INTRODUCTION

CHAPTER-01

INTRODUCTION

1.1 Introduction of the project:

“Kisan Mitra Mobile Apps Which provide Agricultural Information to farmer community .and fulfill the need of farmers community in the area of Agriculture, Horticulture, Veterinary. Information of In this aaps is about the all the crops and crops related issues like weather which favorable to the particular crop, irrigation period , pesticide and more.

New feature is available is that farmer can directly chat with experts for any personal information or problem solution. farmer simply have to register his/her number on apps and he/she gets groups of different fields. farmer can put his/her question of problem on that group chat board and experts of that group will put solution of farmers.

1.2 Objectives:

- To Reduce migration
- To Organize farmers interest groups.
- To Reduce indebtedness among farmers.
- To Enhamces food security at community level.
- To Promote at source value addition of farm produce .
- To Make agriculture economically viable and sustainable.
- To Eliminate sucides due to agrarian crisis in the project area.
- To Advocate to effective and farmer friendly agricultural policies

LITERATURE SURVEY

CHAPTER-2

LITERATURE SURVEY

2.1 EXISTING SYSTEM:

1. Grofers – provides customers the easy way of buying, but these will source it from the retails.
2. Big basket – similar way of grofers.
3. In the existing system all transactions, dealings of products, purchasing of products were done manually which is time consuming.
4. Reports are prepared manually as and when needed. Maintaining of reports is very tedious task.
5. To buy any product user has to collect information about it either byvisiting the shop or asking people which is the better one.
6. There is no computer system for handling payments.
7. Any internet user can use this existing website to search for any kindof products, select particular products from a wide range of products.
8. Once they make of their mind to purchase any particular thing they
   can place an order and make a payment throw various available payment option

2.2 PROPOSED SYSTEM

- Introducing Farmers to sell the crops to mediator based on the market price.
- Farmers will get to know about the current market price.
- Enables farmers to take off the crops from the field based on high market price.
- Prior information of stock available on stock to mediator.
- Easy of managing the stocks and price to mediator
- Auto generated consumer price.
- Consumers will not be cheated by paying more money.
- Mobile app to shop groceries.

2.3 MODULES

2.3.1 User Management

- User Registration
- Verification
- PreProduction
- PostProduction
- Blogs
- Cropinfo

2.3.2 Admin Module

- Admin Registration
- Verification
- List of Numbers (numbers of farmer who puts their questions)
- Examine the problems
- Gives proper solutions of problems

CHAPTER-3

TECHNOLOGY INFRASTRUCTURE

3.1 Java

Java is a programming language originally developed by James Gosling at Sun Microsystems (which is now a subsidiary of Oracle Corporation) and released in 1995 as a core component of Sun Microsystems Java platform. The language derives much of its syntax from C and C++ but has a simplerobject model and fewer low-level facilities. Java applications are typically compiled to bytecode (class file) that can run on any Java Virtual Machine (JVM) regardless of computer architecture. Java is a general-purpose, concurrent, class-based, object-oriented language that is specifically designed to have as few implementation dependencies as possible. It is intended to let application developers "write once, run anywhere". Java is currently one of the most popular programming languages in use, and is widely used from application software to web applications.

The original and reference implementation Java compilers, virtual machines, and class libraries were developed by Sun from 1995. As of May 2007, in compliance with the specifications of the Java Community Process, Sun relicensed most of its Java technologies under the GNU General Public License. Others have also developed alternative implementations of these Sun technologies, such as the GNU Compiler for Java, GNU Classpathand Dalvik.

James Gosling, Mike Sheridan, and Patrick Naughton initiated the Java language project in June 1991. Java was originally designed for interactive television, but it was too advanced for the digital cable television industry at the time. The language was initially called Oakafteran oak tree that stood outside Gosling's office; it went by the name Green later, and was later renamed Java, from a list of random words. Gosling aimed to implement a virtual machine and a language that had a familiar C/C++ style of notation.

Sun Microsystems released the first public implementation as Java 1.0 in 1995. It promised "Write Once, Run Anywhere”, providing no-cost run-times on popular platforms. Fairly secure and featuring configurable security, it allowed network- and file-access restrictions. Major web browsers soon incorporated the ability to runJava applets within web pages, and Java quickly became popular.With the advent of Java 2 (released initially as J2SE 1.2 in December 1998–1999), new versions had multiple configurations built for different types of platforms. For example, J2EE targeted enterprise applications and the greatly stripped-down version J2MEfor mobile applications (Mobile Java). J2SE designated the Standard Edition. In 2006, for marketing purposes, Sun renamed new J2 versions as Java EE, Java ME, andJava SE, respectively.

On November 13, 2006, Sun released much of Java as open source software under the terms of the GNU General Public License (GPL). On May 8, 2007, Sun finished the process, making all of Java's core code available under free software/open-source distribution terms, aside from a small portion of code to which Sun did not hold the copyright.Sun's vice-president Rich Green has said that Sun's ideal role with regards to Java is as an "evangelist."

Following Oracle Corporation's acquisition of Sun Microsystems in 2009–2010, Oracle has described itself as the "steward of Java technology with a relentless commitment to fostering a community of participation and transparency".

3.1.1 PRINCIPLES

There were five primary goals in the creation of the Java language

1. It should be "simple, object oriented and familiar".
2. It should be "robust and secure".
3. It should have "an architecture-neutral and portable environment".
4. It should execute with "high performance".
5. It should be "interpreted, threaded, and dynamic".

3.2 JAVA FEATURES

A)PlatformIndependent

The concept of Write-once-run-anywhere (known as the Platform independent) is one of the important key feature of java language that makes java as the most powerful language. Not even a single language is idle to this feature but java is closer to this feature. The programs written on one platform can run on any platform provided the platform must have the JVM.

B) Simple

There are various features that make the java as a simple language. Programs are easy to write and debug because java does not use the pointers explicitly. It is much harder to write the java programs that can crash the system but we cannot say about the other programming languages. Java provides the bug free system due to the strong memory management. It also has the automatic memory allocation and de allocation system.

C) Object-Oriented

To be an Object Oriented language, any language must follow at least the four characteristics.

- Inheritance: It is the process of creating the new classes and using the behavior of the existing classes by extending them just to reuse the existing code and adding the additional features as needed.
- Encapsulation: It is the mechanism of combining the information and providing the abstraction.
- Polymorphism: As the name suggest one name multiple form, Polymorphism is the way of providing the different functionality by the functions having the same name based on the signatures of the methods.
- Dynamic binding: Sometimes we don't have the knowledge of objects about their specific types while writing our code. It is the way of providing the maximum functionality to a program about the specific type at runtime.

As the languages like Objective C, C++ fulfills the above four characteristics yet they are not fully object oriented languages because they are structured as well as object oriented languages. But in case of java, it is a fully Object Oriented language because object is at the outer most level of data structure in java. No stand alone methods, constants, and variables are there in java. Everything in java is object even the primitive data types can also be converted into object by using the wrapper class.

D) Robust

Java has the strong memory allocation and automatic garbage collection mechanism. It provides the powerful exception handling and type checking mechanism as compare to other programming languages. Compiler checks the program whether there any error and interpreter checks any run time error and makes the system secure from crash. All of the above features make the java language robust.

E) Distributed

The widely used protocols like HTTP and FTP are developed in java. Internet programmers can call functions on these protocols and can get access the files from any remote machine on the internet rather than writing codes on their local system.

F) Portable

The feature Write-once-run-anywhere makes the java language portable provided that the system must have interpreter for the JVM. Java also has the standard data size irrespective of operating system or the processor. These features make the java as a portable language.

G) Dynamic

While executing the java program the user can get the required files dynamically from a local drive or from a computer thousands of miles away from the user just by connecting with the Internet.

H) Secure

Java does not use memory pointers explicitly. All the programs in java are run under an area known as the sand box. Security manager determines the accessibility options of a class like reading and writing a file to the local disk. Java uses the public key encryption system to allow the java applications to transmit over the internet in the secure encrypted form. The bytecode Verifier checks the classes after loading.

I) Performance

Java uses native code usage, and lightweight process called threads. In the beginning

interpretation of byte code resulted the performance slow but the advance version of JVM uses the adaptive and just in time compilation technique that improves the performance.

J) Multithreaded

As we all know several features of Java like Secure, Robust, Portable, dynamic etc; you will be more delighted to know another feature of Java which is Multithreaded. Java is also a multithreaded programming language. Multithreading means a single program having different threads executing independently at the same time. Multiple threads execute instructions according to the program code in a process or a program. Multithreading works the similar way as multiple processes run on one computer. Multithreading programming is a very interesting concept in Java. In multithreaded programs not even a single thread disturbs the execution of other thread. Threads are obtained from the pool of available ready to run threads and they run on the system CPUs. This is how Multithreading works in Java which you will soon come to know in details in later chapters.

K) Interpreted

We all know that Java is an interpreted language as well. With an interpreted language such as Java, programs run directly from the source code. The interpreter program reads the source code and translates it on the fly into computations. Thus, Java as an interpreted language depends on an interpreter program. The versatility of being platform independent makes java to outshine from other languages. The source code to be written and distributed is platform independent. Another advantage of Java as an interpreted language is its error debugging quality. Due to this any error occurring in the program gets traced. This is how it is different to work with Java.

3.3APACHE TOMCAT

Apache Tomcat (or Jakarta Tomcat or simply Tomcat) is an open source servlet container developed by the Apache Software Foundation (ASF). Tomcat implements the Java Servlet and the JavaServer Pages (JSP) specifications from Sun Microsystems, and provides a "pure Java" HTTP web server environment for Java code to run.

Tomcat should not be confused with the Apache web server, which is a C implementation of an HTTP web server; these two web servers are not bundled together. Apache Tomcat includes tools for configuration and management, but can also be configured by editing XML configuration files.

Tomcat started off as a servlet reference implementation by James Duncan Davidson, a software architect at Sun Microsystems. He later helped make the project open source and played a key role in its donation by Sun to the Apache Software Foundation. The Apache Ant software build automation tool was developed as a side-effect of the creation of Tomcat as an open source project.

3.3.1 FEATURES:

- Implements the Servlet 2.4 and JSP 2.0 specifications.
- Reduced garbage collection, improved performance and scalability.
- Native Windows and UNIX wrappers for platform integration.
- Faster JSP parsing.

3.4 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 a modified version of the Linux kernel. Google and other members of the Open Handset Alliance collaborated on Android's development and release] The Android Open Source Project (AOSP) is tasked with the maintenance and further development of Android. The Android operating system is the world's best-selling Smartphoneplatform.

Android has a large community of developers writing applications ("apps") that extend the functionality of the devices. 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. Developers write primarily in the Java language, controlling the device via Google-developed Java libraries.

The unveiling of the Android distribution on 5 November 2007 was announced with the founding of the Open Handset Alliance, a consortium of 80hardware, software, and telecom companies devoted to advancing open standards for mobile devices. Google released most of the Android code under the Apache License, a free software and open source license.

The Android open-source software stack consists of Java applications running on a Java-based, object-oriented application framework on top of Java core libraries running on a Dalvik virtual machine featuring JIT compilation. Libraries written in C include the surface manager, OpenCore media framework, SQLite relational database management system, OpenGL ES 2.0 3D graphics API, WebKit layout engine, SGL graphics engine, SSL, and Bionic libc. The Android operating system, including the Linux kernel, consists of roughly 12 million lines of code including 3 million lines of XML, 2.8 million lines ofC, 2.1 million lines of Java, and 1.75 million lines of C++.

3.5 ECLIPSE

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 and can be used to develop applications in Java and, by means of various plugins,other programminglanguagesincluding Ada, C, C++, COBOL, Perl, PHP, Python, Ruby (including Rub y on Rails framework), Scala, Clojure, and Scheme. The IDE is often called Eclipse ADT for Ada, Eclipse CDT for C/C++, Eclipse JDT for Java, and Eclipse PDT for PHP.

The initial codebase originated from Visual Age. In its default form it is meant for Java developers, consisting of the Java Development Tools (JDT). Users can extend its abilities by installing plug-ins written for the Eclipse software framework, 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 is free and open source software. It was one of the first IDEs to run under GNU Classpath and it runs without issues under IcedTea.

3.5.1 HISTORY

Eclipse began as an IBM Canada project. It was developed by Object Technology International (OTI) as a Java-based replacement for the Smalltalk based Visual Age family of IDE products, which itself had been developed by OTI. In November 2001, a consortium was formed to further the development of Eclipse as open source. In January 2004, the Eclipse Foundation was created.

Eclipse 3.0 (released on 21 June 2004) selected the OSGi Service Platform specifications as the runtime architecture. Eclipse was originally released under the Common Public License, but was later relicensed under the Eclipse Public License. The Free Software Foundation has said that both licenses are free software licenses, but are incompatible with the GNU General Public License (GPL). Mike Milinkovich, of the Eclipse Foundation commented that moving to the GPL would be considered when version 3 of the GPL was released.

3.5.2 ARCHITECTURE

Eclipse employs plug-ins in order to provide all of its functionality on top of (and including) the runtime system, in contrast to some other applications where functionality is typically hard coded. The runtime system of Eclipse is based on Equinox, an OSGi standard compliant implementation.

This plug-in mechanism is a light weight software component framework. In addition to allowing Eclipse to be extended using other programming languages such as C and Python, the plug-in framework allows Eclipse to work with typesetting languages like LaTeX,networking applications such as telnet, and database management systems. The plug-in architecture supports writing any desired extension to the environment, such as for configuration management. Java and CVS support is provided in the Eclipse SDK, with Subversion support provided by third-party plug-ins. With the exception of a small run-time kernel, everything in Eclipse is a plug-in. This means that every plug-in developed integrates with Eclipse in exactly the same way as other plug-ins; in this respect, all features are "created equal". Eclipse provides plug-ins for a wide variety of features, some of which are through third parties using both free and commercial models. Examples of plug-ins include a UML plug-in for Sequence and other UML diagrams, a plug-in for DB Explorer, and many others.

The Eclipse SDK includes the Eclipse Java Development Tools (JDT), offering an IDE with a built-in incremental Java compiler and a full model of the Java source files. This allows for advanced refactoring techniques and code analysis. The IDE also makes use of a *workspace*, in this case a set of metadata over a flat file space allowing external file modifications as long as the corresponding workspace "resource" is refreshed afterwards.

Eclipse implements widgets through a widget toolkit for Java called SWT, unlike most Java applications, which use the Java standard Abstract Window Toolkit (AWT) or Swing. Eclipse's user interface also uses an intermediate GUI layer called JFace, which simplifies the construction of applications based on SWT. Language packs provide translations into over a dozen natural languages.

3.5.3 RICH CLIENT PLATFORM

Eclipse provides the Eclipse Rich Client Platform (RCP) for developing general purpose applications. The following components constitute the rich client platform:

- Equinox OSGi – a standard bundling framework
- Core platform – boot Eclipse, run plug-ins
- Standard Widget Toolkit (SWT) – a portable widget toolkit
- JFace – viewer classes to bring model view controller programming to SWT, file buffers, text handling, text editors.
- Eclipse Workbench – views, editors, perspectives, wizards

3.5.4 ECLIPSE: THE IDE FOR UNISYS CLEARPATH DEVELOPERS

By design, Eclipse itself is a deliberately skinny foundation with minimal functionality. It is built to be extended via the concept of plug-ins, which are code bundles that are loaded and used according to a specified configuration. The basic Eclipse structure defines extension points, which allow new modules (plug-ins) that extend the menus and offer more services.

Developers working within the Eclipse IDE can pick and choose the features they want to use from the wide, open-source world of plug-in options. The choices are nearly endless. Unisys is making Eclipse a major focal point for development of ClearPath applications. In fact, we plan to eventually replace current, proprietary solutions, such as Programmer's Workbench, with this open, industry-standard environment.

We have several objectives with our Eclipse offering. The first is to provide industry-standard IDE assistance for developing Java applications that use ClearPath assets. To this end, Unisys provides industry-standard Resource Adapters (RAs) for ClearPath database and transaction access and the Eclipse IDE offers features that assist with development of these applications. For both ClearPath platforms, there is an RA that supports Distributed Transaction Processing Services (DTP RA). For ClearPath OS 2200 environments, we support access to Business Information Server, DMS and RDMS databases, as well as to TIP/HVTIP transactions. For ClearPath MCP, there are RAs for Enterprise Database Server (DMS II) and COMS.

Our second major objective with Eclipse is to provide the ability to develop other 3GL applications for ClearPath environments using the same framework that can be used for Java and other non-ClearPath hosted applications. For ClearPath OS 2200 environments, we enable development of COBOL, Java, and PLUS applications, as well as the use of TelNet and CMplus. For MCP, we support COBOL 74/85 and ALGOL, as well as the use of WFL. And by moving ClearPath development activities into Eclipse, programming teams will gain the additional benefits of a streamlined build and test process with a common IDE for all projects, regardless of language.

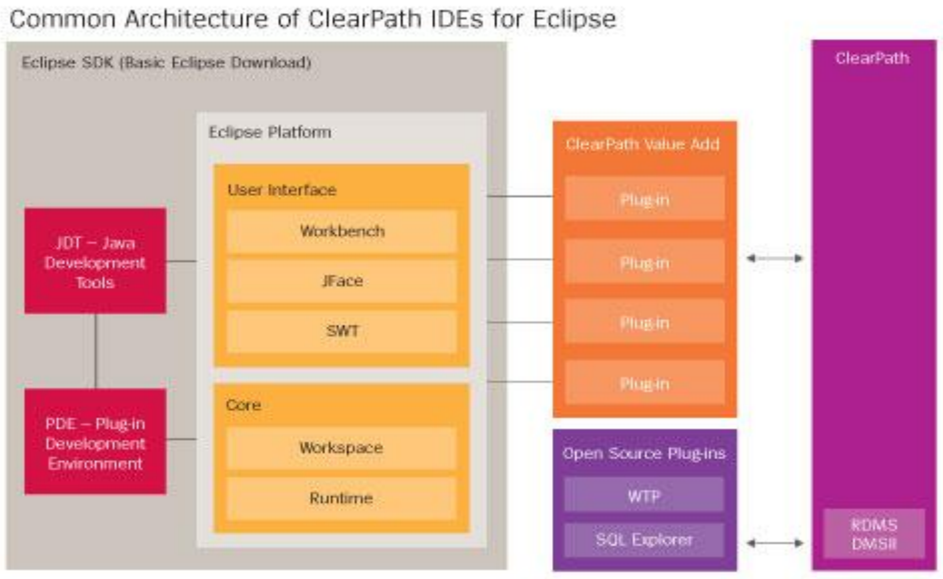


Figure 2.1: Architecture of Eclipse

The Unisys engineering team is fully invested in contributing to the open source movement in ways that benefit our ClearPath customers. We are directing our resources to creating ClearPath specific plug-ins that naturally build on the Eclipse IDE foundation and provide capabilities that make sense for developers on

ClearPath systems. To that end, Unisys has built and released Eclipse plug-ins that enable development for ClearPath OS 2200 and MCP operating environments using the Eclipse IDE, as well as integration of ClearPath based assets in a service-oriented environment. So, what is Unisys providing exactly for Eclipse? An All-in-One package comprised of the following: The basic, currently released Eclipse IDE

- A selection of open-source plug-ins that we see as particularly valuable in a ClearPath development environment, such as the Eclipse Data Tools Project and Web Tools Project.
- Unisys plug-ins, some of which are modified versions of open source plug-ins and others built by Unisys, which enable the development of applications specifically for ClearPath and access to ClearPath assets (databases and transactions).

An Application Development Guide to help you get started Equally important, the components of the Unisys Eclipse All-in-One package are:

- Integrated and tested by Unisys engineering
- Packaged and released as a part of the OS 2200 and MCP IOEs
- Supported via Unisys Support at no additional charge.

For more detailed information about the ClearPath OS 2200 IDE for Eclipse, see our Tech Corner article in this issue of ClearPath Connection. (And, look for a similarly in-depth article about the ClearPath MCP IDE for Eclipse in the next issue of this newsletter).

3.5.6 EASY, INTEGRATED, AND INNOVATIVE

With the Unisys All-in-One packages, it’s never been easier to get started using the Eclipse IDE. Whether you’re an experienced ClearPath programmer or a new college grad just starting out, it’s time to start exploring the technical and business benefits of this powerful toolset. You can take advantage of the many plug-ins available from the open source community and Unisys own plug-ins that are designed to help you get maximum value from your current ClearPath investment.

TOOLS/ENVIRONMENT USED

CHAPTER-4

SYSTEM REQUIREMENT ANALYSIS

3.1 Software requirements Analysis:

A software requirements definition is an abstract description of the services .Which the system should provide and the constraints under which the system must operate. It should only specify only the external behaviour of the systems and is not concerned with system design characteristics .It is the solution, in a natural language plus diagram, of what services the system is expected to provide and the constraints under which it must operate.

3.2 Hardware requirements Analysis:

Hardware requirements analysis is to define and analyse a complete set of functional, operational, performance, interface, quality factors and design, critically and test requirements.

3.3 Software requirements:

- **A**ndroid SDK 4.0 or more
- Eclipse 3.7 or more
- Java 1.6 or more
- Android development tools

3.4 Hardware requirements:

- Processor: Armv7 processor
- Ram: 1GB
- Android Device
- Data flow diagram

ANALYSIS DOCUMENT

CHAPTER-5

ANAYLYSIS DOCUMENT

4.1 Dataflow Diagram:

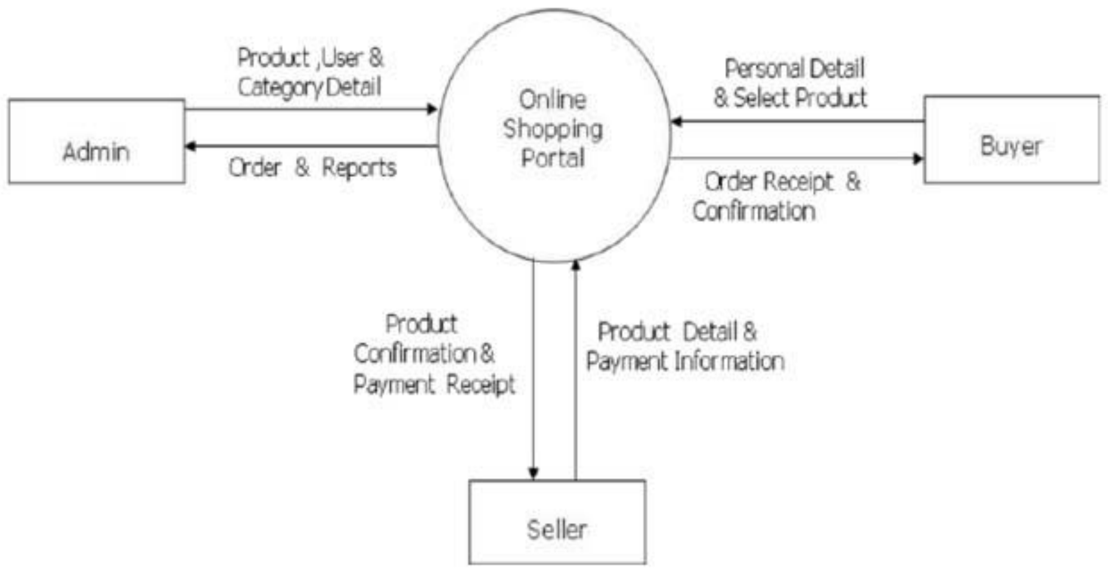

Admin
Product ,User & Category Detail
Order & Reports
Online Shopping Portal
Personal Detail & Select Product
Buyer
Order Receipt & Confirmation
Product Confirmation & Payment Receipt
Product Detail & Payment Information
Seller

Fig 4.1 User and admin architecture

4.2 Level 1 DFD:

a. Admin Dashboard:-

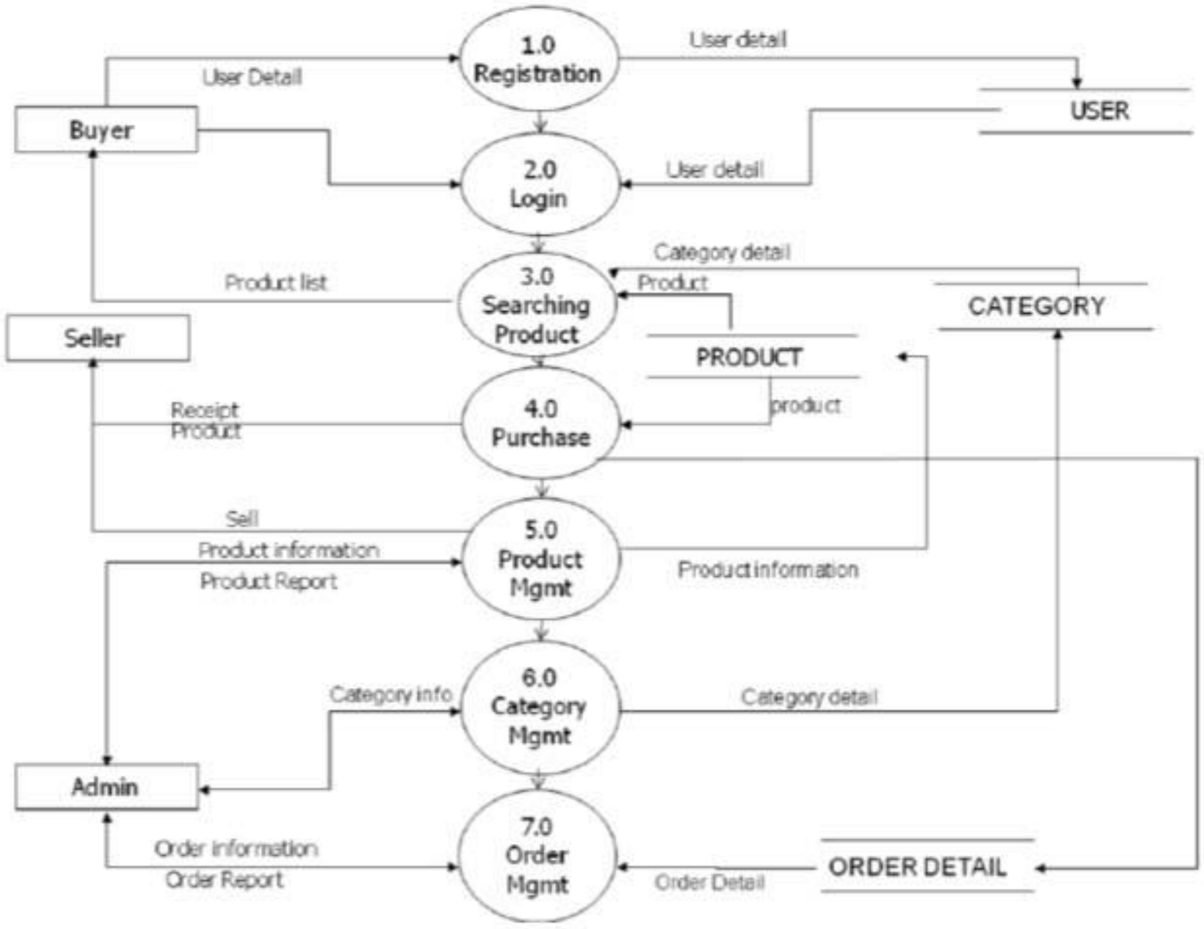


Fig 4.2 Admin architecture

- **Admin users.where admin users can manage the information places in website.**

b. User

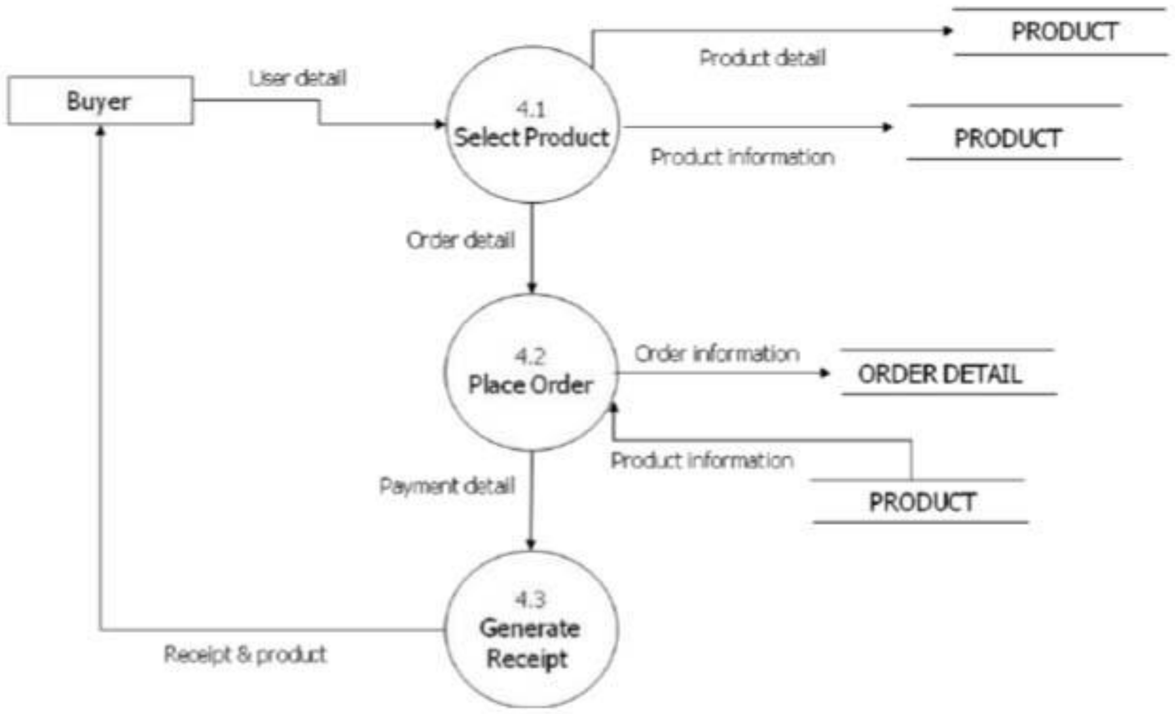


Fig 4.2 User architecture

- **Users are the member of chaloexplore website; here users get the daily notifications of website. Can also view the places.**

DESIGN

CHAPTER-6

DESIGN

- Design Patterns. In software engineering, a design pattern is a general repeatable solution to a commonly occurring problem in software design. A design pattern isn't a finished design that can be transformed directly into code.

5.1 Input design

- Once the analysis of the system has been done, it would be necessary to identify the data that is required to produce outputs.
- Input design features can ensure the reliability of the system and generate correct reports from the accurate data.
- The input design also determines whether the user can interact efficiently with the system. Keeping the process simple since, JAVA has been chosen for this system, the user should easily understand the screens designed.
- The validations are carried out easily and the user will have no difficulty in adding a new entry in the coding page.

5.2 Output design

- Computer output is the most important and direct source of information to the user.
- Efficient, intelligible output design should improve the systems relationship with the user and help in the decision making.
- A major form of output is viewing the data in the destination place.

5.3 Code design

- When large volumes of data are being handled, it is important that the items to be stored, select easily and quickly.
- The purpose of codes is to facilitate the identification and retrieval of item of information. The system analysis will find code structures will not always be the most suitable for efficient computer processing principles of code design. When large volumes of data are being handled, selected easily and quickly. To accomplish this, each data item must have unique identification and must be related to the other items.
- To accomplish each data item must have a unique specification and must be related to other forms or items of data of the same type.

5.4 Procedural design

- The purpose of the process design is, to establish the communication between the data and the code.
- The process takes the data from the database and checks these data with the user-input data or any other data.
- Once the process has been completed, the output of the report will be generated based on the processed results.

SOURCE CODE

CHAPTER-6

SOURCE CODE

Home Activity.java

```
package com.smart.ticketing.agriculture;

import android.content.Intent;

import android.os.Bundle;

import android.support.v7.app.AppCompatActivity;

import android.view.View;

import android.widget.Button;

import com.smart.ticketing.R;

import butterknife.BindView;

import butterknife.ButterKnife;

import butterknife.Unbinder;

public class HomeActivity extends AppCompatActivity {

@BindView(R.id.price)

Button btnTransfer;

@BindView(R.id.prep)

Button btnPreProduction;

@BindView(R.id.pop)

Button btnPostProduction;
```

```
@BindView(R.id.production)

Button btnBalance;

@BindView(R.id.vieworders)

Button btnViewOrders;

@BindView(R.id.viewotherlands)

Button btnViewLands;

@BindView(R.id.viewItems)

Button btnCropsInfo;

private String TAG = "ViewBooksActivity";

private Unbinder unbinderknife;

    @Override

    protected void onCreate(Bundle savedInstanceState) {

        super.onCreate(savedInstanceState);

        setContentView(R.layout.activity_agai_home);

        unbinderknife = ButterKnife.bind(this);

        if (LoginActivity.userType.equalsIgnoreCase("customer")) {

            btnCropsInfo.setVisibility(View.VISIBLE);

//            btnViewLands.setVisibility(View.GONE);

            btnViewOrders.setVisibility(View.GONE);

            btnTransfer.setVisibility(View.GONE);

            btnBlogs.setVisibility(View.GONE);
            btnPreProduction.setVisibility(View.GONE);

            btnPostProduction.setVisibility(View.GONE);

//            btnBalance.setVisibility(View.GONE);

        } else {

            btnCropsInfo.setVisibility(View.GONE);

            btnViewLands.setVisibility(View.GONE);

            btnViewOrders.setVisibility(View.GONE);

            btnTransfer.setVisibility(View.GONE);

            btnBlogs.setVisibility(View.VISIBLE);

            btnBlogs.setText("Blogs");

            btnPreProduction.setVisibility(View.VISIBLE);

            btnPostProduction.setVisibility(View.VISIBLE);

            btnBalance.setVisibility(View.GONE);

            btnBalance.setText("Balance: "+LoginActivity.balance);

            btnTransfer.setText("Transfer");

            btnTransfer.setVisibility(View.GONE);

            btnCropsInfo.setVisibility(View.VISIBLE);

            btnCropsInfo.setText("Crops Info");

        }

        btnCropsInfo.setOnClickListener(new View.OnClickListener() {

            @Override
            public void onClick(View view) {
                Intent intent = new Intent(HomeActivity.this, CropDetailsList.class);
                startActivity(intent);
            }
        });
        btnViewLands.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                Intent intent = new Intent(HomeActivity.this, ViewLandsActivity.class);
                startActivity(intent);
            }
        });
        btnPreProduction.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                Intent intent = new Intent(HomeActivity.this, RegisterCropActivity.class);
                startActivity(intent);
            }
        });
       /* btnBalance.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {

                Intent intent = new Intent(HomeActivity.this, LandPledgeActivity.class);

                startActivity(intent);

            }

        });*/

        btnPostProduction.setOnClickListener(new View.OnClickListener() {

            @Override

            public void onClick(View view) {

                Intent intent = new Intent(HomeActivity.this, PublishCropsActivity.class);

                startActivity(intent);

            }

        });

        btnBlogs.setOnClickListener(new View.OnClickListener() {

            @Override

            public void onClick(View view) {

                Intent intent = new Intent(HomeActivity.this, ViewBlogsActivity.class);

                startActivity(intent);

            }

        });

        btnTransfer.setOnClickListener(new View.OnClickListener() {

            @Override
```

```
            public void onClick(View view) {
                Intent intent = new Intent(HomeActivity.this, TransferActivity.class);
                startActivity(intent);
            }
        });
        btnViewOrders.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                Intent intent = new Intent(HomeActivity.this, ViewOrdersActivity.class);
                startActivity(intent);
            }
        });
    }
    @Override
    public void onResume(){
        super.onResume();
        btnBalance.setText("Balance: "+LoginActivity.balance);
    }
    @Override
    public void onDestroy() {
        super.onDestroy();
        unbinderknife.unbind();

    }

}
```

Login Activity.java

```
package com.smart.ticketing.agriculture;

import android.content.Intent;

import android.os.Bundle;

import android.support.v7.app.AppCompatActivity;

import android.util.Log;

import android.view.View;

import android.widget.Button;

import android.widget.EditText;

import com.backendless.Backendless;

import com.backendless.IDataStore;

import com.backendless.async.callback.AsyncCallback;

import com.backendless.exceptions.BackendlessFault;

import com.backendless.persistence.DataQueryBuilder;

import com.smart.ticketing.R;

import com.smart.ticketing.Utils;

import com.smart.ticketing.agriculture.data.Users;

import com.smart.ticketing.backendless.SplashActivity;
import java.util.List;

import butterknife.BindView;

import butterknife.ButterKnife;

import butterknife.Unbinder;

public class LoginActivity extends AppCompatActivity {

    @BindView(R.id.editText)

    EditText etUsernmae;

    @BindView(R.id.editText2)

    EditText etPassword;

    @BindView(R.id.button1)

    Button btnLogin;

    @BindView(R.id.button2)

    Button btnRegister;

    private Unbinder unbinderknife;

    public static String name = "", username, password, phone, address, industry, userType = "";

    public static Users user;

    public static int balance = 100;

    private String TAG = "LoginActivity";

    @Override

    protected void onCreate(Bundle savedInstanceState) {

        super.onCreate(savedInstanceState);
```

```
setContentView(R.layout.activity_home);

unbinderknife = ButterKnife.bind(this);

btnRegister.setOnClickListener(new View.OnClickListener() {

   @Override

   public void onClick(View v) {

       Intent intent = new Intent(LoginActivity.this, RegisterActivity.class);

       startActivity(intent);

   }

});

btnForgotpassword.setOnClickListener(new View.OnClickListener() {

   @Override

   public void onClick(View v) {

Intent intent= newIntent(LoginActivity.this,com.smart.ticketing.agriculture.ForgotPasswordActivity.class);

       startActivity(intent);

   }

});

btnForgotpassword.setVisibility(View.GONE);

btnLogin.setOnClickListener(new View.OnClickListener() {

   @Override

   public void onClick(View v) {

       String username = etUsernmae.getText().toString();
```

```
            String password = etPassword.getText().toString();
//          username = "daya@gmail.com";
//          password = "1234";
            if (username.equals("")) {
                Utils.showToast(getApplicationContext(), "Enter username");
            } else if (password.equals("")) {
                Utils.showToast(getApplicationContext(), "Enter password");
            } else {
                IDataStore<Users> querydata = Backendless.Data.of(Users.class);
                DataQueryBuilder query = DataQueryBuilder.create();
                query.setWhereClause("username='" + username + "' and pass='" + password + "' and
userType='farmer'");
                querydata.find(query, new AsyncCallback<List<Users>>() {
                    @Override
                    public void handleResponse(List<Users> users) {
                        if (users.size() > 0) {
                            Utils.showToast(LoginActivity.this, "Login successfull");
                            Intent intent = new Intent(LoginActivity.this, SplashActivity.sClass);
                            startActivity(intent);
                            name = users.get(0).getName();
                            LoginActivity.username = users.get(0).getUsername();
                            LoginActivity.password = users.get(0).getPassword();
                            phone = users.get(0).getPhone();

                            address = users.get(0).getAddress();

                            industry = users.get(0).getIndustry();

                            userType = users.get(0).getUserType();

                            balance = users.get(0).getBalance();

                            user = users.get(0);

                        } else {

                            Utils.showToast(getApplicationContext(), "Invalid credentials");

                        }

                    }

                    @Override

                    public void handleFault(BackendlessFault fault) {

                        Log.i(TAG, "Login error: "+fault.getMessage());

                        Utils.showToast(getApplicationContext(), "Login error: " + fault);

                    }

                });

            }

        }

      });

    }

    @Override
```

```
    public void onDestroy() {

        super.onDestroy();

        unbinderknife.unbind();

    }

}
```

Registration Activity .java

```
package com.smart.ticketing.agriculture;

import android.os.Bundle;

import android.support.v7.app.AppCompatActivity;

import android.util.Log;

import android.view.View;

import android.widget.AdapterView;

import android.widget.ArrayAdapter;

import android.widget.Button;

import android.widget.EditText;

import android.widget.Spinner;

import com.backendless.Backendless;

import com.backendless.IDataStore;

import com.backendless.async.callback.AsyncCallback;

import com.backendless.exceptions.BackendlessFault;

import com.backendless.persistence.DataQueryBuilder;
import com.smart.ticketing.R;

import com.smart.ticketing.Utils;

import com.smart.ticketing.common.TexttoSpeechHandler;

import com.smart.ticketing.qless.RegisterCrop;

import java.util.ArrayList;

import java.util.List;

import butterknife.BindView;

import butterknife.ButterKnife;

import butterknife.Unbinder;

public class RegisterCropActivity extends AppCompatActivity {

@BindView(R.id.editText)

EditText etName;

@BindView(R.id.editText2)

EditText etTotalArea;

@BindView(R.id.button1)

Button btnLogin;

@BindView(R.id.spin)

Spinner spinner;

private Unbinder unbinderknife;

public static String name = "", username, password, phone, address;

TexttoSpeechHandler tts ;
```

```
@Override

protected void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

setContentView(R.layout.activity_registercrop);

unbinderknife = ButterKnife.bind(this);

tts = new TexttoSpeechHandler(RegisterCropActivity.this);

etName.setVisibility(View.GONE);

fetchData();

btnLogin.setOnClickListener(new View.OnClickListener() {

@Override

public void onClick(View v) {

String name = etName.getText().toString();

final String area = etTotalArea.getText().toString();

if (selectedCropName.equals("")) {

Utils.showToast(getApplicationContext(), "Enter cropname");

} else if (area.equals("")) {

Utils.showToast(getApplicationContext(), "Enter area");

} else {

IDataStore<RegisterCrop> querydata = Backendless.Data.of(RegisterCrop.class);

DataQueryBuilder query = DataQueryBuilder.create();

query.setWhereClause("name='" + selectedCropName + "'");
```

```
querydata.find(query, new AsyncCallback<List<RegisterCrop>>() {

@Override

public void handleResponse(List<RegisterCrop> response) {

if (response.size() > 0) {

RegisterCrop crop = response.get(0);

int totalArea = crop.getTotalArea() + Integer.parseInt(area);

if (totalArea > crop.getMaximumArea()) {

tts.speak("Threshhold reached, can't guaratee amount. Please try another crop");

Utils.showToast(getApplicationContext(), "Threshhold reached, can't guaratee    amount. Please try another
crop");

} else {

crop.setTotalArea(crop.getTotalArea() + Integer.parseInt(area));

Backendless.Data.save(crop, new AsyncCallback<RegisterCrop>() {

@Override

public void handleResponse(RegisterCrop response) {

Utils.showToast(getApplicationContext(), "Registration successfull");

}

@Override

public void handleFault(BackendlessFault fault) {

Log.i("Registercrop", "fault: " + fault.getMessage());

}

});
}
} else {
Utils.showToast(getApplicationContext(), "Crop not found");
}
}
@Override
public void handleFault(BackendlessFault fault) {
Log.i("Registercrop", "fault: " + fault.getMessage());
}
});
}
}
});
spinner.setOnItemSelectedListener(new AdapterView.OnItemSelectedListener() {
@Override
public void onItemSelected(AdapterView<?> adapterView, View view, int i, long l) {
selectedCropName = nameList.get(i);
}
@Override
public void onNothingSelected(AdapterView<?> adapterView) {
}
```

```
});
}
@Override
public void onDestroy() {
super.onDestroy();
unbinderknife.unbind();
if(tts != null){
tts.stop();
}
}
List<String> nameList = new ArrayList<>();
String selectedCropName = "";
public void fetchData() {
IDataStore<RegisterCrop> querydata = Backendless.Data.of(RegisterCrop.class);
DataQueryBuilder query = DataQueryBuilder.create();
query.setPageSize(100);
querydata.find(query, new AsyncCallback<List<RegisterCrop>>() {
@Override
public void handleResponse(List<RegisterCrop> response) {
if (response.size() > 0) {
nameList.clear();
for (RegisterCrop crop : response) {

nameList.add(crop.getName());

}

ArrayAdapter<String>aa=newArrayAdapter<String>(RegisterCropActivity.this,android.R.layout.simple_dropd
own_item_1line, nameList);

spinner.setAdapter(aa);

}

}

@Override

public void handleFault(BackendlessFault fault) {

Log.i("Registercrop", fault.toString());

}

});

}

}
```

Registration Crop Activity.Java

```
package com.smart.ticketing.agriculture;

import android.os.Bundle;

import android.support.v7.app.AppCompatActivity;

import android.util.Log;

import android.view.View;
import android.widget.AdapterView;

import android.widget.ArrayAdapter;

import android.widget.Button;

import android.widget.EditText;

import android.widget.Spinner;

import com.backendless.Backendless;

import com.backendless.IDataStore;

import com.backendless.async.callback.AsyncCallback;

import com.backendless.exceptions.BackendlessFault;

import com.backendless.persistence.DataQueryBuilder;

import com.smart.ticketing.R;

import com.smart.ticketing.Utils;

import com.smart.ticketing.common.TexttoSpeechHandler;

import com.smart.ticketing.qless.RegisterCrop;

import java.util.ArrayList;

import java.util.List;

import butterknife.BindView;

import butterknife.ButterKnife;

import butterknife.Unbinder;

public class RegisterCropActivity extends AppCompatActivity {

    @BindView(R.id.editText)
    EditText etName;

    @BindView(R.id.editText2)

    EditText etTotalArea;

    @BindView(R.id.button1)

    Button btnLogin;

    @BindView(R.id.spin)

    Spinner spinner;

    private Unbinder unbinderknife;

    public static String name = "", username, password, phone, address;

    TexttoSpeechHandler tts ;

    @Override

    protected void onCreate(Bundle savedInstanceState) {

        super.onCreate(savedInstanceState);

        setContentView(R.layout.activity_registercrop);

        unbinderknife = ButterKnife.bind(this);

        tts = new TexttoSpeechHandler(RegisterCropActivity.this);

        etName.setVisibility(View.GONE);

        fetchData();

        btnLogin.setOnClickListener(new View.OnClickListener() {

            @Override

            public void onClick(View v) {
```

```
        String name = etName.getText().toString();
        final String area = etTotalArea.getText().toString();
        if (selectedCropName.equals("")) {
            Utils.showToast(getApplicationContext(), "Enter cropname");
        } else if (area.equals("")) {
            Utils.showToast(getApplicationContext(), "Enter area");
        } else {
            IDataStore<RegisterCrop> querydata = Backendless.Data.of(RegisterCrop.class);
            DataQueryBuilder query = DataQueryBuilder.create();
            query.setWhereClause("name='" + selectedCropName + "'");
            querydata.find(query, new AsyncCallback<List<RegisterCrop>>() {
                @Override
                public void handleResponse(List<RegisterCrop> response) {
                    if (response.size() > 0) {
                        RegisterCrop crop = response.get(0);
                        int totalArea = crop.getTotalArea() + Integer.parseInt(area);
                        if (totalArea > crop.getMaximumArea()) {
                            tts.speak("Threshhold reached, can't guaratee amount. Please try another crop");
                            Utils.showToast(getApplicationContext(), "Threshhold reached, can't guaratee amount.
Please try another crop");
                        } else {
                            crop.setTotalArea(crop.getTotalArea() + Integer.parseInt(area));
                            Backendless.Data.save(crop, new AsyncCallback<RegisterCrop>() {

                                @Override

                                public void handleResponse(RegisterCrop response) {

                                    Utils.showToast(getApplicationContext(), "Registration successfull");

                                }

                                @Override

                                public void handleFault(BackendlessFault fault) {

                                    Log.i("Registercrop", "fault: " + fault.getMessage());

                                }

                            });

                        }

                    } else {

                        Utils.showToast(getApplicationContext(), "Crop not found");

                    }

                }

                @Override

                public void handleFault(BackendlessFault fault) {

                    Log.i("Registercrop", "fault: " + fault.getMessage());

                }

            });

        }
```

```
            }
        });
        spinner.setOnItemSelectedListener(new AdapterView.OnItemSelectedListener() {
            @Override
            public void onItemSelected(AdapterView<?> adapterView, View view, int i, long l) {
                selectedCropName = nameList.get(i);
            }
            @Override
            public void onNothingSelected(AdapterView<?> adapterView) {
            }
        });
    }
    @Override
    public void onDestroy() {
        super.onDestroy();
        unbinderknife.unbind();
        if(tts != null){
            tts.stop();
        }
    }
    List<String> nameList = new ArrayList<>();
    String selectedCropName = "";

    public void fetchData() {

        IDataStore<RegisterCrop> querydata = Backendless.Data.of(RegisterCrop.class);

        DataQueryBuilder query = DataQueryBuilder.create();

        query.setPageSize(100);

        querydata.find(query, new AsyncCallback<List<RegisterCrop>>() {

            @Override

            public void handleResponse(List<RegisterCrop> response) {

                if (response.size() > 0) {

                    nameList.clear();

                    for (RegisterCrop crop : response) {

                        nameList.add(crop.getName());

                    }

                    ArrayAdapter<String> aa = new ArrayAdapter<String>(RegisterCropActivity.this,
android.R.layout.simple_dropdown_item_1line, nameList);

                    spinner.setAdapter(aa);

                }

            }

            @Override

            public void handleFault(BackendlessFault fault) {

                Log.i("Registercrop", fault.toString());

            }
        });

    }

}
```

Blogs Details Activity.java

```
package com.smart.ticketing.agriculture;
import android.content.Intent;
import android.net.Uri;
import android.os.Bundle;
import android.support.v7.app.AppCompatActivity;
import android.view.View;
import android.widget.Button;
import android.widget.TextView;
import com.backendless.Backendless;
import com.backendless.async.callback.AsyncCallback;
import com.backendless.exceptions.BackendlessFault;
import com.smart.ticketing.R;
import com.smart.ticketing.Utils;
import com.smart.ticketing.agriculture.data.Blogs;
import java.util.ArrayList;
import java.util.List;

public class BlogsDetailsActivity extends AppCompatActivity {

    TextView tvTopic, tvDesc, tvAuthor, tvLikeCount, tvDislikeCount, tvLink;
    Button btnLike, btnDislike, btnNewBlog;

    private String TAG = "ViewBooksActivity";

    List<Blogs> blogsList = new ArrayList<>();
```

```
List<String> nameList = new ArrayList<>();

Blogs blog;

@Override

protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_agri_blogs_view);
    blog = (Blogs) getIntent().getSerializableExtra("blog");
    tvTopic = findViewById(R.id.topic);
    tvDesc = findViewById(R.id.description);
    tvAuthor = findViewById(R.id.author);
    btnLike = findViewById(R.id.like);
    btnDislike = findViewById(R.id.dislike);
    btnNewBlog = findViewById(R.id.addnewblog);
    tvLikeCount = findViewById(R.id.likecount);
    tvLink = findViewById(R.id.link);
    tvDislikeCount = findViewById(R.id.dislikecount);
    tvTopic.setText(blog.getTopic());
    tvDesc.setText(blog.getDescription());
    tvDislikeCount.setText(blog.getDislikeCount() + "");
    tvLikeCount.setText(blog.getLikeCount() + "");
    tvAuthor.setText(blog.getUsername());
    if (blog.getLink() == null || blog.getLink().equals("")) {
        tvLink.setVisibility(View.GONE);
    } else {
        tvLink.setText(blog.getLink());
    }

    tvLink.setOnClickListener(new View.OnClickListener() {
```

```
    @Override

    public void onClick(View view) {
        Uri uri = Uri.parse(tvLink.getText().toString());
        uri = Uri.parse( "vnd.youtube:" + uri.getQueryParameter( "v" ) );
        startActivity( new Intent( Intent.ACTION_VIEW, uri ) );

    }
});

btnLike.setOnClickListener(new View.OnClickListener() {

    @Override

    public void onClick(View view) {
        btnLike.setEnabled(false);
        int like = blog.getLikeCount() + 1;
        blog.setLikeCount(like);
        tvLikeCount.setText(like + "");
        Backendless.Persistence.save(blog, new AsyncCallback<Blogs>() {

            @Override

            public void handleResponse(Blogs response) {
                Utils.showToast(getApplicationContext(), "like updated");
            }

            @Override

            public void handleFault(BackendlessFault fault) {
                Utils.showToast(getApplicationContext(), "like update error : " + fault.getMessage());
            }
        });
    }
});

btnDislike.setOnClickListener(new View.OnClickListener() {

    @Override

    public void onClick(View view) {
        btnDislike.setEnabled(false);
        int like = blog.getDislikeCount() + 1;
        blog.setDislikeCount(like);
        tvDislikeCount.setText(like + "");
        Backendless.Persistence.save(blog, new AsyncCallback<Blogs>() {



            @Override

            public void handleResponse(Blogs response) {
                Utils.showToast(getApplicationContext(), "Dislike updated");
            }

            @Override

            public void handleFault(BackendlessFault fault) {
                Utils.showToast(getApplicationContext(), "dislike update error : " + fault.getMessage());
            }
        });
    }
```

```
        });

        btnNewBlog.setOnClickListener(new View.OnClickListener() {

            @Override

            public void onClick(View view) {
                Intent intent = new Intent(BlogsDetailsActivity.this, AddBlogActivity.class);
                startActivity(intent);
                finish();
            }
        });
    }
}
```

Add Blogs Details.java

```
package com.smart.ticketing.agriculture;
import android.os.Bundle;
import android.support.v7.app.AppCompatActivity;
import android.view.View;
import android.widget.Button;
import android.widget.EditText;
import com.backendless.Backendless;
import com.backendless.async.callback.AsyncCallback;
import com.backendless.exceptions.BackendlessFault;
import com.smart.ticketing.R;
import com.smart.ticketing.Utils;
import com.smart.ticketing.agriculture.data.Blogs;
import java.util.ArrayList;
import java.util.List;
```

```
public class AddBlogActivity extends AppCompatActivity {
    EditText etTopic, etDesc, etLinks;
    Button btnSubmit;
    private String TAG = "ViewBooksActivity";
    List<Blogs> blogsList = new ArrayList<>();
    List<String> nameList = new ArrayList<>();

    @Override

    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_agri_blogs_add);
        etTopic = (EditText) findViewById(R.id.topic);
        etDesc = (EditText) findViewById(R.id.desc);
        etLinks = (EditText) findViewById(R.id.links);
        btnSubmit = (Button) findViewById(R.id.submit);
        btnSubmit.setOnClickListener(new View.OnClickListener() {

            @Override

            public void onClick(View view) {
                String name = etTopic.getText().toString();
                String desc = etDesc.getText().toString();
                String links = etLinks.getText().toString();

                if (name.equals("")) {
                    Utils.showToast(getApplicationContext(), "Enter topic");
                    return;
                }

                if (desc.equals("")) {
```

```
                Utils.showToast(getApplicationContext(), "Enter description");
                return;
            }

            Blogs blog = new Blogs();
            blog.setTopic(name);
            blog.setDescription(desc);
            blog.setUsername(LoginActivity.username);
            blog.setName(LoginActivity.name);
            blog.setLikeCount(0);
            blog.setDislikeCount(0);
            blog.setLink(links == null ? "" : links);

            Backendless.Persistence.save(blog, new AsyncCallback<Blogs>() {
                @Override
                public void handleResponse(Blogs response) {
                    Utils.showToast(getApplicationContext(), "Blogs posted successfully");
                    finish();
                }

                @Override
                public void handleFault(BackendlessFault fault) {
                    Utils.showToast(getApplicationContext(), "Blogs post error");
                }
            });

        }
    });


}
```

TESTING

CHAPTER-7

TESTING

- The process or method of finding error/s in a software application or program so that the application functions according to the end user's requirement is called software testing.

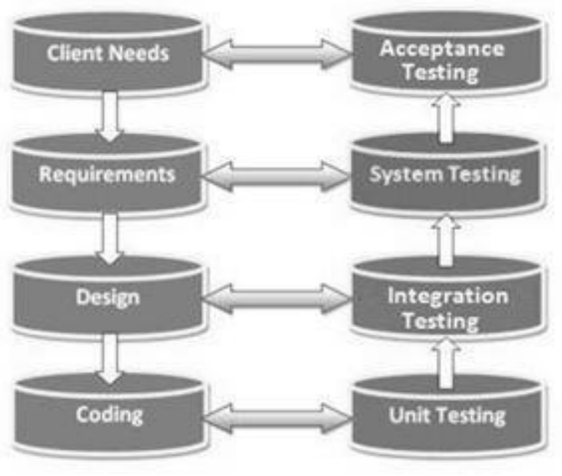


Fig 7.1 Levels of testing

7.1 Unit testing

Unit testing involves the design of the test case that validates the internal program logic is functioning properly, and that program input procedures 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 the knowledge of its construction and is invasive. Unit test ensure that each unique path of a business process performs accurately to the documented specifications and contains clearly defined inputs and expected results.

Unit testing is a level of software testing where individual units/ components of a software are tested. The purpose is to validate that each unit of the software performs as designed. A unit is the smallest testable part of software. It usually has one or a few inputs and usually a single output. In procedural programming a unit may be an individual program, function, procedure, etc. In object-oriented programming, the smallest unit is a method, which may belong to a base/ super class, abstract class or derived/ child class. (Some treat a module of an application as a unit. This is to be discouraged as there will probably be many individual units within that module.) Unit testing frameworks, drivers, stubs, and mock/ fake objects are used to assist in unit testing.

7.2 Integration testing

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.

Integration testing is a software testing methodology used to test individual software components or units of code to verify interaction between various software components and detect interface defects. Components are tested as a single group or organized in an iterative manner. After the integration testing has been performed on the components, they are readily available for system testing.

Integration is a key software development life cycle (SDLC) strategy. Generally, small software systems are integrated and tested in a single phase, whereas larger systems involve several integration phases to build a complete system, such as integrating modules into low-level subsystems for integration with larger subsystems.

Integration testing encompasses all aspects of a software system's performance, functionality and reliability. Most unit-tested software systems are comprised of integrated components that are tested for error isolation due to grouping. Module details are presumed accurate, but prior to integration testing, each module is separately tested via partial component implementation, also known as a stub.

7.3 System testing

System testing ensures that the entire integrated software system meets requirements. It tests a configuration to ensure known and predictable results. An example of system testing is the configuration oriented system integration test. System testing is based on process description and flows, emphasizing pre-driven links and integration points. System Testing is the testing of a complete and fully integrated software product.

Usually software is only one element of a larger computer based system. Ultimately, software is interfaced with other software/hardware systems.
System testing of software or hardware is testing conducted on a complete, integrated system to evaluate the system's compliance with its specified requirements.

7.4 Acceptance testing.

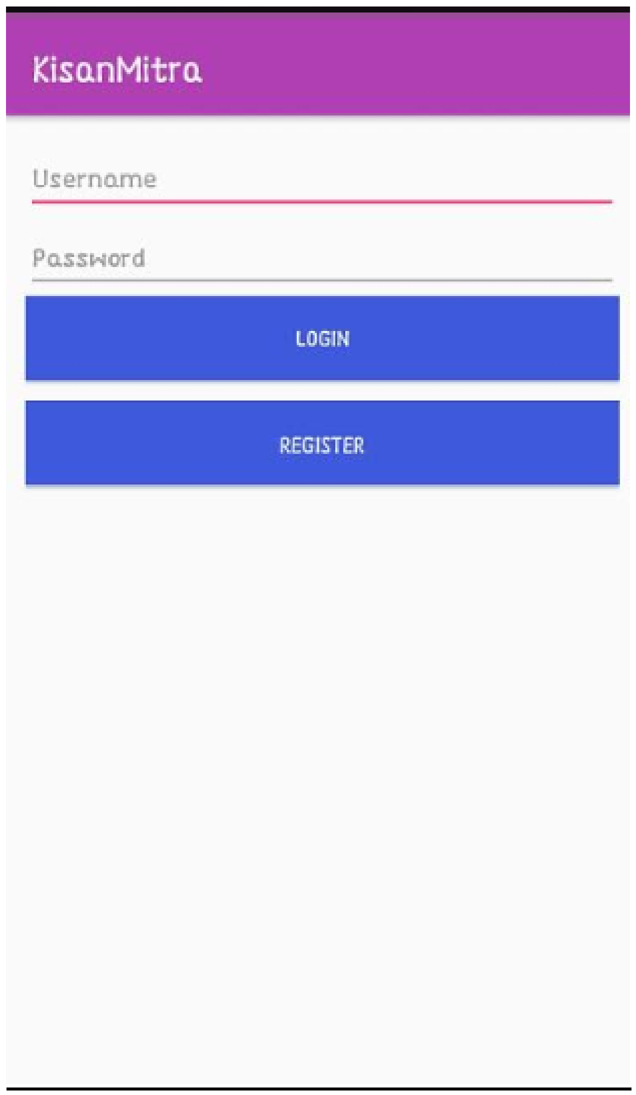
Acceptance testing is the most important phase of testing as this decides whether the client approves the application/software or not. It may involve functionality, usability, performance, and U.I of the application. It is also known as user acceptance testing (UAT), operational acceptance testing (OAT), and end-user testing. It is one of the final stages of the software’s testing cycle and often occurs before a client or customer accepts the new application. Acceptance tests are black-box system tests. Users of the system perform tests in line with what would occur in real-time scenarios and verify whether or not the software/application meets all specifications

SCREEN SHOTS

CHAPTER-8

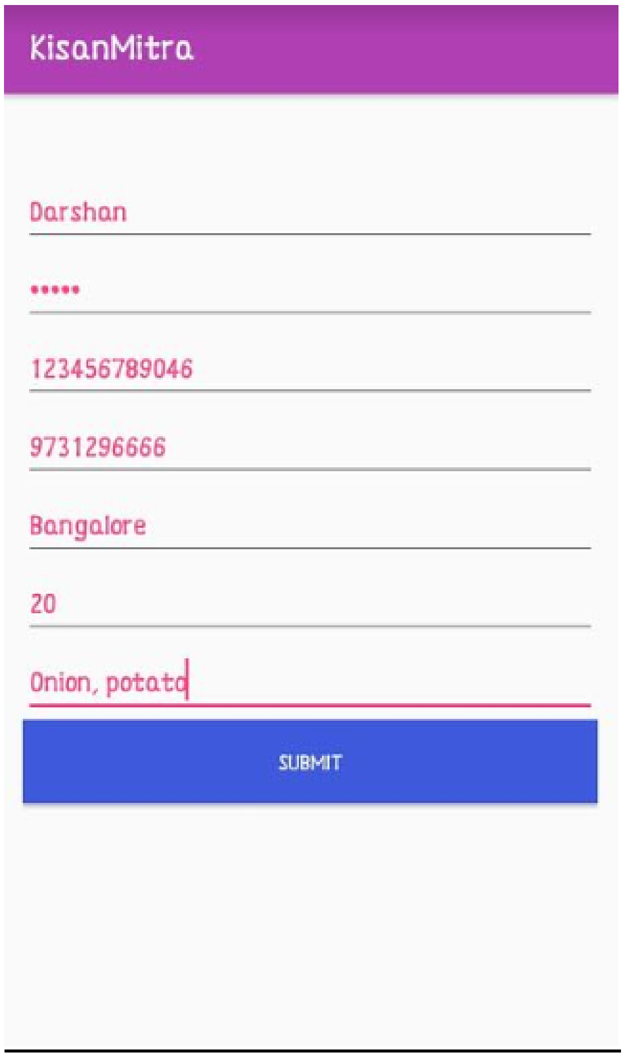
SCREENSHOTS

8.1 Signup:



- **Where User going to login for the app.**

8.2 Register:



KisanMitra

Darshan

•••••

123456789046

9731296666

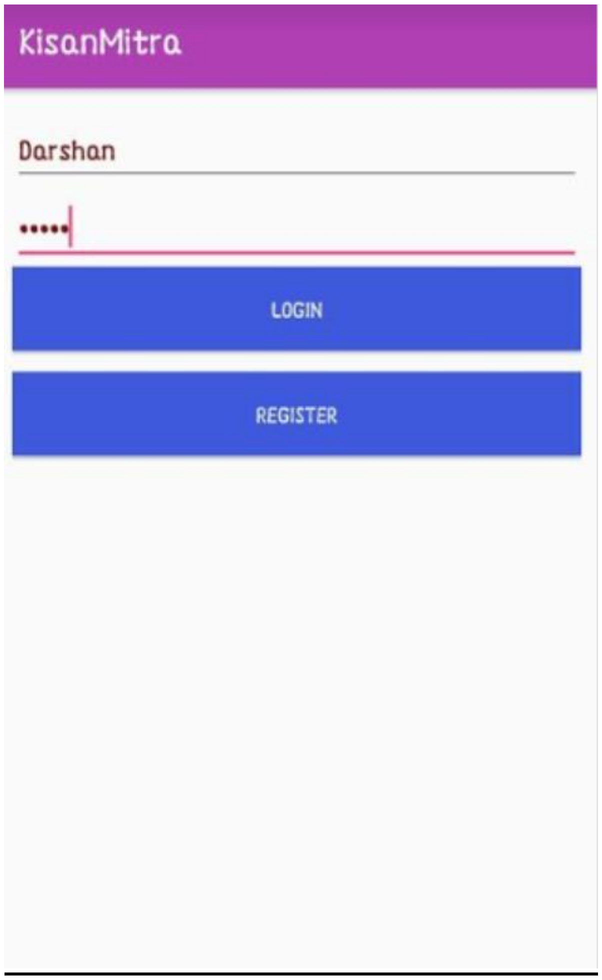
Bangalore

20

Onion, potato

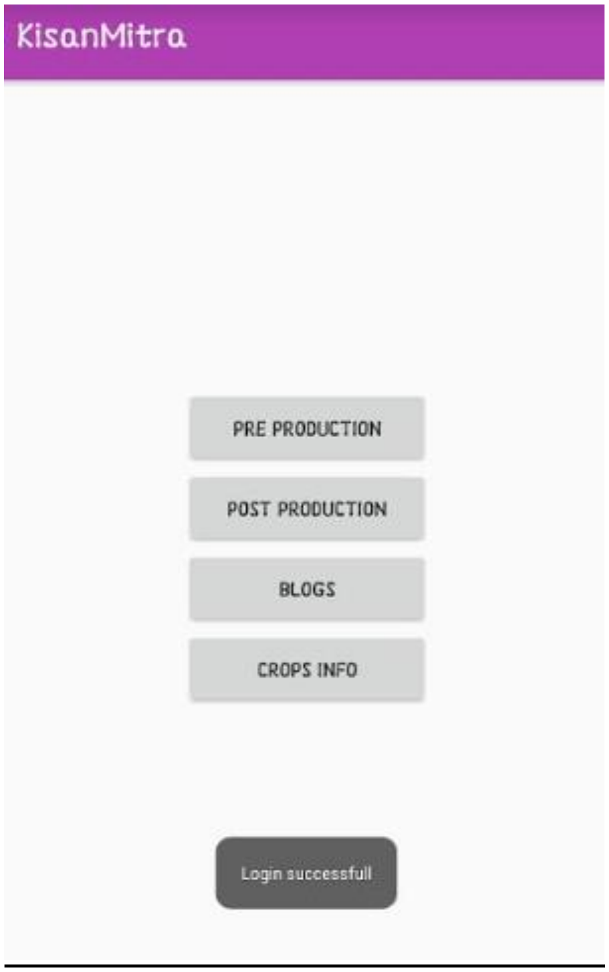
- **Where User must enroll their details.**

<u>8.3 Login:</u>



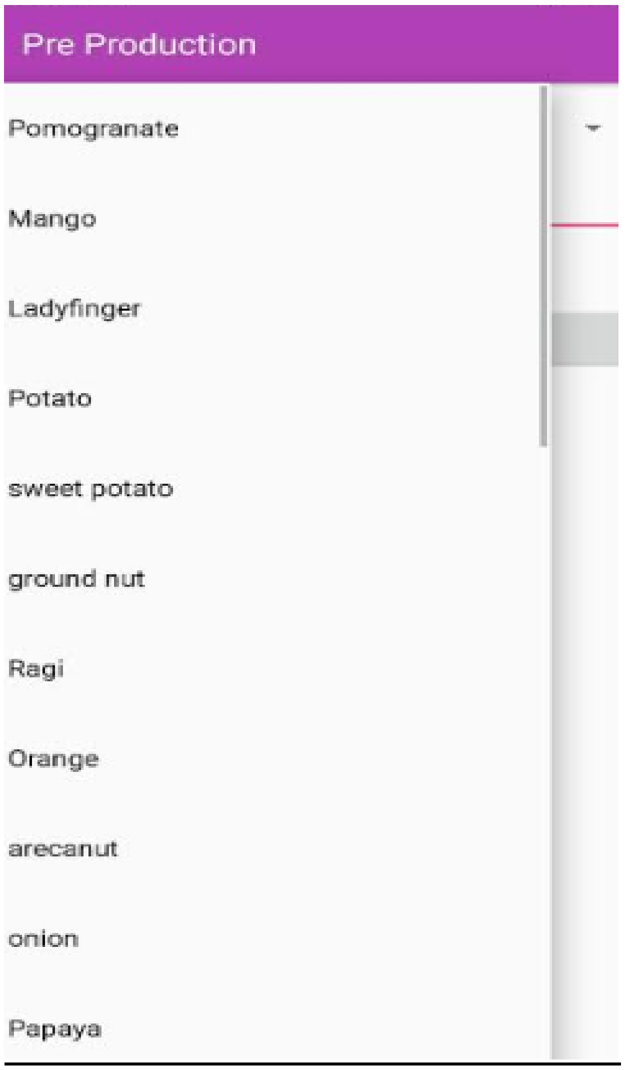
- **After registration compeletion the user login's with <u>username</u> and <u>password</u>.**

8.4 Home Activity:



- **After login is successful the user will be provided to perform different activity.**

8.5 Pre Production Page activity:



Pre Production

Pomogranate

Mango

Ladyfinger

Potato

sweet potato

ground nut

Ragi

Orange

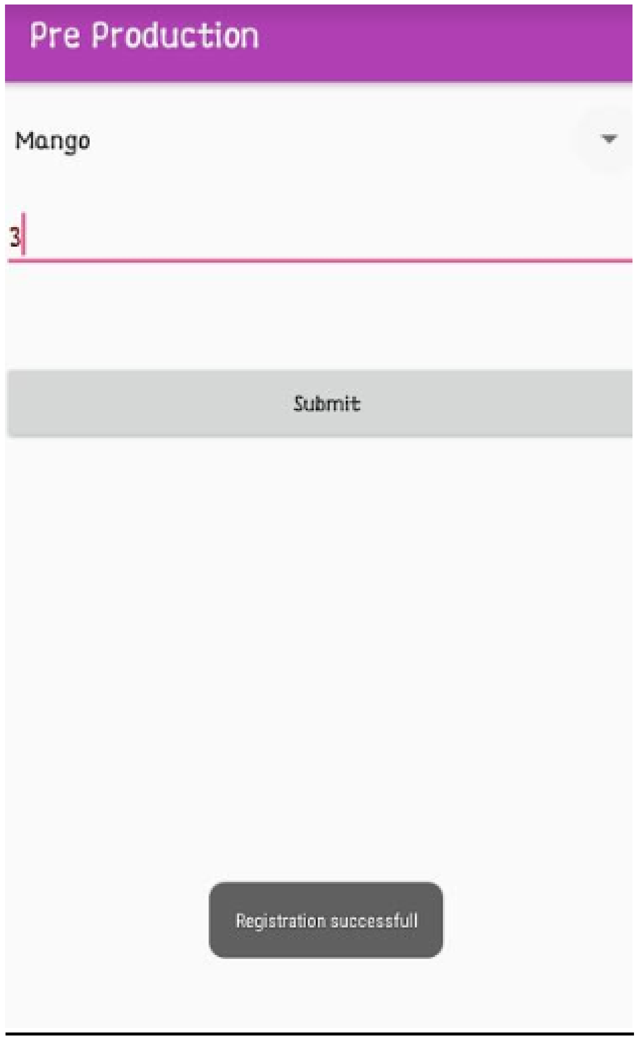
arecanut

onion

Papaya

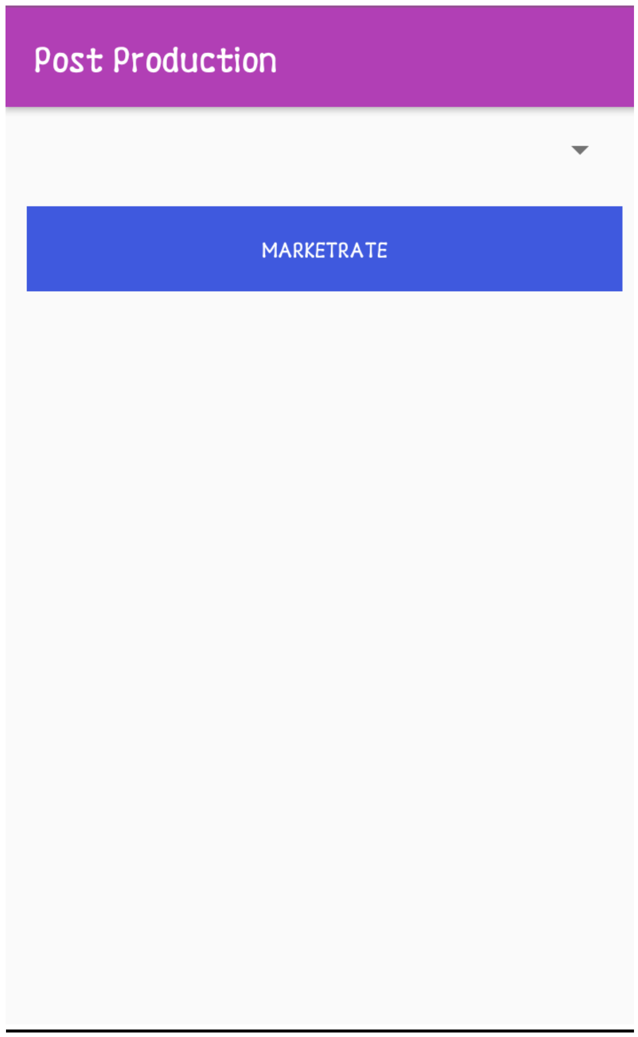
- **The user will select some desired items. In the pre-production activity.**

8.6 Pre Production item selection:



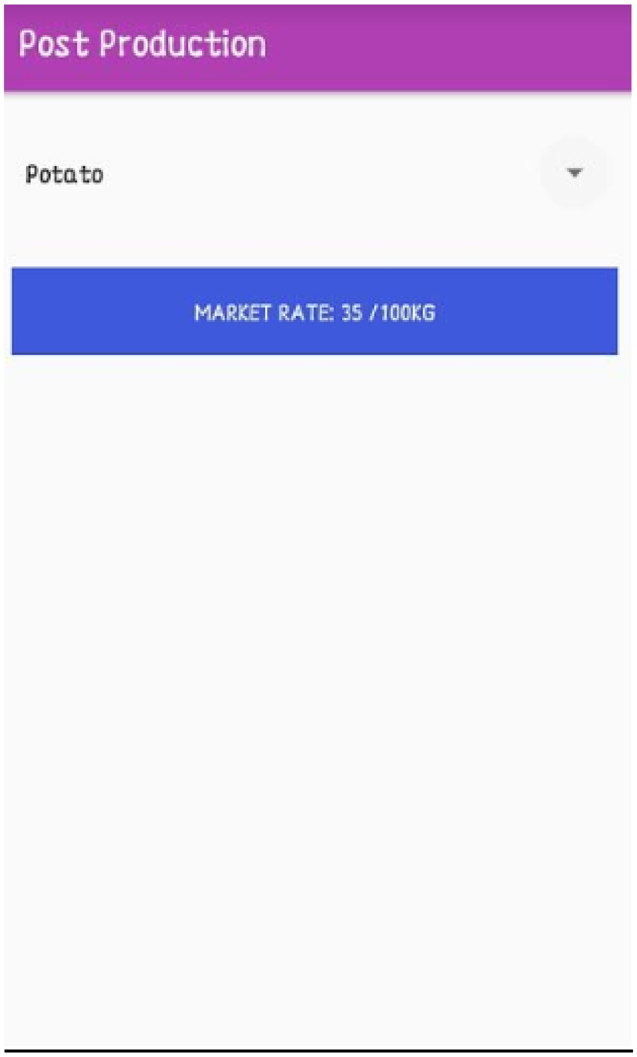
- **The user will enter the total area to harvest the crops.**

8.7 Post Production Page activity:



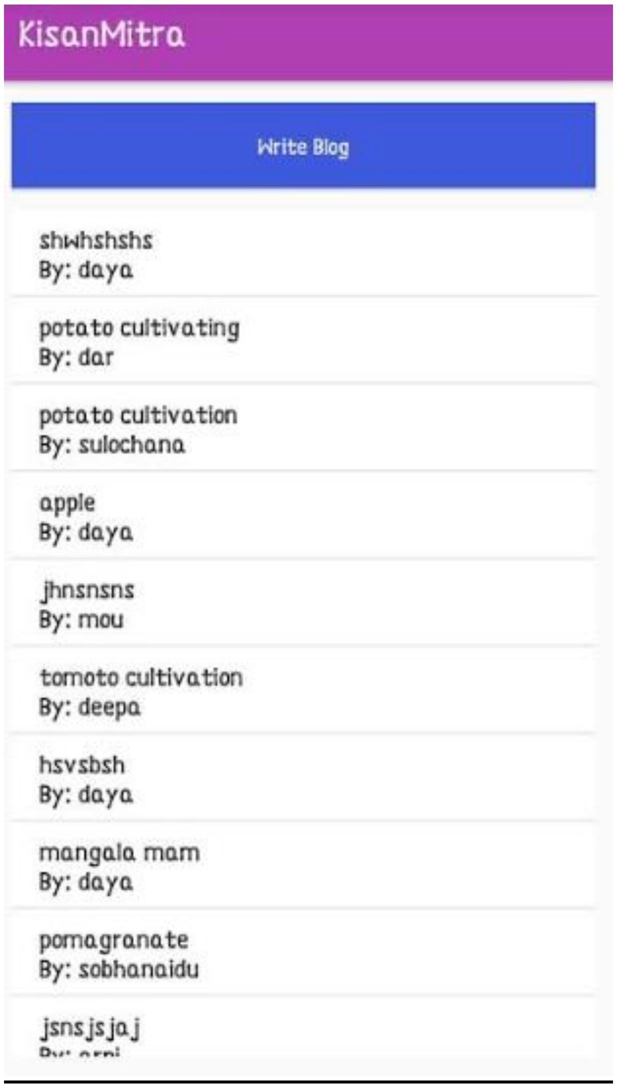
- **The user will enter to pre-production activity.**

<u>8.8 Post Production items market rate:</u>



- **User selects the desired items and he will get the market rate.**

8.9 Blog Activity:



shwhshshs
By: daya

potato cultivating
By: dar

potato cultivation
By: sulochana

apple
By: daya

jhnsnsns
By: mou

tomoto cultivation
By: deepa

hsvsbsh
By: daya

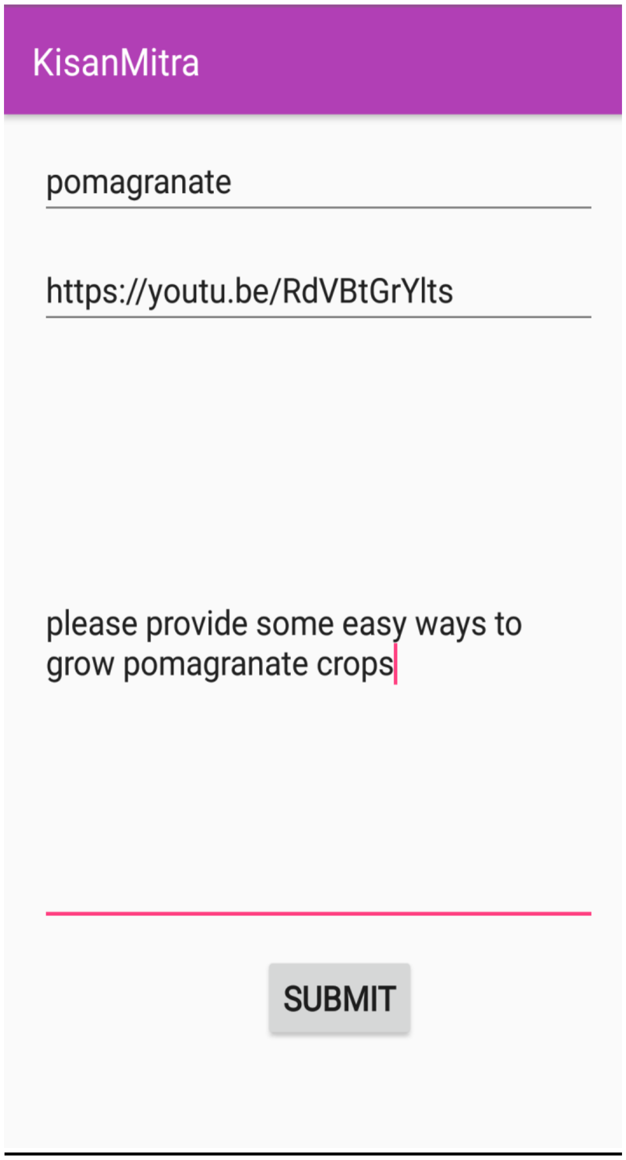
mangala mam
By: daya

pomagranate
By: sobhanaidu

jsnsjsjaj

- **User enters the blog activity.**

8.10 Blog Activity 1:



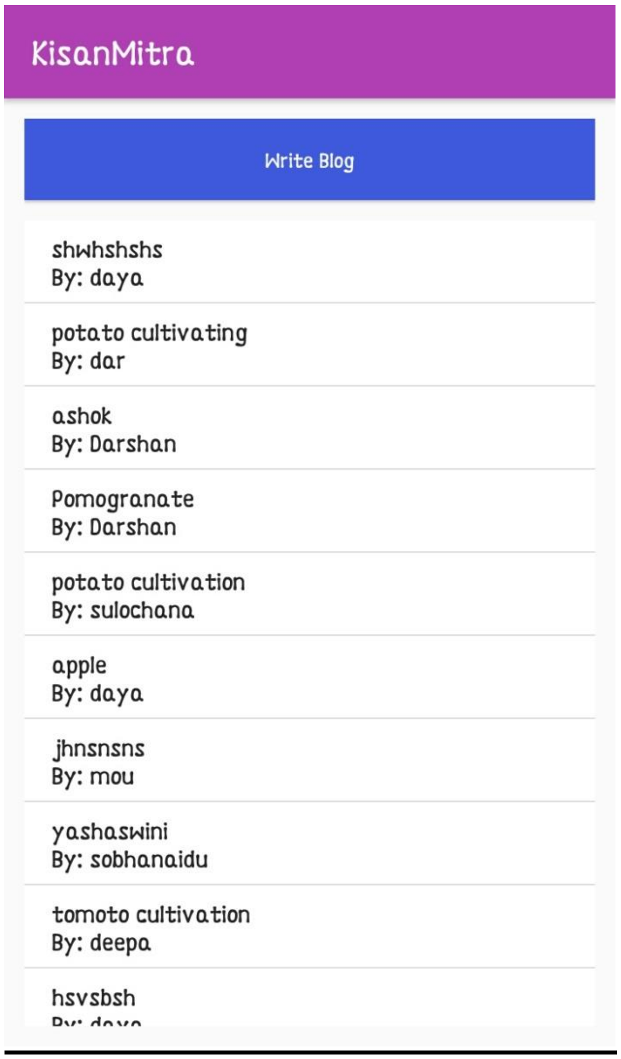
pomagranate

https://youtu.be/RdVBtGrYlts

please provide some easy ways to grow pomagranate crops

- **The user writes the topic name and suggestions , then he submits it.**

8.11 Blog Activity 2:


KisanMitra

shwhshshs
By: daya

potato cultivating
By: dar

ashok
By: Darshan

Pomogranate
By: Darshan

potato cultivation
By: sulochana

apple
By: daya

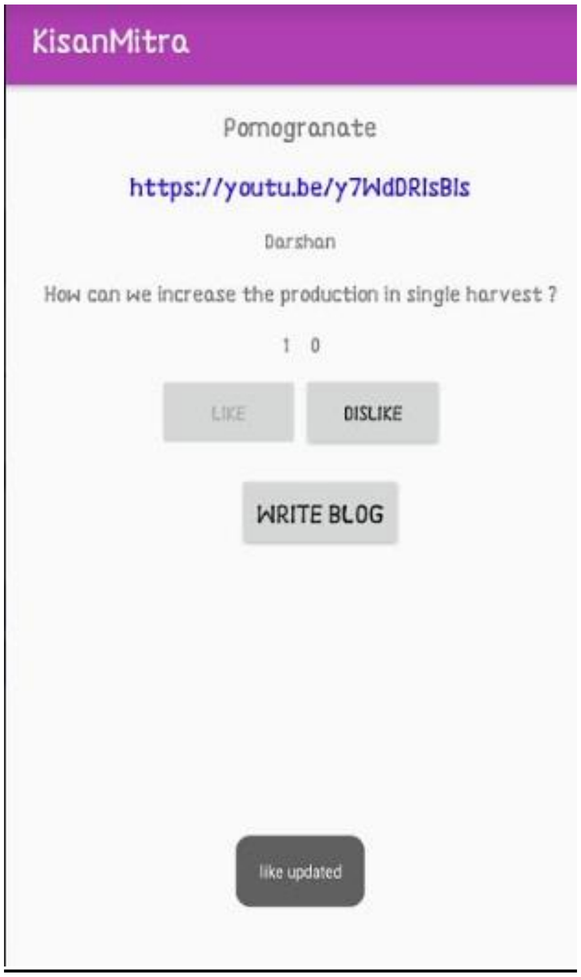
jhnsnsns
By: mou

yashaswini
By: sobhanaidu

tomoto cultivation
By: deepa

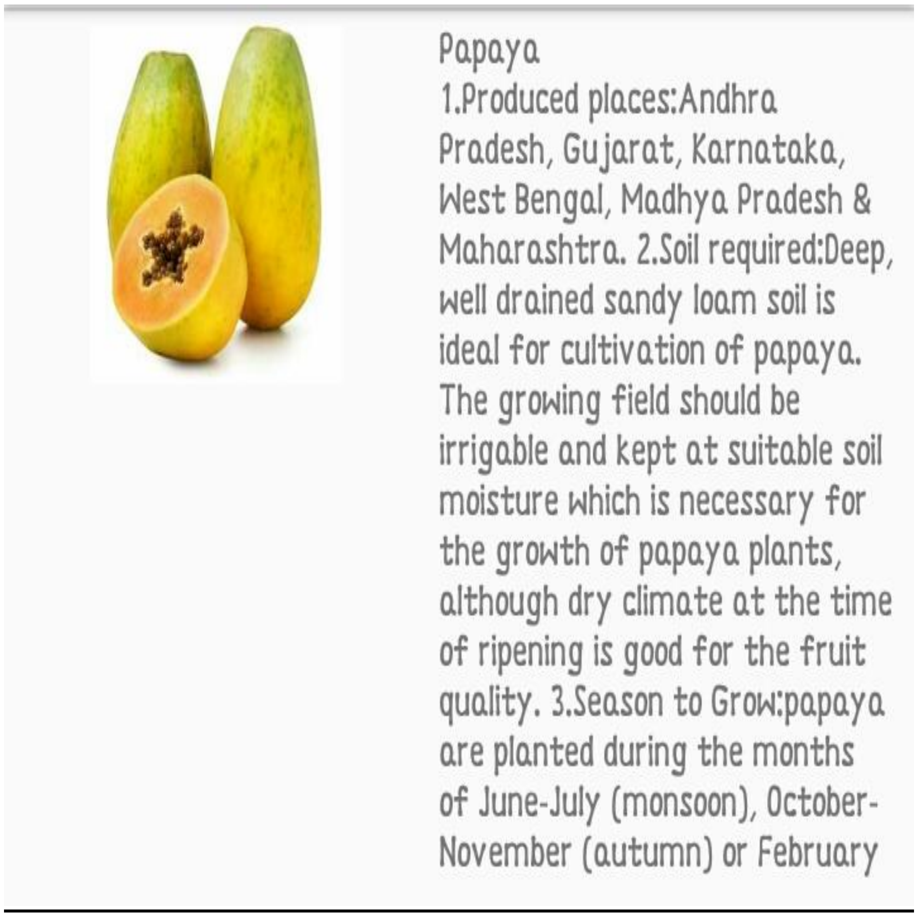
hsvsbsh

- **The user views the other comments.**

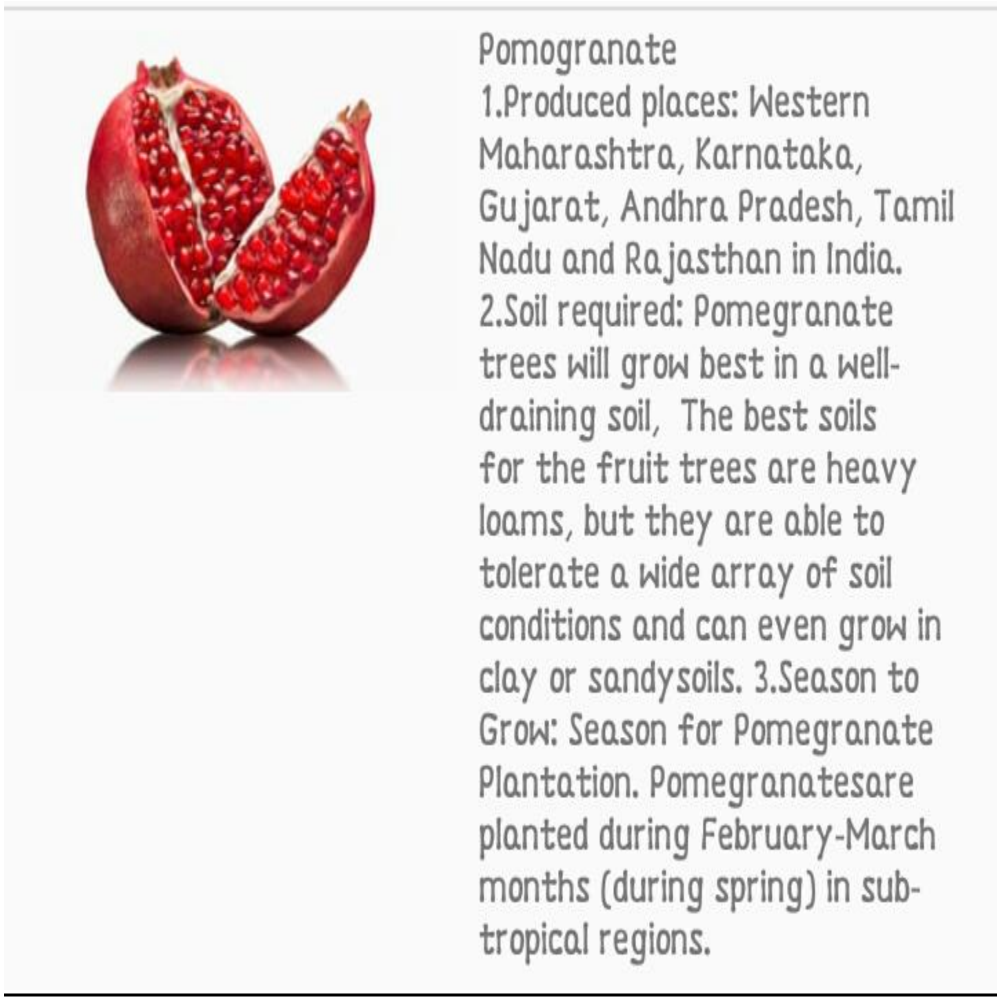

KisanMitra
Pomogranate
https://youtu.be/y7WdDRIsBls
Darshan
How can we increase the production in single harvest ?
1 0
LIKE
DISLIKE
WRITE BLOG
like updated

- **User will like (or) Dislike the comments.And he clicks the link suggested by the other user.**

8.12 Crop Details:



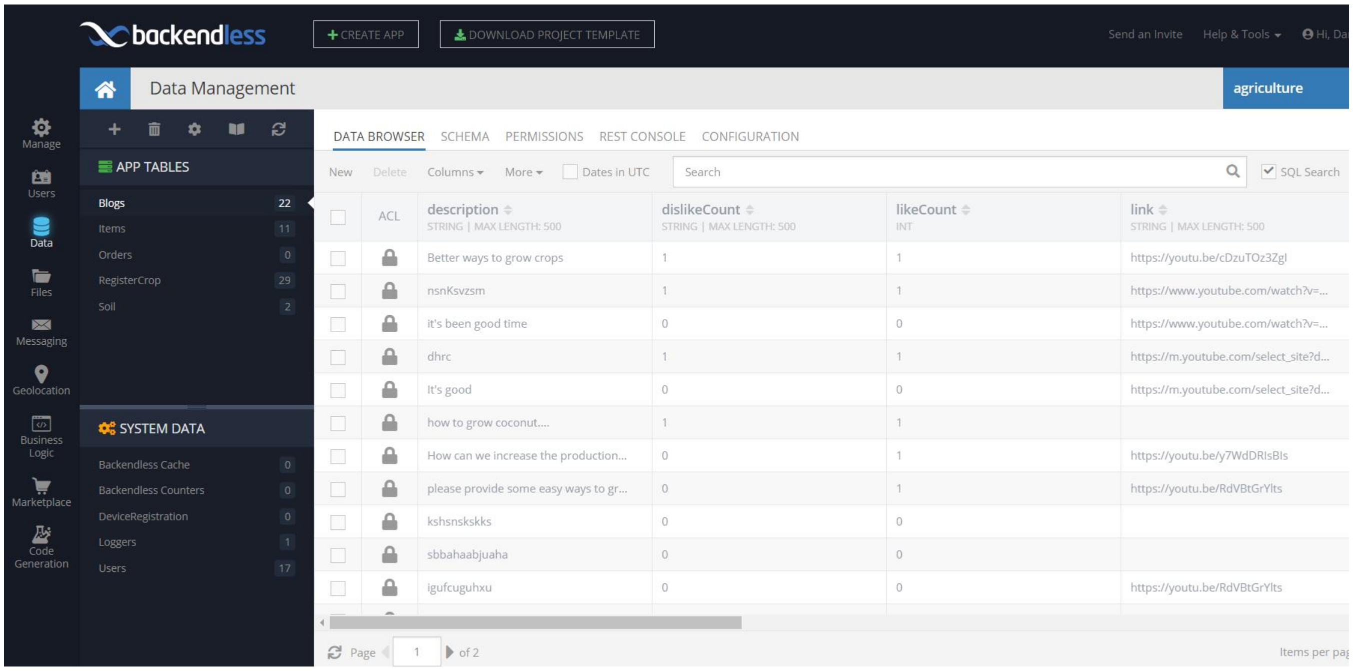
Papaya
1.Produced places:Andhra Pradesh, Gujarat, Karnataka, West Bengal, Madhya Pradesh & Maharashtra. 2.Soil required:Deep, well drained sandy loam soil is ideal for cultivation of papaya. The growing field should be irrigable and kept at suitable soil moisture which is necessary for the growth of papaya plants, although dry climate at the time of ripening is good for the fruit quality. 3.Season to Grow:papaya are planted during the months of June-July (monsoon), October-November (autumn) or February



Pomogranate
1.Produced places: Western Maharashtra, Karnataka, Gujarat, Andhra Pradesh, Tamil Nadu and Rajasthan in India.
2.Soil required: Pomegranate trees will grow best in a well-draining soil, The best soils for the fruit trees are heavy loams, but they are able to tolerate a wide array of soil conditions and can even grow in clay or sandysoils. 3.Season to Grow: Season for Pomegranate Plantation. Pomegranatesare planted during February-March months (during spring) in sub-tropical regions.

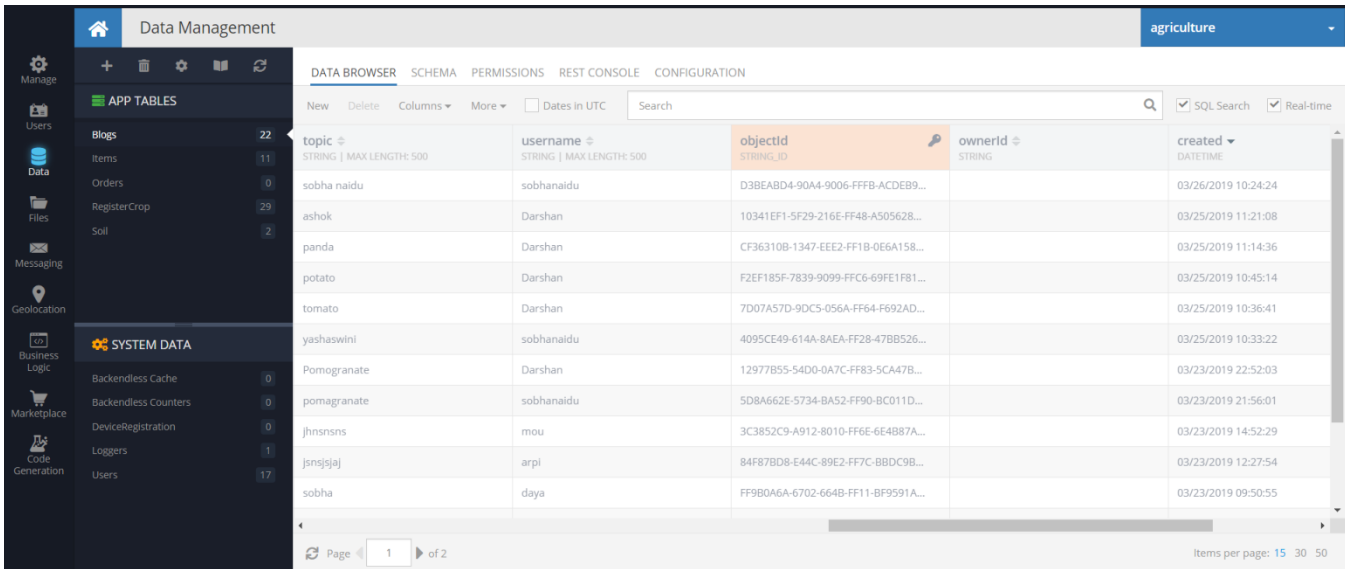
- **When user clicks on the crop details blogs, he gets the crop details (produced places,soil required,season to grow)**

8.13 Admin blog:



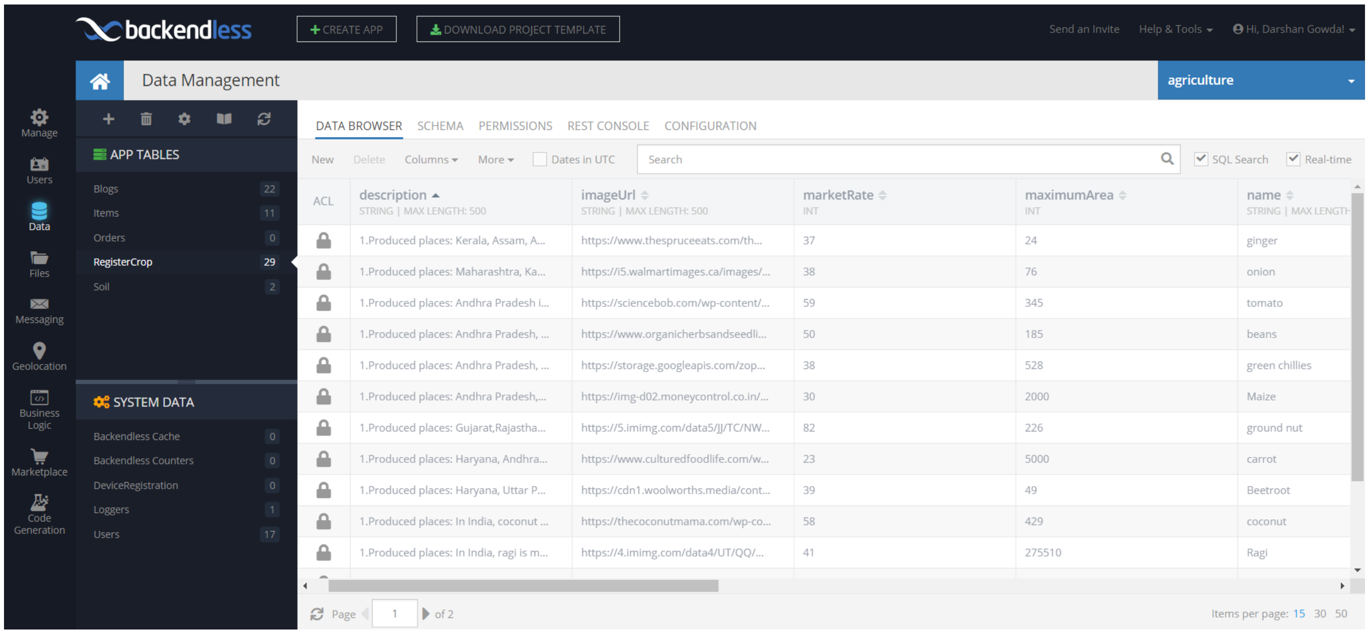
ACL	description STRING | MAX LENGTH: 500	dislikeCount STRING | MAX LENGTH: 500	likeCount INT	link STRING | MAX LENGTH: 500
🔒	Better ways to grow crops	1	1	https://youtu.be/cDzuTOz3Zgl
🔒	nsnKsvzsm	1	1	https://www.youtube.com/watch?v=...
🔒	it's been good time	0	0	https://www.youtube.com/watch?v=...
🔒	dhrc	1	1	https://m.youtube.com/select_site?d...
🔒	It's good	0	0	https://m.youtube.com/select_site?d...
🔒	how to grow coconut....	1	1	
🔒	How can we increase the production...	0	1	https://youtu.be/y7WdDRIsBIs
🔒	please provide some easy ways to gr...	0	1	https://youtu.be/RdVBtGrYlts
🔒	kshsnskskks	0	0	
🔒	sbbahaabjuaha	0	0	
🔒	igufcuguhxu	0	0	https://youtu.be/RdVBtGrYlts

- **Admin will be able to view the comments, likes/dislikes which is given by the user.**



topic STRING | MAX LENGTH: 500	username STRING | MAX LENGTH: 500	objectId STRING_ID	ownerId STRING	created DATETIME
sobha naidu	sobhanaidu	D3BEABD4-90A4-9006-FFFB-ACDEB9...		03/26/2019 10:24:24
ashok	Darshan	10341EF1-5F29-216E-FF48-A505628...		03/25/2019 11:21:08
panda	Darshan	CF36310B-1347-EEE2-FF1B-0E6A158...		03/25/2019 11:14:36
potato	Darshan	F2EF185F-7839-9099-FFC6-69FE1F81...		03/25/2019 10:45:14
tomato	Darshan	7D07A57D-9DC5-056A-FF64-F692AD...		03/25/2019 10:36:41
yashaswini	sobhanaidu	4095CE49-614A-8AEA-FF28-47BB526...		03/25/2019 10:33:22
Pomogranate	Darshan	12977B55-54D0-0A7C-FF83-5CA47B...		03/23/2019 22:52:03
pomagranate	sobhanaidu	5D8A662E-5734-BA52-FF90-BC011D...		03/23/2019 21:56:01
jhnsnsns	mou	3C3852C9-A912-8010-FF6E-6E4B87A...		03/23/2019 14:52:29
jsnsjsjaj	arpi	84F87BD8-E44C-89E2-FF7C-BBDC9B...		03/23/2019 12:27:54
sobha	daya	FF9B0A6A-6702-664B-FF11-BF9591A...		03/23/2019 09:50:55

- **Admin can also get to know the user names and topics.**



APP TABLES	
Blogs	22
Items	11
Orders	0
RegisterCrop	29
Soil	2

SYSTEM DATA	
Backendless Cache	0
Backendless Counters	0
DeviceRegistration	0
Loggers	1
Users	17

ACL	description STRING | MAX LENGTH: 500	imageUrl STRING | MAX LENGTH: 500	marketRate INT	maximumArea INT	name STRING | MAX LENGTH
	1.Produced places: Kerala, Assam, A...	https://www.thespruceeats.com/th...	37	24	ginger
	1.Produced places: Maharashtra, Ka...	https://i5.walmartimages.ca/images/...	38	76	onion
	1.Produced places: Andhra Pradesh i...	https://sciencebob.com/wp-content/...	59	345	tomato
	1.Produced places: Andhra Pradesh, ...	https://www.organicherbsandseedli...	50	185	beans
	1.Produced places: Andhra Pradesh, ...	https://storage.googleapis.com/zop...	38	528	green chillies
	1.Produced places: Andhra Pradesh,...	https://img-d02.moneycontrol.co.in/...	30	2000	Maize
	1.Produced places: Gujarat,Rajastha...	https://5.imimg.com/data5/JJ/TC/NW...	82	226	ground nut
	1.Produced places: Haryana, Andhra...	https://www.culturedfoodlife.com/w...	23	5000	carrot
	1.Produced places: Haryana, Uttar P...	https://cdn1.woolworths.media/cont...	39	49	Beetroot
	1.Produced places: In India, coconut ...	https://thecoconutmama.com/wp-co...	58	429	coconut
	1.Produced places: In India, ragi is m...	https://4.imimg.com/data4/UT/QQ/...	41	275510	Ragi

Page 1 of 2 Items per page: 15 30 50

- **Here, admin can change the market rate according to the present rates.**

CONCLUSION

CHAPTER-9

CONCLUSION

It was a great experience for us to make project on subject "Kishan Mitra Chat application". We had a some good and worst points in a project. But we tried to make every possible effort to make system perfect and reduce the all weak points from a project. From beginning of the project we have try to do something different and we think we succeed in that up to some extent

9.1 LIMITATIONS OF THE PROJECT

- Database cannot be used.
- Only the static can be used.
- Dynamic cannot be used.

9.2 FUTURE ENHANCEMENTS

- We will also put feature of upload images , video , audio so conversation become more convenient..
- One to one chat so that farmer can share their experience to their farmer friends.
- In future application provide search option.
- We will put some features which will update contents of crops automatically when device will connect to the network.
- We will put visual information like video of youtube for farmer. Which is very easy to understand by regional farmer community.

REFERENCES/BIBILOGRAPHY

CHAPTER-10

REFERENCES/BIBILOGRAPH

Websites

[1] www.wikipedia.com

[2] https://www.scribd.com/doc/88753639/Online-Shooping-Project-Report

[3] http://www.androiddevloper.com

[4] http://www.javatpoint.com

[5] http://www.tutorialpoint.com