

**VISVESVARAYA TECHNOLOGICAL UNIVERSITY**  
**Jnana Sangama, Belagavi-590010**



**MINI PROJECT REPORT**  
**ON**  
**“PLANT NURSERY MANAGEMENT SYSTEM”**

Submitted in partial fulfillment for the requirements of the fifth semester curriculum

**BACHELOR OF ENGINEERING**  
**IN**  
**COMPUTER SCIENCE AND ENGINEERING**

For the Academic year 2019-2020

Submitted by:

**SRISHTI NEMA**  
**PRATILIP AICH**

**1MV17CS110**  
**1MV17CS128**

Project carried out at:

**Sir M. Visvesvaraya Institute of Technology**  
Bengaluru-562157

Under the guidance of:

**Dr. Sheela Kathavate**

Associate Professor, Department of CSE  
Sir M. Visvesvaraya Institute of Technology, Bengaluru



**DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING**  
**SIR. M VISVESVARAYA INSTITUTE OF TECHNOLOGY**  
HUNASAMARANAHALLI, BENGALURU-562157

**SIR M. VISVESVARAYA INSTITUTE OF TECHNOLOGY  
BENGALURU -562157**

**DEPARTMENT OF COMPUTER SCIENCE AND  
ENGINEERING**



**CERTIFICATE**

It is certified that the project work entitled “**PLANT NURSERY MANAGEMENT SYSTEM**” is a bona fide work carried out by Srishti Nema (1MV17CS110) and Pratilipi Aich (1MV17VS128) in partial fulfillment for the requirements of mini project for the V semester curriculum, Bachelor of Engineering in Computer Science and Engineering of the Visvesvaraya Technological University, Belagavi during the academic year 2019-2020. It is certified that all corrections and suggestions indicated for internal assessment have been incorporated in the report. The project report has been approved as it satisfies the academic requirements in respect of project work prescribed for the course of Bachelor of Engineering.

Name & Signature of Guide

**Dr. Sheela Kathavate**

Associate Prof. & Internal Guide

Dept. Of CSE, Sir MVIT

Bengaluru -562157

Name & Signature of HOD

**Dr. Bhanu Prakash G C**

HOD, Dept of CSE

Sir MVIT Bengaluru -562157

External Examination:  
Name of the Examiners

Signature with date

1)

2)

## ACKNOWLEDGMENT

It gives us immense pleasure to express our sincere gratitude to the management of Sir M. Visvesvaraya Institute of Technology, Bangalore for providing the opportunity and the resources to accomplish our project work in their premises.

On the path of learning, the presence of an experienced guide is indispensable and we would like to thank our guide **Dr. Sheela Kathavate**, Associate Professor, Dept. of CSE, for her invaluable help and guidance.

We would also like to convey our regards and sincere thanks to **Dr. G. C. Bhanu Prakash**, HOD, Dept. of CSE for his suggestions, constant support and encouragement, Heartfelt and sincere thanks to **Dr. V.R. Manjunath**, Principal, Sir. MVIT for providing us with the infrastructure and facilities needed to develop our project.

We would also like to thank the staff of Department of Computer Science and Engineering and lab-in-charges for their co-operation and suggestions. Finally, we would like to thank all our friends for their help and suggestions without which completing this project would not have been possible.

- Srishti Nema (1MV17CS110)

- Pratilipi Aich (1MV17CS128)

## **DECLARATION**

We hereby declare that the entire mini project work embodied in this dissertation has been carried out by us and no part has been submitted for any degree or diploma of any institution previously.

Place: Bengaluru

Date:

Signature of Students:

SRISHTI NEMA (1MV17CS110)

PRATILIP AICH (1MV17CS128)

## **ABSTRACT**

This project aims at building a plant nursery management system in order to simplify the procedure of building a home garden. When planning about building a full-fledged garden, decorating the house, a kitchen garden or just some flowers to beautify your backyard, the plant nursery system proves to be really helpful to plan out what plants to buy. All the plants in this system have been classified based on their categories and hence customers have the ease to look for a particular category for their garden. We have also provided a short description about each plant along with an image for the beginners in gardening.

Every customer registers into the plant nursery system initially, after which they can view the wide variety of plants, choose what's best for them, check out the price and place an order for the desired quantity. One supplier has been assigned for each plant that is responsible for managing the delivery of the orders made for that particular plant. The admin has the rights to add, update, delete and view all the plants, orders and categories.

## TABLE OF CONTENTS

<b>CERTIFICATE</b>	<b>ii</b>
<b>ACKNOWLEDGEMENT</b>	<b>iii</b>
<b>DECLARATION</b>	<b>iv</b>
<b>ABSTRACT</b>	<b>v</b>
<b>TABLE OF CONTENTS</b>	<b>vi</b>
<b>LIST OF FIGURES</b>	<b>vii</b>
<b>Chapter 1 INTRODUCTION</b>	<b>1</b>
1.1 Introduction of the project	1
1.2 Need for the project	2
1.3 Objective of the project	2
<b>Chapter 2 FRONTEND AND BACKEND</b>	<b>3</b>
2.1 About the frontend	3
2.2 About the backend	4
<b>Chapter 3 SPECIFICATIONS</b>	<b>6</b>
3.1 Hardware Requirements	6
3.2 Software Requirements	6
3.3 Functional Requirements	6
3.4 Non-Functional Requirements	8
<b>Chapter 4 SYSTEM DESIGN</b>	<b>9</b>

4.1 Introduction	9
4.2 Data Flow Diagram	9
4.3 Schema Design	10
4.4 ER Diagram	11
4.5 Design strategy	12
<b>Chapter 5 IMPLEMENTATION</b>	<b>19</b>
5.1 Table design	19
<b>Chapter 6 RESULTS AND SNAPSHOTS</b>	<b>24</b>
<b>CONCLUSION</b>	<b>31</b>
<b>REFERENCES</b>	<b>32</b>
8.1 Books	32
8.2 Links	33

## **LIST OF FIGURES**

Figure 4.1 Data Flow Diagram	13
Figure 4.2 Login Data Flow	14
Figure 4.3 Admin Data Flow	15
Figure 4.4 Database Schema	16
Figure 4.5 ER Diagram	17



## CHAPTER 1

### INTRODUCTION

#### 1.1 INTRODUCTION TO PROJECT

The nursery management system aims at solving the gardening related problems for everyone who is ready to build a home garden or kitchen garden. Learning that very few people around the world have gardening as their hobby and due to the lack of information and resources they tend to lose contact with gardening is worth pondering about. For all those aesthetics out there who want to decorate their homes with beautiful flora and other houseplants, this is the perfect source to gain the knowledge, sort out their plans and budgets and get what they want easily and efficiently. This project also tends to convey an environmental related message since in today's world there aren't many trees and plants left which is most needed element for our survival.

The plant nursery management system takes into consideration all the basic needs of a customer while planning for a garden starting from listing a variety of plants based on their categories along with their ideal season, affordable prices and a short description to let them decide on what is best for them. Categories ranging from herbal plants, house decorative plants, flora, fruits and vegetables have been given thought to in this system. The management system provides a well and good user experience by letting all the customers register to the system and then log into the system using the registered username and password. By doing so they can have access to view and pick from a list of plants. There is a separate login provided specifically for the administrator to allow him/her to access the nursery system. The admin is granted to edit the information available on the system that is, he/she can add new plants, suppliers or categories, update the values for plants, suppliers, orders, categories or customers and also delete any of these contents.

## **1.2 NEED FOR THE PROJECT**

A plant information management system should fulfill two important goals. These are to collect and archive as much data needed to operate the plant, and allow for effective troubleshooting. While having a lot of data is beneficial to operating a manufacturing process; collecting and storing data, does not yield any functional results. It is in the presentation of the data which leads to informed, productive, and cost-effective decisions. The objective of plant maintenance is to achieve minimum breakdown and to keep the plant in good working condition at the lowest possible cost. The structure of plant management depends basically on the size and the specific production type of the enterprise. Such a system will provide better-organized production data.

## **1.3 OBJECTIVE OF THE PROJECT**

The main objective while implementing the project - Plant Nursery Management System – were to minimize the work done and at the same time increase the speed of the work done while deciding and purchasing plants for gardening purposes.

This new system is built with the following objectives:

- 1) Easier to retrieve information.
- 2) Maintenance of the database as well as the overall project will become efficient.
- 3) Reduction of data redundancy.
- 4) Security is provided by maintaining the login of username and the password.

## CHAPTER 2

### FRONT END AND BACK END

#### 2.1 ABOUT THE FRONT END

##### 2.1.1 PYTHON

Python is an interpreted, object-oriented, high-level programming language with dynamic semantics. Its high-level built in data structures, combined with dynamic typing and dynamic binding, make it very attractive for Rapid Application Development, as well as for use as a scripting or glue language to connect existing components together. Python's simple, easy to learn syntax emphasizes readability and therefore reduces the cost of program maintenance. Python supports modules and packages, which encourages program modularity and code reuse. The Python interpreter and the extensive standard library are available in source or binary form without charge for all major platforms, and can be freely distributed.

Often, programmers fall in love with Python because of the increased productivity it provides. Since there is no compilation step, the edit-test-debug cycle is incredibly fast. Debugging Python programs is easy: a bug or bad input will never cause a segmentation fault. Instead, when the interpreter discovers an error, it raises an exception. When the program doesn't catch the exception, the interpreter prints a stack trace. A source level debugger allows inspection of local and global variables, evaluation of arbitrary expressions, setting breakpoints, stepping through the code a line at a time, and so on. The debugger is written in Python itself, testifying to Python's introspective power. On the other hand, often the quickest way to debug a program is to add a few print statements to the source: the fast edit-test-debug cycle makes this simple approach very effective.

### **2.1.2 TKINTER**

Tkinter is a Python binding to the Tk GUI toolkit. It is the standard Python interface to the Tk GUI toolkit. Tkinter is included with standard Linux, Microsoft Windows and Mac OS X installs of Python. The name Tkinter comes from Tk interface. Tkinter was written by Fredrik Lundh. Tkinter is free software released under a Python license. The Tkinter module (“Tk interface”) is the standard Python interface to the Tk GUI toolkit from Scriptics (formerly developed by Sun Labs). Both Tk and Tkinter are available on most UNIX platforms, as well as on Windows and Macintosh systems. Starting with the 8.0 release, Tk offers native look and feel on all platforms.

Tkinter is Python’s standard GUI (Graphical User Interface) package. It is a thin object-oriented layer on top of standard GUI package, TCL/TK. It is one among the many GUI toolkits for Python; however, it is the most commonly used toolkit out there. Python when combined with Tkinter provides a fast and easy way to create GUI applications. Tkinter provides a powerful object-oriented interface to the TK GUI toolkit. Creating a GUI Application using Tkinter is simple - Import the Tkinter module, Create the GUI application main window, Add one or more of the above-mentioned widgets to the GUI application and enter the main event loop to take action against each event triggered by the user.

Tkinter consists of a number of modules. The Tk interface is provided by a binary extension module named `_tkinter`. This module contains the low-level interface to Tk, and should never be used directly by application programmers. It is usually a shared library (or DLL), but might in some cases be statically linked with the Python interpreter. The public interface is provided through a number of Python modules. The most important interface module is the Tkinter module itself.

## **2.2 ABOUT THE BACKEND**

### **2.2.1 SQLITE**

SQLite is a software library that provides a relational database management

system. The lite in SQLite means light weight in terms of setup, database administration, and required resource. SQLite has the following noticeable features: self-contained, serverless, zero-configuration, transactional. It is Serverless. SQLite is self-contained means it requires minimal support from the operating system or external library. This makes SQLite usable in any environments especially in embedded devices like iPhones, Android phones, game consoles, handheld media players, etc.

SQLite is developed using ANSI-C. The source code is available as a big `sqlite3.c` and its header file `sqlite3.h`. If you want to develop an application that uses SQLite, you just need to drop these files into your project and compile it with your code. All transactions in SQLite are fully ACID-compliant. It means all queries and changes are Atomic, Consistent, Isolated, and Durable. In other words, all changes within a transaction take place completely or not at all even when an unexpected situation like application crash, power failure, or operating system crash occurs. SQLite uses dynamic types for tables. It means you can store any value in any column, regardless of the data type. SQLite allows a single database connection to access multiple database files simultaneously. This brings many nice features like joining tables in different databases or copying data between databases in a single command. SQLite is capable of creating in-memory databases which are very fast to work with.

## CHAPTER 3

### SPECIFICATIONS

#### 3.1 HARDWARE REQUIREMENTS

- ☐ PROCESSOR : Intel Pentium or Higher Version
- ☐ RAM : Minimum 1GB
- ☐ HARD DISK : 60GB and above

#### 3.2 SOFTWARE REQUIREMENTS

- ☐ SOFTWARE : Python 3.3 or greater, SQLITE3, DB Browser
- ☐ EDITORS : Any code editor of your choice. (Sublime Text, Visual Studio Code, Pycharm, etc)
- ☐ Windows Vista\* or Newer Version, or MACos, or Linux (32/64 bit)

#### 3.3 FUNCTIONAL REQUIREMENTS

The Functional Requirements Specification documents the Operations and activities that a system must be able to perform. Functional Requirements include:

- ☐ Addition, Deletion and Modification of records
- ☐ Customer (Name, Contact, Address)
- ☐ Descriptions of work-flows performed by the system
- ☐ Orders (Cus\_ID, Plant\_ID)
- ☐ Login (Username, Password)
- ☐ Who can enter the data into the system
- ☐ How the system meets applicable regulatory requirements

The Functional Requirements Specifications is designed to be read by a general audience. Readers should understand the system, but no particular technical knowledge should be required to understand the document.

These are the functional requirements specification documents for the project analysis. A software requirement specification helps to attenuate the time and energy needed by the developers to attain their desired goals and additionally minimizes the value of development.

**Following Factors are used to measure software development quality:**

Each attribute may be accustomed measure of the product performance. These attributes may be used for Quality assurance similarly as quality control. Quality assurance activities are directed towards prevention of introduction of defects and internal control activities are aimed toward detecting defects in product and services.

**1. Reliability**

Measure if product is reliable enough to sustain in any condition. Give systematically correct results. Product dependability is measured in terms of operation of project underneath different operating atmosphere and different conditions.

**2. Maintainability**

Different versions of the product ought to be easy to maintain. For development it ought to be easy to feature code to existing system, ought to be easy to upgrade for brand new options and new technologies time to time. Maintenance ought to be value effective and simple. System be easy to take care of and correcting defects or making a change within the software system.

**3. Usability**

This can be measured in terms of ease of use. Application should be user friendly. Easy to use for input preparation, operation and also for interpreting of output.

**4. Portability**

This can be measured in terms of Costing issues related to porting, Technical issues related to porting, Behavioural issues related to porting.

### **3.4 NON-FUNCTIONAL REQUIREMENTS**

Satisfactoriness will probably not be assessed on the system where the program is developed, tested or first installed. Nonfunctional requirements describe how a system must behave and establish constraints of its functionality. Security requirements are ensured so that the software is protected from unauthorized access to the system and its stored data. Reliability is maintained to measure how likely it is for the software to work without failure for a given period of time.



## **CHAPTER 4**

### **SYSTEM DESIGN**

#### **4.1 INTRODUCTION**

System design is the first design stage for devising the basic approach to solving the problem. During system design, developers decide the overall structures and styles. The system architecture determines the organization of the system into subsystems. In addition, the architecture provides the context for the detailed decisions that are made in later stages .during design, developers make decisions about how the problem will be solved, first at the high level and then with more detail.

#### **4.2 DATA FLOW DIAGRAM**

A Data Flow Diagram (DFD) is a graphical representation of the “flow” of data through an information system. Data Flow modules are used to show how data flows through a sequence of processing steps. The data is transformed at each step before moving on to the next stage. These processing steps or transformations are program functions when Data Flow Diagrams are used to document a software design. Data flow modules are an intuitive way of showing how data is processed by a system. At the analysis level, they should be used to module the way in which data is processed in the existing system. The notation used in these modules represents functional processing, data stores and data movements between functions. With a dataflow diagram, users are able to visualize how the system will operate, what the system will accomplish and how the system will be implemented. Old system dataflow diagrams can be drawn up and compared with the new system dataflow.

These are several common modelling rules to be followed while creating DFD's are as follows:

- All processes must have at least one data flow in and one data flow out.
- All processes should modify the incoming data, producing a new form of outgoing data.
- Each data store must be involved with at least one data flow.
- Each external entity must be involved with at least one data flow.
- A data flow must be attached to at least one process.

### **4.3 SCHEMA DIAGRAM**

A database schema represents the logical configuration of all or part of a relational database. It can exist both as a visual representation and as a set of formulas known as integrity constraints that govern a database. These formulas are expressed in a data definition language, such as SQL. As part of a data dictionary, a database schema indicates how the entities that make up the database relate to one another, including tables, views, stored procedures, and more.

Typically, a database designer creates a database schema to help programmers whose software will interact with the database. The process of creating a database schema is called data modeling. When following the three-schema approach to database design, this step would follow the creation of a conceptual schema. Conceptual schemas focus on an organization's informational needs rather than the structure of a database.

There are two main kinds of database schema:

1. A logical database schema conveys the logical constraints that apply to the stored data. It may define integrity constraints, views, and tables.
2. A physical database schema lays out how data is stored physically on a storage system in terms of files and indices.

A database schema is the skeleton structure that represents the logical view of the entire database. It defines how the data is organized and how the relations among them are associated. It formulates all the constraints that are to be applied on the data. A database schema defines its entities and the relationship among them. It contains a descriptive detail of the database, which can be depicted by means of schema diagrams. It's the database designers who design the schema to help programmers understand the database and make it useful.

## 4.4 ER DIAGRAM

Entity relationship diagram displays the relationships of entity set stored in a database. In other words, we can say that ER diagrams help you to explain the logical structure of databases. At first look, an ER diagram looks very similar to the flowchart. However, ER Diagram includes many specialized symbols, and its meanings make this model unique.

An entity relationship diagram (ERD) shows the relationships of entity sets stored in a database. An entity in this context is an object, a component of data. An entity set is a collection of similar entities. These entities can have attributes that define its properties. By defining the entities, their attributes, and showing the relationships between them, an ER diagram illustrates the logical structure of databases. ER diagrams are used to sketch out the design of a database.

This model is based on three basic concepts:

- ☐ Entities
- ☐ Attributes
- ☐ Relationships

Rectangle: Represents Entity sets.

Ellipses: Attributes

Diamonds: Relationship Set

Lines: They link attributes to Entity Sets and Entity sets to Relationship Set

Double Ellipses: Multivalued Attributes

Dashed Ellipses: Derived Attributes

Double Rectangles: Weak Entity Sets

Double Lines: Total participation of an entity in a relationship set

## 4.5 DESIGN STRATEGY

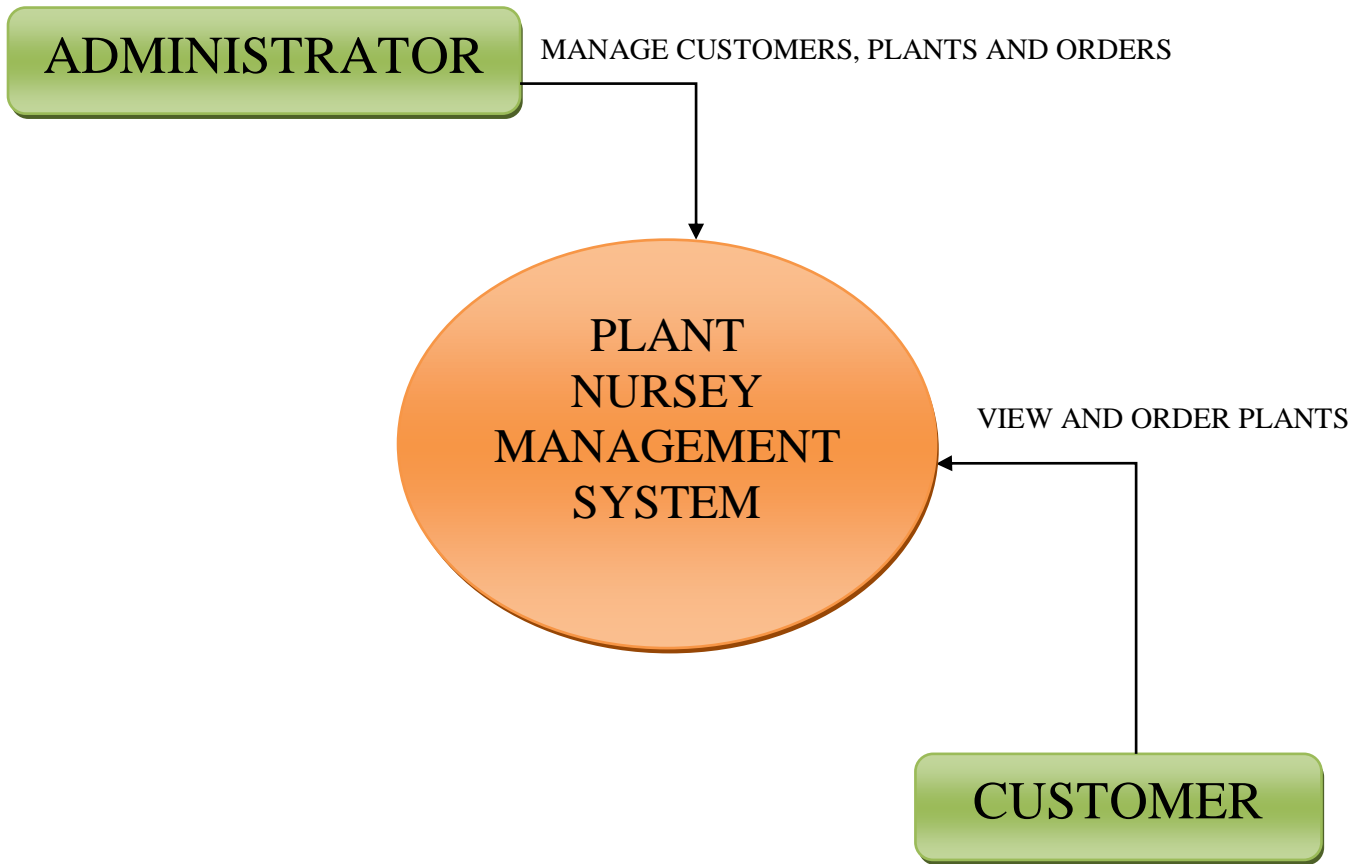
The design strategy is a vital aspect of the system to be developed. The design of the software reflects the basic understanding of the problem. For designing a good system what we have to do is get the correct definition of the problem and analyze it thoroughly.

The design of a system should be such that if a small portion is changed, the rest of the system should be unaffected. This is the flexibility of the system. Greater the flexibility greater will be the system reliability. While carrying out the job of designing of a new system, one has to consider many factors. These factors include the drawbacks and limitations of the present manual system as well as of the features and advantages of the proposed system. It should be designed in such a manner that even a layman can run it without any difficulty. An important quality of software that must be present is “user friendliness”. This can be achieved in many ways such as providing menu, giving context sensitive help, doing automatic validation to input data, etc. another main factor is speed efficiency. In order to achieve speed efficiency, the program should be designed accordingly and the user is provided with a compiled copy of the software package with necessary data file format rather than source code.

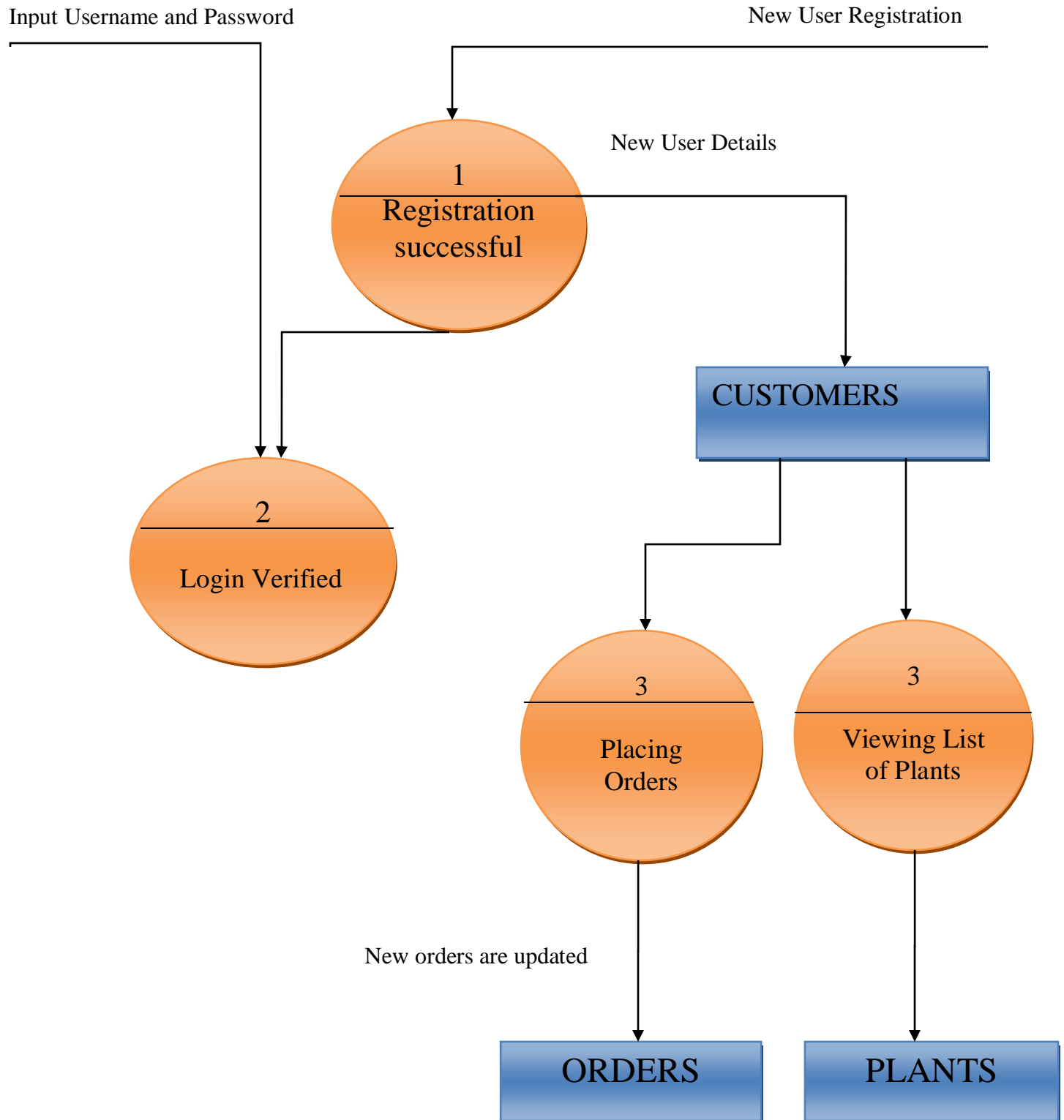
Design of input and output formats is equally important for any design. The output format should be designed in such a way that it must reflect all the required information in detail and the design of the database itself such as type of data stored, size of data, etc. Some of the decisions made during database design are:

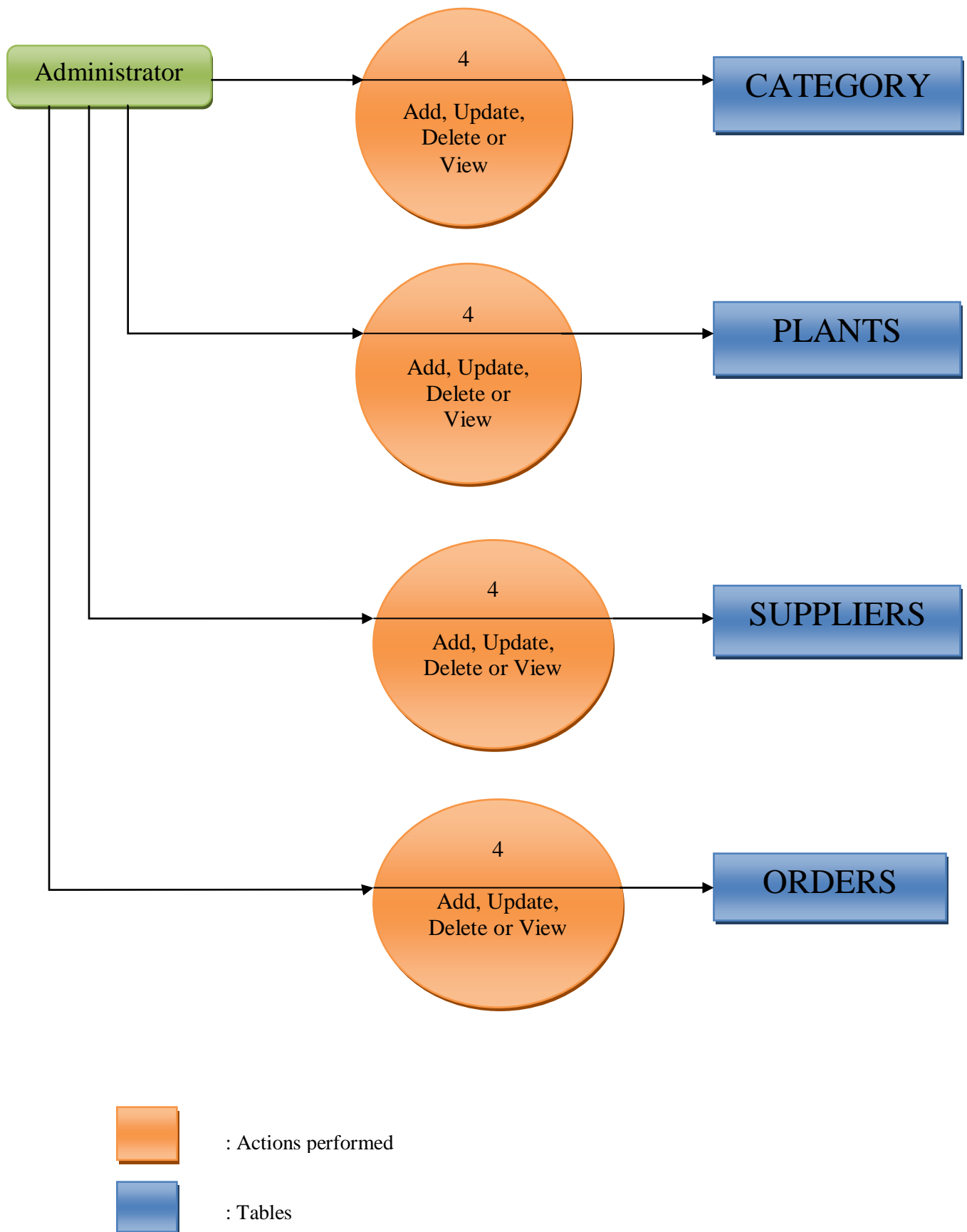
- ☐ Which data items are to be recorded and in which database.
- ☐ Length of each record based on the characteristics of the data items on which it is based.
- ☐ Unauthorized change of data must be prevented.
- ☐ Data, which must be avoided from redundancy.
- ☐ Maintenance of data integrity.
- ☐ Avoid over writings.
- ☐ Prevents invalid data access and changes.

Having all this and a positive interaction with the client at every stage of development is the core around which the software is built.



**Figure 4.1: Data Flow Diagram**

**Figure 4.2: Login Data Flow**

**Figure 4.3: Admin Data Flow**

**CATEGORY**

CAT_ID	CAT_NAME
--------	----------

PK

**CUSTOMERS**

CID	NAME	PHONE	ADDRESS	USERNAME	PASSWORD
-----	------	-------	---------	----------	----------

PK

**PLANT**

FK

PID	PLANT_NAME	SEASON	CATEGORY_NAME	DESCRIPTION	PRICE	SID
-----	------------	--------	---------------	-------------	-------	-----

PK

FK

**SUPPLIER**

SID	SUP_NAME	SUP_PHONE	SUP_ADDRESS
-----	----------	-----------	-------------

PK

**ORDERS**

FK

OID	CID	PID	QUANTITY
-----	-----	-----	----------

FK

**Figure 4.4: Database Schema**



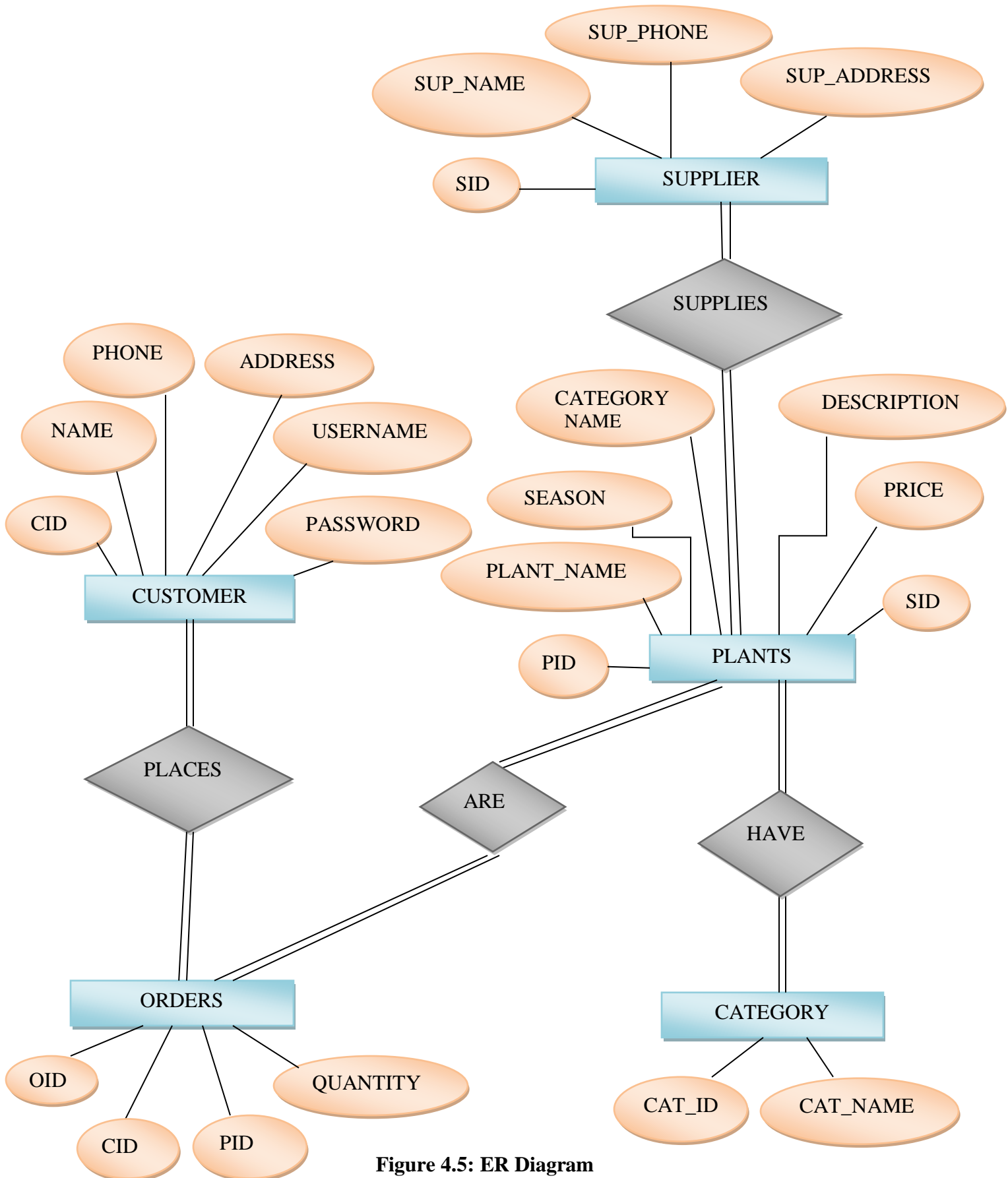


Figure 4.5: ER Diagram

## CHAPTER 5

## IMPLEMENTATION

### 5.1 TABLE DESIGN

#### 5.1.1 CUSTOMERS

<u>CID</u>	TEXT
NAME	TEXT
PHONE	INTEGER
ADDRESS	TEXT
USERNAME	TEXT
PASSWORD	TEXT

```
c.execute("INSERT INTO customer VALUES (:cid, :name, :phone, :address, :username, :password) ",
```

```
    {  
        'cid': cid.get(),  
        'name': name.get(),  
        'phone': phone.get(),  
        'address': address.get(),  
        'username': username.get(),  
        'password': password.get()  
    }
```

Admin can update the values of customer entity stored in the GARDEN.db database. Referencing to a particular customer is done using the primary key CID.

```
def update():  
    conn=sqlite3.connect('garden.db')  
  
    #create cursor  
    c=conn.cursor()  
  
    record_id=select_box.get()  
    c.execute("""UPDATE customer SET  
        cid=:cid,  
        name=:name,  
        phone=:phone,  
        address=:address,  
        ussername=:ussername,  
        password=:password  
  
        WHERE cid=:cid""",  
        {  
            'cid': cid_editor.get(),  
            'name':name_editor.get(),  
            'phone':phone_editor.get(),  
            'address':address_editor.get(),  
            'ussername':ussername_editor.get(),  
            'password':password_editor.get()  
        })
```

### 5.1.2 PLANT

<u>PID</u>	INTEGER
NAME	TEXT
SEASON	TEXT
PLANT_CATEGORY	TEXT
DESCRIPTION	TEXT
PRICE	FLOAT
SID	INTEGER

```
c.execute("""UPDATE plant SET
    pid= :pid,
    name=:name,
    season=:season,
    category=:category,
    description=:description,
    price=:price,
    sid=:sid

    WHERE pid=:pid""",
    {
        'pid': pid_editor.get(),
        'name':name_editor.get(),
        'season':season_editor.get(),
        'category':category_editor.get(),
        'description':description_editor.get(),
        'price':price_editor.get(),
        'sid':sid_editor.get(),
        'pid':record_id
    })
```

### 5.1.3 CATEGORY

<u>CAT_ID</u>	INTEGER
CAT_NAME	TEXT

```
c.execute("DELETE from category WHERE cat_id="+ select_box.get())
***
c.execute("SELECT *,cat_id FROM category")
records=c.fetchall()
```

Inserting new category into the database.

```
def submit():
    conn=sqlite3.connect('garden.db')

    #create cursor
    c=conn.cursor()

    #insert into table
    c.execute("INSERT INTO category VALUES(:cat_id,:cat_name)",
              {
                  'cat_id': cat_id.get(),
                  'cat_name': cat_name.get()
              }
    )

    conn.commit()

    #close connection
    conn.close()
```

### 5.1.4 ORDERS

<u>OID</u>	INTEGER
PID	INTEGER
CID	INTEGER
QUANTITY	INTEGER

```
c.execute("INSERT INTO orders VALUES(:oid,:cid,:pid,:quantity)",
        {
            'oid': oid.get(),
            'cid': cid.get(),
            'pid': pid.get(),
            'quantity': quantity.get()
        })
```

```
def update():
    conn=sqlite3.connect('garden.db')

    #create cursor
    c=conn.cursor()

    record_id=select_box.get()
    c.execute("""UPDATE orders SET
                oid= :oid,
                cid=:cid,
                pid=:pid,
                quantity=:quantity

                WHERE oid=:oid""",
        {
```

```

        'oid': oid_editor.get(),
        'cid':cid_editor.get(),
        'pid':pid_editor.get(),
        'quantity':quantity_editor.get(),
        'oid':record_id
    })
conn.commit()
conn.close()

```

### 5.1.5 SUPPLIER

SID	INTEGER
SUP_NAME	TEXT
SUP_PHONE	INTEGER
SUP_ADDRESS	TEXT

```

c.execute("""UPDATE supplier SET
        sid=:sid,
        sup_name=:sup_name,
        sup_phone=:sup_phone,
        sup_address=:sup_address

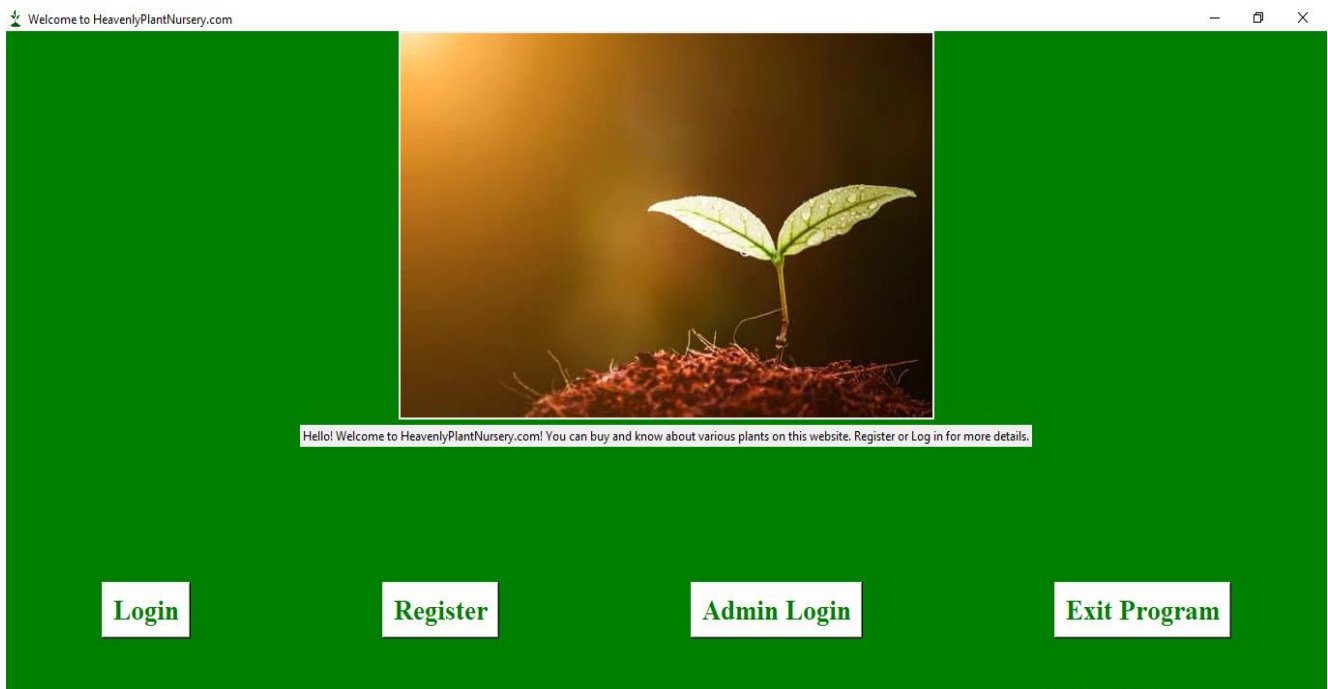
        WHERE sid=:sid""",
        {
            'sid': sid_editor.get(),
            'sup_name':sup_name_editor.get(),
            'sup_phone':sup_phone_editor.get(),
            'sup_address':sup_address_editor.get(),
            'sid':record_id
        })

```

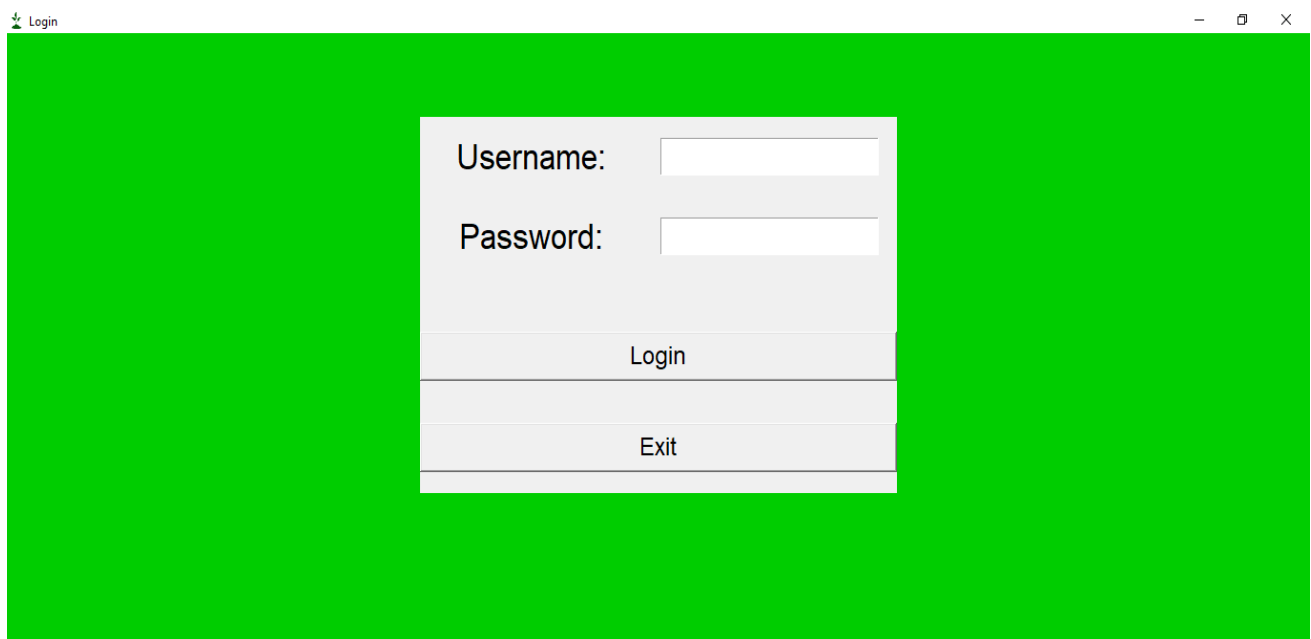
## CHAPTER 6

## RESULTS AND SNAPSHOT

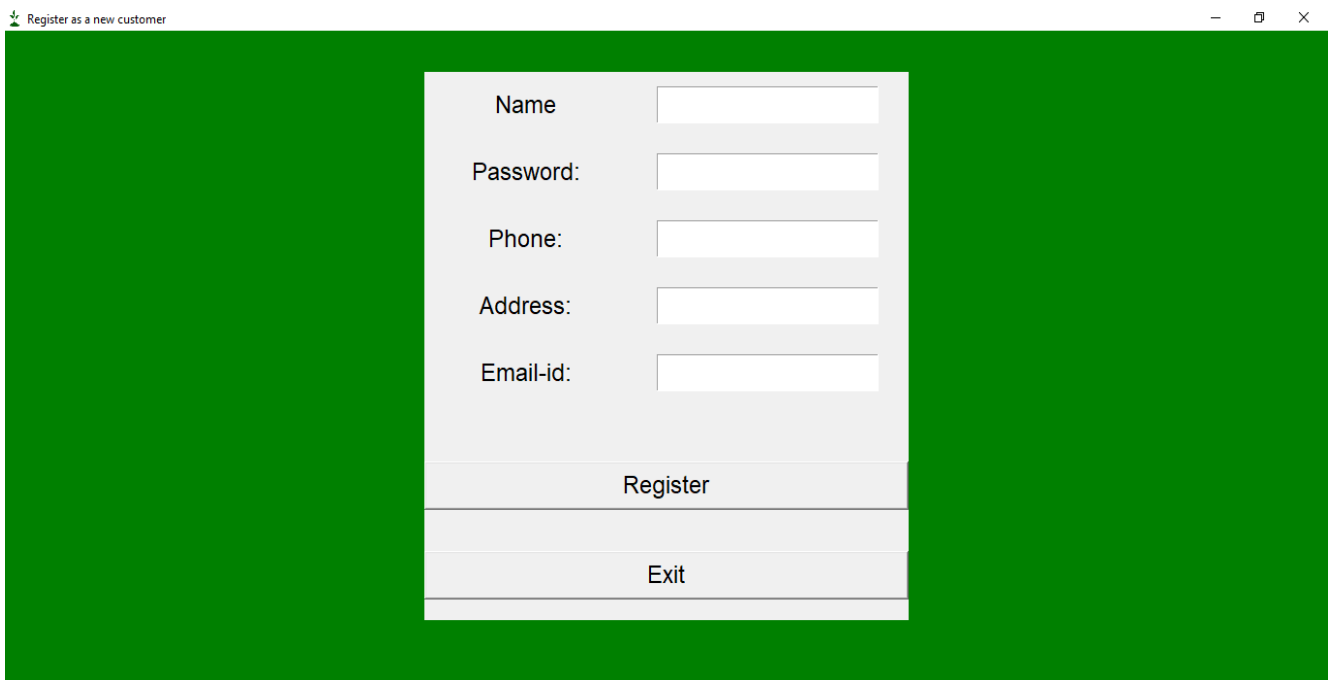
### 6.1 HOME PAGE:



### 6.2 LOGIN PAGE:

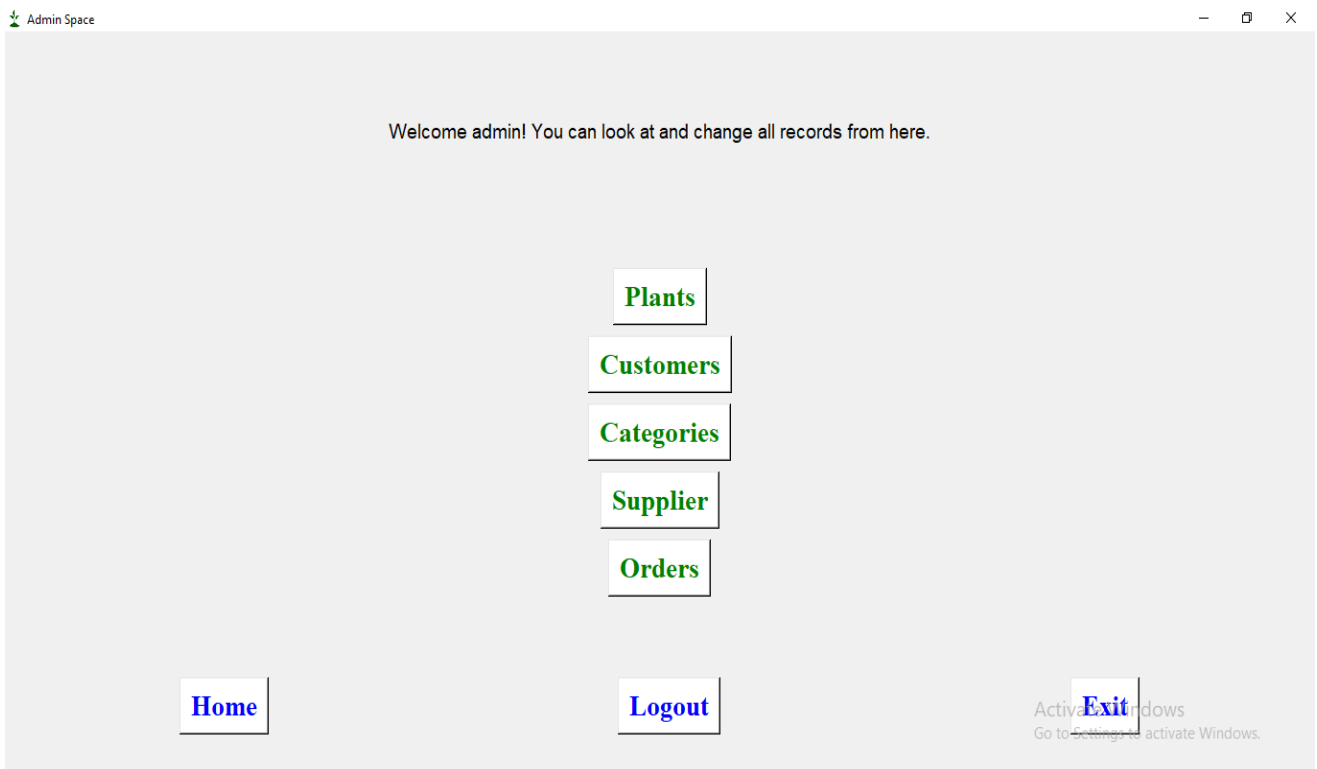




**6.3 REGISTRATION PAGE:**

The screenshot shows a registration form titled "Register as a new customer" on a green background. The form is a light gray box with the following fields and buttons:

- Name:
- Password:
- Phone:
- Address:
- Email-id:
- Register button
- Exit button

**6.4 ADMIN LOGIN:**

The screenshot shows the "Admin Space" dashboard. At the top, it says "Welcome admin! You can look at and change all records from here." Below this, there is a vertical stack of buttons: Plants, Customers, Categories, Supplier, and Orders. At the bottom left are buttons for Home and Logout. At the bottom right is an Exit button. A Windows activation watermark is visible in the bottom right corner.

## 6.5 ADMIN CAN ACCESS

Plants(As Admin)

Plant ID

Name of plant

Price

About the plant

Season of growth

Category ID of table

Supplier ID of table

Select ID Number to delete/update

Activate Windows

Categories(As admin)

Enter category id

Enter category name

Select ID Number to delete/update

Activate Windows

Supplier Details(As Admin)

Supplier ID

Name of supplier

Phone number of supplier

Address of supplier

[Add record to database](#)

[Show record of plants](#)

Select ID Number to delete/update

[Delete record](#)

[Update record](#)

[Home](#) [Logout](#) [Exit](#)

Activate Windows  
Go to Settings to activate Windows.

Orders(As admin)

Enter order id

Enter customer id

Enter plant id

Enter quantity

[Add record to database](#)

[Show record of all orders](#)

Select ID Number to delete/update

[Delete record](#)

[Update record](#)

[Home](#) [Logout](#) [Exit](#)

Activate Windows  
Go to Settings to activate Windows.

## 6.6 ADMIN CAN ADD/UPDATE:

✚ Edit a plant record

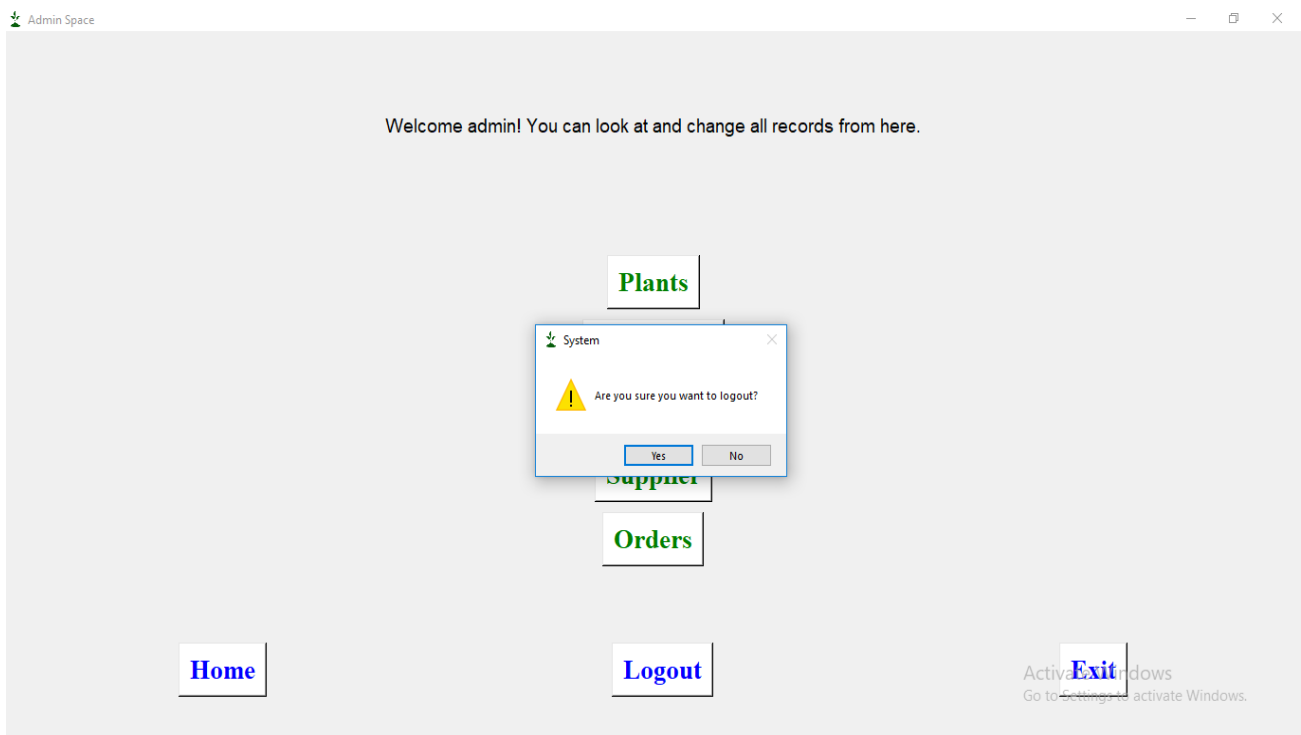
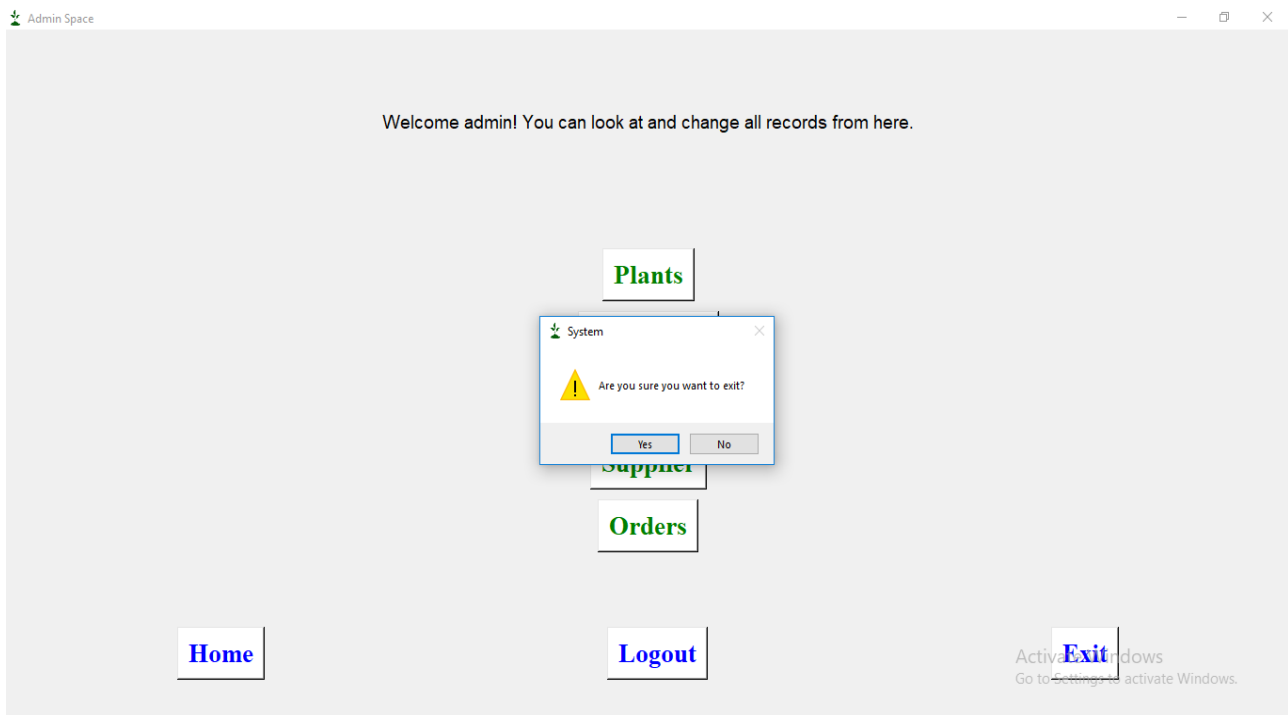
Plant ID	<input type="text" value="3"/>
Name of plant	<input type="text" value="Cilantro"/>
Season	<input type="text" value="Summer-Bloom"/>
Category of plant	<input type="text" value="Herb"/>
Description about the plant	<input type="text" value="With bright green, fern-textured st"/>
Price	<input type="text" value="65.0"/>
Supplier ID	<input type="text" value="y0ya 0JFIF 00 0 0 yb ,CREA"/>

Save record

✚ Edit a record

Order ID	<input type="text" value="111"/>
Customer ID	<input type="text" value="1"/>
Plant id ordered	<input type="text" value="3"/>
Quantity ordered	<input type="text" value="4"/>

Save record

**6.7 PROMPTS TO CONFIRM:**

## CONCLUSION

A plant nursery management system is vital for stores/small scale projects related to the sales of plants, in order to manage the information about a variety of plants and keep track of the number of customers and orders placed by customers. Better information management systems are needed to improve the current practices of retail markets, thus increasing efficiency and minimizing costs. The following are the major achievements of the new system developed:

- ☐ A fully menu-driven user friendly system, for the user to enter data, update records or delete records with great ease.
- ☐ Automated operations lets us avoid a lot of manual work.
- ☐ Additional checks have been incorporated to avoid duplications of data as far as possible.

The Plant Nursery Management System serves as a great platform to manage plant information along with allowing customers to view and choose according to their priorities. The user has to first register himself/herself. Then he/she can go onto viewing the list of plants to pick one or two for their garden. The admin has the rights to view all the plants, orders, categories, customers and suppliers. Also administrator can add, update and delete any of these records.

For further improvement, a platform for users to place orders for their chosen plants online covering steps from entering card details, making payment and entering delivery information to tracking their orders can be implemented. There is no space for customer feedback. This can be incorporated in further versions. Better security can be provided. An alternate method for situations where the user does not remember the password can be incorporated. Certain offers can be proposed on specific plants including discount offers and free gardening tools. . A fully integrated approach will better improve communication and minimize gaps in information flow among all the parties and departments involved.

## REFERENCES

### BOOKS

- [1] Database Management Systems, Ramakrishnan and Gehrke, 3<sup>rd</sup> Edition, 2014, McGrawHill.
  
- [2] Fundamentals of Database Systems, Ramez Elmasri and Shamkant B. Navathe, 7<sup>th</sup> Edition, 2017, Pearson.
  
- [3] Python GUI Programming with Tkinter, Alan D. Moore, Packt Publishing
  
- [4] Tkinter GUI Application Development Cookbook: A practical solution to your GUI development problems with Python and Tkinter, Alejandro Rodas de Paz
  
- [5] Using SQLite, Jay Kreibich
  
- [6] The SQL Guide to SQLite, Rick F. Van Der Lans
  
- [7] Python – The Fastest Growing Programming Language, International Research Journal of Engineering and Technology (IRJET)
  
- [8] PyDraw: a GUI drawing generator based on Tkinter and its design concept, Jinwei Lin and Aimin Zhou

## LINKS

- [1] <https://www.sqlite.org/index.html>
- [2] <https://www.google.co.in/>
- [3] <https://likegeeks.com/python-sqlite3-tutorial/>
- [4] <https://stackoverflow.com/>
- [5] <https://codemy.com/>
- [6] <https://www.youtube.com/>
- [7] <https://tkdocs.com/tutorial/>
- [8] [https://www.python-course.eu/python\\_tkinter.php](https://www.python-course.eu/python_tkinter.php)
- [9] [http://www.learntosolveit.com/python/software\\_engineering\\_sqlite3.html](http://www.learntosolveit.com/python/software_engineering_sqlite3.html)
- [10] <https://www.sqlitetutorial.net/sqlite-resources/>