**FRESH FIELDS**

**A Project Report**

***Submitted by***

**RIYA ADSUL**

**RISHABH TULSHYAN**

**AARYAN SARNAIK**

**DEVIKA SURYAWANSHI**

***Under the Guidance of***

**Prof. Kamal Mistry**

***in partial fulfillment for the award of the degree of***

**BTECH**

**COMPUTER ENGINEERING**

At



**MUKESH PATEL SCHOOL OF TECHNOLOGY, MANAGEMENT AND ENGINEERING, MUMBAI**

**APRIL, 2020**

**DECLARATION**

I, Aaryan Sarnaik, Devika Suryawanshi, Rishabh Tulshyan , Riya Adsul Roll No. B044,B073,B074,B100 BTech (Computer Engineering), IV semester understand that plagiarism is defined as anyone or combination of the following:

1. Un-credited verbatim copying of individual sentences, paragraphs or illustration (such as graphs, diagrams, etc.) from any source, published or unpublished, including the internet.
2. Un-credited improper paraphrasing of pages paragraphs (changing a few words phrases, or rearranging the original sentence order)
3. Credited verbatim copying of a major portion of a paper (or thesis chapter) without clear delineation of who did wrote what. ( Source: IEEE, The institute, Dec. 2004)
4. I have made sure that all the ideas, expressions, graphs, diagrams, etc., that are not a result of my work, are properly credited. Long phrases or sentences that had to be used verbatim from published literature have been clearly identified using quotation marks.
5. I affirm that no portion of my work can be considered as plagiarism and I take full responsibility if such a complaint occurs. I understand fully well that the guide of the seminar/ project report may not be in a position to check for the possibility of such incidences of plagiarism in this body of work.

Signature of the Students: \_\_\_\_\_\_\_\_\_\_, \_\_\_\_\_\_\_\_\_\_\_\_\_, \_\_\_\_\_\_\_\_\_\_\_\_, \_\_\_\_\_\_\_\_\_\_\_\_\_\_

Names Aaryan Sarnaik, Devika Suryawanshi, Rishabh Tulshyan , Riya Adsul

Roll Nos. : B044,B073,B074,B100,B101

Place: Mumbai

Date: 06/04/2021

**CERTIFICATE**

This is to certify that the project entitled “FRESH FIELDS” is the bonafide work carried out by Aaryan Sarnaik, Devika Suryawanshi, Rishabh Tulshyan , Riya Adsul of BTECH Tech, MPSTME (NMIMS), Mumbai, during the IV semester of the academic year 2020-2021, in partial fulfillment of the requirements for the Course Database Management Systems.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Prof. Kamal Mistry

Internal Mentor

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Examiner 1 Examiner 2

**Table of Contents**

**Chapter 1**: Introduction to the system

1.1 Introduction

1.2 Problem Statement

1.3 Users of the system

**Chapter 2**: System Design and constraints

2.1 ER Model

2.2 Reduction of ER model to Relational Model

2.3 Schema Diagram

2.4 Constraints

2.5 Normalization techniques applied on relational model

**Chapter 3**: Implementation

3.1 Hardware and Software details (Front end and Back-end details)

3.2 Tools or library used

3.3 Screenshots and description

3.4 code implemented for the database

**Chapter 4**: Conclusion and Future work

* 1. **Introduction**

In today’s world, or any item to reach from the farmer to the consumer involves a lot of steps and middlemen which can cause the process to slow down and also cause hoarding of prices where only the farmer and consumer are at a loss.

Hoarding of the agricultural production leads to a hike in the price of the crop/end product. Hence, there is a need for a platform that will help the authorities keep a track of the trade, and take down any such hoarding practices being performed at any stage.

Our Idea was to create such a platform for the farmers connecting them directly to the consumers and workers thus expanding their business. This platform will allow these stakeholders to interact with each other and will facilitate the hiring of labor and purchase and sale of raw materials and produce respectively. Fresh Fields enables farmers to sell their products online. We provide technology and services to the farmers, merchants, and farm laborers, thus providing them with a wider great experience with an easy-to-use application.

* 1. **Problem Statement**

Creating a digital environment for the food supply chain to eliminate hording and other unnecessary influence and providing an easy-to-use interface which directly connects the farmers to the consumers and workers.

* 1. **Users**

Farmers can sell their produce online which can be purchased by wholesalers and retailers, and can also purchase farming products.

• There are three types of users: farmers, wholesalers/retailers/consumer and workers. The login ID and password is required to login to the system.

• The articles and statistics sections help farmers to gain knowledge.

• It has a history section which help the farmers keep a track of their uploaded produce and the customer to view the produce he/she has bought.

• It also helps the farm laborer’s in finding jobs.

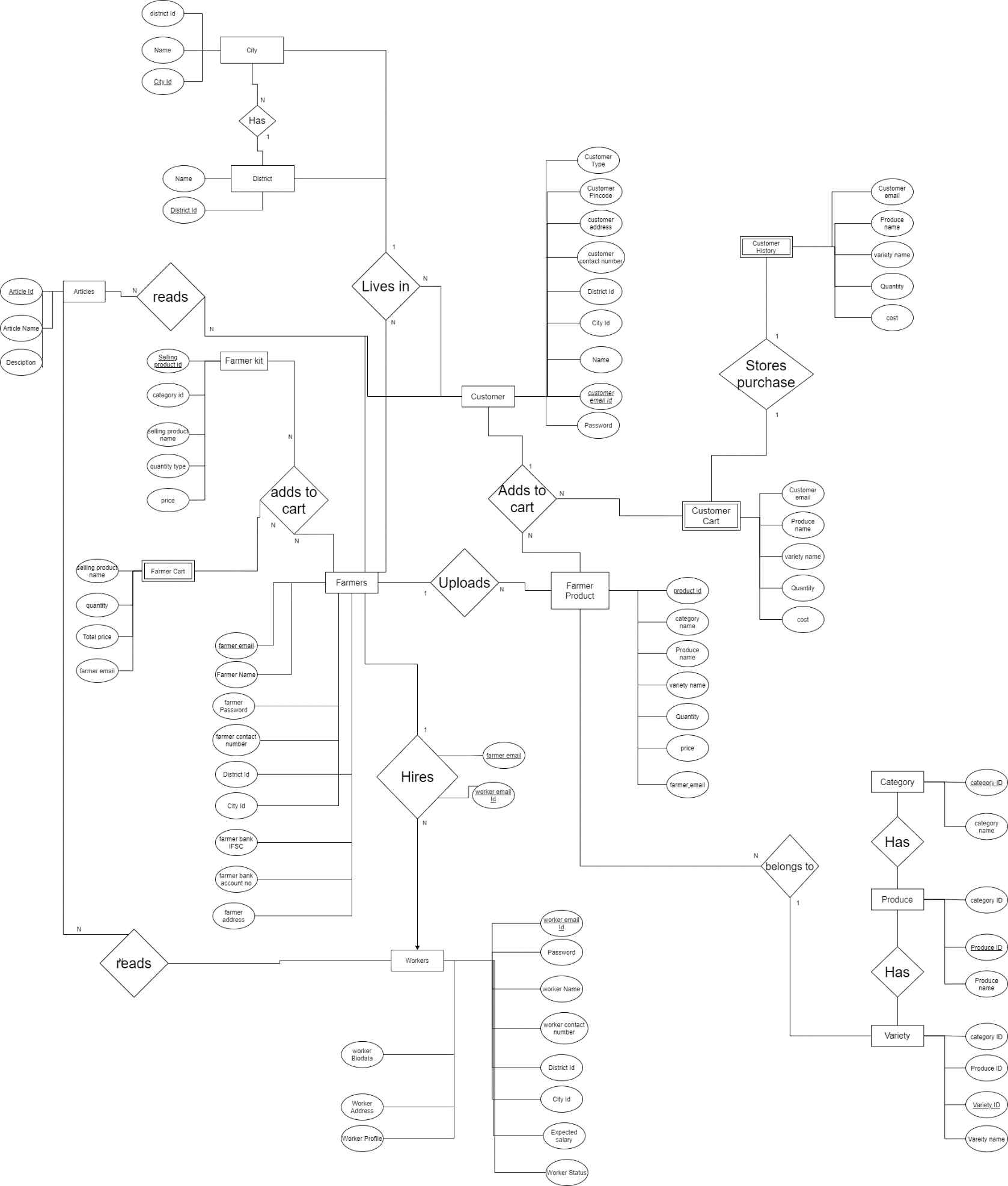
The scope of this application currently includes all the cities and districts present in the state of Maharashtra

**2) Database Tables**

The database for fresh fields is organized into 15 tables:

1. article
2. category
3. produce
4. variety
5. city
6. district
7. customer
8. customer\_cart
9. customer\_history
10. farmer\_product
11. farmer\_cart
12. farmer\_worker
13. farmer
14. farmer\_kit
15. worker

**2.1) ER DIAGRAM**



**2.2 Reduction of ER model to Relational Model**

**ARTICLE**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| article\_name | **article\_id** | Article\_description | Publish\_date | title | status |
|  |  |  |  |  |  |

**CATEGORY**

|  |  |  |
| --- | --- | --- |
| **Category\_id** | Category\_name | Category\_type |
|  |  |  |

**CITY**

|  |  |  |
| --- | --- | --- |
| **City\_id** | District\_id | City\_name |
|  |  |  |

**CUSTOMER**

|  |  |  |  |
| --- | --- | --- | --- |
| Customer\_name | Customer\_contact\_number | **Customer\_email** | Customer\_password |
|  |  |  |  |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| District\_id | City\_id | Customer\_pincode | Customer\_type | Customer\_address |
|  |  |  |  |  |

**CUSTOMER\_CART**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Product\_name | Variety\_name | quantity | cost | Customer\_email |
|  |  |  |  |  |

**CUSTOMER\_HISTORY**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Product\_name | Variety\_name | quantity | cost | Customer\_email |
|  |  |  |  |  |

**FARMER**

|  |  |  |  |
| --- | --- | --- | --- |
| Farmer\_name | Farmer\_password | **Farmer\_email** | Farmer\_contact\_number |
|  |  |  |  |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| District\_id | City\_id | Farmer\_bank\_IFSC | Farmer\_\_bank\_acno | Farmer\_\_address |
|  |  |  |  |  |

**DISTRICT**

|  |  |
| --- | --- |
| **District\_id** | District\_name |
|  |  |

**FARMER\_CART**

|  |  |  |  |
| --- | --- | --- | --- |
| Product\_name | Quantity | Total\_price | Farmer\_email |
|  |  |  |  |

**FARMER\_KIT**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Selling\_product\_id** | Category\_id | Selling\_product\_name | Kit\_quantity\_type | Kit\_price |
|  |  |  |  |  |

**FARMER\_PRODUCT**

|  |  |  |  |
| --- | --- | --- | --- |
| **Product\_id** | Category\_name | Produce\_name | Variety\_name |
|  |  |  |  |

|  |  |  |
| --- | --- | --- |
| Quantity | Product\_price | Farmer\_email |
|  |  |  |

**FARMER\_WORKER**

|  |  |
| --- | --- |
| **Worker\_email** | Farmer\_email |
|  |  |

**PRODUCE**

|  |  |  |
| --- | --- | --- |
| Category\_id | **Produce\_id** | Produce\_name |
|  |  |  |

**VARIETY**

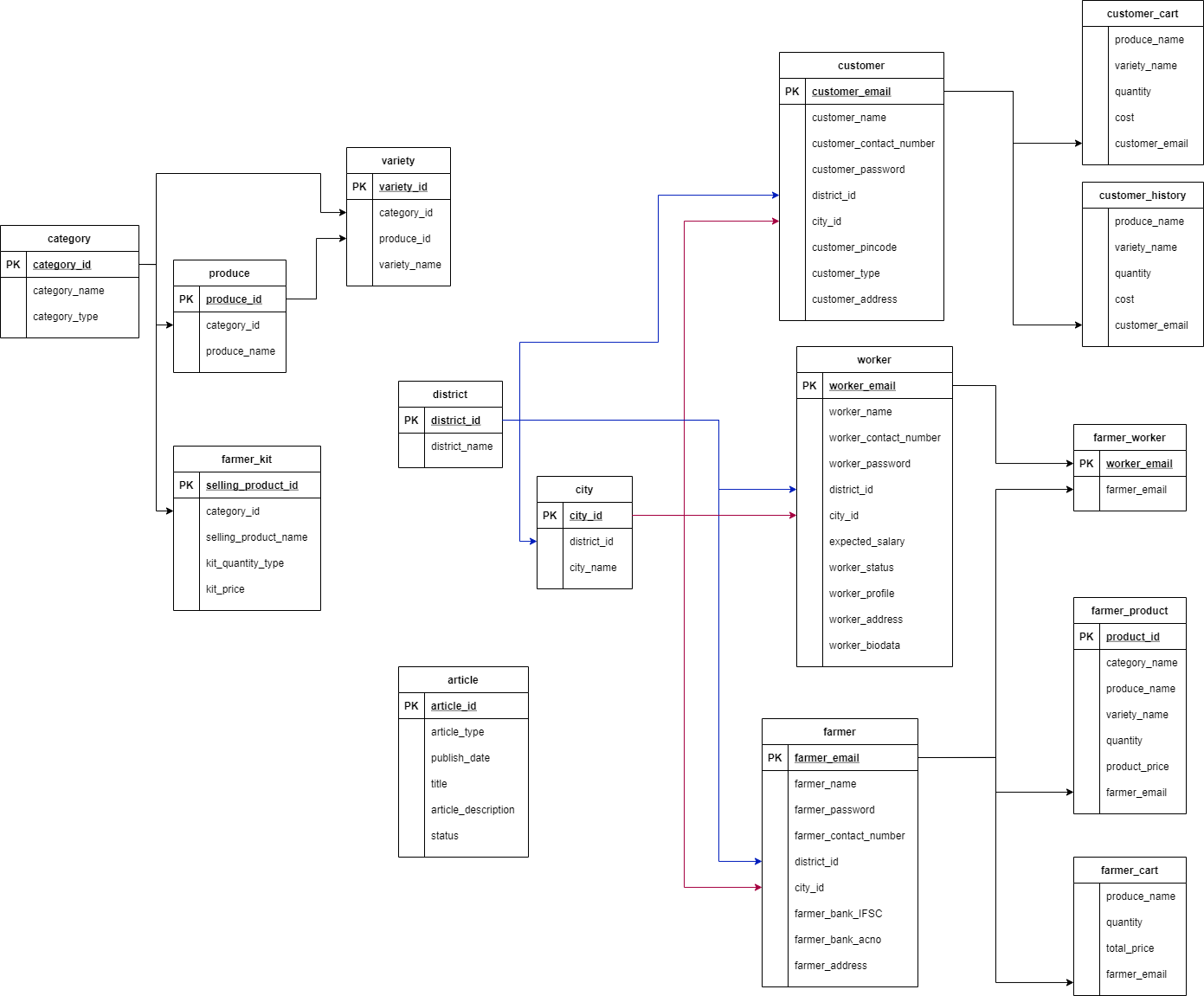
|  |  |  |  |
| --- | --- | --- | --- |
| Category\_id | Produce\_id | **Variety\_id** | Variety\_name |
|  |  |  |  |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Worker\_name | Worker\_contact\_number | **Worker\_email** | Worker\_password | District\_id | City\_id |
|  |  |  |  |  |  |

**WORKER**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Expected\_salary | Worker\_status | Worker\_profile | Worker\_address | Worker\_biodata |
|  |  |  |  |  |

**2.3) Schema diagram**



**2.4) Constraints:**

|  |  |  |  |
| --- | --- | --- | --- |
|  | TABLES | PRIMARY KEY | FOREIGN KEY |
| 1 | article | article\_id | nil |
| 2 | category | category\_id | nil |
| 3 | produce | produce\_id | category\_id |
| 4 | variety | variety\_id | category\_id, produce\_id |
| 5 | city | city\_id | district\_id |
| 6 | district | district\_id | nil |
| 7 | customer | customer\_email | district\_id, city\_id |
| 8 | customer\_cart  (volatile table, clears after customers presses buy now) | Not a relation | customer\_email |
| 9 | customer\_history | customer\_history\_id | customer\_email |
| 10 | farmer\_product | produce\_id | farmer\_email |
| 11 | farmer\_cart  (volatile table, clears after farmer presses buy now) | Not a relation | farmer\_email |
| 12 | farmer\_worker | worker\_email | farmer\_email |
| 13 | farmer | farmer\_email | district\_id, city\_id |
| 14 | farmer\_kit | selling\_product\_id | category\_id |
| 15 | worker | worker\_email | district\_id, city\_id |

**2.5) Normalization techniques applied on relational model**

1. **article**

* 1NF
* Each table cell contains a single value
* Each record of the cell is unique
* 2NF
* It is already in 1NF
* Primary key: (article\_id) consists of only one attribute
* 3NF
* It is already in 2NF
* It has no functional dependencies

1. **Category**

* 1NF
* Each table cell contains a single value
* Each record of the cell is unique
* 2NF
* It is already in 1NF
* Primary key: (category\_id) consists of only one attribute
* 3NF
* It is already in 2NF
* It has no functional dependencies

1. **Produce**

* 1NF
* Each table cell contains a single value
* Each record of the cell is unique
* 2NF
* It is already in 1NF
* Primary key: (produce\_id) consists of only one attribute
* 3NF
* It is already in 2NF
* It has no functional dependencies

1. **Variety**

* 1NF
* Each table cell contains a single value
* Each record of the cell is unique
* 2NF
* It is already in 1NF
* Primary key: (variety\_id) consists of only one attribute
* 3NF
* It is already in 2NF
* It has no functional dependencies

1. **City**

* 1NF
* Each table cell contains a single value
* Each record of the cell is unique
* 2NF
* It is already in 1NF
* Primary key: (city\_id) consists of only one attribute
* 3NF
* It is already in 2NF
* It has no functional dependencies

1. **District**

* 1NF
* Each table cell contains a single value
* Each record of the cell is unique
* 2NF
* It is already in 1NF
* Primary key: (district\_id) consists of only one attribute
* 3NF
* It is already in 2NF
* It has no functional dependencies

1. **Customer**

* 1NF
* Each table cell contains a single value
* Each record of the cell is unique
* 2NF
* It is already in 1NF
* Primary key: (customer\_ email) consists of only one attribute

It is not in 3NF as district id can be derived from city id.

Therefore, transitivity exists, so it is in 2NF

To remove transitivity, we have introduced a new table city which has city\_id and district\_id

1. **customer\_history**

* 1NF
* Each table cell contains a single value
* Each record of the cell is unique
* 2NF
* It is already in 1NF
* Primary key: (customer\_ email) consists of only one attribute
* 3NF
* It is already in 2NF
* It has no functional dependencies

1. **farmer\_product**

* 1NF
* Each table cell contains a single value
* Each record of the cell is unique
* 2NF
* It is already in 1NF
* Primary key: (product\_id) consists of only one attribute
* 3NF
* It is already in 2NF
* It has no functional dependencies

1. **farmer\_worker**

* 1NF
* Each table cell contains a single value
* Each record of the cell is unique
* 2NF
* It is already in 1NF
* Primary key: (worker\_ email) consists of only one attribute
* 3NF
* It is already in 2NF
* It has no functional dependencies

1. **Farmer**

* 1NF
* Each table cell contains a single value
* Each record of the cell is unique
* 2NF
* It is already in 1NF
* Primary key: (farmer\_email) consists of only one attribute

It is not in 3NF as district id can be derived from city id.

Therefore transitivity exists, so it is in 2NF

To remove transitivity, we have introduced a new table city which has city\_id and district\_id

1. **farmer\_kit**

* 1NF
* Each table cell contains a single value
* Each record of the cell is unique
* 2NF
* It is already in 1NF
* Primary key: (product\_id) consists of only one attribute
* 3NF
* It is already in 2NF
* It has no functional dependencies

1. **Worker**

* 1NF
* Each table cell contains a single value
* Each record of the cell is unique
* 2NF
* It is already in 1NF
* Primary key: (worker\_ email) consists of only one attribute

It is not in 3NF as district id can be derived from city id.

Therefore transitivity exists, so it is in 2NF

To remove transitivity, we have introduced a new table city which has city\_id and district\_id

**3) Implementation**

**3.1) SOFTWARES AND MODULES USED**

**SOFTWARE REQUIREMENTS**

* Operating system – **Windows 10** is used as the operating system as it is
* stable and supports more features and is more user friendly.
* Database Management System – **DB Browser for SQLite** (DB4S) is a high quality, visual, open-source tool to create, design, and edit database files compatible with SQLite.
* English language. The queries are easy to understand and to write.
* Development tools and Programming language **– Python** is used to write

the whole code and **Python Tkinter** is used for designing and styling of

front end.

**3.2) TOOLS AND LIBRARIES USED**

**TOOLS**

* PyCharm IDE is used for writing the code and developing the system.
* **DB Browser for SQLite** Libraries
* **Canva-** Used to edit images used in the background and on buttons

**LIBRARIES**

The following python libraries have been used:

* **tkinter** – Tkinter is the standard GUI (Graphic User Interface) library for

Python. Tkinter provides a fast and easy way to create GUI applications.

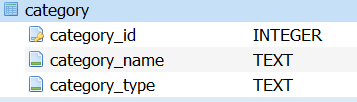
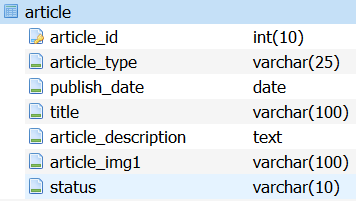
* **PIL(Image, ImageTk):** Python Imaging Library is a free and open-source additional library for the Python programming language that adds support for opening, manipulating, and saving many different image file formats.
* **Pandas** (DataFrame) :

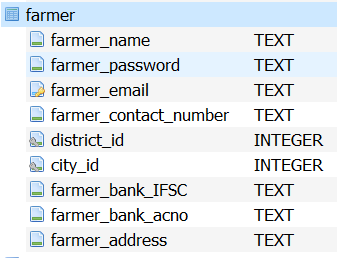
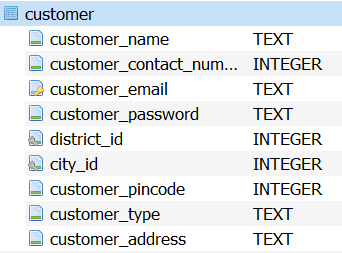
**pandas** is a fast, powerful, flexible and easy to use open source data analysis and manipulation tool, built on top of the **Python** programming language.

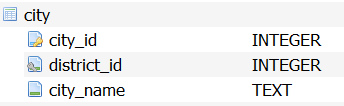
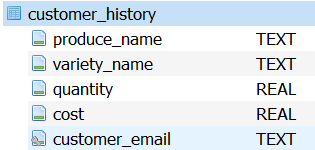
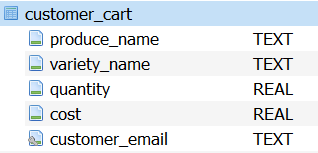
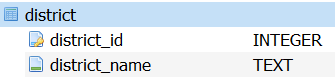
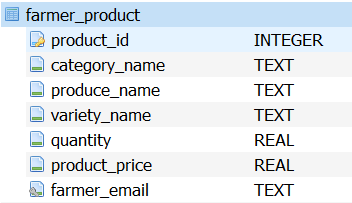
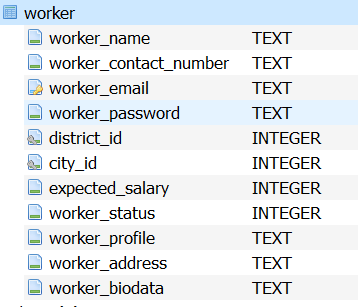
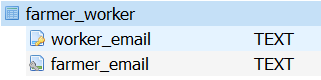
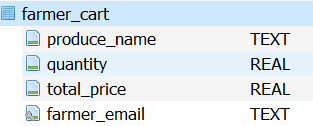
* **Matplotlib:** It is a comprehensive library for creating static, animated, and interactive visualizations in **Python**

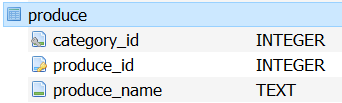
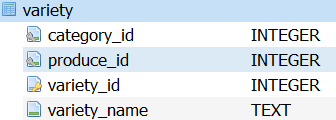
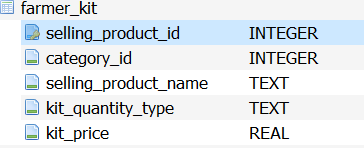
**SQLite 3** : SQLite is a relational database management system contained in a C library. In contrast to many other database management systems, SQLite is not a client–server database engine**.**

**3.3) Screenshots and description**





**3.4) Code implemented for the database**

**Data manipulation queries are present in the code (main.py file)**

**Insertion queries (used to make the data base)**

import os  
import sqlite3  
import pandas as pd  
  
connection = sqlite3.connect("FreshFields.db")  
cursor = connection.cursor()  
  
cursor.execute('CREATE TABLE IF NOT EXISTS category (category\_id INTEGER NOT NULL PRIMARY KEY, category\_name TEXT NOT NULL,category\_type TEXT NOT NULL)')  
  
cursor.execute('CREATE TABLE IF NOT EXISTS produce (category\_id INTEGER NOT NULL, produce\_id INTEGER NOT NULL PRIMARY KEY, produce\_name TEXT NOT NULL,FOREIGN KEY (category\_id) REFERENCES category (category\_id))')  
  
cursor.execute('CREATE TABLE IF NOT EXISTS variety (category\_id INTEGER NOT NULL, produce\_id INTEGER NOT NULL,variety\_id INTEGER NOT NULL PRIMARY KEY, variety\_name TEXT NOT NULL ,FOREIGN KEY (category\_id) REFERENCES category (category\_id),FOREIGN KEY (produce\_id) REFERENCES produce (produce\_id))')  
  
cursor.execute('CREATE TABLE IF NOT EXISTS district (district\_id INTEGER NOT NULL PRIMARY KEY, district\_name TEXT NOT NULL)')  
  
cursor.execute('CREATE TABLE IF NOT EXISTS city (city\_id INTEGER NOT NULL PRIMARY KEY,district\_id INTEGER NOT NULL, city\_name TEXT NOT NULL,FOREIGN KEY (district\_id) REFERENCES district (district\_id))')  
  
cursor.execute('CREATE TABLE IF NOT EXISTS customer (customer\_name TEXT NOT NULL,customer\_contact\_number INTEGER NOT NULL,customer\_email TEXT NOT NULL PRIMARY KEY,customer\_password TEXT NOT NULL,district\_id INTEGER NOT NULL,city\_id INTEGER NOT NULL, customer\_pincode INTEGER NOT NULL ,customer\_type TEXT NOT NULL,customer\_address TEXT NOT NULL,FOREIGN KEY (district\_id) REFERENCES district (district\_id),FOREIGN KEY (city\_id) REFERENCES city (city\_id) )')  
  
cursor.execute('CREATE TABLE IF NOT EXISTS farmer (farmer\_name TEXT NOT NULL,farmer\_password TEXT NOT NULL,farmer\_email TEXT NOT NULL PRIMARY KEY,farmer\_contact\_number TEXT NOT NULL,district\_id INTEGER NOT NULL,city\_id INTEGER NOT NULL,farmer\_bank\_IFSC TEXT NOT NULL,farmer\_bank\_acno TEXT NOT NULL,farmer\_address TEXT NOT NULL,FOREIGN KEY (district\_id) REFERENCES district (district\_id),FOREIGN KEY (city\_id) REFERENCES city (city\_id) )')  
  
cursor.execute('CREATE TABLE IF NOT EXISTS worker (worker\_name TEXT NOT NULL,worker\_contact\_number TEXT NOT NULL, worker\_email TEXT NOT NULL PRIMARY KEY,worker\_password TEXT NOT NULL,district\_id INTEGER NOT NULL,city\_id INTEGER NOT NULL,expected\_salary INTEGER NOT NULL,worker\_status INTEGER NOT NULL DEFAULT 0, worker\_profile TEXT NOT NULL,worker\_address TEXT NOT NULL,worker\_biodata TEXT, FOREIGN KEY (district\_id) REFERENCES district (district\_id),FOREIGN KEY (city\_id) REFERENCES city (city\_id) )')  
  
cursor.execute('CREATE TABLE IF NOT EXISTS farmer\_product (product\_id INTEGER PRIMARY KEY AUTOINCREMENT,category\_name TEXT NOT NULL, produce\_name TEXT NOT NULL,variety\_name TEXT NOT NULL,quantity REAL NOT NULL,product\_price REAL NOT NULL, farmer\_email TEXT NOT NULL, FOREIGN KEY (farmer\_email) REFERENCES farmer (farmer\_email))')  
  
cursor.execute('CREATE TABLE IF NOT EXISTS farmer\_cart (produce\_name TEXT NOT NULL, quantity REAL NOT NULL,total\_price REAL NOT NULL,farmer\_email TEXT NOT NULL ,FOREIGN KEY (farmer\_email) REFERENCES farmer (farmer\_email))')  
  
cursor.execute('CREATE TABLE IF NOT EXISTS farmer\_kit (selling\_product\_id INTEGER NOT NULL PRIMARY KEY,category\_id INTEGER NOT NULL, selling\_product\_name TEXT NOT NULL, kit\_quantity\_type TEXT NOT NULL,kit\_price REAL NOT NULL)')  
  
cursor.execute('CREATE TABLE IF NOT EXISTS farmer\_worker (worker\_email TEXT NOT NULL PRIMARY KEY, farmer\_email TEXT NOT NULL,FOREIGN KEY (worker\_email) REFERENCES worker (worker\_email) ON DELETE CASCADE ON UPDATE CASCADE,FOREIGN KEY (farmer\_email) REFERENCES farmer (farmer\_email) ON DELETE CASCADE ON UPDATE CASCADE)')  
  
cursor.execute('CREATE TABLE IF NOT EXISTS customer\_cart (produce\_name TEXT NOT NULL,variety\_name TEXT NOT NULL,quantity REAL NOT NULL, cost REAL NOT NULL, customer\_email TEXT NOT NULL,FOREIGN KEY (customer\_email) REFERENCES customer (customer\_email))')  
  
cursor.execute('CREATE TABLE IF NOT EXISTS customer\_history (customer\_history\_id INTEGER PRIMARY KEY NOT NULL AUTOINCREMENT ,produce\_name TEXT NOT NULL,variety\_name TEXT NOT NULL,quantity REAL NOT NULL, cost REAL NOT NULL, customer\_email TEXT NOT NULL,FOREIGN KEY (customer\_email) REFERENCES customer (customer\_email))')  
  
# Creating Articles Table  
  
cursor.execute("""  
 CREATE TABLE article (  
 article\_id int(10) NOT NULL PRIMARY KEY,  
 article\_type varchar(25) NOT NULL,  
 publish\_date date NOT NULL,  
 title varchar(100) NOT NULL,  
 article\_description text NOT NULL,  
 article\_img1 varchar(100) NOT NULL,  
 status varchar(10) NOT NULL  
)  
""")  
  
connection.commit()  
  
df = pd.read\_csv("database/kit.csv", usecols = ['selling\_product\_id','category\_id', 'selling\_product\_name', 'kit\_quantity\_type','kit\_price'])  
df.to\_sql('farmer\_kit', connection, if\_exists='append', index=False)  
df = pd.read\_csv("database/category.csv", usecols = ['category\_id', 'category\_name', 'category\_type'])  
df.to\_sql('category', connection, if\_exists='append', index=False)  
df = pd.read\_csv("database/produce.csv", usecols = ['category\_id', 'produce\_id', 'produce\_name'])  
df.to\_sql('produce', connection, if\_exists='append', index=False)  
df = pd.read\_csv("database/variety.csv", usecols = ['category\_id', 'produce\_id','variety\_id','variety\_name'])  
df.to\_sql('variety', connection, if\_exists='append', index=False)  
df = pd.read\_excel("database/city.xlsx", usecols = ['city\_id', 'district\_id','city\_name'])  
df.to\_sql('city', connection, if\_exists='append', index=False)  
df = pd.read\_excel("database/district.xlsx", usecols = ['district\_id','district\_name'])  
df.to\_sql('district', connection, if\_exists='append', index=False)  
df = pd.read\_excel("database/farmer.xlsx", usecols = ["farmer\_name" ,"farmer\_password","farmer\_email","farmer\_contact\_number","district\_id","city\_id","farmer\_bank\_IFSC","farmer\_bank\_acno","farmer\_address"])  
df.to\_sql('farmer', connection, if\_exists='append', index=False)  
df = pd.read\_excel("database/customer.xlsx", usecols = ["customer\_name","customer\_contact\_number","customer\_email","customer\_password","district\_id","city\_id","customer\_pincode","customer\_type","customer\_address"])  
df.to\_sql('customer', connection, if\_exists='append', index=False)  
df = pd.read\_excel("database/worker.xlsx", usecols = ["worker\_name","worker\_contact\_number","worker\_email","worker\_password","district\_id","city\_id","expected\_salary","worker\_status","worker\_profile","worker\_address","worker\_biodata"])  
df.to\_sql('worker', connection, if\_exists='append', index=False)

**4) Conclusion and Future Scope**

This Project will thus pave the way for an efficient means to carry out the buying and selling of the products. Farmers will earn money as per the work they have done and will not suffer losses. Also, the system it completely online (webapp/app) thereby reducing the price aspect of the system tremendously. This system is proposed to replace the existing system where the farmer has to suffer between the manufacturers and the traders. Also, the main advantage of this project is that it uses Information Technology. The User only needs basic products like a Computer and an internet connection.

Our application can we converted into an android/ios application thus making it easily accessible to the majority population of India.

There are many ways in which this application could be updated to be more efficient and have a wider scope in the future. Some of them are:

1. Readability of the application as a whole in India could be improved by adding a feature to translate its pages into local languages like Hindi and Marathi.
2. Connectivity across the users on the application could be enhanced by adding modules like Photo Uploading to allow the farmers to display their crops, or a messaging portal where the users could communicate with each other. There could also be Audio/Video calls that would be made available to the users.
3. Transportation, travel and other miscellaneous commodities could be incorporated into the app itself so the users would not need to refer to third-parties for it.
4. In app crop testing and quality check facility can be integrated in the app

**Societal Applications**

Our application helps to provide the hardworking farmers of our country the business and clients that they need and most of all, deserve. It is free and open-source and easily accessible to even someone who is not technically sound. It helps the wholesalers and retailers in buying produce directly from a large number of farmers. Thereby, it enables the wholesalers and retailers in expanding their business. It features online shopping for fertilizers, pesticides, machinery & tools, etc. It enables farmers to hire laborers, which in turn, will help the farm laborers to find small jobs by having a work profile on the website. As a whole, ‘Fresh Fields’ provides a concept of virtual agricultural trade to its users.

SCREENSHOTS OF PROJECT ARE PROVIDED IN THE POWERPOINT PRESENTATION