**Farmer Buddy Project**

**By- Rishabh Verma (1719213083)**

**Vivek Anand (1719213124)**

**Vishwajeet Saini(1719213123)**

**Project Report of Farmer Buddy**

*Introduction of the Project Farmer buddy:*

The "Farmer Buddy" has been developed to override the problems prevailing in the practicing manual system. This software is supported to eliminate and, in some cases, reduce the hardships faced by this existing system. Moreover, this system is designed for the particular need of the company to carry out operations in a smooth and effective manner.

The application is reduced as much as possible to avoid errors while entering the data. It also provides error messages while entering invalid data. No formal knowledge is needed for the user to use this system. Thus, by this all it proves it is user-friendly. Farmer Buddy, as described above, can lead to error free, secure, reliable and fast management systems. It can assist the user to concentrate on their other activities rather than to concentrate on the record keeping. Thus, it will help organizations in better utilization of resources.

Every organization, whether big or small, has challenges to overcome and manage the information of Equipment's, Crops, Pesticides, System User, Customer. Every Farmer Buddy system has different Crops needs; therefore we design exclusive employee management systems that are adapted to your managerial requirements.

This is designed to assist in strategic planning, and will help you ensure that your organization is equipped with the right level of information and details for your future goals. Also, for those busy executives who are always on the go, our systems come with remote access features, which will allow you to manage your workforce anytime, at all times. These systems will ultimately allow you to better manage resources.

**Abstract of the Project Farmer Buddy:**

The purpose of Farmer Buddy is to automate the existing manual system by the help of computerized equipment and full-fledged computer software, fulfilling their requirements, so that their valuable data/information can be stored for and hardware are easily available and easy to work with.

Farmer Buddy, as described above, can lead to error free, secure, reliable and fast management systems. It can assist the user to concentrate on their other activities rather than to concentrate on record keeping. Thus, it will help organizations in better utilization of resources. The organization can maintain computerized records without redundant entries. That means that one need not be distracted by information that is not relevant, while being able to reach the information.

The aim is to automate its existing manual system by the help of computerized equipment and full-fledged computer software, fulfilling their requirements, so that their valuable data/information can be stored for a longer period with easy accessing and manipulation of the same. Basically, the project describes how to manage for good performance and better services for the clients.

**Objective of Project on Farmer Buddy:**

* The main objective of the Project on Farmer Buddy is to manage
* the details of Crops, Equipment's, Insecticides, Pesticides, Customer. It manages all
* the information about Crops, System User, Customer, Crops. The project is totally built
* at the administrative end and thus only the administrator is guaranteed the access. The
* purpose of the project is to build an application program to reduce the manual work for
* managing the Crops, Equipment, System User, Insecticides. It tracks all the details
* about the Insecticides, Pesticides, Customer.
* Functionalities provided by Farmer Buddy are as follows:
* Provides the searching facilities based on various factors. Such as Crops, Insecticides, Pesticides, Customer
* Farmer Buddy also manages the System User details online for Pesticides details, Customer details, Crops.
* It tracks all the information of Equipment's, System User, Pesticides etc
* Manage the information of Equipments
* Shows the information and description of the Crops, Insecticides
* To increase efficiency of managing the Crops, Equipments
* It deals with monitoring the information and transactions of Pesticides.
* Manage the information of Crops
* Editing, adding and updating of Records is improved which results in proper resource management of Crops data.
* Manage the information of Pesticides
* Integration of all records of Customers.

**Scope of the project Farmer Buddy**

It may help collect perfect management in detail. In a very short time, the collection will be obvious, simple and sensible. It will help a person to know the management of the past year perfectly and vividly. It also helps in current all works relative to the Farm Management System. It will also reduce the cost of collecting the management & collection procedure so we will go smoothly.

Our project aims at Business process automation, i.e. We have tried to computerize various processes of the Farmer Buddy.

* In the computer system the person has to fill the various forms & number of copies of the forms can be easily generated at a time.
* In the computer system, it is not necessary to create the manifest but we can directly print it, which saves our time.
* To assist the staff in capturing the effort spent on their respective working areas.
* To utilize resources in an efficient manner by increasing their productivity through automation.
* The system generates types of information that can be used for various purposes.
* It satisfy the user requirement
* Be easy to understand by the user and operator
* Be easy to operate
* Have a good user interface
* Be expandable
* Delivered on schedule within the budget.

**Reports of Farmer Buddy:**

.

* It generates the report on Crops, Equipments, System User
* Provide filter reports on Insecticides, Pesticides, Customer

You can easily export PDF for the Crops, System User, Pesticides

* Application also provides excel export for Equipments, Insecticides, Customer
* You can also export the report into csv format for Crops, Equipments, Customer

**Modules of Farmer Buddy**:

* Crops Management Module: Used for managing the Crops details.
* Customer Module : Used for managing the details of Customer
* System User Module : Used for managing the details of System User
* Equipment Management Module: Used for managing the information and details of the Equipments.
* Insecticides Module: Used for managing the Insecticides details
* Pesticides Module : Used for managing the Pesticides information
* Login Module: Used for managing the login details
* Users Module : Used for managing the users of the system

**Input Data and Validation of Project on Farmer Buddy**

* All the fields such as Crops, Insecticides, Customer are validated and does not take invalid values
* Each form for Crops, Equipments, System User can not accept blank value fields
* Avoiding errors in data
* Controlling amount of input
* Integration of all the modules/forms in the system.
* Preparation of the test cases.
* Preparation of the possible test data with all the validation checks.
* Actual testing done manually.
* Recording of all the reproduced errors.
* Modifications done for the errors found during testing.
* Prepared the test result scripts after rectification of the errors.
* Functionality of the entire module/forms.
* Validations for user input.
* Checking of the Coding standards to be maintained during coding.
* Testing the module with all the possible test data.
* Testing of the functionality involving all types of calculations etc.
* Commenting standard in the source files.

**The software quality plan we will use the following SQA Strategy:**

* In the first step, we will select the test factors and rank them. The selected test factors such as reliability, maintainability, portability or etc, will be placed in the matrix according to their ranks.
* The second step is for identifying the phases of the development process. The phase should be recorded in the matrix.
* The third step is identifying the business risks of the software deliverables.
* The risks will be ranked into three ranks such as high, medium and low.

**Features of the project Farmer Buddy:**

* Product and Component based
* Creating and Changing issue at ease
* Query issue list to any depth
* Reporting and charting in more comprehensive way
* User Accounts to control the access and maintain security
* Simple status and Resolutions
* Multi-level Priorities and Severities
* Accuracy in work
* Easy and fast retrieval of information
* Easy to update information
* Well designed reports
* Robust database back end
* It contain better storage capacity
* Work becomes very speedy

**Software Requirement Specification**

The Software Requirements Specification is produced at the culmination of the analysis task. The function and performance allocated to software as part of system engineering are refined by establishing a complete information description, a detailed functional and behavioral description, an indication of performance requirements and design constraints, appropriate validation criteria, and other data pertinent to requirements.

The proposed system has the following requirements:

* System needs to store information about new entries of Crops.
* System needs to help the internal staff to keep information on equipment and find them as per various queries.
* System needs to maintain a quantity record.
* System needs to keep the record of Insecticides.
* System needs to update and delete the record.
* System also needs a search area.
* It also needs a security system to prevent data.

**Identification of need:**

The old manual system was suffering from a series of drawbacks. Since the whole of the system was to be maintained with hands the process of keeping, maintaining and retrieving the information was very tedious and lengthy. The records were never used in a systematic order. there used to be lots of difficulties in associating any particular transaction with a particular context. If any information was to be found it was required to go through the different registers, documents there would never exist anything like report generation. There would always be unnecessary consumption of time while entering records and retrieving records. One more problem was that it was very difficult to find errors while entering the records. Once the records were entered it was very difficult to update these records.

The reason behind it is that there is a lot of information to be maintained and has to be kept in mind while running the business. For this reason we have provided features Present system is partially automated (computerized), actually the existing system is quite laborious as one has to enter the same information at three different places.

**Following points should be well considered:**

* Documents and reports that must be provided by the new system: there can also be few reports, which can help management in decision-making and cost controlling, but since these reports do not get required attention, such kinds of reports and information were also identified and given required attention.
* Details of the information needed for each document and report.
* The required frequency and distribution for each document.
* Probable sources of information for each document and report.
* With the implementation of computerized systems, the task of keeping records in an organized manner will be solved. The greatest of all is the retrieval of information, which will be at the click of the mouse. So the proposed system helps in saving the time in different operations and making information flow easy giving valuable reports.

**Feasibility Study:**

After doing the project Farm Management System, study and analyzing all the existing or required functionalities of the system, the next task is to do the feasibility study for the project. All projects are feasible - given unlimited resources and infinite time.

Feasibility study includes consideration of all the possible ways to provide a solution to the given problem. The proposed solution should satisfy all the user requirements and should be flexible enough so that future changes can be easily done based on the future upcoming requirements.

**A. Economical Feasibility**

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

• All hardware and software cost has to be borne by the organization.

• Overall we have estimated that the benefits the organization is going to receive from the proposed system will surely overcome the initial costs and the later on running cost for the system.

**B. Technical Feasibility**

This included the study of function, performance and constraints that may affect the ability to achieve an acceptable system. For this feasibility study, we studied complete functionality to be provided in the system, as described in the System Requirement Specification (SRS), and checked if everything was possible using different types of frontend and backend platforms.

**C. Operational Feasibility**

No doubt the proposed system is fully GUI based that is very user friendly and all inputs to be taken are all self-explanatory even to a layman. Besides, a proper training has been conducted to let the user know the essence of the system to the users so that they feel comfortable with the new system. As far our study is concerned the clients are comfortable and happy as the system has cut down their loads and doing.

**System Design of Farm Management System:**

In this phase, a logical system is built which fulfils the given requirements. Design phase of software development deals with transforming the clients requirements into a logically working system. Normally, design is performed in the following in the following two steps:

**1. Primary Design Phase:**

In this phase, the system is designed at block level. The blocks are created on the basis of analysis done in the problem identification phase. Different blocks are created for different functions emphasis is put on minimising the information flow between blocks. Thus, all activities which require more interaction are kept in one block.

**2. Secondary Design Phase:**

In the secondary phase the detailed design of every block is performed.

The general tasks involved in the design process are the following:

1. Design various blocks for overall system processes.

2. Design smaller, compact and workable modules in each block.

3. Design various database structures.

4. Specify details of programs to achieve desired functionality.

5. Design the form of inputs, and outputs of the system.

6. Perform documentation of the design.

7. System reviews.

**User Interface Design**

User Interface Design is concerned with the dialogue between a user and the computer. It is concerned with everything from starting the system or logging into the system to the eventually presentation of desired inputs and outputs. The overall flow of screens and messages are called a dialogue.

The following steps are various guidelines for User Interface Design:

1. The system user should always be aware of what to do next.

2. The screen should be formatted so that various types of information, instructions and messages always appear in the same general display area.

3. Messages, instructions or information should be displayed long enough to allow the system user to read them.

4. Use display attributes sparingly.

5. Default values for fields and answers to be entered by the user should be specified.

6. A user should not be allowed to proceed without correcting an error.

7. The system user should never get an operating system message or fatal error.

**Preliminary Product Description:**

The first step in the system development life cycle is the preliminary investigation to determine the feasibility of the system. The purpose of the preliminary investigation is to evaluate project requests. It is not a design study nor does it include the collection of details to describe the business system in all respects. Rather, it is the collecting of information that helps committee members to evaluate the merits of the project request and make an informed judgment about the feasibility of the proposed project. Analysts working on the preliminary investigation should accomplish the following objectives:

* Clarify and understand the project request
* Determine the size of the project.
* Assess costs and benefits of alternative approaches.
* Determine the technical and operational feasibility of alternative approaches.
* Report the findings to management, with recommendations outlining the acceptance or rejection of the proposal.

**Benefit to Organization**

The organization will obviously be able to gain benefits such as savings in operating cost, reduction in paperwork, better utilization of human resources and more presentable image increasing goodwill.

**The Initial Cost**

The initial cost of setting up the system will include the cost of hardware software(OS, add-on software, utilities) & labour (setup & maintenance). The same has to be borne by the organization.

**Running Cost**

Besides, the initial cost the long term cost will include the running cost for the system including the AMC, stationary charges, cost for human resources, cost for update/renewal of various related software.

**Need for Training**

The users along with the administrator need to be trained at the time of

implementation of the system for smooth running of the system. The client will provide the training site.

We talked to the management people who were managing the financial issues of the center, the staff who were keeping the records in lots of registers and the reporting manager regarding their existing system, their requirements and their expectations from the new proposed system. Then, we did the system study of the entire system based on their requirements and the additional features they wanted to incorporate in this system.

Reliable, accurate and secure data was also considered to be a complex task without this proposed system. Because there was no such record for keeping track of all the activities, which was done by the Farm Management System on a daily basis.

The new system proposed and then developed by me will ease the task of the

organization in consideration. It will be helpful in generating the required reports by the staff, which will help them to track their progress and services.

Thus, it will ease the task of Management to a great extent as all the major activities to be performed are computerized through this system.

**Project Category**

Relational Database Management System (RDBMS): This is an RDBMS based project which is currently using MySQL for all the transaction statements. MySQL is an open source RDBMS System.

***Brief Introduction about RDBMS :***

A relational database management system (RDBMS) is a database management system (DBMS) that is based on the relational model as invented by E. F. Codd, of IBM's San Jose Research Laboratory. Many popular databases currently in use are based on the relational database model.

RDBMSs have become a predominant choice for the storage of information in new databases used for financial records, manufacturing and logistical information, personnel data, and much more since the 1980s. Relational databases have often replaced legacy hierarchical databases and network databases because they are easier to understand and use. However, relational databases have been challenged by object databases, which were introduced in an attempt to address the object-relational impedance mismatch in relational databases, and XML databases.

**Implementation Methodology:**

Model View Controller or MVC as it is popularly called, is a software design pattern for developing web applications. A Model View Controller pattern is made up of the following three parts:

• Model - The lowest level of the pattern which is responsible for maintaining data.

• View - This is responsible for displaying all or a portion of the data to the user.

• Controller - Software Code that controls the interactions between the Model and View.

MVC is popular as it isolates the application logic from the user interface layer and supports separation of concerns. Here the Controller receives all requests for the application and then works with the Model

to prepare any data needed by the View. The View then uses the data prepared by the Controller to generate a final presentable response. The MVC abstraction can be graphically represented as follows.

MVC (Model View Controller Flow) Diagram



**Project Planning:**

Software project plan can be viewed as the following:

1) **Within the organization**: How the project is to be implemented? What are

various constraints (time, cost, staff)? What is market strategy?

2) **With respect to the customer:** Weekly or timely meetings with the customer with presentations on status reports. Customer feedback is also taken and further modification and developments are done. Project milestones and deliverables are also presented to the customer.

**For a successful software project. the following steps can be followed**:

**1. Select a project**

* Identifying project's aims and objectives
* Understanding requirements and specification
* Methods of analysis, design and implementation
* Testing techniques
* Documentation

.

**2.Project milestones and deliverables**

* Budget allocation
* Exceeding limits within control
* Project Estimates
* Cost
* Time
* Size of code
* Duration
* Resource Allocation
* Hardware
* Software
* Previous relevant project information
* Digital Library
* Risk Management
* Risk avoidance
* Risk detection

**Cost estimation of the project:**

Software cost comprises a small percentage of overall computer-based system cost.

There are a number of factors, which are considered, that can affect the ultimate cost of the software such as - human, technical, Hardware and Software availability etc.

The main point that was considered during the cost estimation of the project was its sizing. In spite of complete software sizing, function point and approximate lines of code were also used to "size" each element of the Software and their costing.

The cost estimation done by me for the Project also depends upon the baseline metrics collected from past projects and these were used in conjunction with estimation variables to develop cost and effort projections.

We have basically estimated this project mainly on two bases -

1) Effort Estimation - This refers to the total man-hours required for the development of the project. It even includes the time required for doing documentation and user manuals.

2) Hardware Required Estimation - This includes the cost of the PCs and the hardware cost required for development of this project.

**Tools/Platform, Hardware and Software Requirement specifications**:

Software Requirements:

**Name of component Specification:**

**Operating System Windows 7, Windows 10.**

**Language Java 2 Runtime Environment**

**Database MySQL Server**

**Browser Any of Mozilla, Opera, Chrome**

**Web Server Tomcat 7**

**Software Development Kit Java JDK 1.7 or Above**

**Scripting Language Enable JSP (Java Server Pages)**

**Database JDBC Driver MySQL Connector**

**Hardware Requirements:**

**Name of component Specification**

**Processor Pentium III 630MHz**

**RAM 128 MB**

**Hard disk 20 GB**

**Monitor 15" color monitor**

**Keyboard 122 keys**

**Project Profile**

There has been continuous effort to develop tools, which can ease the process of software development. But, with the evolving trend of different programming paradigms today's software developers are really challenged to deal with the changing technology. Among other issues, software re-engineering is being regarded as an important process in the software development industry. One of the major tasks here is to understand software systems that are already developed and to transform them to a different software environment. Generally, this requires a lot of manual effort in going through a program that might have been developed by another programmer. This project makes a novel attempt to address the issue of program analysis and generation of diagrams, which can depict the structure of a program in a better way. Today, UML is being considered as an industrial standard for the software engineering design process. It essential provides several diagramming tools that can express different aspects/characteristics of program such as

**Use cases:** Elicit requirement from users in meaningful chunks. Construction planning is built around delivering some use cases on each interaction basis for system testing.



**Class diagrams:** shows static structure of concepts, types and class. Concepts how users think about the world; type shows interfaces of software components; classes show implementation of software components.



**Entity relation diagrams:** A basic ER model is composed of entity types and specifies relationships that can exist between entities



**Package diagram:** show group of classes and dependencies among them.

**State diagram:** show how a single object behaves across many use cases.

**Activity diagram:** shows behavior with control structure. Can show many objects over many uses, many objects in a single use case, or implementations methods encourage parallel behavior, etc.

The end-product of this project is a comprehensive tool that can parse any vb.net program and extract most of the object-oriented features inherent in the program such as polymorphism, inheritance, encapsulation and abstraction.

**What is UML?**

UML stands for Unified Modeling Language is the successor to the wave of Object-Oriented Analysis and Design (OOA&D) methods that appeared in the late 80's.

It most directly unifies the methods of Booch, Rumbaugh (OMT) and Jacobson. The UML is called a modeling language, not a method. Most methods consist at least in principle, of both a modeling language and a process. The Modeling language is that notation that methods used to express design.

**Notations and meta-models:**

The notation is the graphical stuff; it is the syntax of the modeling language. For instance, class diagram notation defines how items are concepts such as class, association, and multiplicity is represented. These are:

**Class Diagram:** The class diagram technique has become truly central within object-oriented methods. Virtually every method has included some variation on this technique.

Class diagrams are also subject to the greatest range of modeling concepts. Although the basic elements are needed by everyone, advanced concepts are used less often. A class diagram describes the types of objects in the system and the various kinds of static relationship that exist among them. There are two principal kinds of static relationship:

• Association

* Subtype

Class diagrams also show the attributes and operations of a class and the constraints that apply to the way objects are connected.

**Association:** Association represents between instances of class. From the conceptual perspective, association represents conceptual relations between classes. Each association has two roles. Each role is a direction on the association. A role also has multiplicity, which is an indication of how many objects may participate in the given relationship.

**Generalization:** A typical example of generalization evolves the personal and corporate customer of a business. They have differences but also many similarities. The similarities can be placed in generalization with personal customer and corporate customer subtype.

**Aggregation:** aggregation is the part of a relationship. It is like saying a car has an engine and wheels as its parts. This sounds good, but the difficult thing to consider is the difference between aggregation and association.

**Interaction:** interaction diagrams are models that describe how groups of objects collaborate in some behavior.

Typically, an interaction diagram captures the behavior of a single use case. The diagram shows a number of example objects and the messages that are passed between these objects in use cases. These are following approaches with a simple use case that exhibits the following behavior.

Objects can send a message to another. Each message is checked with a given stock item. There are two diagrams: Sequence and Collaboration diagram.

**Package Diagram:** One of the oldest questions in software methods is: how do you break down a large system into smaller systems? It becomes difficult to understand and the changes we make to them.

Structured methods used functional decomposition in which the overall system was mapped as a function broken down into sub function, which is further broken down into sub function and so forth. The separation of process data is gone, functional decomposition is gone, but the old question is still remaining. One idea is to group the classes together into higher-level units. This idea, applied very loosely, appears in many objects. In UML, this grouping mechanism is a package. The term package diagram for a diagram that shows packages of classes and the dependencies among them.

A dependency exists between two elements if changes to the definition of one element may cause another. With classes, dependencies exist for various reasons: one class sends a message to another, one class has another as part of its data; one class mentions another as a parameter to an operation. A dependency between two packages exists; and any dependencies exist between any two classes in the package.

**State diagram:** State diagrams are a familiar technique to describe the behavior of a system. They describe all the possible states a particular object can get into and how the object's state changes as a result of events that reach the objects. In most techniques, state diagrams are drawn for a single class to show the lifetime behavior of a single object. There are many forms of state diagram, each with slightly different semantics. The most popular one used in 0 technique is based on David Harel's statechart.

**Use Case Model of the Project:**

The use case model for any system consists of use cases". Use cases represent different ways in which the system can be used by the user. A simple way to find all the use cases of a system is to ask the questions "What the user can do using the system?"

The use cases partition the system behavior into transactions such that each transaction performs some useful action from the users' point of view.

The purpose of the use case is to define a piece of coherent behavior without revealing the internal structure of the system. A use case typically represents a sequence of interaction between the user and the system. These interactions consist of one main line sequence that represents the normal interaction between the user and the system. The use case model is an important analysis and design artifact (task). Use cases can be represented by drawing a use case diagram and writing an accompanying text elaborating the drawing.

In the use case diagram, each use case is represented by an ellipse with the name of use case written inside the ellipse. All the ellipses of the system are enclosed within a rectangle which represents the system boundary. The name of the system being modeled appears inside the rectangle. The different users of the system are represented by using a stick person icon. The stick person icon is normally referred to as an Actor. The line connecting the actor and the use cases is called the communication relationship. When a stick person icon represents an external system, it is annotated by the stereotype<<external system>>.

**About ER Diagram:**

**Entity Relationship Diagram**

E-R Model is a popular high level conceptual data model. This model and its variations are frequently used for the conceptual design of database application and many database design tools employ its concept.

A database that confirms to an E-R diagram can be represented by a collection of tables in the relational system. The mapping of E-R diagram to the entities are:

* Attributes
* Relations
* Many-to-many
* Many-to-one
* One-to-many
* One-to-one
* Weak entities
* Subtype and supertype

The entities and their relationships between them are shown using the following conventions.

* An entity is shown in a rectangle.
* A diamond represents the relationship among a number of entities.
* The attributes shown as ovals are connected to the entities or relationship by lines.
* Diamond, oval and relationships are labeled
* Model is an abstraction process that hides super details while highlighting details relation to application at end.
* A data model is a mechanism that provides this abstraction for database application.
* Data modeling is used for representing entities and their relationship in the database.
* Entities are the basic units used in modeling database entities that can have concrete existence or constitute ideas or concepts.
* Entity type or entity set is a group of similar objects concern to an organization for which it maintains data,
* Properties are characteristics of an entity also called as attributes.
* A key is a single attribute or combination of 2 or more attributes of an
* entity set is used to identify one or more instances of the set.
* In relational models we represent the entity by a relation and use tuples to represent an instance of the entity.
* Relationship is used in data modeling to represent association between an entity set.
* An association between two attributes indicates that the values of the associated attributes are independent.

**Security Testing of the Project**

Testing is vital for the success of any software. no system design is ever perfect. Testing is also carried in two phases, first phase is during the software engineering that is during the module creation, second phase is after the completion of software, this is system testing which verifies that the whole set of programs changed together.

**White Box Testing:**

In this technique, the close examination of the logical parts through the software are tested by cases that exercise species sets of conditions or loops. all logical parts of the software checked once. errors that can be corrected using this technique are typographical errors, logical expressions which should be executed once may be getting executed more than once and errors resulting by using wrong controls and loops. When the box testing tests all the independent parts within a module a logical decision on their true and the false side are exercised, all loops and bounds within their operational bounds were exercised and internal data structure to ensure their validity were exercised once.

**Black Box Testing:**

This method enables the software engineer to device sets of input techniques that fully exercise all functional requirements for a program. black box testing tests the input, the output and the external data. it checks whether the input data is correct and whether we are getting the desired output

**Alpha Testing:**

Acceptance testing is also sometimes called alpha testing. Be spoke systems are developed for a single customer. The alpha testing proceeds until the system developer and the customer agree that the provided system is an acceptable implementation of the system requirements.

**Beta Testing:**

On the other hand, when a system is to be marked as a software product, another process called beta testing is often conducted. During beta testing, a system is delivered among a number of potential users who agree to use it. The customers then report problems to the developers. This provides the product for real use and detects errors which may not have been anticipated by the system developers.

**Unit Testing:**

Each module is considered independently. it focuses on each unit of software as implemented in the source code. It is white box testing.

**Integration Testing:**

Integration testing aims at constructing the program structure while at the same constructing tests to uncover errors associated with interfacing the modules. modules are integrated by using the top down approach.

**Validation Testing:**

Validation testing was performed to ensure that all the functional and performance requirements are met.

**System Testing:**

It is executing programs to check logical changes made in it with the intention of finding errors. a system is tested for online response, volume of transaction, recovery from failure etc. System testing is done to ensure that the system satisfies all the user requirements.

**Implementation and Software Specification Testing's**

**Detailed Design of Implementation**

This phase of the systems development life cycle refines hardware and software specifications, establishes programming plans, trains users and implements extensive testing procedures, to evaluate design and operating specifications and/or provide the basis for further modification.

**Technical Design**

This activity builds upon specifications produced during new system design, adding detailed technical specifications and documentation.

**Test Specifications and Planning**

This activity prepares detailed test specifications for individual modules and programs, job streams, subsystems, and for the system as a whole.

**Programming and Testing**

This activity encompasses actual development, writing, and testing of program units or modules.

**User Training**

This activity encompasses writing user procedure manuals, preparation of user training materials, conducting training programs, and testing procedures.

**Acceptance Test**

A final procedural review to demonstrate a system and secure user approval before a system becomes operational

**Installation Phase**

In this phase the new Computerized system is installed, the conversion to new procedures is fully implemented, and the potential of the new system is explored.

**System Installation**

The process of starting the actual use of a system and training user personnel in its operation.

**Review Phase**

This phase evaluates the successes and failures during a systems development project, and to measure the results of a new Computerized TranSystems in terms of benefits and savings projected at the start of the project.

**Development Recap**

A review of a project immediately after completion to find successes and potential problems in future work.

**Post-Implementation Review**

A review, conducted after a new system has been in operation for some time, to evaluate actual system performance against original expectations and projections for cost-benefit improvements. Also identifies maintenance projects to enhance or improve the system.

**THE STEPS IN THE SOFTWARE TESTING**

The steps involved during Unit testing are as follows:

a. Preparation of the test cases.

b. Preparation of the possible test data with all the validation checks.

C. Complete code review of the module.

d. Actual testing done manually.

e. Modifications done for the errors found during testing.

f. Prepared the test result scripts.

**The unit testing done included the testing of the following items:**

1. Functionality of the entire module/forms.

2. Validations for user input.

3. Checking of the Coding standards to be maintained during coding.

4. Testing the module with all the possible test data.

5. Testing of the functionality involving all types of calculations etc.

6. Commenting standard in the source files.

After completing the Unit testing of all the modules, the whole system is integrated with all its dependencies in that module. While System Integration, we integrated the modules one by one and tested the system at each step. This helped in reduction of errors at the time of the system testing.

**The steps involved during System testing are as follows:**

* Integration of all the modules/forms in the system.
* Preparation of the test cases.
* Preparation of the possible test data with all the validation checks.
* Actual testing done manually.
* Recording of all the reproduced errors.
* Modifications done for the errors found during testing.
* Prepared the test result scripts after rectification of the errors.

**The System Testing done included the testing of the following items:**

1. Functionality of the entire system as a whole.

2. User Interface of the system.

3. Testing the dependent modules together with all the possible test data scripts.

4. Verification and Validation testing.

5. Testing the reports with all its functionality.

After the completion of system testing, the next following phase was the Acceptance Testing.

Clients at their end did this and accepted the system with appreciation. Thus, we reached the final phase of the project delivery.

**There are six other tests, which fall under the special category. They are described below:**

**Peak Load Test:** It determines whether the system will handle the volume of activities that occur when the system is at the peak of its processing demand. For example, test the system by activating all terminals at the same time.

**Storage Testing:** It determines the capacity of the system to store transaction data on a disk or in other files.

• **Performance Time Testing:** it determines the length of time system used by the system to process transaction data. This test is conducted prior to implementation to determine how long it takes to get a response to an inquiry, make a backup copy of a file, or send a transmission and get a response.

• **Recovery Testing:** This testing determines the ability of the user to recover data or restart the system after failure. For example, load backup copy of data and resume processing without data or integrity loss.

• **Procedure Testing:** It determines the clarity of documentation on operation and uses of the system by having users do exactly what manuals request. For example, powering down the system at the end of week or responding to paper-out light on the printer.

• **Human Factors Testing:** It determines how users will use the system when processing data or preparing reports.

**System Analysis:**

System analysis is a process of gathering and interpreting facts, diagnosing problems and the information about the Farm Management System to recommend improvements on the system. It is a problem-solving activity that requires intensive communication between the system users and system developers. System analysis or study is an important phase of any system development process. The system is studied to the minutest detail and analyzed. The system analyst plays the role of the interrogator and dwells deep into the working of the present system. The system is viewed as a whole and the input to the system is identified. The outputs from the organizations are traced to the various processes. System analysis is concerned with becoming aware of the problem, identifying the relevant and decisional variables, analyzing and synthesizing the various factors and determining an optimal or at least a satisfactory solution or program of action. A detailed study of the process must be made by various techniques like interviews, questionnaires etc. The data collected by these sources must be scrutinized to arrive at a conclusion. The conclusion is an understanding of how the system functions. This system is called the existing system.

Now the existing system is subjected to close study and problem areas are identified.

The designer now functions as a problem solver and tries to sort out the difficulties that the enterprise faces. The solutions are given as proposals. The proposal is then weighed with the existing system analytically and the best one is selected. The proposal is presented to the user for an endorsement by the user. The proposal is reviewed on user request and suitable changes are made. This loop ends as soon as the user is satisfied with the proposal. Preliminary study is the process of gathering and interpreting facts, using the information for further studies on the system. Preliminary study is problem solving activity that requires intensive communication between the system users and system developers. It does various feasibility studies. In these studies, a rough figure of the system activities can be obtained, from which the decision about the strategies to be followed for effective system study and analysis can be taken.

**Existing System of Farm Management System:**

In the existing system the exams are done only manually but in the proposed system we have to computerize the exams using this application.

* Lack of security of data.
* More man power.
* Time consuming.
* Consumes a large volume of spare work.
* Needs manual calculations.
* No direct role for the higher officials

**Proposed System of Farm Management System:**

The aim of the proposed system is to develop a system of improved facilities. The proposed system can overcome all the limitations of the existing system. The system provides proper security and reduces the manual work.

* Security of data.
* Ensure data accuracies.
* Proper control of the higher officials.
* Minimize manual data entry.
* Minimum time needed for the various processing.
* Greater efficiency
* Better service.
* User friendliness and interactive.
* Minimum time required.

**Data Dictionary:**

This is normally represented as the data about data. It is also termed as metadata sometimes which gives the data about the data stored in the database. It defines each data term encountered during the analysis and design of a new system. Data elements can describe files or the processes.

**Following are some major symbols used in the data dictionary**

* = equivalent to
* + and
* [] either/ or
* () Optional entry

**Following are some rules, which defines the construction of data dictionary entries:**

1. Words should be defined to understand what they need and not the variable needed by which they may be described in the program.

2. Each word must be unique. We cannot have two definitions of the same client.

3. Aliases or synonyms are allowed when two or more entries show the same meaning. For example, a vendor number may also be called a customer number.

4. A self-defining word should not be decomposed. It means that the reduction of any information into subpart should be done only if it is really required, that is it is not easy to understand directly.

Data dictionary includes information such as the number of records in file, the frequency a process will run, security factors like password which the user must enter to get excess information.

**Coding of the Project Farmer Buddy**

**coding for crops.java**

package Model;

import java.util.\*;

import java.sql.\*;

import com.\*;

import java.io.\*;

public class Crops extends Connect

{

/////Function for connect to the MySQL Server Database//////

public Crops()

{

Connect.connect\_mysql();

}

/////Save User Details /////

public String saveCrops(HashMap cropsData)

{

String SQL = "INSERT INTO `crops (crops\_title', 'crops\_month', 'crops\_description',

crops\_image") VALUES (?, ?, ?, ?);";

int record=0;

String error = "";

try

pstmt = connection.prepareStatement(SQL);

pstmt.setString(1,(String) crops Data.get("crops\_title"));

pstmt.setString(2,(String) cropsData.get("crops\_month"));

pstmt.setString(3,(String) cropsData.get("crops\_description"));

pstmt.setString(4,(String) cropsData.get("crops\_image"));

record = pstmt.execute Update();

pstmt.close();

connection.close();

}

catch(Exception e)

{

StringWriter writer = new StringWriter();

Print Writer printWriter = new Print Writer( writer);

e.printStackTrace (printWriter );

printWriter.flush();

String stackTrace = writer.toString();

error+="Error : "+stackTrace;

System.out.println("Error : "+e.toString());

}

return error;

}

////////Function for getting Users Details////////

public HashMap getCrops Details(int crops\_id)

{

HashMap results = new HashMap();

int count=0;

try

{

String SQL = "SELECT \* FROM `crops WHERE crops\_id = "+crops\_id;

statement = connection.createStatement();

rs = statement.executeQuery(SQL);

while(rs.next())

{

results.put("crops\_title",rs.getString("crops\_title"));

results.put("crops\_month",rs.getString("crops\_month"));

results.put("crops\_description",rs.getString("crops\_description"));

results.put("crops\_image",rs.getString("crops\_image"));

results.put("crops\_id",rs.getString("crops\_id"));

count++;

}

if(count==0)

{

results.put("crops\_title","");

results.put("crops\_month", "");

results.put("crops\_description","");

results.put("crops\_image", "");

results.put("crops\_id","");

}

}

catch(Exception e)

{

System.out.println("Error is: "+e);

}

return results;

}

/// Update the Crops ////

public String updateCrops(HashMapcropsData)

{

String SQL = "UPDATE 'crops' SET 'crops\_title' = ?, 'crops\_month'= ?,

'crops\_description' = ?, 'crops\_image' = ? WHERE 'crops\_id' = ?;";

String error = "";

int record=0;

try

{

pstmt = connection.prepareStatement(SQL);

pstmt.setString(1,(String) cropsData.get("crops\_title"));

pstmt.setString(2,(String) cropsData.get("crops\_month"));

pstmt.setString(3,(String) cropsData.get("crops\_description"));

pstmt.setString(4,(String) cropsData.get("crops\_image"));

pstmt.setString(5,(String) cropsData.get("crops\_id"));

record = pstmt.execute Update();

pstmt.close();

connection.close();

}

catch(Exception e)

{

StringWriter writer = new StringWriter();

Print Writer print Writer = new Print Writer( writer);

e.printStackTrace( print Writer );

printWriter.flush();

String stackTrace = writer.toString();

error+="Error: "+stackTrace;

System.out.println("Error : "+e.toString());

}

return error;

}

//////////Function for getting all the Airport Details/////////

public ArrayList getAllCrops(String title)

int count=0;

String SQL = "";

SQL = "SELECT \* FROM `crops';

if(title != "") {

SQL = "SELECT \* FROM `crops' WHERE 'crops\_month' LIKE '%"+title+"%' OR crops\_title LIKE '%"+title+"%' OR crops\_description LIKE '%"+title+"%"';

}

ArrayList resultArray = new ArrayList();

try

{

statement = connection.createStatement();

rs = statement.executeQuery(SQL);

while(rs.next())

{

HashMap results = new HashMap();

results.put("crops\_title",rs.getString("crops\_title"));

results.put("crops\_month",rs.getString("crops\_month"));

results.put("crops\_description",rs.getString("crops\_description"));

results.put("crops\_image",rs.getString("crops\_image"));

results.put("crops\_id",rs.getString("crops\_id"));

count++;

resultArray.add(results);

}

}

catch(Exception e)

{

System.out.println("Error is: "+e);

}

return resultArray;

}

}

**Coding for Customer.java**

package Model;

import java.util.\*;

import java.sql.\*;

import com.\*;

import java.io.\*;

public class Customer extends Connect

{

///////Function for connect to the MySQL Server Database///////

public Customer()

{

Connect.connect\_mysql();

}

//////////Save User Details /////

public String saveCustomer(HashMap customerData)

{

String SQL = "INSERT INTO 'customer' ('customer\_name', 'customer\_mobile','customer\_email', 'customer\_password', "customer\_address', 'customer\_city,' customer\_state', 'customer\_pincode') VALUES (?, ?, ?, ?, ?, ?, ?, ?);";

int record=0;

String error = "";

try

{

pstmt = connection.prepareStatement(SQL);

pstmt.setString(1,(String) customerData.get("customer\_name"));

pstmt.setString(2.(String) customerData.get("customer\_mobile"));

pstmt.

String(3,(String) customerData.get("customer\_email"));

pstmt.setString(4,(String) customerData.get("customer\_password"));

pstmt.setString(5,(String) customerData.get("customer\_address"));

pstmt.setString(6,(String) customerData.get("customer\_city"));

pstmt.setString(7,(String) customerData.get("customer\_state"));

pstmt.setString(8,(String) customerData.get("customer\_pincode"));

record = pstmt.executeUpdate();

pstmt.close();

connection.close();

}

catch(Exception e)

{

StringWriter writer = new StringWriter();

Print Writer print Writer = new Print Writer( writer);

e.printStackTrace( printWriter );

printWriter.flush();

String stackTrace = writer.toString();

error+="Error : "+stackTrace;

System.out.println("Error : "+e.toString());

}

return error;

}

/////////Function for getting Users Details//////////

public HashMap getCustomerDetails(int customer\_id)

Hash Map results = new HashMap();

int count=0;

try

{

String SQL = "SELECT \* FROM 'customer' WHERE customer\_id = "+customer\_id;

statement = connection.createStatement();

rs = statement.executeQuery(SQL);

while(rs.next())

{

results.put("customer\_name",rs.getString("customer\_name"));

results.put("customer\_mobile",rs.getString("customer\_mobile"));

results.put("customer\_email",rs.getString("customer\_email"));

results.put("customer\_password",rs.getString("customer\_password"));

results.put("customer\_address",rs.getString("customer\_address"));

results.put("customer\_city",rs.getString("customer\_city"));

results.put("customer\_state",Integer.parseInt(rs.getString("customer\_state")));

results.put("customer\_pincode",rs.getString("customer\_pincode"));

results.put("customer\_id",rs.getString("customer\_id"));

count++;

}

if(count==0)

{

results.put("customer\_name","");

results.put("customer\_mobile","");

results.put("customer\_email", "");

results.put("customer\_password", "");

results.put("customer\_address", "");

results.put("customer\_city","");

results.put("customer\_state",0);

results.put("customer\_pincode","");

results.put("customer\_id","");

}

}

catch(Exception e)

System.out.println("Error is: "+e);

}

return results;

}

/// Update the Customer ////

public String updateCustomer(HashMap customerData)

String SQL = "UPDATE 'customer' SET 'customer\_name' = ?, 'customer\_mobile' = ?,'customer\_email '= ?, 'customer\_password' = ?, 'customer\_address' = ?, 'customer\_city' = ?,'customer\_state = ?, customer\_pincode = ? WHERE customer\_id' = ?;";

String error = "";

int record=0;

try

{

pstmt = connection.prepareStatement(SQL);

pstmt.setString(1,(String) customerData.get("customer\_name"));

pstmt.setString(2,(String) customerData.get("customer\_mobile"));

pstmt.setString(3,(String) customerData.get("customer\_email"));

pstmt.setString(4,(String) customerData.get("customer\_password"));

pstmt.setString(5,(String) customerData.get("customer\_address"));

pstmt.setString(6,(String) customerData.get("customer\_city"));

pstmt.setString(7.(String) customerData.get("customer\_state"));

pstmt.setString(8,(String) customerData.get("customer\_pincode"));

pstmt.setString(9.(String) customerData.get("customer\_id"));

record = pstmt.executeUpdate();

pstmt.close();

connection.close();

}

catch(Exception e)

{

StringWriter writer = new StringWriter();

Print Writer print Writer = new PrintWriter( writer);

e.printStackTrace(print Writer);

print Writer.flush();

String stackTrace = writer.toString();

error+="Error: "+stackTrace;

System.out.println("Error : "+e.toString());

}

return error;

}

//////////Function for getting all the Airport Details//////////

public ArrayList getAllCustomer()

{

String SQL = "SELECT \* FROM `customer'";

int count=0;

ArrayList resultArray = new ArrayList();

try

{

statement = connection.createStatement();

rs = statement.executeQuery(SQL);

while(rs.next()

{

HashMap results = new HashMap();

results.put("customer\_name",rs.getString("customer\_name"));

results.put("customer\_mobile",rs.getString("customer\_mobile"));

results.put("customer\_email",rs.getString("customer\_email"));

results.put("customer\_password",rs.getString("customer\_password"));

results.put("customer\_address",rs.getString("customer\_address"));

results.put("customer\_city",rs.getString("customer\_city"));

results.put("customer\_state", Integer.parseInt(rs.getString("customer\_state')));

results.put("customer\_pincode",rs.getString("customer\_pincode"));

results.put("customer\_id",rs.getString("customer\_id"));

count++;

resultArray.add(results);

}

}

catch(Exception e)

{

System.out.println("Error is: "+e);

}

return resultArray;

}

/////Function for Getting the List///////

public String getStateOption(Integer SellD)

{

int selectedID = SellD.intValue();

return

Connect.getOptionList("state","state\_id","state\_name","state\_id, state\_name", selectedID,"1");

}

|||||//Function for getting Login Details///////

public HashMap getLogin Details(String login\_user, String login\_password)

{

HashMap resultsArray = new HashMap();

int count=0;

try

{

String SQL = "SELECT \* FROM customer WHERE customer\_email = '"+login\_user+"' AND

customer\_password = '"+login\_password+""';

statement = connection.createStatement();

rs = statement.executeQuery(SQL);

while(rs.next())

{

resultsArray.put("customer\_id",rs.getString("customer\_id"));

resultsArray.put("customer\_name",rs.getString("customer\_name"));

resultsArray.put("login\_level",4);

count++;

}

if(count==0)

{

resultsArray.put("customer\_id","");

resultsArray.put("customer\_name","");

resultsArray.put("login\_level",0);

}

}

catch(Exception e)

{

System.out.println("Error is: "+e);

}

return resultsArray;

}

//////////Function for checking the existing username/////////

public int checkUsernameExits(String login\_user, int type)

{

Hash Map resultsArray = new HashMap();

int exits=0;

try

{

String SQL = "";

if(type == 1) {

SQL = "SELECT \* FROM customer WHERE customer\_email =

""+login\_user+"";

}

statement = connection.createStatement();

rs = statement.executeQuery(SQL);

while(rs.next())

{

exits++;

}

}

catch(Exception e)

{

System.out.println("Error is: "+e);

}

return exits;

}

////////Function for geting the Single Airport Details////////

public boolean checkLogin(String login\_user,String login\_password)

{

int count=0;

try

{

String SQL = "SELECT \* FROM customer WHERE customer\_email = '''+login\_user+''' AND

customer\_password = '"+login\_password+'"";

statement = connection.createStatement();

rs = statement.executeQuery(SQL);

while(rs.next()) count++;

}

catch(Exception e)

{

System.out.println("Error is: "+e);

}

if(count==0)

return false;

return true;

}

}

**Coding for Employee.java**

package Model;

import java.util.\*;

import java.sql.\*;

import com.\*;

import java.io.\*;

public class Employee extends Connect

{

/////Function for connect to the MySQL Server Database//////

public Employee()

{

Connect.connect\_mysql();

}

/////////Save User Details //////////

public String saveEmployee(HashMap employeeData)

{

String SQL = "INSERT INTO employee (employee\_sal, employee\_first\_name,employee\_middle\_name, employee\_last\_name, employee\_gender, employee\_address,employee\_village, employee\_state,employee\_country, employee\_landline, employee\_mobile,employee\_email, employee\_status, employee\_deparment, employee\_dob, employee\_nationalty,employee\_manager\_id, employee\_role) VALUES (?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?);";

int record=0,last\_inserted\_id=0;

String error = "";

try

{

pstmt = connection.prepareStatement(SQL, Statement.RETURN\_GENERATED\_KEYS);

pstmt.setString(1,(String) employeeData.get("employee\_sal"));

pstmt.setString(2,(String) employeeData.get("employee\_first\_name"));

pstmt.setString(3,(String) employeeData.get("employee\_middle\_name"));

pstmt.setString(4,(String) employeeData.get("employee\_last\_name"));

pstmt.setString(5,(String) employeeData.get("employee\_gender"));

pstmt.setString(6,(String) employeeData.get("employee\_address"));

pstmt.setString(7,(String) employeeData.get("employee\_village"));

pstmt.setString(8,(String) employeeData.get("employee\_state"));

pstmt.setString(9,(String) employee Data.get("employee\_country"));

pstmt.setString(10,(String) employeeData.get("employee\_landline"));

pstmt.setString(11,(String) employeeData.get("employee\_mobile"));

pstmt.setString(12,(String) employee Data.get("employee\_email"));

pstmt.setString(13,(String) employee Data.get("employee\_status"));

pstmt.setString(14,(String) employeeData.get("employee\_deparment"));

pstmt.setString(15,(String) employeeData.get("employee\_dob"));

pstmt.setString(16,(String) employeeData.get("employee\_nationalty'));

pstmt.setString(17,(String) employeeData.get("employee\_manager\_id"));

pstmt.setString(18,(String) employeeData.get("employee\_role"));

record = pstmt.executeUpdate();

/// Get the Last Insert ID ///

rs = pstmt.getGeneratedKeys();

if(rs.next())

{

last\_inserted\_id = rs.getInt(1);

}

pstmt.close();

}

catch(Exception e)

{

StringWriter writer = new StringWriter();

Print Writer print Writer = new Print Writer( writer);

e.printStackTrace print Writer);

print Writer.flush();

String stackTrace = writer.toString();

error+="Error : "+stackTrace;

System.out.println("Error : "+e.toString());

}

/// Save Credentials to Login ////

SQL = "INSERT INTO login (login\_emp\_id, login\_user, login\_password, login\_email, login\_level)

VALUES (?, ?, ?, ?, ?);";

try

{

pstmt = connection.prepareStatement(SQL);

pstmt.setInt(1,last\_inserted\_id);

pstmt.setString(2,(String) employeeData.get("employee\_user"));

pstmt.setString(3,(String) employeeData.get("employee\_password"));

pstmt.setString(4,(String) employeeData.get("employee\_email"));

pstmt.setString(5,(String) employeeData.get("employee\_role"));

record = pstmt.executeUpdate();

pstmt.close();

connection.close();

}

catch(Exception e)

{

String Writer writer = new StringWriter();

Print Writer print Writer = new Print Writer( writer);

e.printStackTrace( print Writer);

print Writer.flush();

String stackTrace = writer.toString();

error+="Error : "+stackTrace;

System.out.println("Error : "+e.toString());

}

return error;

}

///////Function for getting Users Details//////

public HashMap getEmployeeDetails(int employee\_id)

{

HashMap resultsArray = new HashMap();

int count=0;

try

{

String SQL = "SELECT \* FROM employee WHERE employee\_id = "+employee\_id;

statement = connection.createStatement();

rs = statement.executeQuery(SQL);

while(rs.next())

{

resultsArray.put("employee\_id",rs.getString("employee\_id"));

resultsArray.put("employee\_manager\_id", Integer.parseInt(rs.getString("employee\_manager\_id

resultsArray.put("employee\_role",Integer.parseInt(rs.getString("employee\_role")));

resultsArray.put("employee\_sal",Integer.parseInt(rs.getString("employee\_sal")));

resultsArray.put("employee\_first\_name",rs.getString("employee\_first\_name"));

resultsArray.put("employee\_middle\_name",rs.getString("employee\_middle\_name"));

resultsArray.put("employee\_last\_name",rs.getString("employee\_last\_name"));

resultsArray.put("employee\_gender",rs.getString("employee\_gender"));

resultsArray.put("employee\_address",rs.getString("employee\_address"));

resultsArray.put("employee\_village",rs.getString("employee\_village"));

resultsArray.put("employee\_state",Integer.parseInt(rs.getString("employee\_state')));

resultsArray.put("employee\_country",Integer.parseInt(rs.getString("employee\_country')));

resultsArray.put("employee\_landline",rs.getString("employee\_landline"));

resultsArray.put("employee\_mobile",rs.getString("employee\_mobile"));

resultsArray.put("employee\_email",rs.getString("employee\_email"));

resultsArray.put("employee\_status", Integer.parseInt(rs.getString("employee\_status")));

resultsArray.put("employee\_deparment",Integer.parseInt(rs.getString("employee\_deparment"))

);

resultsArray.put("employee\_dob",rs.getString("employee\_dob"));

resultsArray.put("employee\_nationalty",rs.getString("employee\_nationalty"));

count++;

}

if(count==0)

{

resultsArray.put("employee\_id","");

resultsArray.put("employee\_manager\_id",0);

resultsArray.put("employee\_role",0);

resultsArray.put("employee\_user", "");

Project Report of Farm Management System

resultsArray.put("employee\_sal",0);

resultsArray.put("employee\_first\_name", "");

resultsArray.put("employee\_middle\_name","");

resultsArray.put("employee\_last\_name","");

resultsArray.put("employee\_gender","");

resultsArray.put("employee\_address","");

resultsArray.put("employee\_village","");

resultsArray.put("employee\_state",0);

resultsArray.put("employee\_country",0);

resultsArray.put("employee\_landline","");

resultsArray.put("employee\_mobile","");

resultsArray.put("employee\_email","");

resultsArray.put("employee\_status",0);

resultsArray.put("employee\_deparment",0);

resultsArray.put("employee\_dob","");

resultsArray.put("employee\_nationalty","");

}

}

catch(Exception e)

{

System.out.println("Error is: "+e);

}

return resultsArray;

}

public String updateEmployee(HashMap employeeData)

String SQL = "UPDATE employee SET employee\_sal = ?, employee\_first\_name = ?,

employee\_middle\_name = ?, employee\_last\_name = ?, employee\_gender = ?, employee\_address = ?,

employee\_village = ?, employee\_state = ?, employee\_country = ?, employee\_landline = ?,

employee\_mobile = ?, employee\_email = ?, employee\_status = ?, employee\_deparment = ?,

employee\_dob = ?, employee\_nationalty = ?, employee\_manager\_id = ?, employee\_role = ? WHERE

employee\_id = ?";

String error = "";

int record=0;

try

{

pstmt = connection.prepareStatement(SQL);

pstmt.setString(1,(String) employeeData.get("employee\_sal"));

pstmt.setString(2,(String) employeeData.get("employee\_first\_name"));

pstmt.setString(3,(String) employeeData.get("employee\_middle\_name"));

pstmt.setString(4,(String) employeeData.get("employee\_last\_name"));

pstmt.setString(5,(String) employeeData.get("employee\_gender"));

pstmt.setString(6,(String) employeeData.get("employee\_address"));

pstmt.setString(7,(String) employeeData.get("employee\_village"));

pstmt.setString(8,(String) employeeData.get("employee\_state"));

pstmt.setString(9.(String) employeeData.get("employee\_country"));

pstmt.setString(10,(String) employeeData.get("employee\_landline"));

pstmt.setString(11,(String) employeeData.get("employee\_mobile"));

pstmt.setString(12,(String) employeeData.get("employee\_email"));

pstmt.setString(13,(String) employeeData.get("employee\_status"));

pstmt.setString(14,(String) employeeData.get("employee\_deparment"));

pstmt.setString(15,(String) employeeData.get("employee\_dob"));

pstmt.setString(16,(String) employeeData.get("employee\_nationalty"));

pstmt.setString(17,(String) employeeData.get("employee\_manager\_id"));

pstmt.setString(18,(String) employeeData.get("employee\_role"));

pstmt.setString(19,(String) employeeData.get("employee\_id"));

record = pstmt.execute Update();

pstmt.close();

connection.close();

}

catch(Exception e)

{

StringWriter writer = new StringWriter();

PrintWriter printWriter = new PrintWriter( writer );

e.printStackTrace(printWriter );

printWriter.flush();

String stackTrace = writer.toString();

error+="Error: "+stackTrace;

System.out.println("Error : "+e.toString());

}

return error;

}

public boolean delete Employee(int employee\_id)

{

return true;

}

/////Function for Getting the List//////////

public String getStateOption(Integer SellD)

{

int selectedID = Selld.intValue();

return

Connect.getOptionList("state", "state\_id","state\_name","state\_id, state\_name", selectedID,"1");

}

/////Function for Getting the List///////

public String getEmployeeOption(Integer SellD)

{

int selectedID = Selid.intValue();

return

Connect.getOptionList("employee","employee\_id","employee\_id","employee\_id, employee\_id", selected

ID, "1");

}

/////Function for Getting the List/|||||||||

public String getRoleOption(Integer SellD)

{

int selectedID = SellD.intValue();

return Connect.getOptionList("roles","role\_id","role\_title","role\_id, role\_title", selectedID,"1");

}

/////Function for Getting the List///////////

public String getCountryOption(Integer SellD)

{

int selectedID = SellD.intValue();

return

Connect.getOptionList("country","country\_id","country\_name","country\_id, country\_name" selectedID,

"1");

}

III//Function for Getting the List//11

public String getSalution Option(Integer Sello)

{

int selectedID = Selld.intValue();

return Connect.getOptionList("salutions","sl\_id","sl\_name","sl\_id,sl\_name", selectedID,"1");

}

/////Function for Getting the List////

public String getStatusOption(Integer SellD)

{

int selectedID = Selid.intValue();

return

Connect.getOptionList("status","status\_id","status\_name","status\_id,status\_name", selectedID, "1");

}

/////Function for Getting the List//////

public String getDepartmentOption(Integer SellD)

{

int selectedID = SellD.intValue();

return

Connect.getOptionList("department","dept\_id","dept\_name","dept\_id,dept\_name", selectedID, "1");

}

///Function for getting all the Airport Details//

public ArrayList getAllEmployee(String managerlD)

int count=0;

String error = "";

String SQL = "SELECT \* FROM employee";

ArrayList resultArray = new ArrayList();

try

{

if(!managerID.equals("0"))

{

SQL = "SELECT \* FROM employee WHERE employee\_manager\_id =

"+managerlD;

}

statement = connection.createStatement();

rs = statement.executeQuery(SQL);

while(rs.next())

{

HashMap results = new HashMap();

results.put("employee\_id",rs.getString("employee\_id"));

results.put("employee\_role",rs.getString("employee\_role"));

results.put("employee\_sal",rs.getString("employee\_sal"));

results.put("employee\_first\_name",rs.getString("employee\_first\_name"));

results.put("employee\_middle\_name",rs.getString("employee\_middle\_name"));

results.put("employee\_last\_name",rs.getString("employee\_last\_name"));

results.put("employee\_gender",rs.getString("employee\_gender"));

results.put("employee\_address",rs.getString("employee\_address"));

results.put("employee\_village",rs.getString("employee\_village"));

results.put("employee\_state",rs.getString("employee\_state"));

results.put("employee\_country",rs.getString("employee\_country"));

results.put("employee\_landline",rs.getString("employee\_landline"));

results.put("employee\_mobile",rs.getString("employee\_mobile"));

results.put("employee\_email",rs.getString("employee\_email"));

results.put("employee\_status", Integer.parseInt(rs.getString("employee\_status"));

results.put("employee\_deparment", Integer.parseInt(rs.getString("employee\_deparment"));

results.put("employee\_dob",rs.getString("employee\_dob"));

results.put("employee\_nationalty",rs.getString("employee\_nationalty"));

results.put("employee\_manager\_id", Integer.parseInt(rs.getString("employee\_manager\_id"));

count++;

resultArray.add(results);

}

}

catch(Exception e)

{

StringWriter writer = new StringWriter();

Print Writer printWriter = new Print Writer( writer);

e.printStackTracel print Writer);

printWriter.flush();

String stackTrace = writer.toString();

error+="Error : "+stackTrace:

System.out.println("Error : "+e.toString());

}

return resultArray;

}

public String getDepartment(int dept\_id)

HashMap results = new HashMap();

String SQL = "";

String value = "";

int count=0;

try

{

SQL = "SELECT dept\_name FROM department WHERE dept\_id = "+dept\_id;

statement = connection.createStatement();

rs = statement.executeQuery(SQL);

while(rs.next())

{

value = rs.getString("dept\_name");

}

}

catch(Exception e)

{

System.out.println("Error is: "+e);

}

return value;

}

public String getStatus(int status\_id)

{

HashMap results = new HashMap();

String SQL = "";

String value = "";

int count=0;

try

{

SQL = "SELECT status name FROM status WHERE status\_id = "+status\_id;

statement = connection.createStatement();

rs = statement.executeQuery(SQL);

while(rs.next())

value = rs.getString("status\_name");

}

}

catch(Exception e)

System.out.println("Error is: "+e);

}

return value;

}

}

**Coding for Equipment.java**

package Model;

import java.util.\*;

import java.sql.\*;

import com.\*;

import java.io.\*;

public class Equipment extends Connect

{

/////Function for connect to the MySQL Server Database/////

public Equipment()

{

Connect.connect\_mysql();

}

//Save User Details /////

public String saveEquipment(HashMap equipmentData)

{

String SQL = "INSERT INTO equipment' ('equipment\_title', 'equipment\_cost\_range',

'equipment\_description', 'equipment\_image') VALUES (?, ?, ?, ?);";

int record=0;

String error = "";

try

{

pstmt = connection.prepareStatement(SQL);

pstmt.setString(1,(String) equipmentData.get("equipment\_title"));

pstmt.setString(2,(String) equipmentData.get("equipment\_cost\_range"));

pstmt.setString(3,(String) equipmentData.get("equipment\_description"));

pstmt.setString(4,(String) equipmentData.get("equipment\_image"));

record = pstmt.execute Update();

pstmt.close();

connection.close();

}

catch(Exception e)

{

StringWriter writer = new StringWriter();

Print Writer print Writer = new Print Writer( writer);

e.printStackTrace(printWriter );

printWriter.flush();

String stackTrace = writer.toString();

error+="Error: "+stackTrace;

System.out.println("Error : "+e.toString());

}

return error;

}

///Function for getting Users Details//////////

public HashMap getEquipment Details(int equipment\_id)

{

Hash Map results = new HashMap();

int count=0;

try

{

String SQL = "SELECT \* FROM 'equipment' WHERE equipment\_id ="+equipment\_id;

statement = connection.createStatement();

rs = statement.executeQuery(SQL);

while(rs.next())

{

results.put("equipment\_title",rs.getString("equipment\_title"));

results.put("equipment\_cost\_range",rs.getString("equipment\_cost\_range"));

results.put("equipment\_description",rs.getString("equipment\_description"));

results.put("equipment\_image",rs.getString("equipment\_image"));

results.put("equipment\_id",rs.getString("equipment\_id"));

count++;

}

if(count==0)

{

results.put("equipment\_title","");

results.put("equipment\_cost\_range","");

results.put("equipment\_description","");

results.put("equipment\_image","");

results.put('equipment\_id","");

}

}

catch(Exception e)

{

System.out.println("Error is: "+e);

}

return results;

}

/// Update the Equipment ////

public String update Equipment(HashMap equipmentData)

{

String SQL = "UPDATE 'equipment SET 'equipment\_title' = ?, equipment\_cost\_range" =?, 'equipment\_description' = ?, 'equipment\_image' = ? WHERE 'equipment\_id' = ?;":

String error = "";

int record=0;

try

{

pstmt = connection.prepareStatement(SQL);

pstmt.setString(1,(String) equipmentData.get("equipment\_title"));

pstmt.setString(2,(String) equipmentData.get("equipment\_cost\_range"));

pstmt.setString(3,(String) equipmentData.get("equipment\_description"));

pstmt.setString(4,(String) equipmentData.get("equipment\_image"));

pstmt.setString(5,(String) equipmentData.get("equipment\_id"));

record = pstmt.execute Update();

pstmt.close();

connection.close();

}

catch(Exception e)

{

StringWriter writer = new StringWriter();

PrintWriter print Writer = new PrintWriter( writer );

e.printStackTrace( print Writer);

printWriter.flush();

String stackTrace = writer.toString();

error+="Error : "+stackTrace;

System.out.println("Error : "+e.toString());

}

return error;

}

/////////Function for getting all the Airport Details//////////

public ArrayList getAllEquipment(String title)

{

int count=0;

String SQL = "";

SQL = "SELECT \* FROM equipment";

if(title != "") {

SQL = "SELECT \* FROM 'equipment' WHERE equipment\_title LIKE '%"+title+"%'

OR equipment\_description LIKE '%"+title+"%"',

}

ArrayList resultArray = new ArrayList();

try

{

statement = connection.createStatement();

rs = statement.executeQuery(SQL);

while(rs.next())

{

HashMap results = new HashMap();

results.put("equipment\_title",rs.getString("equipment\_title"));

results.put("equipment\_cost\_range",rs.getString("equipment\_cost\_range"));

results.put("equipment\_description",rs.getString("equipment\_description"));

results.put("equipment\_image",rs.getString("equipment\_image"));

results.put("equipment\_id",rs.getString("equipment\_id"));

count++;

resultArray.add(results);

}

count++;

resultArray.add(results);

}

}

catch(Exception e)

{

System.out.println("Error is: "+e);

}

return resultArray;

}

**Coding for Insecticides.java**

package Model;

import java.util.\*;

import java.sql.\*;

import com.\*;

import java.io.\*;

public class Insecticides extends Connect

{

/////Function for connect to the MySQL Server Database////////

public Insecticides()

{

Connect.connect\_mysql();

}

//// Save User Details /////

public String savelnsecticides(HashMap insecticidesData)

{

String SQL = "INSERT INTO `insecticides' ('insecticides\_title', 'insecticides\_cost\_range','insecticides\_description', 'insecticides\_image') VALUES (?, ?, ?, ?);";

int record=0;

String error = "";

try

{

pstmt = connection.prepareStatement(SQL);

pstmt.setString(1,(String) insecticidesData.get("insecticides\_title"));

pstmt.setString(2,(String) insecticidesData.get("insecticides\_cost\_range"));

pstmt.setString(3,(String) insecticides Data.get("insecticides\_description"));

pstmt.setString(4,(String) insecticidesData.get("insecticides\_image"));

record = pstmt.executeUpdate();

pstmt.close();

connection.close();

}

catch(Exception e)

{

StringWriter writer = new String Writer();

PrintWriter print Writer = new PrintWriter( writer);

e.printStackTrace(print Writer );

printWriter.flush();

String stackTrace = writer.toString();

error+="Error : "+stackTrace;

System.out.println("Error : "+e.toString();

}

return error;

}

//Function for getting Users Details//////////

public HashMap getinsecticides Details(int insecticides\_id)

{

HashMap results = new HashMap();

int count=0;

try

{

String SQL = "SELECT \* FROM 'insecticides' WHERE insecticides\_id ="+insecticides\_id;

statement = connection.createStatement();

rs = statement.executeQuery(SQL);

while(rs.next())

{

results.put("insecticides\_title",rs.getString("insecticides\_title"));

results.put("insecticides\_cost\_range",rs.getString("insecticides\_cost\_range"));

results.put("insecticides\_description",rs.getString("insecticides\_description"));

results.put("insecticides\_image",rs.getString("insecticides\_image"));

results.put("insecticides\_id",rs.getString("insecticides\_id"));

count++;

}

if(count==0)

results.put("insecticides\_title","");

results.put("insecticides\_cost\_range","");

results.put("insecticides\_description","");

results.put("insecticides\_image","");

results.put("insecticides\_id","");

}

}

catch(Exception e)

System.out.println("Error is: "+e);

}

return results;

}

/// Update the Insecticides ////

public String updateInsecticides(HashMap insecticides Data)

String SQL = "UPDATE 'insecticides' SET 'insecticides\_title' = ?, 'insecticides\_cost\_range'= ?, 'insecticides\_description' = ?, 'insecticides\_image' = ? WHERE `insecticides\_id' = ?;";

String error = ";

int record=0;

try

{

pstmt = connection.prepareStatement(SQL);

pstmt.setString(1,(String) insecticides Data.get("insecticides\_title"));

pstmt.setString(2,(String) insecticidesData.get("insecticides\_cost\_range"));

pstmt.setString(3,(String) insecticides Data.get("insecticides\_description"));

pstmt.setString(4.(String) insecticidesData.get("insecticides\_image"));

pstmt.setString(5,(String) insecticidesData.get("insecticides\_id"));

record = pstmt.execute Update();

pstmt.close();

connection.close();

}

catch(Exception e)

{

StringWriter writer = new StringWriter();

PrintWriter printWriter = new PrintWriter( writer);

e.printStackTrace( printWriter );

print Writer.flush();

String stackTrace = writer.toString();

error+="Error: "+stackTrace;

System.out.println("Error : "+e.toString());

}

return error;

}

///////Function for getting all the Airport Details//

public ArrayList getAllinsecticides(String title)

{

int count=0;

String SQL = "";

SQL = "SELECT \* FROM 'insecticides";

if(title != "") {

SQL = "SELECT \* FROM 'insecticides' WHERE insecticides\_title LIKE '%"+title+"% ' OR insecticides\_description LIKE '%"+title+"%'";

}

ArrayList resultArray = new ArrayList();

try

{

statement = connection.createStatement();

rs = statement.executeQuery(SQL);

while(rs.next())

{

HashMap results = new HashMap();

results.put("insecticides\_title",rs.getString("insecticides\_title"));

results.put("insecticides\_cost\_range",rs.getString("insecticides\_cost\_range"));

results.put("insecticides\_description",rs.getString("insecticides\_description"));

results.put("insecticides\_image",rs.getString("insecticides\_image"));

results.put("insecticides\_id",rs.getString("insecticides\_id"));

count++;

resultArray.add(results);

}

}

catch(Exception e)

{

System.out.println("Error is: "+e);

}

return resultArray;

}

}

**Coding for Login.java**

package Model;

import java.util.\*;

import java.sql.\*;

import com.\*;

import java.io.\*;

public class Login extends Connect

{

/////Function for connect to the MySQL Server Database//

public Login()

{

Connect.connect\_mysql();

}

////////Function for Update the airport/////////////

public boolean change Password(String old\_password, String new\_password, int login\_id)

{

String SQL;

int count = 0;

try

{

FROM login WHERE login\_password = ''+old\_password+" AND

SQL = "SELECT login\_id = "+login\_id;

statement = connection.createStatement();

rs = statement.executeQuery(SQL);

while(rs.next())

count++;

if(count==1)

{

SQL = "UPDATE login SET login\_password=? WHERE login\_id=?";

int record=0;

pstmt = connection.prepareStatement(SQL);

pstmt.setString(1,new\_password);

pstmt.setint(2, login\_id);

record = pstmt.execute Update();

pstmt.close();

connection.close();

}

}

catch(Exception e)

{

System.out.println("Error : "+e.toString());

}

if(count==0)

return false;

return true;

}

///////Function for geting the Single Airport Details////////

public boolean checkLogin(String login\_user,String login\_password)

{

int count=0;

try

{

String SQL = "SELECT \* FROM login WHERE login\_user='+login\_user+'" AND login\_password =

"+login\_password+"",

statement = connection.createStatement();

rs = statement.executeQuery(SQL);

while(rs.next()) count++;

}

catch(Exception e)

{

System.out.println("Error is: "+e);

}

if(count==0)

return false;

return true;

}

//////////Function for getting Login Details////////

public HashMap getLoginDetails(String login\_user, String login\_password)

{

HashMap resultsArray = new HashMap();

int count=0;

try

{

String SQL = "SELECT \* FROM login WHERE login\_user = "+login\_user+" AND login\_password =

+login\_password+"";

statement = connection.createStatement();

rs = statement.executeQuery(SQL);

while(rs.next())

{

resultsArray.put("login\_id",rs.getString("login\_id"));

resultsArray.put("login\_emp\_id",rs.getString("login\_emp\_id"));

resultsArray.put("login\_user",rs.getString("login\_user"));

resultsArray.put("login\_level",rs.getString("login\_level"));

count++;

}

if(count==0)

{

resultsArray.put("login\_id","");

resultsArray.put("login\_emp\_id", "");

resultsArray.put("login\_user","");

resultsArray.put("login\_level", "");

}

}

catch(Exception e)

{

System.out.println("Error is: "+e);

}

return resultsArray;

}

///Function for getting Login Details////////

public int checkUsernameExits(String login\_user, int type)

{

HashMap resultsArray = new HashMap();

int exits=0;

try

{

String SQL = "";

if(type == 1) {

SQL = "SELECT \* FROM login WHERE login\_user="'login\_user+""';

}

if(type == 2) {

SQL = "SELECT \* FROM login WHERE login\_email = "+login\_user+""";

}

statement = connection.createStatement();

rs = statement.executeQuery(SQL);

while(rs.next())

{

exits++;

}

}

catch(Exception e)

{

System.out.println("Error is: "+e);

}

return exits;

}

**Coding for Prestcides.java**

package Model;

import java.util.\*;

import java.sql.\*;

import com.;

import java.io.\*;

public class Pesticides extends Connect

//////////Function for connect to the MySQL Server Database///////////

public Pesticides()

{

Connect.connect\_mysql();

)

///////Save User Details /////

public String savePesticides(HashMap pesticides Data)

{

String SQL = "INSERT INTO 'pesticides' ('pesticides\_title', 'pesticides\_cost\_range','pesticides\_description', 'pesticides\_image') VALUES (?, ?, ?, ?);";

int record=0;

String error = "";

try

{

pstmt = connection.prepareStatement(SQL);

pstmt.setString(1,(String) pesticidesData.get("pesticides\_title"));

pstmt.setString(2,(String) pesticidesData.get("pesticides\_cost\_range"));

pstmt.setString(3,(String) pesticidesData.get("pesticides\_description"));

pstmt.setString(4,(String) pesticidesData.get("pesticides\_image"));

record = pstmt.execute Update();

pstmt.close();

connection.close();

}

catch(Exception e)

{

StringWriter writer = new StringWriter();

PrintWriter printWriter = new PrintWriter writer();

e.printStackTrace(printWriter );

printWriter.flush();

String stackTrace = writer.toString();

error+="Error: "+stackTrace;

System.out.println("Error: "+e.toString();

}

return error;

}

/////////Function for getting Users Details/////////

public HashMap getPesticides Details(int pesticides\_id)

{

HashMap results = new HashMap();

int count=0;

try

{

String SQL = "SELECT \* FROM ' pesticides 'WHERE pesticides\_id = "+pesticides\_id

statement = connection.createStatement();

rs = statement.executeQuery(SQL);

while(rs.next())

results.put("pesticides\_title",rs.getString("pesticides\_title"));

results.put("pesticides\_cost\_range",rs.getString("pesticides\_cost\_range"));

results.put("pesticides\_description",rs.getString("pesticides\_description"));

results.put("pesticides\_image",rs.getString("pesticides\_image"));

results.put("pesticides\_id",rs.getString("pesticides\_id"));

count++;

}

if(count==0)

{

results.put("pesticides\_title","");

results.put("pesticides\_cost\_range", "");

results.put("pesticides\_description","");

results.put("pesticides\_image","");

results.put("pesticides\_id","");

}

catch(Exception e)

{

System.out.println("Error is: "+e);

}

return results;

}

/// Update the Pesticides ////

public String update Pesticides (HashMap pesticides Data)

{

String SQL = "UPDATE pesticides SET 'pesticides\_title' = ?, 'pesticides\_cost\_range' = ?,

pesticides\_description = ?, 'pesticides\_image" = ? WHERE pesticides\_id' = ?;";

String error = "";

int record=0;

try

{

pstmt = connection.prepare Statement(SQL);

pstmt.setString(1,(String) pesticides Data.get("pesticides\_title"));

pstmt.setString(2,(String) pesticides Data.get("pesticides\_cost\_range"));

pstmt.setString(3,(String) pesticides Data.get("pesticides\_description"));

pstmt.setString(4,(String) pesticides Data.get("pesticides\_image"));

pstmt.setString(5,(String) pesticides Data.get("pesticides\_id"));

record = pstmt.execute Update();

pstmt.close();

connection.close();

}

catch(Exception e)

{

StringWriter writer = new StringWriter();

PrintWriter printWriter = new PrintWriter( writer);

e.printStackTrace(print Writer);

printWriter.flush();

String stackTrace = writer.toString();

error+="Error : "+stackTrace;

System.out.println("Error : "+e.toString());

}

return error;

}

////////////////Function for getting all the Airport Details///////////////

public ArrayList getAllPesticides(String title)

{

int count=0;

String SQL = "";

SQL = "SELECT \* FROM `pesticides";

if(title != "")

{

SQL = "SELECT \* FROM `pesticides' WHERE 'pesticides\_title' LIKE '%"+title+"%' OR 'pesticides\_description LIKE '%"title+"%"';

}

ArrayList resultArray = new ArrayList();

try

{

statement = connection.createStatement();

rs = statement.executeQuery(SQL);

while(rs.next()

Hash Map results = new Hash Map();

results.put("pesticides\_title",rs.getString("pesticides\_title"));

results.put("pesticides\_cost\_range",rs.getString("pesticides\_cost\_range"));

results.put("pesticides\_description",rs.getString("pesticides\_description"));

results.put("pesticides\_image",rs.getString("pesticides\_image"));

results.put("pesticides\_id",rs.getString("pesticides\_id"));

count++;

resultArray.add(results);

}

}

catch(Exception e)

{

System.out.println("Error is: "+e);

}

return resultArray;

}

}

**Coding for Crops.jsp**

<%@ include file="includes/header.jsp" %>

<%@ page import="java.util.\*"%>

<%@ page import="Model.\*"%>

Crops crops Details = new Crops();

String crops\_title = "";

if(request.getParameterMap().containsKey("crops\_title")) {

crops\_title = request.getParameter("crops\_title").toString();

}

ArrayList allCrops = crops Details.getAllCrops(crops\_title);

%>

<div class="wrapper row3">

<div class="rounded">

<main class="container clear">

<!-- main body -->

<!--#####################################################-->

<div class="group btmspace-30">

<!-- Middle Column -->

<div class="one\_half" style="width:68%">

<!--#####################################################-->

<h2>All Cropss</h2>

<form action="Crop.jsp" enctype="multipart/form-data">

<table>

<tr>

<td>Search Tearm : </td>

<td><input type="text" name="crops\_title" id="crops\_title" size="22" style="width:300px;"></td>

<td><input name="submit" type="submit" value="Search Crops"></td>

</tr>

</table>

</form>

<ul class="nospace listing" style="padding:1px 1px; margin:1px 1px">

<% for(int i=0;i<allCrops.size();i++)

{

HashMap Crops Details = new HashMap();

Crops Details = (HashMap)allCrops.get(i);

%>

<li class="clear" style="padding:1px 1px; margin:1px 1px">

<div class="img borderedbox"><a href="#"><img src="cropsImages/<%out.print(Crops Details.get("crops\_image")); %>" style="height:80px; width:80px;"></a></div>

<p class="nospace btmspace-15"><a href="#" style="text-decoration:underline;font-

weight:bold; font-size:15px;"><%=Crops Details.get("crops\_title")%></a></p>

<p style="color:#000000; margin-top:-12px;"><b>Month</b>:

<%=Crops Details.get("crops\_month")%></p>

<p style="color:#000000; margin-top:-12px;"><%=Crops Details.get("crops\_description")%></p>

</li>

}

%>

</ul>

<!--#####################################################-->

</div>

<!-- / Middle Column -->

<div style="float: right">

<div><img src="images/save\_1.jpg" style="width: 250px"></div><br>

<div><img src="images/save\_2.jpg" style="width: 250px"></div>

</div>

</div>

<!-- / main body -->

<div class="clear"></div>

</main>

</div>

</div>

<%@ include file="includes/footer.jsp" %>

**Coding for Crops.jsp**

<%@ include file="includes/header.jsp" %>

<%@ page import="java.util.\*"%>

<%@ page import="Model.\*" %>

Crops crops Details = new Crops();

int crops\_id = Integer.parseInt(request.getParameter ("crops\_id"));

Hash Map Values = crops Details.getCropsDetails(crops\_id);

%>

<script>

function validimage() {

var val = $("#crops\_image").val();

var id = $("#crops\_id").val();

if(id == " || val != ")

{

if(val == ") {

alert('Choose the Crops Image");

return false;

}

if (!val.match(/(?:gif|jpg| png |bmp)$/)) {

// inputted file path is not an image of one of the above types

alert("inputted file path is not an image!");

return false;

}

}

return true;

}

</script>

<div class="wrapper row3">

<div class="rounded">

<main class="container clear">

<!-- main body -->

<div id="comments" style="width: 70%; float:left; margin-right:30px">

<h2>Crops Form</h2>

<form method="post" action="Upload Crops" enctype="multipart/form-data"

onsubmit="return validimage()">

<div class="half\_width">

<label for="email"> Crops Name<span>\*</span></label>

<input type="text" name="crops\_title" id="crops\_title" value="<%

out.print(Values.get("crops\_title")); %>" size="22" style="width:300px;" required>

</div>

<div class="half\_width">

<label for="email">Month<span>\*</span></label>

<input type="text" name="crops\_month" id="crops\_month" value="<%

out.print(Values.get("crops\_month")); %>" size="22" style="width:300px;" required>

</div>

<div class="half\_width">

<label for="email"> Crops Image<span>\*</span></label>

<input type="file" name="crops\_image" id="crops\_image" style="width:300px">

</div>

<div class="half\_width">

<label for="email">Description<span>\*</span></label>

<textarea style="width:300px; height:100px;" name="crops\_description"

required><% out.print(Values.get("crops\_description")); %></textarea>

</div>

<% if(!Values.get("crops\_image").equals("")) { %>

<div class="half\_width">

<img src="cropsImages/<% out.print(Values.get("crops\_image")); %>"

style="height:100px; width:100px;">

</div>

<% } %>

<div class="block clear"></div>

<div>

<input name="submit" type="submit" value="Save Crops">

&nbsp;

<input name="reset" type="reset" value="Reset Form">

</div>

<input type="hidden" name="image\_name" value="<%

out.print(Values.get("crops\_image")); %>" />

<input type="hidden" name="act" value="Save" />

<input type="hidden" name="type\_id" value="0" />

<input type="hidden" id="crops\_id" name="crops\_id" value="<%

out.print(Values.get("crops\_id")); %>"/>

</form>

</div>

<div style="float: left">

<div><img src="images/save\_1.jpg" style="width: 250px"></div><br>

<div><img src="images/save\_2.jpg" style="width: 250px"></div>

</div>

<div class="clear"></div>

</main>

</div>

</div>

<%@ include file="includes/footer.jsp" %>

**Coding for Customer.jsp**

<%@ include file="includes/header.jsp" %>

<%@ page import="java.util.\*"%>

<%@ page import="Model.\*" %>

<%

Customer customer Details = new Customer();

int customer\_id = Integer.parseInt(request.getParameter ("customer\_id"));

HashMap Values = customer Details.getCustomer Details(customer\_id);

%>

<div class="wrapper row3">

<div class="rounded">

<% if(request.getParameter("msg") != null) { %>

<div class="msg"><%=request.getParameter("msg") %></div>

<%} %>

<main class="container clear">

<!-- main body -->

<div id="comments" style="width: 70%; float:left; margin-right:30px">

<h2>Registration Form</h2>

<h4>Personal Details</h4>

<form method="post" action="model/customer.jsp">

<div class="half\_width">

<label for="email>Name<span>\*</span></label>

<input type="text" name="customer\_name" id="customer\_name" value="<%

out.print(Values.get("customer\_name')); %>" size="22" style="width:300px;" required>

</div>

<div class="half\_width">

<label for="email">Mobile<span>\*</span></label>

<input type="text" name="customer\_mobile" id="customer\_mobile" value="<%

out.print(Values.get("customer\_mobile")); %>" size="22" style="width:300px;" required>

</div>

<div id="password\_row">

<div class="half\_width">

<label for="email">Passwordkspan>\*</span></label>

<input type="password" name="customer\_password"

id="customer\_password" value="<% out.print(Values.get("customer\_password")); %>" size="22"

style="width:300px;" required>

</div>

<div class="half\_width">

<label for="email">Confirm Password<span> </span></label>

<input type="password" name="customer\_confirm\_password"

id="customer\_confirm\_password" value="<% out.print(Values.get("customer\_password")); %>"

size="22" style="width:300px;" required>

</div>

</div>

<div class="half\_width">

<label for="email">Email<span>\*</span></label>

<input type="text" name="customer\_email" id="customer\_email" value="<%

out.print(Values.get("customer\_email")); %>" size="22" style="width:300px;" required>

</div>

<div style="clear:both"></div>

<h4>Address Details</h4>

<div class="half\_width">

<label for="email">City<span>\*</span></label>

<input type="text" name="customer\_city" id="customer\_city" value="<%

out.print(Values.get("customer\_city")); %>" size="22" style="width:300px;" required>

</div>

<div class="half\_width">

<label for="url">State</label>

<select style="height: 40px; width:300px" name="customer\_state" required>

<% out.print(customer Details.getState Option((Integer)

Values.get("customer\_state"))); %>

</select>

</div>

<div>

<label for="email">Full Address<span></span></label>

<textarea style="width:300px; height:100px;" name="customer\_address" required><%

out.print(Values.get("customer\_address")); %></textarea>

</div>

<div>

<label for="email">Pincode<span>\*</span></label>

<input type="text" name="customer\_pincode" id="customer\_pincode" value="<%

out.print(Values.get("customer\_pincode")); %>" size="22" style="width:300px;" required>

</div>

<div class="block clear"></div>

<div>

<input name="submit" type="submit" value="Register">

&nbsp;

<input name="reset" type="reset" value="Reset Form">

</div>

<input type="hidden" name="image\_name" value="<%

out.print(Values.get("customer\_image")); %>" />

<input type="hidden" name="act" value="Save" />

<input type="hidden" id="customer\_id" name="customer\_id" value="<%

out.print(Values.get("customer\_id")); %>"/>

</form>

</div>

<div style="float: left">

<div><img src="images/save\_1.jpg" style="width: 250px"></div><br>

<div><img src="images/save\_2.jpg" style="width: 250px"></div>

</div>

<div class="clear"></div>

</main>

</div>

</div>

<% if(session.getAttribute("login\_level") != null && session.getAttribute("login\_level").equals("1")) { %>

<script>

jQuery('#password\_row').hide();

</script>

<%} %>

<%@ include file="includes/footer.jsp" %>

**Coding for Employee.jsp**

<%@ include file="includes/header.jsp" %>

<%@ page import="java.util.\*"%>

<%@ page import="Model.\*"%>

<%

Employee employeeDetails = new Employee();

int employee\_id = Integer.parseInt(request.getParameter ("employee\_id"));

String male = "";

String female = "";

HashMap Values = employee Details.getEmployeeDetails employee\_id);

if(Values.get("employee\_gender").equals("Male"))

{

male = "Selected";

}

else if(Values.get("employee\_gender").equals("Female"))

{

female = "Selected";

}

%>

<script>

jQuery(function() {

jQuery( "#employee\_dob").datepicker({

changeMonth: true,

change Year:true,

yearRange: "-50:-18",

dateFormat: 'dd ,MM, yy'

});

});

</script>

<div class="wrapper row3">

<div class="rounded">

<main class="container clear">

<!-- main body -->

<div id="comments" style="width: 70%; float:left; margin-right:30px">

<h2>Registration Form</h2>

<% if(request.getParameter("msg") != null) { %>

<div class="msg"><%=request.getParameter("msg") %></div>

<%} %>

<form action="model/employee.jsp" method="post">

<div class="one\_third first">

<label for="email">Salution <span>\*</span></label>

<select style="height: 40px; width:200px" name="employee\_sal" required>

<% out.print(employeeDetails.getSalution Option((Integer) Values.get("employee\_sal")); %>

</select>

</div>

<div class="one\_third">

<label for="email">Employee Status<span>\*</span></label>

<select style="height: 40px; width:200px" name="employee\_status" id = "employee\_status"

required><% out.print(employee Details.getStatusOption((Integer)

Values.get("employee\_status"))); %>

</select>

</div>

<div class="one\_third">

<label for="email">Department<span>\*</span></label>

<select style="height: 40px; width:200px" name = "employee\_deparment" id =

"employee\_deparment" required>

<% out.print(employeeDetails.getDepartment Option((Integer)

Values.get("employee\_deparment"))); %>

</select>

</div>

<div class="one\_third first">

<label for="email">First Name <span>\*</span></label>

<input type="text" name="employee\_first\_name" id="employee\_first\_name" value="<%

out.print(Values.get("employee\_first\_name")); %>" size="22" required>

</div>

<div class="one\_third">

<label for="email">Middle Name <span>\*</span></label>

<input type="text" name="employee\_middle\_name" id="employee\_middle\_name" value="<%

out.print(Values.get("employee\_middle\_name")); %>" size="22" required>

</div>

<div class="one\_third">

<label for="url">Last Name</label>

<input type="text" name="employee\_last\_name" id="employee\_last\_name" value="<%

out.print(Values.get("employee\_last\_name")); %>" size="22" required>

</div>

<div class="one\_third first">

<label for="url">Gender</label>

<select style="height: 40px; width:200px" name="employee\_gender" required>

<option value="0">Please Select</option>

<option value="Male" <% out.print(male); %>>Male</option>

<option value="Female" <% out.print(female); %>>Female</option>

</select>

</div>

<div class="one\_third">

<label for="url">Date of Birth</label>

<input type="text" name="employee\_dob" id="employee\_dob" value="<%

out.print(Values.get("employee\_dob")); %>" size="22" required>

</div>

<div class="one\_third">

<label for="url">Nationality</label>

<input type="text" name="employee\_nationalty" id="employee\_nationalty" value="<%

out.print(Values.get("employee\_nationalty"); %>" size="22" required>

</div>

<div class="one\_third first">

<label for="url">E-mail</label>

<input type="text" name="employee\_email" id="employee\_email" value="<%

out.print(Values.get("employee\_email")); %>" size="22" required>

</div>

<div class="one\_third">

<label for="url">Landline</label>

<input type="text" name="employee\_landline" id="employee\_landline" value="<%

out.print(Values.get("employee\_landline")); %>" size="22" required>

</div>

<div class="one\_third">

<label for="url">Mobile</label>

<input type="text" name="employee\_mobile" id="employee\_mobile" value="<%

out.print(Values.get("employee\_mobile")); %>" size="22" required>

</div>

<div style="clear:both"></div>

<h4>Employee Address Details</h4>

<div class="one\_third first">

<label for="url">Address</label>

<input type="text" name="employee\_address" id="employee\_address" value="<%

out.print(Values.get("employee\_address")); %>" size="22" required>

</div>

<div class="one\_third">

<label for="url">Village/City/Town</label>

<input type="text" name="employee\_village" id="employee\_village" value="<%

out.print(Values.get("employee\_village")); %>" size="22" required>

</div>

<div class="one\_third">

<label for="url">State</label>

<select style="height: 40px; width:200px" name="employee\_state" required>

<% out.print(employeeDetails.getStateOption((Integer)

Values.get("employee\_state"))); %>

</select>

</div>

<div class="one\_third first">

<label for="url">Country</label>

<select style="height: 40px; width:200px" name="employee\_country" required>

<% out.print(employeeDetails.getCountryOption((Integer)

Values.get("employee\_country'))); %>

</select>

</div>

<% if(session.getAttribute("login\_level") != null && session.getAttribute("login\_level").equals("1"))

{%>

<div class="block clear"></div>

<h4>Employee Role & Manager</h4>

<div class="one\_third first">

<label for="email">Employee Role<span>\*</span></label>

<select style="height: 40px; width:200px" name="employee\_role" id ="employee\_role"

required>

<% out.print(employee Details.getRoleOption((Integer)

Values.get("employee\_role"))); %>

</select>

</div>

<div class="one\_third">

<label for="email">Employee Manager Code<span>\*</span></label>

<select style="height: 40px; width:200px" name = "employee\_manager\_id" id =

"employee\_manager\_id" required>

<% out.print(employee Details.getEmployeeOption((Integer)

Values.get("employee\_manager\_id"))); %>

</select>

</div>

<%} %>

<div class="block clear"></div>

<% if(employee\_id == 0) { %>

<h4>Employee Login Details</h4>

<div class="one\_third first">

<label for="url">Username</label>

<input type="text" name="employee\_user" id="employee\_user" value="<%

out.print(Values.get("employee\_user")); %>" size="22">

</div>

<div class="one\_third">

<label for="url">Password</label>

<input type="password" name="employee\_password" id="employee\_password" value=""

size="22">

</div>

<div class="one\_third ">

<label for="url">Confirm Password</label>

<input type="password" name="employee\_confirm\_password"

id="employee\_confirm\_password" value="" size="22">

</div>

<%} %>

<div class="block clear"></div>

<div>

<input name="submit" type="submit" value="Save Form">

&nbsp;

<input name="reset" type="reset" value="Reset Form">

</div>

<input type="hidden" name="act" value="Save" />

<input type="hidden" name="employee\_id" value="<%

out.print(Values.get("employee\_id")); %>"/>

</form>

</div>

<div style="float: left">

<div><img src="images/save\_1.jpg" style="width: 250px"></div><br>

<div><img src="images/save\_2.jpg" style="width: 250px"></div><br>

<div><img src="images/save\_3.jpg" style="width: 250px"></div>

</div>

<div class="clear"></div>

</main>

</div>

</div>

<%@ include file="includes/footer.jsp"%>

**Coding for Equipment.jsp**

<%@ include file="includes/header.jsp" %>

<%@ page import="java.util.\*"%>

<%@ page import="Model.\*"%>

<%

Equipment equipmentDetails = new Equipment();

int equipment\_id = Integer.parseInt(request.getParameter ("equipment\_id"));

HashMap Values = equipmentDetails.getEquipment Details (equipment\_id);

%>

<script>

function validimage() {

var val = $("#equipment\_image").val();

var id = $("#equipment\_id").val();

if(id == "|| val != ")

{

if(val == ") {

alert('Choose the Equipment Image');

return false;

}

if (!val.match(/(?:gif|jpg|png|bmp)$/)) {

// inputted file path is not an image of one of the above types

alert("inputted file path is not an image!");

return false;

}

}

return true;

}

</script>

<div class="wrapper row3">

<div class="rounded">

<main class="container clear">

<!-- main body -->

<div id="comments" style="width: 70%; float:left; margin-right:30px">

<h2>Equipment Form</h2>

<form method="post" action="UploadEquipment" enctype="multipart/form-data"

onsubmit="return validimage()">

<div class="half\_width">

<label for="email">Equipment Name<span>\*</span></label>

<input type="text" name="equipment\_title" id="equipment\_title" value="<%

out.print(Values.get("equipment\_title")); %>" size="22" style="width:300px;" required>

</div>

<div class="half\_width">

<label for="email">Cost Range<span>\*</span></label>

<input type="text" name="equipment\_cost\_range" id="equipment\_cost\_range" value="<%

out.print(Values.get("equipment\_cost\_range")); %>" size="22" style="width:300px;" required>

</div>

<div class="half\_width">

<label for="email">Equipment Image<span>\*</span></label>

<input type="file" name="equipment\_image" id="equipment\_image" style="width:300px">

</div>

<div class="half\_width">

<label for="email">Description<span>\*</span></label>

<textarea style="width:300px; height:100px;"

name="equipment\_description" required><% out.print(Values.get("equipment\_description"));

%></textarea>

</div>

<% if(!Values.get("equipment\_image").equals("")) { %>

<div class="half\_width">

<img src="equipmentImages/<% out.print(Values.get("equipment\_image"));

%>" style="height:100px; width:100px;">

</div>

<% } %>

<div class="block clear"></div>

<div>

<input name="submit" type="submit" value="Save Equipment">

&nbsp;

<input name="reset" type="reset" value="Reset Form">

</div>

<input type="hidden" name="image\_name" value="<%

out.print(Values.get("equipment\_image")); %>" />

<input type="hidden" name="act" value="Save" />

<input type="hidden" name="type\_id" value="0" />

<input type="hidden" id="equipment\_id" name="equipment\_id" value="<%

out.print(Values.get("equipment\_id")); %>"/>

</form>

</div>

<div style="float: left">

<div><img src="images/save\_1.jpg" style="width: 250px"></div><br>

<div><img src="images/save\_2.jpg" style="width: 250px"></div>

</div>

<div class="clear"></div>

</main>

</div>

</div>

<%@ include file="includes/footer.jsp"%>

**Conclusion of the Project Farmer Buddy:**

Our project is only a humble venture to satisfy the needs to manage their project work. Several user friendly coding have also been adopted. This package shall prove to be a powerful package in satisfying all the requirements of the school. The objective of software planning is to provide a framework that enables the manger to make reasonable estimates made within a limited time frame at the beginning of the software

project and should be updated regularly as the project progresses. At the end it is concluded that we have made an effort on the following points...

* A description of the background and context of the project and its relation to work already done in the area.

• Made a statement of the aims and objectives of the project.

• The description of Purpose, Scope, and applicability.

• We define the problem on which we are working in the project.

• We describe the requirement Specifications of the system and the actions that can be done on these things.

• We understand the problem domain and produce a model of the system, which describes operations that can be performed on the system.

• We included features and operations in detail, including screen layouts.

• We designed the user interface and security issues related to the system.

* Finally the system is implemented and tested according to test cases.

**Future Scope of the Project:**

In a nutshell, it can be summarized that the future scope of the project circles around maintaining information regarding:

* We can add a printer in future.
* We can give more advance software for Farm Management System including more facilities
* We will host the platform on online servers to make it accessible worldwide
* Integrate multiple load balancers to distribute the loads of the system
* Create the master and slave database structure to reduce the overload of the database queries
* Implement the backup mechanism for taking backup of codebase and database on regular basis on different servers
* The above mentioned points are the enhancements which can be done to increase the applicability and usage of this project. Here we can maintain the records of Crops and Equipment. Also, as it can be seen that now-a-days the players are versatile, i.e.so there is a scope for introducing a method to maintain the Farm Management System.
* Enhancements can be done to maintain all the Crops, Equipments, Insecticides,Pesticides, Customer
* We have left all the options open so that if there is any other future requirement in the system by the user for the enhancement of the system then it is possible to implement them.In the last we would like to thank all the persons involved in the development of the system directly or indirectly. We hope that the project will serve its purpose for which it is developed there by underlining the success of the process.

**Limitation of Project on Farmer Buddy**

Although I have put my best efforts to make the software flexible, easy to operate but limitations cannot be ruled out even by me. Though the software presents a broad range of options to its users some intricate options could not be covered into it; partly because of logistics and partly due to lack of sophistication. Paucity of time was also a major constraint, thus it was not possible to make the software foolproof and dynamic.

Lack of time also compelled me to ignore some parts such as storing old results of the candidate etc.

Considerable efforts have made the software easy to operate even for the people not related to the field of computers but it is acknowledged that a layman may find it a bit problematic at the first instance. The user is provided help at each step for his convenience in working with the software.

**List of limitations which is available in the Farmer Buddy**

* Excel export has not been developed for Crops, Equipments due to some criticality.
* The transactions are executed in off-line mode, hence on-line data for Insecticides,
* Pesticides capture and modification is not possible.
* Off-line reports of Crops, Customer, Insecticides cannot be generated due to batch mode execution.