other applications. Android contains SQLite database which is frequently used in conjunction with a ContentProvider. The SQLite database would store the data, which would be accessed via ContentProvider.

- **BroadcastReceiver**

BroadcastReceiver can be registered to receive system messages and Intents. A BroadcastReceiver will get notified by the Android system, if the specified situation happens. For example, a BroadcastReceivercould get called once the Android system completed the boot process or if a phone call is received.

7.3.4 Android Development Tools

- **Android SDK**

The Android Software Development Kit (SDK) contains the necessary tools to create, compile and package Android application. Most of these tools are command line based. The Android SDK also provides an Android device emulator, so that Android applications can be tested without a real Android phone. You can create Android virtual devices (AVD) via the Android SDK, which run in this emulator. The Android SDK contains the Android debug bridge (ADB) tool which allows connecting to a virtual or real Android device.

- **ADT**

Google provides the Android Development Tools (ADT) to develop Android applications with Eclipse. ADT is a set of components (plug-ins) which extend the Eclipse IDE with Android development capabilities. ADT contains all required functionalities to create, compile, debug and deploy Android applications from the Eclipse IDE. ADT also allows creating and starting AVDs.

The Android Development Tools (ADT) provides specialized editors for resources files, e.g. layout files. These editors allow switching between the XML representation of the file and a richer user interface via tabs on the bottom of the editor.

- **Dalvik Virtual Machine**

The Android system uses a special virtual machine, i.e. the Dalvik Virtual Machine to run Java based applications. Dalvik uses an own bytecode format which is different from Java bytecode. Therefore, you cannot directly run Java class files on Android; they need to get converted in the Dalvik bytecode format.

7.3.5 How to develop Android Applications?

Android applications are primarily written in the Java programming language. The Java source files are converted to Java class files by the Java compiler. The Android SDK contains a tool called dx which converts Java class files into a .dex (Dalvik Executable) file. All class files of one application are placed in one compressed .dex file. During this conversion process redundant information in the class files are optimized in the .dex file. For example, if the same String is found in different class files, the .dex file contains only once reference of this String. These .dex files are therefore much smaller in size than the corresponding class files.

The .dex file and the resources of an Android project, e.g. the images and XML files, are packed into an .apk (Android Package) file. The program AAPT (Android Asset Packaging Tool) performs this packaging. The resulting .apk file contains all necessary data to run the Android application and can be deployed to an Android device via the ADB tool. The Android Development Tools (ADT) performs these steps transparently to the user. If you use the ADT tooling, you press a button the whole Android application (.apk file) will be created and deployed.

- **Eclipse IDE**

Eclipse is a multi-language software development environment comprising an integrated development environment (IDE) and an extensible plug-in system. It is written mostly in Java. It can be used to develop applications in Java and, by means of various plug-ins, other programming languages including Ada, C, C++, COBOL, Perl, PHP, Python, Ruby (including Ruby on Rails framework), Scala, Clojure, Groovy and Scheme. Development environments include the Eclipse Java development tools (JDT) for Java, Eclipse CDT for C/C++ and Eclipse PDT for PHP, among others.

The initial codebase originated from Visual Age. The Eclipse SDK (which includes the Java development tools) is meant for Java developers. Users can extend its abilities by installing plug-ins written for the Eclipse Platform, such as development toolkits for other programming languages, and can write and contribute their own plug-in modules. Released under the terms of the Eclipse Public License, Eclipse SDK is free and open source software. It was one of the first IDEs to run under GNU Class path and it runs without issues under Iced Tea. Android is the head-to-head competitor for iOS (Apple) created by Google Inc. and Open Handset Alliance. Now a days its becoming more and more popular among the mobile app developers because of its simplicity, reliability and ease of coding. There are many ways to develop Android applications on your PC. The easiest way is integrating the ADT (Android Developing Tools) with the Eclipse IDE.

7.4 Technologies to Be Used

7.4.1 Android with Eclipse

It provides the following:

- The Android project wizard, which generates all the required project files.
- Android-specific resource editors.
- The Android SDK and AVD (Android Virtual Devices) Manager.
- The Eclipse DDMS perspective for monitoring and debugging Android applications.
- Integration with Android Log Cat logging.
- Automated builds and application deployment to Android emulators and handsets.
- Application packaging and code signing tools for release deployment.

7.4.2 Creating Android Projects

The Android Project Wizard creates all the required files for an Android application. Open Eclipse and follow these steps to create a new project:

- Choose File, New, Android Project or click the Android Project reactor icon, which looks like a folder (with the letter *a* and a plus sign :) on the Eclipse toolbar.
- Choose a project name. In this case, name the project Android1.

- Choose a location for the project. Because this is a new project.
- Create New Project in Workspace radio button. Check the Use Default Location checkbox.
- Select a build target for your application. For most applications, you want to select the version of Android most appropriate for the devices used by your target audience and the needs of your application.
- Specify an application name. This name is what users will see. In this case, call the application Android #1.
- Specify a package name, following standard package namespace conventions for Java. Because all code in this book falls under the com.androidbook.namespace, use the package name com. androidbook.Android1.
- Check the Create Activity check box. This will instruct the wizard to create a default launch Activity class for the application. Call your activity Android Activity.
- Confirm that the Min SDK Version field is correct. This field will be set to the API level of the build target. If you want to support older versions of the Android SDK, you need to change this field.
- However, in this case, we can leave it as its default value.
- Click the Next button.
- The Android project wizard allows you to create a test project in conjunction with your Android application. For this example, a test project is unnecessary.
- However, you can always add a test project later by clicking the Android Test Project creator icon, which is to the right of the Android project wizard icon on the Eclipse toolbar.
- Click the Finish button.

7.5 DALVIK VIRTUAL MACHINE

Dalvik is the process virtual machine (VM) in Google's Android operating system. It is the software that runs the apps on Android devices. Dalvik is thus an integral part of Android, which is typically used on mobile devices such as mobile phones and tablet computers as well as more recently on embedded devices such as smart TVs and media streamers. Programs are commonly written in Java and compiled to byte code. They are then converted from Java Virtual Machine-compatible .class files to Dalvik-compatible .dex (Dalvik Executable) files before installation on a device.

The compact Dalvik Executable format is designed to be suitable for systems that are constrained in terms 16 of memory and processor speed. Dalvik is open-source software. It was originally written by Dan Bornstein, who named it after the fishing village of Dalvik in Iceland, where some of his ancestors lived.

7.6 SQLite Database

SQLite is an ACID-compliant embedded relational database management system contained in a small (~275 kB) C programming library. SQLite implements most of the SQL standard, using a dynamically and weakly typed SQL syntax that does not guarantee the domain integrity. In contrast to other database management systems, SQLite is not a separate process that is accessed from the client application, but an integral part of it. SQLite read operations can be multitasked, though writes can only be performed sequentially.

The source code for SQLite is in the public domain. SQLite is a popular choice for local/client storage on web browsers. It has many bindings to programming languages. It is arguably the most widely deployed database engine, as it is used today by several widespread browsers, operating systems, and embedded systems, among others. SQLite is an open source embeddable database engine written in C by D. Richard Hipp. It is entirely self-contained with no external dependencies. It was introduced as an option in PHP V4.3 and is built into PHP V5.

SQLite supports much of the SQL92 standard, runs on all major operating systems, and has support for the major computer languages. SQLite is also surprisingly robust. Its creator

conservatively estimates that it can handle a Web site with a load of up to 10000 hits a day, and there have been cases where SQLite has handled a load 10 times that.

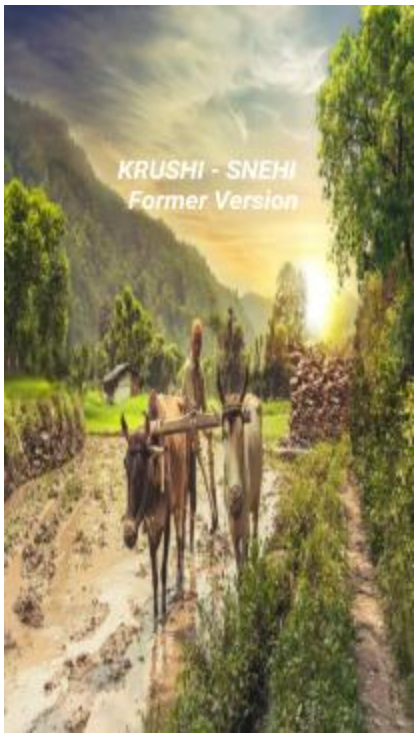
Databases have been an integral part of software applications since the dawn of the commercial application market several decades ago. As crucial as database management systems are, they also come with a large footprint, and considerable overhead in system resources and administration complexity.

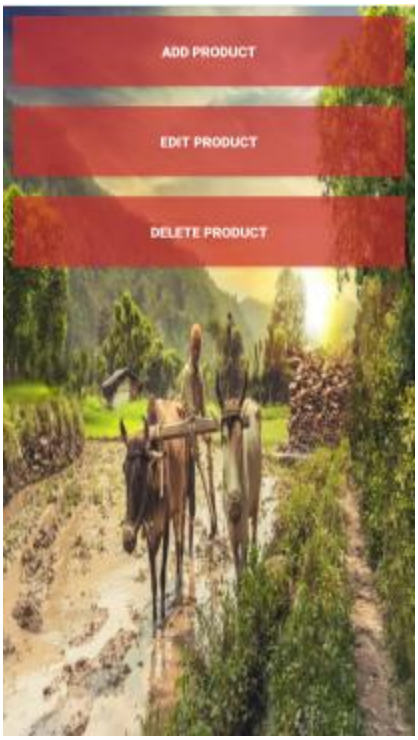
As software applications become less monolithic and more modular, a new type of database can be a better fit than the larger and more complex traditional database management systems. Embeddable databases run directly in the application process, offer zero-configuration run modes, and have very small footprints.

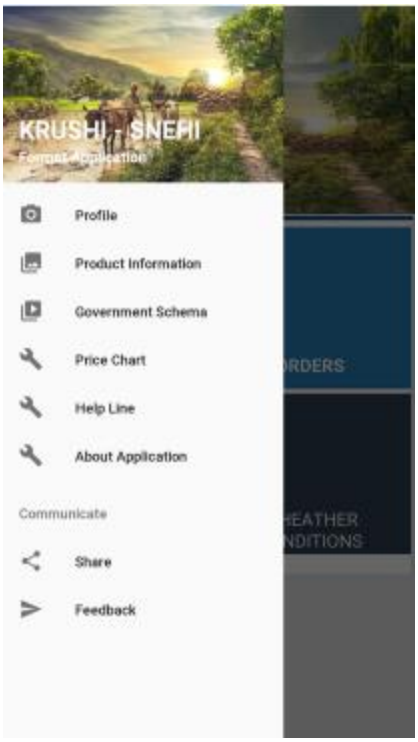
This article introduces the popular SQLite database engine and describes how to use it in application development. SQLite's support of the SQL92 standard includes indices, limitations, triggers, and views. SQLite does not support foreign key constraints, but supports Atomic, Consistent, Isolated, and Durable (ACID) transactions.

Chapter 8

OUTPUT SCREENS

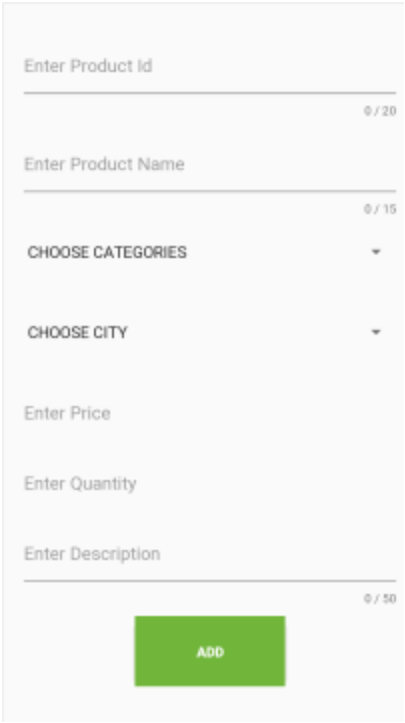



ADD PRODUCT
EDIT PRODUCT
DELETE PRODUCT


KRUSHI - SNEHI
Profile
Product Information
Government Schema
Price Chart
Help Line
About Application
Communicate
Share
Feedback


PRODUCT
ORDERS
PRODUCT INFORMATION
WHEATHER CONDITIONS


FORMER
USER


Enter Product Id
0 / 20
Enter Product Name
0 / 15
CHOOSE CATEGORIES
CHOOSE CITY
Enter Price
Enter Quantity
Enter Description
0 / 50
ADD

Chapter 9

SYSTEM TESTING

9.1 Introduction

The philosophy behind testing is to find errors. The common view of testing is that it is performed to prove that there are no errors in a program. However, it is virtually impossible to prove that no program will be free and clear of errors. Therefore, the most useful approach and practical approach is with the understanding that testing is the process of executing a program with explicit intention of finding errors that is, making the program fail.

9.2 Testing Strategies

9.2.1 Unit Testing:

Unit testing is the process of verifying the logic, functionality, computations, and error handling of a unit. The intent of unit testing is to confirm that the unit provides the capability assigned to it, correctly interfaces with other units and data, and is a faithful implementation of the unit design. Unit testing focuses verification effort on the smallest unit of software design in the module. The important control paths are tested to uncover errors and invalid flow within the boundary of the module. The unit testing is normally structural oriented testing and can be conducted in parallel for multiple modules independently.

Unit testing was done for all the sub entities of the module simultaneously with the development. In all the units the flow of data across the control structures were tested and all independent paths within each module were traversed at least once. All logical decision paths were exercised for both affirmative and negative response to the specified conditions. Also the consistencies of the data that are processed by the unit were verified before and after its execution.

9.2.2 Integration Testing:

Integration testing is a systematic technique for constructing the program structure while conducting tests to uncover errors associated with the interfacing between modules. The individual units are combined with other units to make sure that necessary communication, links and data sharing occur properly. There are three basic integration-testing methods:

- Top-down
- All-at-once
- Bottom-up

The method of integration testing that is used should be appropriate to the design of the system. Top-down testing fits a prototyping environment that establishes an initial skeleton that fills individual modules when they are completed. The testing of this system follows the top-down integration approach since menu-driven systems generally lend themselves to top-down testing. The modules were verified using the existing, previously tested build as a test bed. Units not yet implemented exist in the module as stubs; that is, they contain no executable instructions except to write a message that the unit was entered and has returned control to the calling unit. Using this approach both the module's integration into the growing system and the internal code of the units that it comprises were tested.

9.2.3 Validation Testing

Validation tests are positive tests. They confirm that the system meets the requirements such that the proper inputs give out the desired output. This test focuses on the input-output behavior of the system and checks the validity of the data used. All the data fields entered by the user are checked for validity and invalid user entry is acknowledged with corresponding error messages. Tests are done to check that the desired output is produced and is in relevance with the input given.

Chapter 10

SYSTEM SECURITY

10.1 Introduction

Computer security is a branch of computer technology known as information security as applied to computers and networks. The objective of computer security includes protection of information and property from theft, corruption, or natural disaster, while allowing the information and property to remain accessible and productive to its intended users. The term computer system security means the collective processes and mechanisms by which sensitive and valuable information and services are protected from publication, tampering or collapse by unauthorized activities or untrustworthy individuals and unplanned events respectively. The strategies and methodologies of computer security often differ from most other computer technologies because of its somewhat elusive objective of preventing unwanted computer behavior instead of enabling wanted computer behavior.

10.2 Security in software

Secure Coding

If the operating environment is not based on a secure operating system capable of maintaining a domain for its own execution, and capable of protecting application code from malicious subversion, and capable of protecting the system from subverted code, then high degrees of security are understandably not possible. While such secure operating systems are possible and have been implemented, most commercial systems fall in a 'low security' category because they rely on features not supported by secure operating systems (like portability, and others).

In low security operating environments, applications must be relied on to participate in their own protection. There are 'best effort' secure coding practices that can be followed to make an application more resistant to malicious subversion. In commercial environments, the majority of software subversion vulnerabilities result from a few known kinds of coding defects. Common software defects include buffer overflows, format string vulnerabilities, integer overflow, and code/command injection. It is to be immediately noted that all of the foregoing are specific instances

of a general class of attacks, where situations in which putative "data" actually contains implicit or explicit, executable instructions are cleverly exploited.

Some common languages such as C and C++ are vulnerable to all of these defects (see Sea cord, "Secure Coding in C and C++"). Other languages, such as Java, are more resistant to some of these defects, but are still prone to code/command injection and other software defects which facilitate subversion. Recently another bad coding practice has come under scrutiny; dangling pointers. The first known exploit for this particular problem was presented in July 2007. Before this publication the problem was known but considered to be academic and not practically exploitable.

Unfortunately, there is no theoretical model of "secure coding" practices, nor is one practically achievable, insofar as the variety of mechanisms are too wide and the manners in which they can be exploited are too variegated. It is interesting to note, however, that such vulnerabilities often arise from archaic philosophies in which computers were assumed to be narrowly disseminated entities used by a chosen few, all of whom were likely highly educated, solidly trained academics with naught but the goodness of mankind in mind.

Thus, it was considered quite harmless if, for (fictitious) example, a FORMAT string in a FORTRAN program could contain the J format specified to mean "shut down system after printing." After all, who would use such a feature but a well-intentioned system programmer? It was simply beyond conception that software could be deployed in a destructive fashion.

It is worth noting that, in some languages, the distinction between code (ideally, read-only) and data (generally read/write) is blurred. In LISP, particularly, there is no distinction whatsoever between code and data, both taking the same form: An S-expression can be code, or data, or both, and the "user" of a LISP program who manages to insert an executable LAMBDA segment into putative "data" can achieve arbitrarily general and dangerous functionality. Even something as "modern" as Perl offers the eval () function, which enables one to generate Perl code and submit it to the interpreter, disguised as string data.

Chapter 11

SOURCE CODE

1Flash Page.

```
package com.example.krushi_snehi_farmer_app;


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


public class Flash_page extends AppCompatActivity {


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




        Thread thread=new Thread()
        {
            @Override
```

```
        public void run() {
            try {
                sleep(3000);
                Intent intent=new Intent(getApplication(),MainActivity.class);
                startActivity(intent);
                finish();

            }
            catch (Exception e)
            {

            }
        }
    };
    thread.start();
  }
}
```

2.Login Page

```
package com.example.krushi_snehi_farmer_app;

import android.content.Intent;
import android.os.StrictMode;
import android.support.v7.app.AppCompatActivity;
```

```
import android.os.Bundle;

import android.util.Log;

import android.view.View;

import android.widget.Button;

import android.widget.EditText;

import android.widget.Toast;


import org.apache.http.HttpEntity;

import org.apache.http.HttpResponse;

import org.apache.http.NameValuePair;

import org.apache.http.client.HttpClient;

import org.apache.http.client.entity.UrlEncodedFormEntity;

import org.apache.http.client.methods.HttpPost;

import org.apache.http.impl.client.DefaultHttpClient;

import org.apache.http.message.BasicNameValuePair;


import java.io.BufferedReader;

import java.io.InputStream;

import java.io.InputStreamReader;

import java.util.ArrayList;


public class MainActivity extends AppCompatActivity

{
```

```
EditText editText_mno,editText_password;
Button button_login;
StrictMode.ThreadPolicy  policy = new StrictMode.ThreadPolicy.Builder().permitAll().build();
SharedPrefHandler  sharedPrefHandler;

String s1,s2;
@Override
protected void onCreate(Bundle  savedInstanceState) {
   super.onCreate(savedInstanceState);
   setContentView(R.layout.activity_main);

   editText_mno=(EditText)findViewById(R.id.et_activity_main_mno);
   editText_password=(EditText)findViewById(R.id.et_activity_main_password);
   button_login=(Button)findViewById(R.id.btn_activity_main_login);

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

```
        if(editText_mno.getText().toString()==""  || editText_password.getText().toString()=="")

            Toast.makeText(getApplication(),  "Please Enter Username / Password",
Toast.LENGTH_LONG).show();



        else



            chk_log();



      }

    });



}



private  void chk_log()
```

```
{
    try
    {
        StrictMode.setThreadPolicy(policy);
        InputStream is=null;

        s1 = editText_mno.getText().toString();
        s2 = editText_password.getText().toString();

        sharedPrefHandler.setSharedPreferences("mno",s1);

        String result=null;

        ArrayList<NameValuePair> nvp=new ArrayList<NameValuePair>();

        nvp.add(new BasicNameValuePair("f1", s1));
        nvp.add(new BasicNameValuePair("f2", s2));

        //Toast.makeText(getApplication(), v1, Toast.LENGTH_LONG).show();
        HttpClient hclient=new DefaultHttpClient();
        HttpPost hpost=new HttpPost("https://unhesitating-
flags.000webhostapp.com/Forest/sksvm/admin/login_page.php");
        hpost.setEntity(new UrlEncodedFormEntity(nvp));
```

```
HttpResponse resp=hclient.execute(hpost);

HttpEntity  hent=resp.getEntity();

is=hent.getContent();

BufferedReader rd=new BufferedReader(new InputStreamReader(is,  "iso-8859-1"), 8);

StringBuilder  sb=new StringBuilder();

String  ln=null;

while((ln=rd.readLine())!=null)

{

   sb.append(ln);

}

is.close();

result=sb.toString().trim();

Log.d("respo",result);



//JSONObject  object = new JSONObject(result);

//  String  ch=object.getString("r");

//  Toast.makeText(getApplication(),  ch, Toast.LENGTH_LONG).show();

result=result.substring(1,  result.length()-1);

//Toast.makeText(getApplication(),  result, Toast.LENGTH_LONG).show();

if(!result.trim().equals("Error"))

{

   String[]  r=result.split("-");
```

```
            //usn=r[0].toString();
            //std_name=r[1].toString().toUpperCase();

            Toast.makeText(this, "Login Successfull", Toast.LENGTH_SHORT).show();

            //sharedPrefHandler.setSharedPreferences("email",usn);
            Intent in=new Intent(getApplication(), Main_home.class);
            startActivity(in);
        }
        else
            Toast.makeText(getApplication(), "Invalid Username /Password",
Toast.LENGTH_LONG).show();

    }
    catch(Exception ex)
    {
        Toast.makeText(getApplication(), ex.getMessage(), Toast.LENGTH_LONG).show();
    }

}
```

}

CONCLUSION AND FUTURE WORK

Basic idea of our application is to provide ease to software project management. Our application works on Android, a phone that adds the mobility feature. The application acts as a rich tool for sending predetermined message to the registered contacts including a callfor any help. The user can access the data from anywhere anytime through the mobile phone. It can also estimate the cost of the project. It provides the facility to analyze and control the execution of project. Alerts are automatically sent to the users for the update in data.

In future, more functionality can be added to make this application more robust and more feature rich. With the advent of smart phones, this application, when developed to its fullest, would be able for all to use and make their information much more easily accessable and comfortable.

BIBILOGRAPHY

1. Introducing Basic development of android book named "*Hello, Android"*, third addition written by "Ed. Burnette", published by "The Pragmatic Programmers".
2. *http://www2.paho.org/hq/dmdocuments/violence-against-women-lac*
3. Copyright © 2009-2011 CommonsWare, LLC. All Rights Reserved. Printed in the
    a. United States of America. Printing History:
    b. Mar 2011:Version 3.2 ISBN: 978-0-9816780-4-7
4. M. B. Piedade and M. Y. Santos, "Student relationship management:
5. Concept, practice and technological support," in IEEE International
    a. Engineering Management Conference, 2008, pp. 1–5.
    b. Published by O'Reilly Media, Inc., 1005 Gravenstein Highway North, Sebastopol, CA 95472.
6. Introduction to Android http://developer.android.com/guid/index.html
7. Copyright © 2009-2010 and B.Maccgrall, "College management system using android":