CHAPTER 1

INTRODUCTION

The purpose of Farmer's E Market is to automate the existing manual system by the help of 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. The required software and hardware are easily available and easy to work with Farmer's E Market, as described above, can lead to error free, secure, reliable and fast management system. It can assist the user to concentrate on their other activities rather to concentrate on the record keeping. Thus it will help organization 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. Basically the project describes how to manage for good performance and better services for the clients.

The main objective of Farmer's E Market is to manage the information about Products, Customer Details, Purchase History, Stock Details etc. The project is totally built at administrative end and thus only the administrator is guaranteed the access to database. The purpose of the project is to build an application program to reduce the manual work for managing the Order, Stock, Product Order, Purchase history.

1.1 Company Profile

HACKBAL BUSINESS SOLUTIONS Pvt Ltd is an IT solution provider specialised in the development of software and web applications, aiming to deliver quality services to customers.

We design and develop serious software for ambitious clients. Digital Marketing, Software Development, Designing… over the past years we've been on the leading edge of these and many other trends, all while staying true to our enduring principles.

While we would be building software even if it weren't our job, it is our job, and we take it very seriously. We have a duty to build the best software we can for our clients and their users. At the same time, professional integrity requires that we acknowledge that the software we write can always be better.

We're glad that software engineering and design isn't merely about pretty pictures and algorithms on a whiteboard. The products we build run real businesses and serve real users. We like it that way, because, while we can be pretty geeky, we actually like people, so we want to build useful software that makes them happy.

1.2 Statement of the problem

As we step forward into the modern era of technology, we may find many engineering related applications very beneficial for improvements into the society. This is the world of technology where people use smart phones for completing their daily tasks like shopping, paying bills, managing work and much more. The idea of this project is to add its features into the lives of the people so that the food products which they buy, can be bought directly from the farm so that the profit can reach directly to the farmers. Because in India we follow a supply chain of farm product making things too much indirect for the farmers due to which the farmer still reaming poor and the intermediates are gaining profit which ultimately makes them rich. So in order to break that supply chain of indirect sales, we can make use of this application so that the farmer can be connected directly to the customer and the selling can be done accordingly. Since the farmer will be dealing with the customer directly so the prices of the products offered by the farmer to the customer will also be affordable to customer, which will help both the farmer and the customer where the customer can save some money and the farmer will gain extra profit that he deserved.

CHAPTER 2

SYSTEM ANALYSIS

2.1 Present system

There is no computerized system for the farmer to sell their product. Currently, the farmer goes to nearest market handover his product to a particular agent, agent ask the farmer to visit the market after a specific time to collect the cash earned out of the sold product. Agent sells the product to another agent or a dealer at the cost of that market. Every Agent tries to cuts his commission out of that. There is no way for farmer to know about the deal and the exact amount at which their product was sold. There is no transparency. No facility is present for the farmers to know the product rates at different markets where they can sell their products for achieving high profits. Many times, farmers are not even aware of the schemes and compensation provided by government. In spite of all the opportunities banging the doors the farmers are not able to benefit out of those. Current system does not provide the way of e-learning for farmer that will provide the knowledge of new techniques in farming. So he doesn't get the maximum profit through the current system.

2.2 Limitation of present system

a) Manual process
b) Unwanted hike in Products cost
c) Less Profitable for farmers
d) More human effforts needed

2.3 Proposed system

Farmer's E Market software features are designed to emulate many of the marketing activities involved with live farmers markets. The software's online ordering capabilities and reporting forms facilitate the logistics of complex multi-producer farmers markets or simpler markets managed by a single farm. Farmer's E Market markets come with a customizable website template that allows producers to easily design their online store and begin selling their products to local customers. The software supports up to two separate marketplace accounts for producers and customers. Markets can accept online credit and debit card payments or allow customers to preorder from the online store and pay for their purchase on delivery day.

2.4 Advantages and features of proposed system

a) Farmers has more control over their products price
b) End user gets products at descent price
c) .Fast, reliable and secure
d) Less paper work needed

2.5 Feasibility Study

A feasibility study is an analysis of how successfully a project can be completed, accounting for factors that affect it such as technical, economic, behavioral, operational factors. When a new project is proposed, it normally goes through the feasibility assessment. Feasibility Study is carried out to determine whether the proposed system is possible to develop with available resources & what should be the cost of consideration.

Various types of feasibilities are,

a) Technical Feasibility
b) Economic Feasibility
c) Operational Feasibility

If the proposed system is not feasible to develop, it is rejected at this very step.

2.5 .1 Technical Feasibility

The proposed system uses the language Python. Based on this criteria, we can strongly say that it is technically feasible, since there will not be much difficulty in getting required resources for the development & maintaining system as well. All the resources needed for the development of the software as well as the maintenance of the same is available in the organization. Here we are utilizing the resources which are already available so it's very well technically feasible that we can implement flood detection system.

2.5 .2 Economic Feasibility

It is found that the benefit from our system would be more than the cost and time involved in its development. In our system the implementation cost over production is economically feasible. Economic analysis is the most

frequently used techniques for evaluating the effectiveness of the proposed system more commonly known as cost/benefit analysis the procedure is to determine the benefits and savings that are expected from a proposed system and compare them with costs.

2.5 .3 Operational Feasibility

The proposed system satisfies operational feasibility in the way that the customers needs are satisfied. The system is adaptable to the customers and acceptable to the common people who use this. Operational feasibility assesses the extent to which the required software performs a series of steps to solve business problems and user requirements. This feasibility is dependent on human resources (software development team) and involves visualizing whether the software will operate after it is developed and be operative once it is installed. Operational feasibility also performs the following tasks:

a) Determines whether the problems anticipated in user requirements are of high priority
b) Determines whether the solution suggested by the software development team is acceptable
c) Analyses whether users will adapt to a new software
d) whether the organization is satisfied by the alternative solutions proposed by the software development team.

CHAPTER 3

SYSTEM SPECIFICATION

3.1 Software Requirements

1. Operating System : Windows 8/10
2. Language : Python
3. Framework : Django
4. IDE : Visual Studio Code

3.2 Hardware Requirements

1. Processor : Intel i3 or AMD Ryzen 3
2. RAM : 3 GB
3. Hard Disk Drive : 200 GB
4. Peripherals : Keyboard, Mouse, Monitor, Camera

CHAPTER 4

SYSTEM DESIGN

4.1 Context Level Diagram

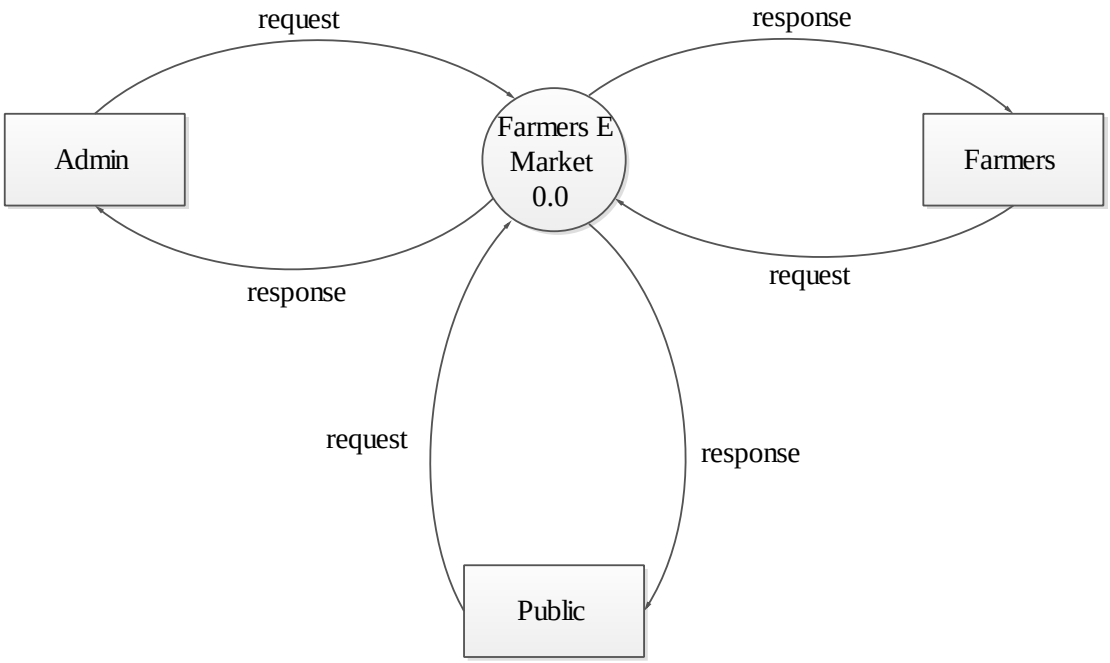


Fig 4.1 Context Level Diagram

4.2 Data Flow Diagram

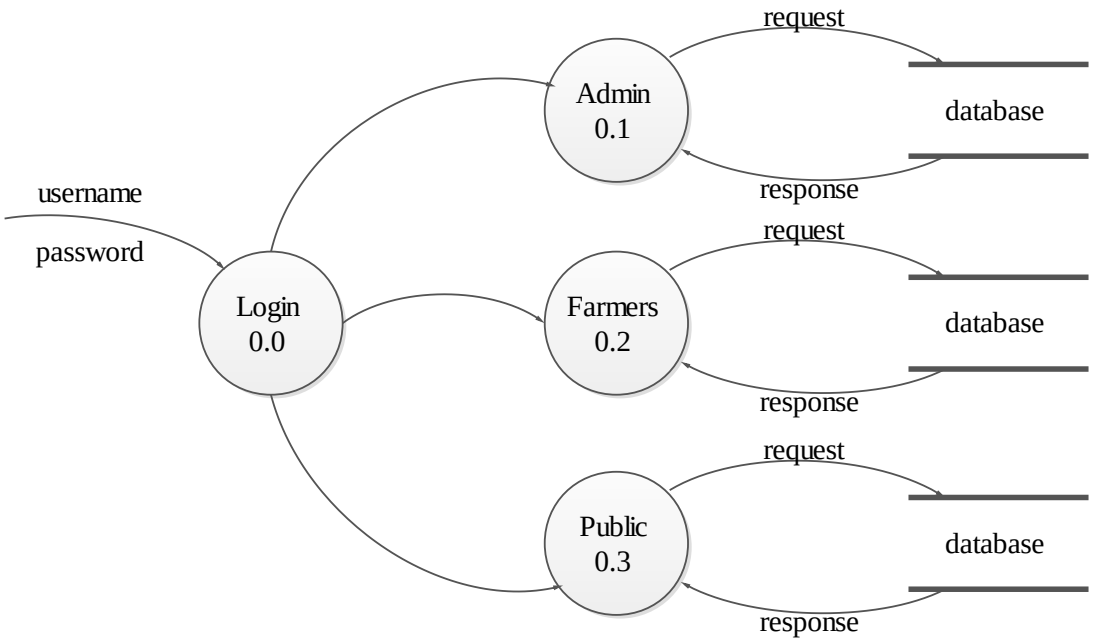


Fig 4.2.1 Level 1 DFD of Farmers E Market

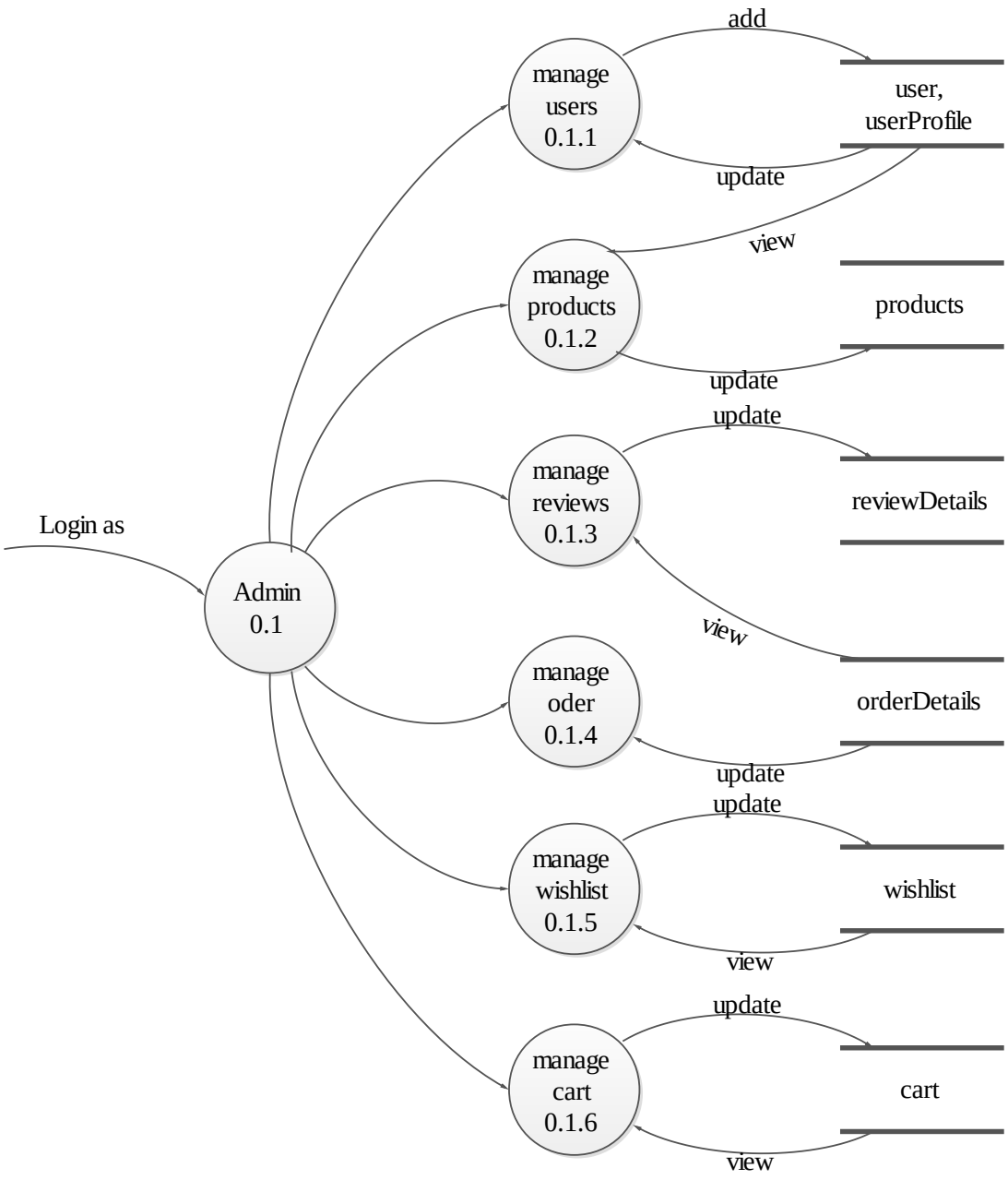


Fig 4.2.2 Level 2 DFD of Admin

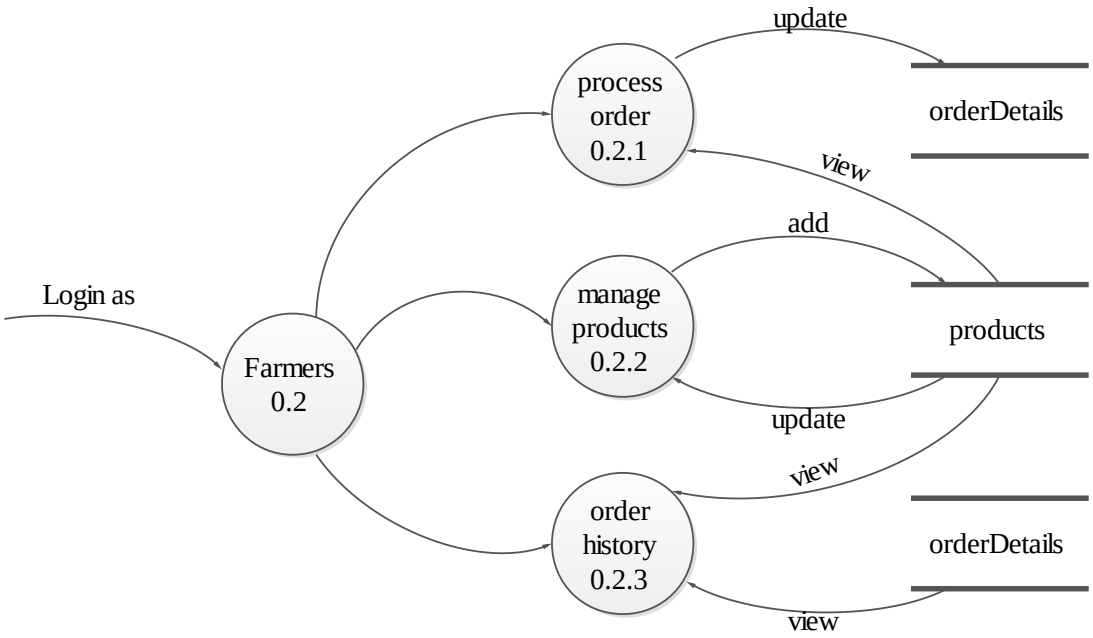


Fig 4.2.3 Level 2 DFD of Farmer

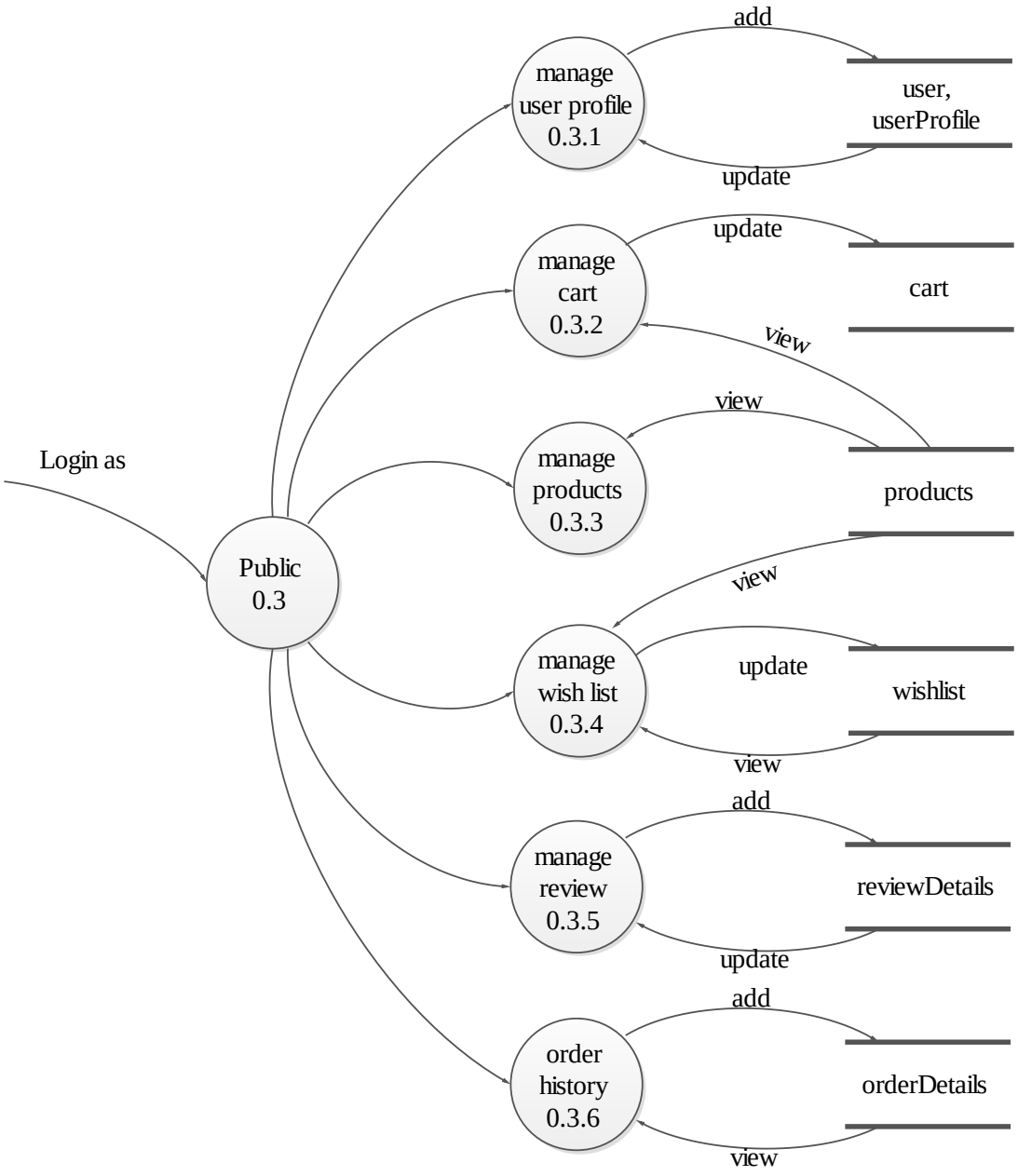
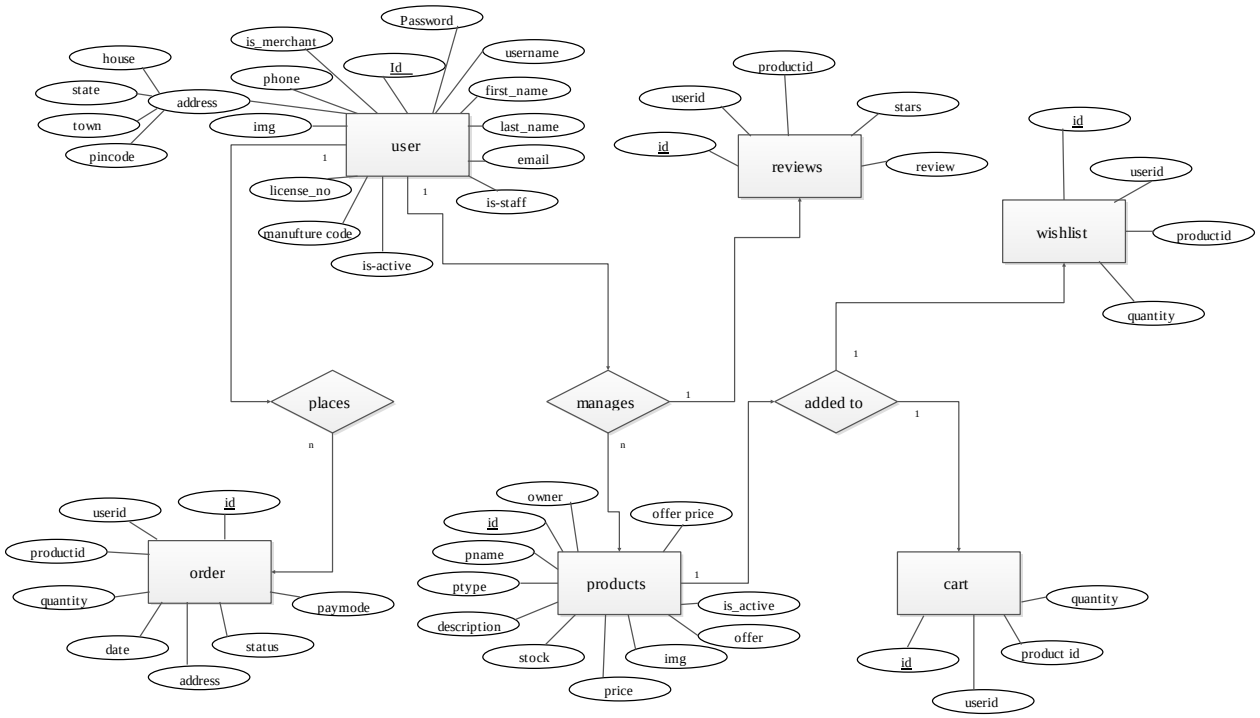


Fig 4.2.4 Level 2 DFD of Public

4.3 ER-Diagram



4.4 Database Design

Table products

Field Name	Data type	Size	Constraints	Description
Id	IntegerField		PrimaryKey	
Pname	CharField	100	Not NULL	
Ptype	CharField	10	Default	
Description	TextField		Not NULL	
Stock	IntegerField		Not NULL	
price	IntegerField		Not NULL	
img	ImageField		Not NULL	
offer	BooleanField		Default	
isactive	BooleanField		Default	
offerprice	IntegerField		Default	
created_at	DateField		Not NULL	
owner	IntegerField		Foreign key	

Table cart

Field Name	Data type	Size	Constraints	Description
Id	IntegerField		PrimaryKey	
userid	IntegerField		Foreign key	
productid	IntegerField		Foreign key	
quantity	IntegerField		Default	

Table wishlist

Field Name	Data type	Size	Constraints	Description
Id	IntegerField		PrimaryKey	
userid	IntegerField		Foreign key	
productid	IntegerField		Foreign key	
quantity	IntegerField		Default	

Table user

Field Name	Data type	Size	Constraints	Description
Id	IntegerField		PrimaryKey	
Password	CharField	128	Not NULL	
last_login	DateField	10		
is_superuser	BooleanField		Not NULL	
username	CharField	150	Not NULL	
first_name	CharField	30	Not NULL	
last_name	CharField	150	Not NULL	
email	CharField	254	Not NULL	
is_staff	BooleanField		Not NULL	
is_active	BooleanField		Not NULL	
date_joined	DateField		Not NULL	

Table userProfile

Field Name	Data type	Size	Constraints	Description
Id	IntegerField		PrimaryKey	
state	CharField	200	Not NULL	
house	CharField	200	Not NULL	
town	CharField	200	Not NULL	
pincode	CharField	12	Not NULL	
phone	CharField	12	Not NULL	
ismerchant	BooleanField		Not NULL	
description	CharField	200	Not NULL	
img	ImageField		Not NULL	
isactive	BooleanField		Not NULL	
license_no	CharField	20	Not NULL	
mafactr_code	CharField	20	Not NULL	
user	IntegerField		Foreign key	

Table orderDetails

Field Name	Data type	Size	Constraints	Description
Id	IntegerField		PrimaryKey	
userid	IntegerField		Foreign key	
productid	IntegerField		Foreign key	
quantity	IntegerField		Not NULL	
date	DateField		Not NULL	
address	CharField		Not NULL	
status	BooleanField		Default	
paymode	CharField	20	Default	

Table reviewDetails

Field Name	Data type	Size	Constraints	Description
Id	IntegerField		PrimaryKey	
userid	IntegerField		Foreign key	
productid	IntegerField		Foreign key	
stars	IntegerField		Default	
review	CharField	200	Default	

4.5 Normalization

Designing a data base is a complex task and the normalization theory is a useful and in this design process.

A bad database design may lead to certain undesirable situations such as;

1. Repetition of information
2. Inability to represent certain information
3. Loss of information

To minimize the anomalies, normalization may be used. In construction and system, here

there is only one database named result. There are tables in database.

First Normal form (1 NF)

A relation is in first normal form (1 NF), if and only if all its attributes are based on a single domain. The objective of normalizing a table is to remove its repeating groups and ensure that all entries of the resulting table have at most single value. The objective of 1NF is to divide the database into logical units called tables. When each table has been designed, primary key is assigned to most or all tables.

Second Normal form (2 NF)

A table is said to be in second normal form (2 NF), when it is in 1NF and it satisfies functional dependency. Functional of 2NF is to take data that is partially dependent on the primary key, enter the data into another table. Now consider the data base of the system. In this there are a total of table and all tables are in second normal form. That is all of these tables satisfies second normal form. No repeated information is stored in any table. All of the table. All of the tables have a separate primary key some of all are auto numbers.

Third Normal form (3 NF)

A table is said to be in 3NF, when it is 2NF and every non-key attribute is functionally

dependent only on the primary key. The objective of 3NF is to remove data in a table that is not dependent on the primary key.

All our tables are in First normalization form

Benefits of Normalization

1. To permit simple retrieval of data in response to query and report requests.
2. Helps to simplify the structure of tables.
3. To structure the data so that there is no repletion of data, that helps in saving space.
4. To simplify the maintenance of data through updates, insertions and deletions.
5. To reduce the need to restructure data when new applications requirements arise.
6. Data consistency within the database.
7. Much more flexible database design.
8. A better handle on database security.
9. Index searching is often faster, since indexes tend to be narrower and shorter.
10. More tables allow better use of segments to control physical placement of data.
11. Fewer null values and less redundant data, making your database more compact.
12. Triggers execute more quickly if you are not maintaining redundant data.

4.6 Design of Each Subsystem

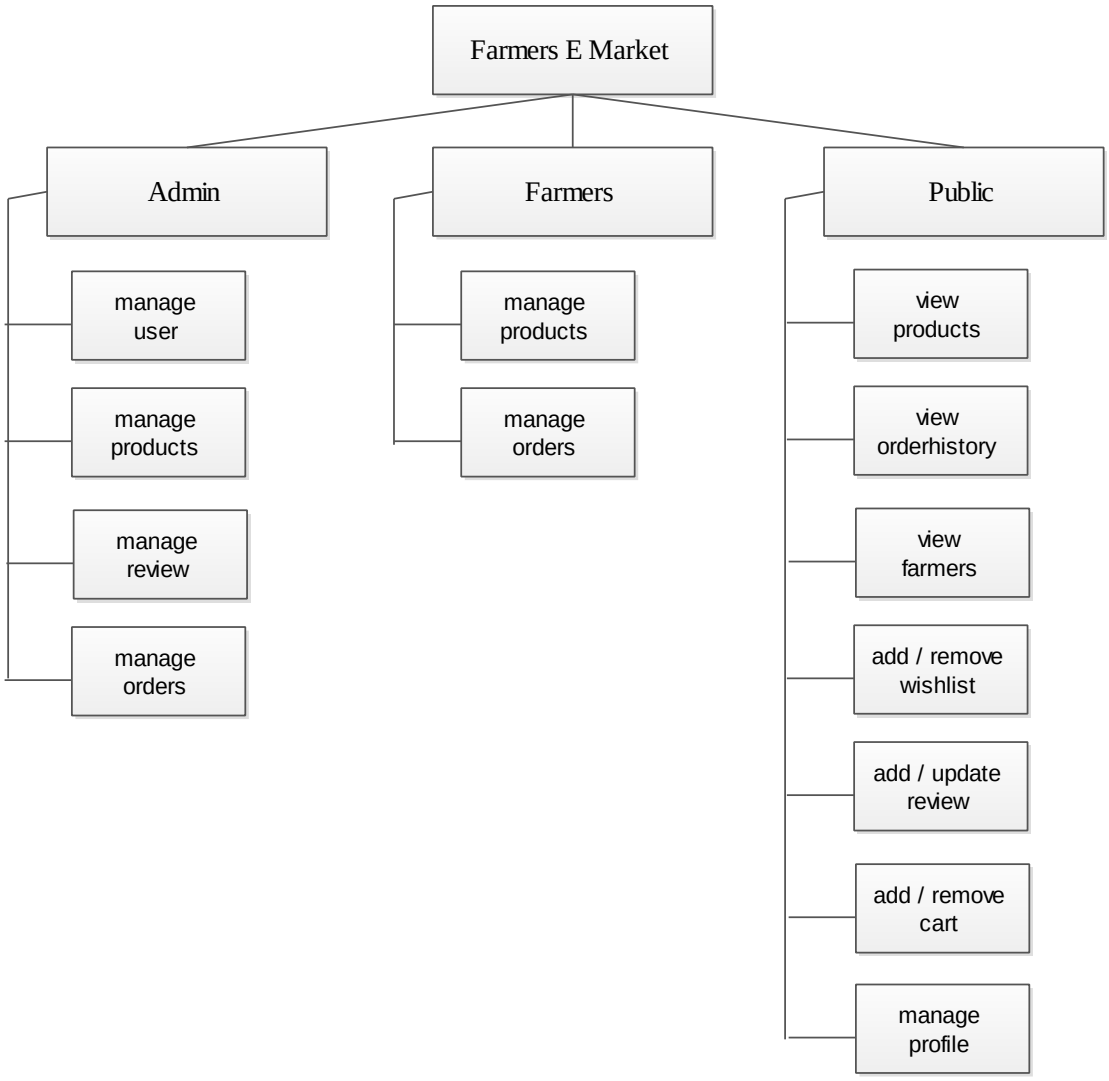


Fig 4.3 Subsystem Design

4.7 UML Diagrams

4.7.1 Use case Diagram

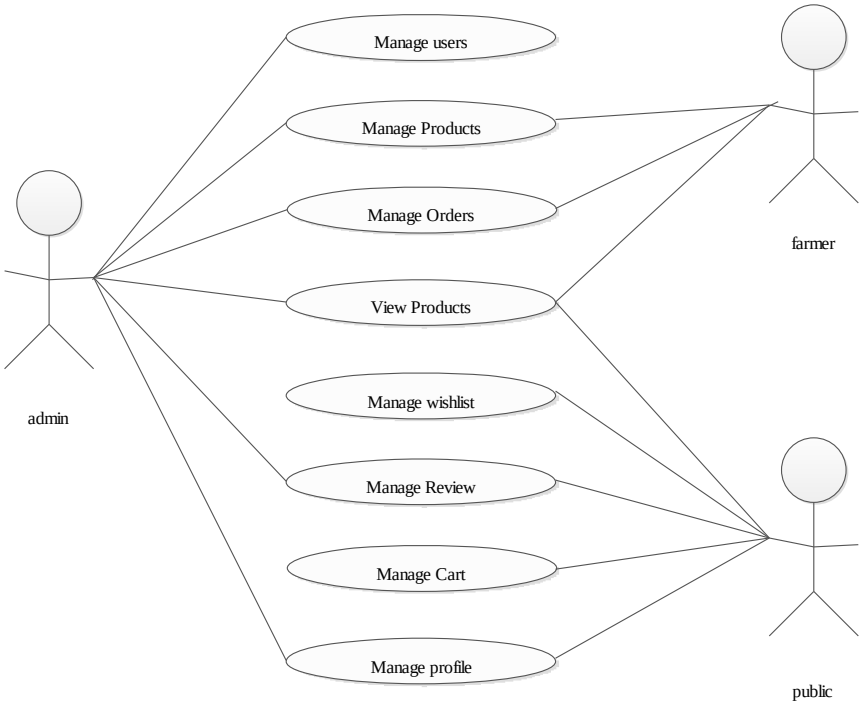


Fig 4.4.1 Use case Diagram

4.7.2 Sequence Diagram

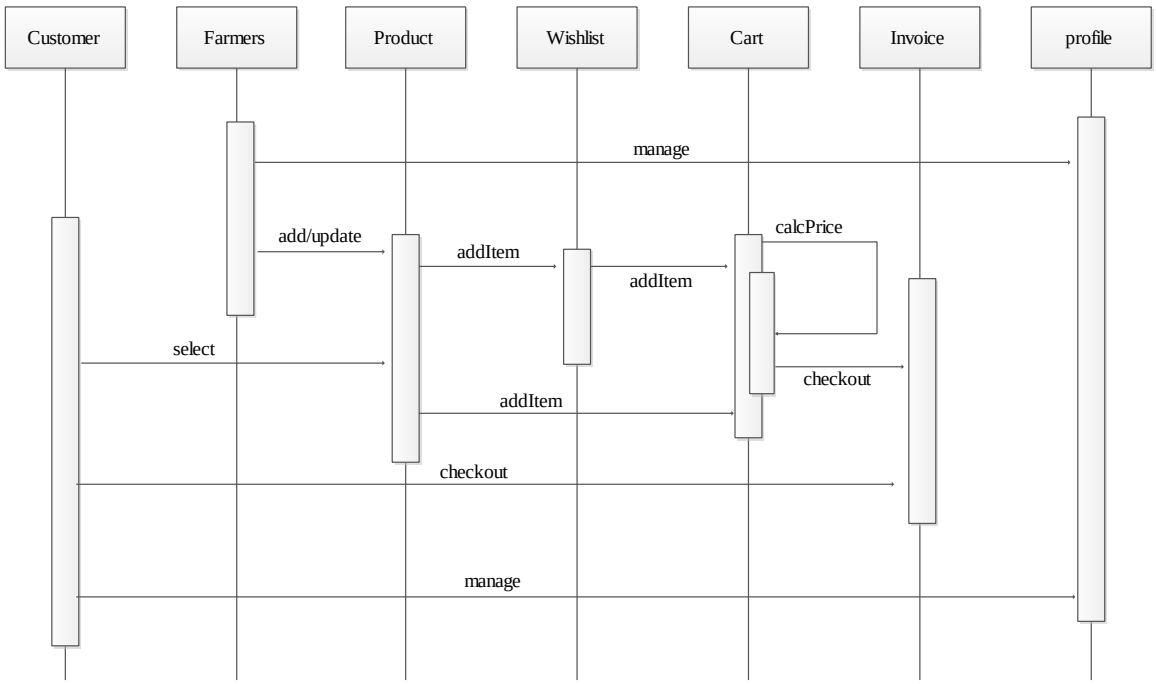


Fig 4.4.2 Sequence Diagram

CHAPTER 5

CODING

5.1 Features of Language

Python Overview

Python is a general-purpose, interpreted, high-level programming language which is widely used nowadays .It is an open source language which was developed by Guido Van Rossum in the late 1980s .Python Software Foundation (PSF), a non-profit organization, holds the intellectual property rights of Python. Python was released in 1991 at Centrum Wiskunde & Informatica (CWI) in the Netherlands as a successor to the ABC language. Rossum named this language after a popular comedy show called 'Monty Python's Flying Circus' (and not after Python-the snake).In the last few years, the popularity of python has increased immensely due to its wide applications. According to most of the tech surveys, Python is in the top ten Most Popular Technologies in 2019.

Python Features

Python is an interpreter-based language, which allows execution of one instruction at a time.

1) Extensive basic data types are supported e.g. numbers (floating point, complex, and unlimited-length long integers), strings (both ASCII and Unicode), lists, and dictionaries.
2) Variables can be strongly typed as well as dynamic typed.
3) Supports object-oriented programming concepts such as class, inheritance, objects, module, namespace etc.
4) Cleaner exception handling support.
5) Supports automatic memory management.

Python Advantages

1) Python provides enhanced readability. For that purpose, uniform indents are used to delimit blocks of statements instead of curly brackets, like in many languages such as C, C++ and Java.
2) Python is free and distributed as open-source software. A large programming community is actively involved in the development and support of Python

libraries for various applications such as web frameworks, mathematical computing and data science.

3) Python is a cross-platform language. It works equally on different OS platforms like Windows, Linux, Mac OSX etc. Hence Python applications can be easily ported across OS platforms.
4) Python supports multiple programming paradigms including imperative, procedural, object-oriented and functional programming styles.

5) Python is an extensible language. Additional functionality (other than what is provided in the core language) can be made available through modules and packages written in other languages (C, C++, Java etc)
6) A standard DB-API for database connectivity has been defined in Python. It can be enabled using any data source (Oracle, MySQL, SQLite etc.) as a backend to the Python program for storage, retrieval and processing of data.
7) Standard distribution of Python contains the Tkinter GUI toolkit, which is the implementation of popular GUI library called Tcl/Tk. An attractive GUI can be constructed using Tkinter. Many other GUI libraries like Qt, GTK, WxWidgets etc. are also ported to Python.
8) Python can be integrated with other popular programming technologies like C, C++, Java, ActiveX and CORBA.

Django

Django is a high-level Python Web framework that encourages rapid development and clean, pragmatic design. Built by experienced developers, it takes care of much of the hassle of Web development, so you can focus on writing your app without needing to reinvent the wheel. It's free and open source.

Django Features

Ridiculously fast:- Django was designed to help developers take applications from concept to completion as quickly as possible.

Fully loaded:- Django includes dozens of extras you can use to handle common Web development tasks. Django takes care of user authentication, content administration, site maps, RSS feeds, and many more tasks — right out of the box.

Reassuringly secure:- Django takes security seriously and helps developers avoid many common security mistakes, such as SQL injection, cross-site scripting, cross-site request forgery and clickjacking. Its user authentication system provides a secure way to manage user accounts and passwords.

Exceedingly scalable:- Some of the busiest sites on the planet use Django's ability to quickly and flexibly scale to meet the heaviest traffic demands.

Incredibly versatile:- Companies, organizations and governments have used Django to build all sorts of things — from content management systems to social networks to scientific computing platforms.

5.2 Functional Description

 # Settings.py

```
"""
Django settings for Farmers_Emarket project.

Generated by 'django-admin startproject' using Django 3.0.2.

For more information on this file, see
https://docs.djangoproject.com/en/3.0/topics/settings/

For the full list of settings and their values, see
https://docs.djangoproject.com/en/3.0/ref/settings/
"""

import os

# Build paths inside the project like this: os.path.join(BASE_DIR, ...)
BASE_DIR = os.path.dirname(os.path.dirname(os.path.abspath(__file__)))



# Quick-start development settings - unsuitable for production
# See https://docs.djangoproject.com/en/3.0/howto/deployment/checklist/

# SECURITY WARNING: keep the secret key used in production secret!
SECRET_KEY = 'r_y&5qgij2n2qmk2#+ge5#p#4f0lk35m_vxakk+t2j9%c_w$$d'

# SECURITY WARNING: don't run with debug turned on in production!
DEBUG = True

ALLOWED_HOSTS = ['*']
```

```
# Application definition

INSTALLED_APPS = [
    'farmers.apps.FarmersConfig',
    'public.apps.PublicConfig',
    'django.contrib.admin',
    'django.contrib.auth',
    'django.contrib.contenttypes',
    'django.contrib.sessions',
    'django.contrib.messages',
    'django.contrib.staticfiles',
]

MIDDLEWARE = [
    'django.middleware.security.SecurityMiddleware',
    'django.contrib.sessions.middleware.SessionMiddleware',
    'django.middleware.common.CommonMiddleware',
    'django.middleware.csrf.CsrfViewMiddleware',
    'django.contrib.auth.middleware.AuthenticationMiddleware',
    'django.contrib.messages.middleware.MessageMiddleware',
    'django.middleware.clickjacking.XFrameOptionsMiddleware',
]

ROOT_URLCONF = 'Farmers_Emarket.urls'

TEMPLATES = [
    {
        'BACKEND': 'django.template.backends.django.DjangoTemplates',
        'DIRS': [os.path.join(BASE_DIR,'template')],
        'APP_DIRS': True,
        'OPTIONS': {
            'context_processors': [
                'django.template.context_processors.debug',
                'django.template.context_processors.request',
```

```
                'django.contrib.auth.context_processors.auth',
                'django.contrib.messages.context_processors.messages',
            ],
        },
    },
]

WSGI_APPLICATION = 'Farmers_Emarket.wsgi.application'



# Database
# https://docs.djangoproject.com/en/3.0/ref/settings/#databases

DATABASES = {
    'default': {
        'ENGINE': 'django.db.backends.sqlite3',
        'NAME': os.path.join(BASE_DIR, 'db.sqlite3'),
    }
}



# Password validation
# https://docs.djangoproject.com/en/3.0/ref/settings/#auth-password-validators

AUTH_PASSWORD_VALIDATORS = [
    {
        'NAME':
'django.contrib.auth.password_validation.UserAttributeSimilarityValidator',
    },
    {
        'NAME': 'django.contrib.auth.password_validation.MinimumLengthValidator',
    },
    {
        'NAME': 'django.contrib.auth.password_validation.CommonPasswordValidator',
```

```
    },
    {
        'NAME': 'django.contrib.auth.password_validation.NumericPasswordValidator',
    },
]



# Internationalization
# https://docs.djangoproject.com/en/3.0/topics/i18n/

LANGUAGE_CODE = 'en-us'

TIME_ZONE = 'UTC'

USE_I18N = True

USE_L10N = True

USE_TZ = True



# Static files (CSS, JavaScript, Images)
# https://docs.djangoproject.com/en/3.0/howto/static-files/

STATIC_URL = '/static0/'
STATICFILES_DIRS = [
    os.path.join(BASE_DIR,'static')
]
STATIC_ROOT = os.path.join(BASE_DIR,'assets')

MEDIA_ROOT= os.path.join(BASE_DIR, 'media')
MEDIA_URL= "/media/"

DJANGORESIZED_DEFAULT_SIZE = [500, 400]
```

```
DJANGORESIZED_DEFAULT_QUALITY = 75
DJANGORESIZED_DEFAULT_KEEP_META = True
DJANGORESIZED_DEFAULT_FORCE_FORMAT = 'JPEG'
DJANGORESIZED_DEFAULT_FORMAT_EXTENSIONS = {'JPEG': ".jpg"}
DJANGORESIZED_DEFAULT_NORMALIZE_ROTATION = True
```

 # urls.py

```
from django.urls import path
from . import views
from django.conf.urls import url

urlpatterns = [
        path('removefromwish', views.removefromwish, name='removefromwish'),
        path('addtowish', views.addtowish, name='addtowish'),
        path('removefromcart', views.removefromcart, name='removefromcart'),
        path('cartcount', views.cartcount, name='cart_count'),
        path('addtocart', views.addtocart, name='cart_create'),
        path('farmerdetails', views.farmerdetails,name='farmerdetails'),
        path('ourfarmers', views.ourfarmers,name='ourfarmers'),
        path('product', views.product,name='product'),
        path('wishlist', views.viewwishlist,name='viewwishlist'),
        path('checkout', views.checkout,name='checkout'),
        path('cart', views.viewcart,name='cart'),
        path('about', views.about,name='about'),
        path('contact', views.contact,name='contact'),
    path('index', views.index,name='index'),
    path('', views.index,name='index'),
    path('shop', views.shop,name='shop'),
    path('plogin', views.login,name='user_login'),
        path('logout', views.logout,name='user_logout'),
    path('signup', views.signup,name='user_signup'),
    path('orderstatus', views.orderstatus,name='orderstatus'),
    path('orderhistory', views.orderhistory,name='order_shistory'),
    path('addreview', views.addreview,name='add_review'),
```

```
    path('changepassword', views.changepassword,name='change_password'),
    path('changeaddress', views.changeaddress,name='change_address'),
]
```

 # views.py

```
from django.shortcuts import render, redirect
from django.http import HttpResponse
from django.contrib.auth.models import User, auth
from farmers.models import products
from django.contrib.auth.forms import PasswordChangeForm
from .models import cart, userProfile, wishlist, orderDetails, check_product_stock,
reviewDetails
from django.forms import ModelForm
from django.db import transaction
from django.db.models import F
from django.contrib import messages
from django.contrib.auth import update_session_auth_hash
from django.core.paginator import Paginator, EmptyPage, PageNotAnInteger

# Create your views here.

class AddcartForm(ModelForm):
    class Meta:
        model = cart
        fields = ['userid','productid','quantity']

def changepassword(request):
        if request.user.is_anonymous:
                return redirect('/')
        if request.method == 'POST':
                form = PasswordChangeForm(request.user, request.POST)
                if form.is_valid():
                        user = form.save()
                        update_session_auth_hash(request, user)
```

```
                        messages.info(request, 'Your password has been changed
successfully!')
                        return redirect('/changepassword')
                else:
                        messages.info(request, 'Please correct the error below.')
        else:
                form = PasswordChangeForm(request.user)
        return render(request, 'public/change_password.html', {
      'form': form
   })

def changeaddress(request):
        if request.user.is_anonymous:
                return redirect('/')
        if request.method == 'POST':
                try:
                        uchange =request.user
                        upchange=request.user.userprofile
                        uchange.first_name=request.POST['first_name']
                        uchange.last_name=request.POST['last_name']
                        uchange.email=request.POST['email']
                        upchange.state=request.POST['state']
                        upchange.house=request.POST['house']
                        upchange.town=request.POST['town']
                        upchange.pincode=request.POST['pincode']
                        upchange.phone=request.POST['phone']
                        uchange.save()
                        upchange.save()
                        print('address Updated')
                        return redirect('/')
                except:
                        print('something went wrong')
                        return redirect('/')
        else:
```

```
                return render(request,"public/change_address.html")

def addreview(request):
        if request.user.is_anonymous:
                return redirect('/')
        pid = request.GET['id']
        star = request.GET['star']
        review = request.GET['review']
        product=products.objects.get(id=pid)
        try:

        chkreview=reviewDetails.objects.get(userid=request.user,productid=product)
        except reviewDetails.DoesNotExist:
                chkreview = None
        if chkreview:
                chkreview.stars=star
                chkreview.review=review
                chkreview.save()
                return HttpResponse('Review Updated')
        else:

        instance=reviewDetails(userid=request.user,productid=product,stars=int(star),
review=review)
                instance.save()
                return HttpResponse('Review Added')

        return HttpResponse('Something went wrong, TryAgain')

def orderhistory(request):
        if request.user.is_anonymous:
                return redirect('/')
        try:

        orderlist=orderDetails.objects.filter(userid_id=request.user,status=True)
```

```
        except orderlist.DoesNotExist:
                orderlist = None
        return render(request,"public/orderhistory.html",{'orderlist':orderlist})

def removefromwish(request):
        if request.user.is_anonymous:
                return redirect('/')
        pid = request.GET['id']
        product=products.objects.get(id=pid)
        try:
                chkwish=wishlist.objects.get(productid=pid,userid=request.user.id)
        except wishlist.DoesNotExist:
                chkwish = None
        if chkwish:
                try:
                        chkwish.delete()
                        return HttpResponse('Product Removed from Wishlist')
                except:
                        return HttpResponse('Something went wrong, TryAgain')
        else:
                return HttpResponse('No such Product in Wishlist')

def addtowish(request):
        if request.user.is_anonymous:
                return redirect('/')
        pid = request.GET['id']
        qty = request.GET['qty']
        product=products.objects.get(id=pid)
        try:
                chkwish=wishlist.objects.get(productid=pid,userid=request.user.id)
        except wishlist.DoesNotExist:
                chkwish = None
        if chkwish:
                return HttpResponse('Product already in Wishlist')
```

```
        else:
                instance =
wishlist(userid=request.user,productid=product,quantity=qty)
                try:
                        instance.save()
                        return HttpResponse('Added to Wishlist')
                except:
                        return HttpResponse('Something went wrong, TryAgain')

def cartcount(request):
        if request.user.is_anonymous:
                return redirect('/')
        try:
                chkcart=cart.objects.filter(userid=request.user.id)
        except cart.DoesNotExist:
                chkcart = None
        if chkcart:
                return HttpResponse(len(chkcart))
        else:
                return HttpResponse('0')

def removefromcart(request):
        if request.user.is_anonymous:
                return redirect('/')
        pid = request.GET['id']
        product=products.objects.get(id=pid)
        try:
                chkcart=cart.objects.get(productid=pid,userid=request.user.id)
        except cart.DoesNotExist:
                chkcart = None
        if chkcart:
                try:
                        chkcart.delete()
                        return HttpResponse('Product Removed from Cart')
```

```
                except:
                        return HttpResponse('Something went wrong, TryAgain')
        else:
                return HttpResponse('No such Product in Cart')

def addtocart(request):
        if request.user.is_anonymous:
                return redirect('/')
        pid = request.GET['id']
        qty = request.GET['qty']
        product=products.objects.get(id=pid)
        try:
                chkcart=cart.objects.get(productid=pid,userid=request.user.id)
        except cart.DoesNotExist:
                chkcart = None
        if chkcart:
                return HttpResponse('Product already in Cart')
        else:
                instance = cart(userid=request.user,productid=product,quantity=qty)
                try:
                        instance.save()
                        return HttpResponse('Added to Cart')
                except:
                        return HttpResponse('Something went wrong, TryAgain')

def index(request):
        return render(request,"public/index.html")
def shop(request):
        try:
                cat = request.GET['cat']
        except:
                cat = 'All'
        if request.user.is_anonymous:
                current_user = request.user
```

```
                Vegetables1=products.objects.filter(isactive=True,ptype=
'Vegetables').order_by('id')
                Fruits1=products.objects.filter(isactive=True,ptype=
'Fruits').order_by('id')
                Product1=products.objects.filter(isactive=True,ptype=
'Products').order_by('id')
                Dried1=products.objects.filter(isactive=True,ptype=
'Dried').order_by('id')
                All1=products.objects.filter(isactive=True).order_by('id')
        else:
                current_user = request.user
                Vegetables1=products.objects.filter(isactive=True,ptype=
'Vegetables',owner__userprofile__pincode__range=[int(request.user.userprofile.pinco
de)-2,int(request.user.userprofile.pincode)+2]).order_by('id')
                Fruits1=products.objects.filter(isactive=True,ptype=
'Fruits',owner__userprofile__pincode__range=[int(request.user.userprofile.pincode)-
2,int(request.user.userprofile.pincode)+2]).order_by('id')
                Product1=products.objects.filter(isactive=True,ptype=
'Products',owner__userprofile__pincode__range=[int(request.user.userprofile.pincode
)-2,int(request.user.userprofile.pincode)+2]).order_by('id')
                Dried1=products.objects.filter(isactive=True,ptype=
'Dried',owner__userprofile__pincode__range=[int(request.user.userprofile.pincode)-
2,int(request.user.userprofile.pincode)+2]).order_by('id')

        All1=products.objects.filter(isactive=True,owner__userprofile__pincode__ran
ge=[int(request.user.userprofile.pincode)-
2,int(request.user.userprofile.pincode)+2]).order_by('id')

        pagev = request.GET.get('vpage', 1)
        pagef = request.GET.get('fpage', 1)
        pagep = request.GET.get('ppage', 1)
        paged = request.GET.get('dpage', 1)
        pagea = request.GET.get('apage', 1)
```

```
        paginatorv = Paginator(Vegetables1, 10)
        paginatorf = Paginator(Fruits1, 10)
        paginatorp = Paginator(Product1, 10)
        paginatord = Paginator(Dried1, 10)
        paginatora = Paginator(All1, 10)
        try:
                Vegetables = paginatorv.page(pagev)
                Fruits = paginatorf.page(pagef)
                Product = paginatorp.page(pagep)
                Dried = paginatord.page(paged)
                All = paginatora.page(pagea)
        except PageNotAnInteger:
                Vegetables = paginatorv.page(1)
                Fruits = paginatorf.page(1)
                Product = paginatorp.page(1)
                Dried = paginatord.page(1)
                All = paginatora.page(1)
        except EmptyPage:
                Vegetables = paginatorv.page(paginatorv.num_pages)
                Fruits = paginatorf.page(paginatorf.num_pages)
                Product = paginatorp.page(paginatorp.num_pages)
                Dried = paginatord.page(paginatord.num_pages)
                All = paginatora.page(paginatora.num_pages)
        return render(request,
'public/shop.html',{'All':All,'Vegetables':Vegetables,'Fruits':Fruits,'Products':Product,'
Dried':Dried,'cat':cat})

def about(request):
        return render(request,"public/about.html")
def contact(request):
        return render(request,"public/contact.html")

def viewcart(request):
        if request.user.is_anonymous:
```

```
            return redirect('/')
        try:
            cartlist=cart.objects.filter(userid=request.user.id)
        except cart.DoesNotExist:
            cartlist = None
        total=0
        subtotal=[]
        if cartlist:
            for item in cartlist:
                if item.productid.offer:

        subtotal.append(item.productid.offerprice*item.quantity)
                        total=total+item.productid.offerprice*item.quantity
                else:
                        subtotal.append(item.productid.price*item.quantity)
                        total=total+item.productid.price*item.quantity
        return
render(request,"public/cart.html",{'cartlist':cartlist,'subtotal':subtotal,'total':total})



        return render(request,"public/cart.html")
def checkout(request):
        if request.user.is_anonymous:
            return redirect('/')
        return render(request,"public/checkout.html")

def viewwishlist(request):
        if request.user.is_anonymous:
            return redirect('/')
        try:
            wishlst=wishlist.objects.filter(userid=request.user.id)
        except wishlist.DoesNotExist:
            wishlst = None
        subtotal=[]
```

```
        if wishlst:
                for item in wishlst:
                        if item.productid.offer:

        subtotal.append(item.productid.offerprice*item.quantity)
                        else:
                                subtotal.append(item.productid.price*item.quantity)
        return
render(request,"public/wishlist.html",{'wishlist':wishlst,'subtotal':subtotal})

def product(request):
        pid = request.GET['id']
        product=products.objects.filter(id=pid)
        for pro in product:
                type=pro.ptype
        rproduct=products.objects.filter(ptype=type,isactive=True).exclude(id =
pid)[:4]
        buyed = False
        tstar=0
        soldcount=0
        try:
                chkreview=reviewDetails.objects.filter(productid=pro)
                for rev in chkreview:
                        tstar+=rev.stars
        except reviewDetails.DoesNotExist:
                chkreview = None
        try:
                rating=tstar/chkreview.count()
        except :
                rating=0.0
        half=rating.is_integer()
        if half:
                nostars=5-int(rating)
        else:
```

```
                nostars=4-int(rating)
        if not request.user.is_anonymous:
                try:

        orderlist=orderDetails.objects.filter(userid_id=request.user,status=True,produc
tid_id=pro)

        soldlist=orderDetails.objects.filter(status=True,productid_id=pro)
                        for sold in soldlist:
                                soldcount+=sold.quantity
                except orderDetails.DoesNotExist:
                        orderlist=None
                if orderlist:
                        buyed=True
        return
render(request,"public/product.html",{'product':product,'rproduct':rproduct,'buyed':bu
yed,'chkreview':chkreview,'trate':rating,'soldcount':soldcount,'nostars':nostars,'half':hal
f})

def ourfarmers(request):
        farmers=User.objects.filter(userprofile__ismerchant=True)
        return render(request,"public/ourfarmers.html",{'farmers':farmers})


def farmerdetails(request):
        scount=0
        fid = request.GET['id']
        farmers=User.objects.filter(userprofile__ismerchant=True,id=fid)
        product=products.objects.filter(isactive=True,owner=fid)
        tcount=product.count()
        soldlist=orderDetails.objects.filter(status=True,userid_id=fid)
        for sold in soldlist:
                scount+=sold.quantity
```

```
        return
render(request,"public/ourfarmers_details.html",{'farmers':farmers,'products':product[
:4],'tcount':tcount,'scount':scount})

def logout(request):
        if request.user.is_anonymous:
                return redirect('/')
        auth.logout(request)
        return redirect('/')

def login(request):
        if not request.user.is_anonymous:
                return redirect('/')
        if request.method == 'POST':
                password=request.POST['password']
                username=request.POST['username']
                user=auth.authenticate(username=username,password=password)
                if user is not None:
                        auth.login(request,user)
                        return redirect('/')
                else :

                        return render(request,"public/login.html",{'status':True})
        else:
                return render(request,"public/login.html")
def signup(request):
        if not request.user.is_anonymous:
                return redirect('/')
        if request.method == 'POST':
                first_name=request.POST['first_name']
                last_name=request.POST['last_name']
                email=request.POST['email']
                password=request.POST['password']
                username=request.POST['email']
```

```
                state=request.POST['state']
                house=request.POST['house']
                town=request.POST['town']
                pincode=request.POST['pincode']
                phone=request.POST['phone']
                try:
                        user
=User.objects.create_user(username=username,first_name=first_name,last_name=last
_name,email=email,password=password)
                        userp=userProfile.objects.create(user=user, phone=phone,
pincode=pincode,town=town, house=house, state=state)
                        user.save()
                        userp.save()
                        print('created')
                        return redirect('/plogin')
                except Exception as e:
                        if str(e)[0:46]=="duplicate key value violates unique
constraint":
                                msg="Email Already Exist.."
                        else:
                                msg="Something went wrong, Try Again..."
                        return
render(request,"public/signup.html",{'status':True,'msg':msg})
        else:
                return render(request,"public/signup.html")

def orderstatus(request):
        if request.user.is_anonymous:
                return redirect('/')
        user = request.user
        paymode=request.POST['optradio']
        flag=False
```

```
        address=user.first_name+user.last_name+","+user.userprofile.house+","+user.
userprofile.town+","+user.userprofile.state+","+str(user.userprofile.pincode)+","+str(
user.userprofile.phone)+","+user.email
        print(address)
        try:
                cartlist=cart.objects.filter(userid=request.user.id)
        except cart.DoesNotExist:
                cartlist = None
        if cartlist:
                with transaction.atomic():
                        for item in cartlist:
                                product=products.objects.get(id=item.productid.id)
                                instance =
orderDetails(userid=request.user,productid=product,quantity=item.quantity,address=a
ddress,paymode=paymode)

user.userprofile.phone)+","+user.email
        print(address)
        try:
                cartlist=cart.objects.filter(userid=request.user.id)
        except cart.DoesNotExist:
                cartlist = None
        if cartlist:
                with transaction.atomic():
                        for item in cartlist:
                                product=products.objects.get(id=item.productid.id)
                                instance =
orderDetails(userid=request.user,productid=product,quantity=item.quantity,address=a
ddress,paymode=paymode)
user.userprofile.phone)+","+user.email
        print(address)
        try:
                cartlist=cart.objects.filter(userid=request.user.id)
        except cart.DoesNotExist:
```

```
            cartlist = None
        if cartlist:
            with transaction.atomic():
                for item in cartlist:
                    product=products.objects.get(id=item.productid.id)
                    instance =
orderDetails(userid=request.user,productid=product,quantity=item.quantity,address=a
ddress,paymode=paymode)
                    product.stock = F('stock')- item.quantity
                    try:
                        product.save()
                        check_product_stock(product.id)
                        instance.save()
                        print("added to o list")
                        #return HttpResponse('Added to Cart')
                    except Exception as e:
                        print(e)
                        print("Something went wrong with ol,
TryAgain")
                        #return HttpResponse('Something went wrong,
TryAgain')
                flag=True

        try:
            cartlist=cart.objects.filter(userid=request.user.id)
        except cart.DoesNotExist:
            cartlist = None
        if cartlist:
            with transaction.atomic():
                for item in cartlist:
                    try:
                        item.delete()
                        print("deleted from cart")
                        #return HttpResponse('Added to Cart')
```

```
                except Exception as e:
                    print(e)
                    print("Something went wrong, TryAgain")
                    #return HttpResponse('Something went wrong,
TryAgain')
                flag=True

        if flag:
            return render(request,"public/order_status.html",{'status':flag})
```

CHAPTER 6
TESTING

Testing is a set of activities that can be planned in advance and conducted systematically. System testing is that stage of implementation that is aimed for ensuring that the system works accurately and efficiently before live operation commences. System testing is the execution of the program to check logical changes and its impact on the output of the system with the intention of finding error. A successful test is the one that uncovers a yet undiscovered error. Testing is vital to the success of the system

A strategy for software testing integrates software test case design methods into well planned series of steps that result in successful completion of the software. A software testing strategy should be flexible enough to promote a customized testing approach. A strategy for testing should accommodate low level test that are necessary to verify implemented as well as high level tests that validate major system functions against customer requirements.

6.1 Levels of Testing

Testing is done in the following levels:

1. Unit Testing
2. Integration Testing
3. Validation Testing
4. Output Testing

Unit Testing

In this level of testing each of the modules were tested one by one individually, before they were integrated. The modules were tested as soon as they were developed. It helped to recognize areas requiring modifications and corrections. In this system, unit testing can be easily accomplished. The project is divided into four modules training data generation, number plate detection, character segmentation and user. The modules includes various activities. In the case of this project, each module was built separately and was tested and verified individually before integrating them to a complete project.

Integration Testing

The modules that are tested individually and confirmed to be working according to specifications are then integrated to form the entire system. The training data generation, number plate detection, character segmentation and user modules are integrated in this testing step and then the entire system is tested. The interface between the modules are thoroughly tested. It is ensured that data and controls are passed properly between modules

Validation Testing

At the end of the integration testing software is completely assembled as a package and interfacing errors have been uncovered and corrected and final series of software validation testing begins

Once the system is integrated and tested to operate properly, it is tested to see if the software developed meets all functional, behavioural and performance requirements. The errors which were uncovered during Integration testing were covered here.

In this system. I have tested the different forms to see whether the inputs given to the forms are stored in the appropriate entries. I also tested the home page with image/video inputs, to see whether the inputs are sent to the system and are validated properly. Hence, I found that the validation testing had carried out successfully.

Output Testing

Here I have tested the system to determine whether it produce the required output or not and to ensure the correctness of the output and its format. In my project testing is implemented in each of the modules. The following test cases are performed to do the testing.

1. Proper opening and closing of window.

2. Appropriate menu bars displayed in appropriate section.

3. A proper working of system is checked for multiple input modes.

Like this all other processes are tested and hence come under the conclusion that there arise no errors. It also found that there arise no modification and corrections in any areas of the project.

CHAPTER 7

IMPLEMENTATION

Implementation is the stage of the project where theoretical design is turned into a working system. If the implementation is not carefully planned and controlled, it can cause chaos and confusion. Proper implementation is essential to provide a reliable system to meet organization requirements. Successful implementation may not guarantee improvement in the organization using the system, but proper installation will prevent it. The process of putting the developed system in actual use is called system implementation. The system can only be implemented after thorough testing is done and if it is found to be working according to the specifications.

The implementation stage involves following tasks:

1. Careful Planning
2. Investigation of system and constraints.
3. Design of methods to achieve the changeover.
4. Training of the staff in the changeover phase.
5. Evaluation of the changeover method.

In order to have the system running of a specific machine, the following components are needed.

1) Python
2) A preferable operating system like windows 8 or windows 10
3) Visual studio
4) Android Phone

Parallel run is done and both the computerized and manual systems are executed in parallel manual result can be compared with the result of computerized system. For the case of demonstration of the success of this system, it was implemented with successfully running; manual systems results are verified.

CHAPTER 8

SECURITY, BACKUP AND RECOVERY MECHANISMS

Security is an important consideration in desktop application. The first step in securing our application is deciding where we need security and what will be needed to protect.

Security concepts:

1. Authentication
2. Authorization

Authentication

This is the process of determining users identify and forcing users to prove they are who they claim to be usually this involves entering username and password in login page.

Authorization

Once the user is authenticated, authorization is the process of determining whether the user has sufficient permission to perform a given action or not, such as viewing a page or retrieving information from the database.

CHAPTER 9

CONCLUSION

Our system Farmer's E Market, as described above, can lead to error free, secure, reliable and fast management system. It can assist the user to concentrate on their other activities rather to concentrate on the record keeping. Thus it will help farmers in better utilization of resources. The farmers 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. Basically the project describes how to manage for good performance and better services for the clients.

CHAPTER 10

FUTURE ENHANCEMENT

Farmer's E Market project has a very vast scope in the future. The model may be expanded to add functions like payment gateway, real time order tracking etc. In future mobile app can be updated with the same functionalities so that every remote user can access our system without any issues. This enhancement will make it a better utility for mobile users as well in future

APPENDIX

Input and Output Forms

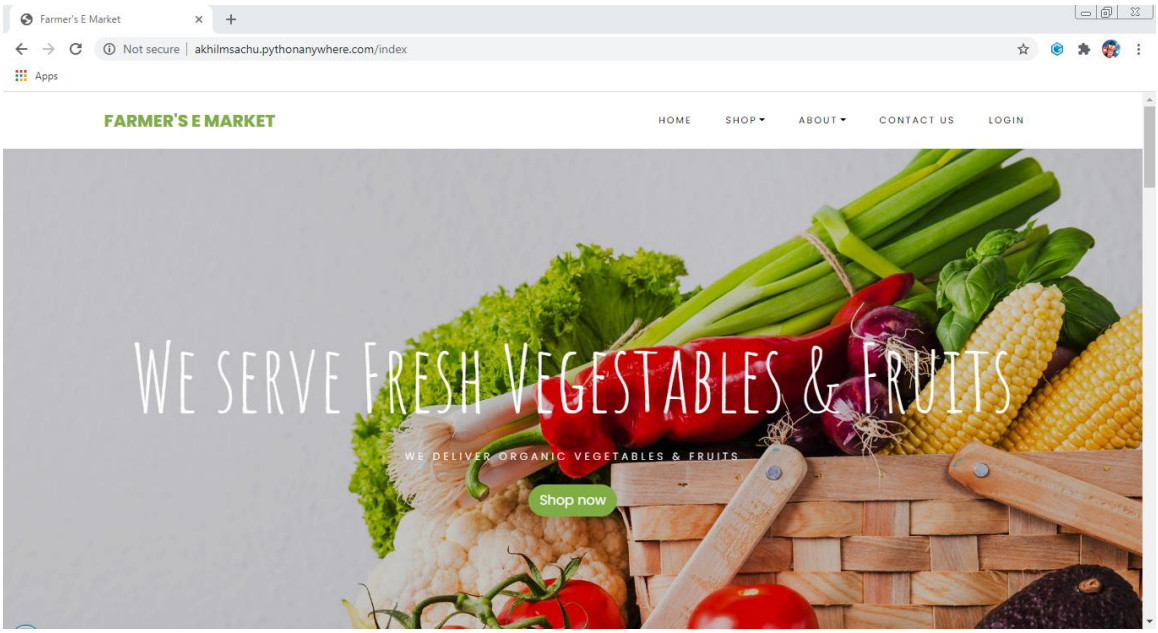


Fig 1: Index Page

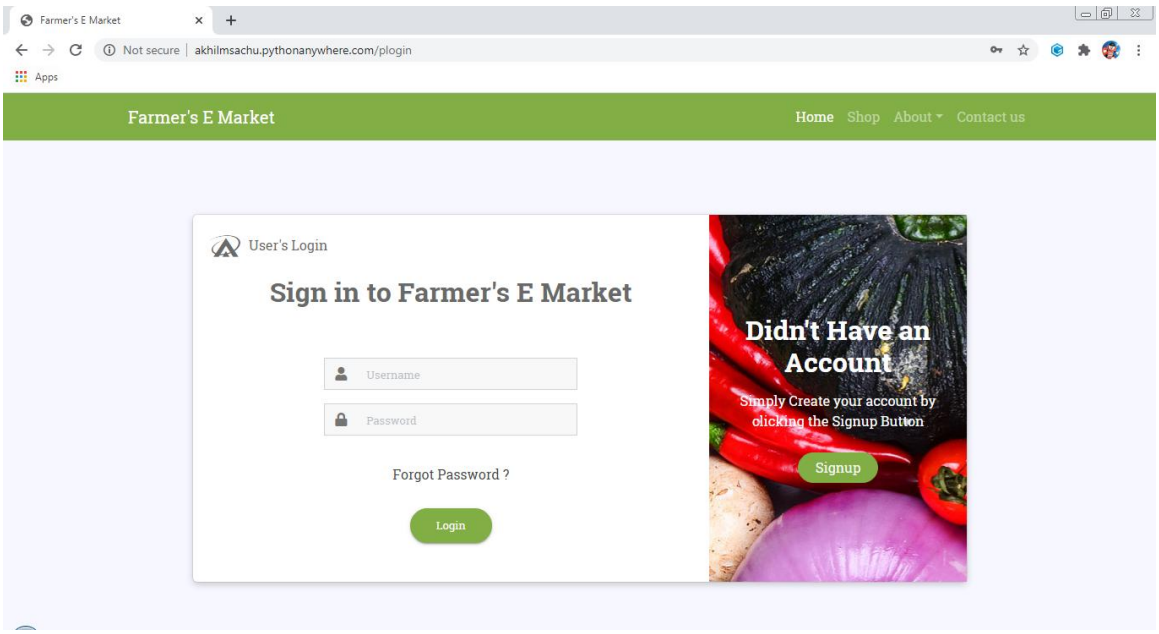


Fig 2: User Login Page

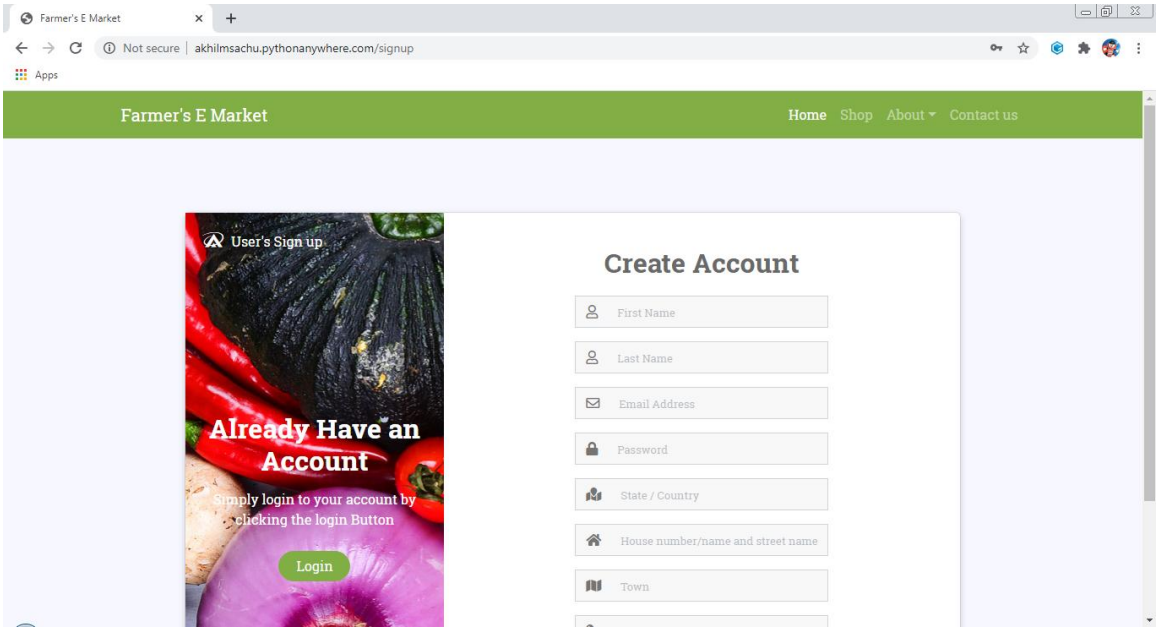


Fig 3: User Signup Page

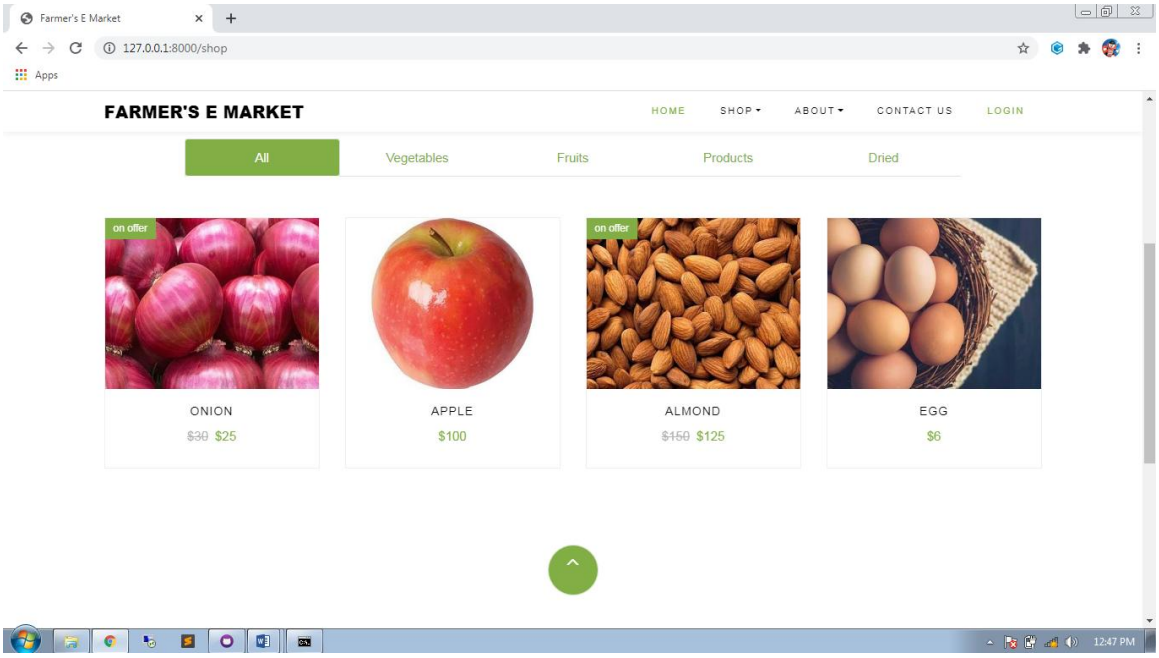


Fig 4: Shop Page

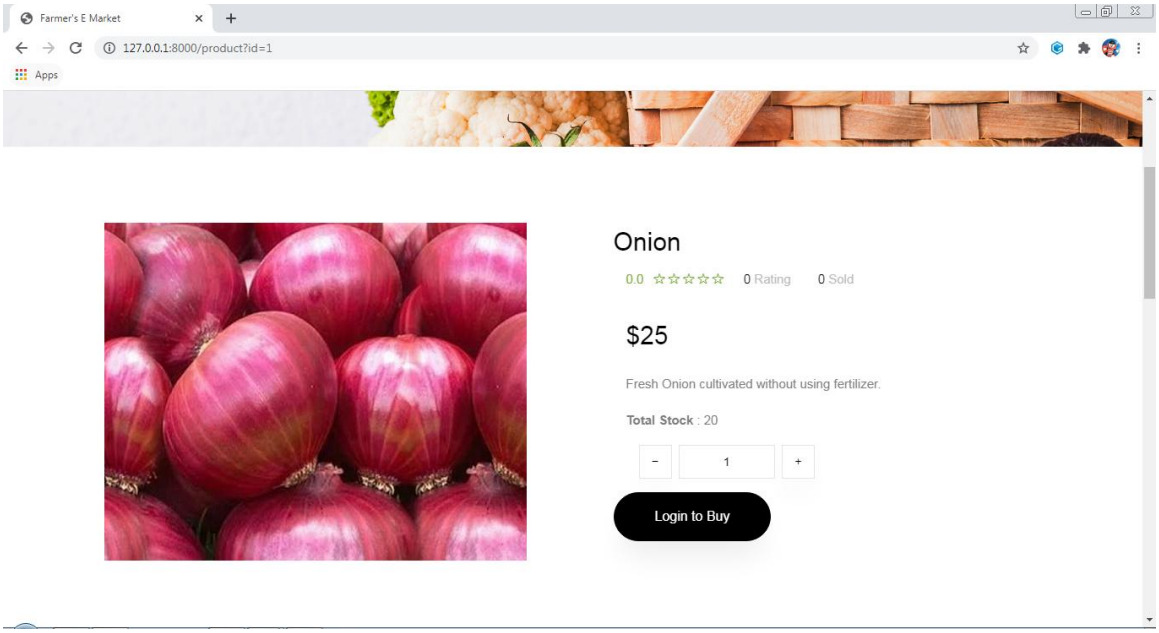


Fig 5: Product Page

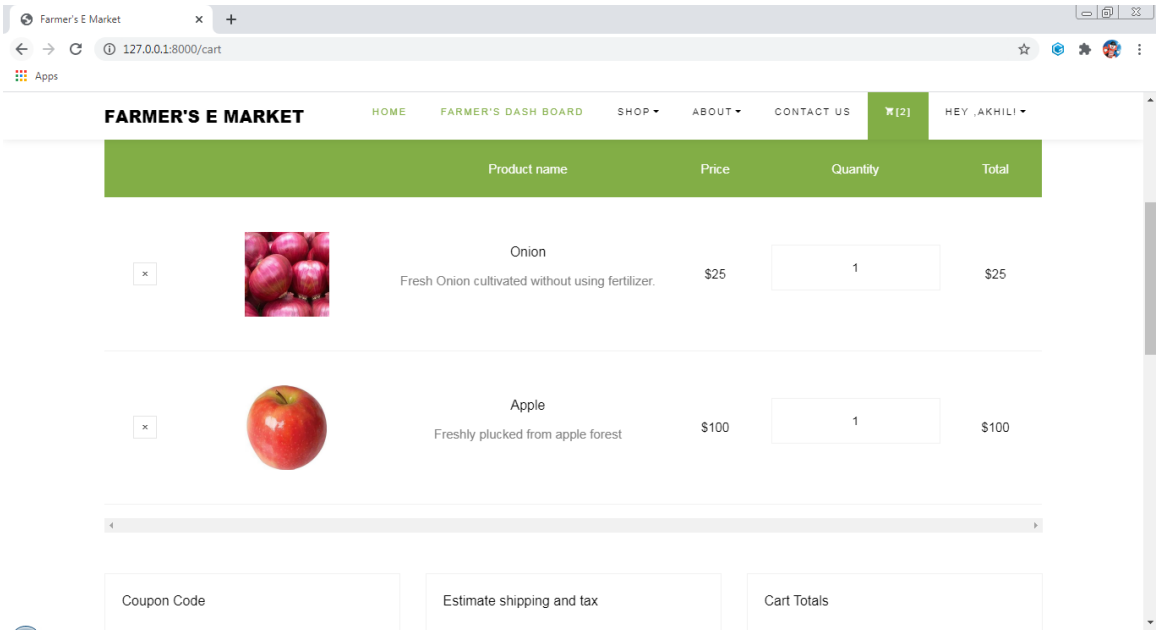


Fig 6: Cart Page

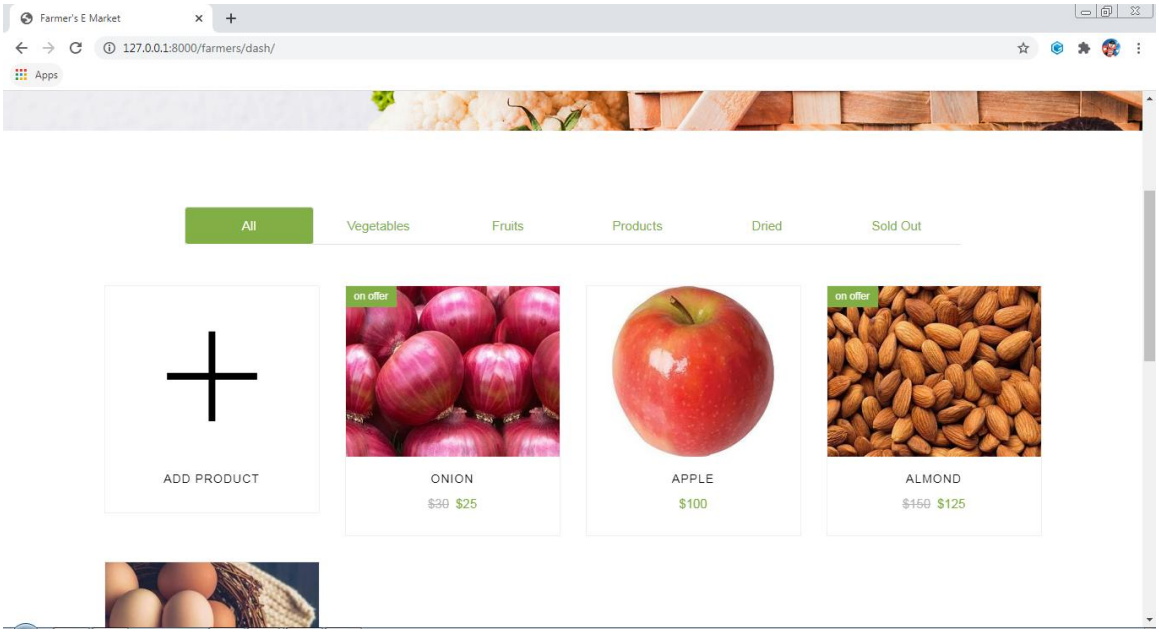


Fig 3: Farmers Dash Page

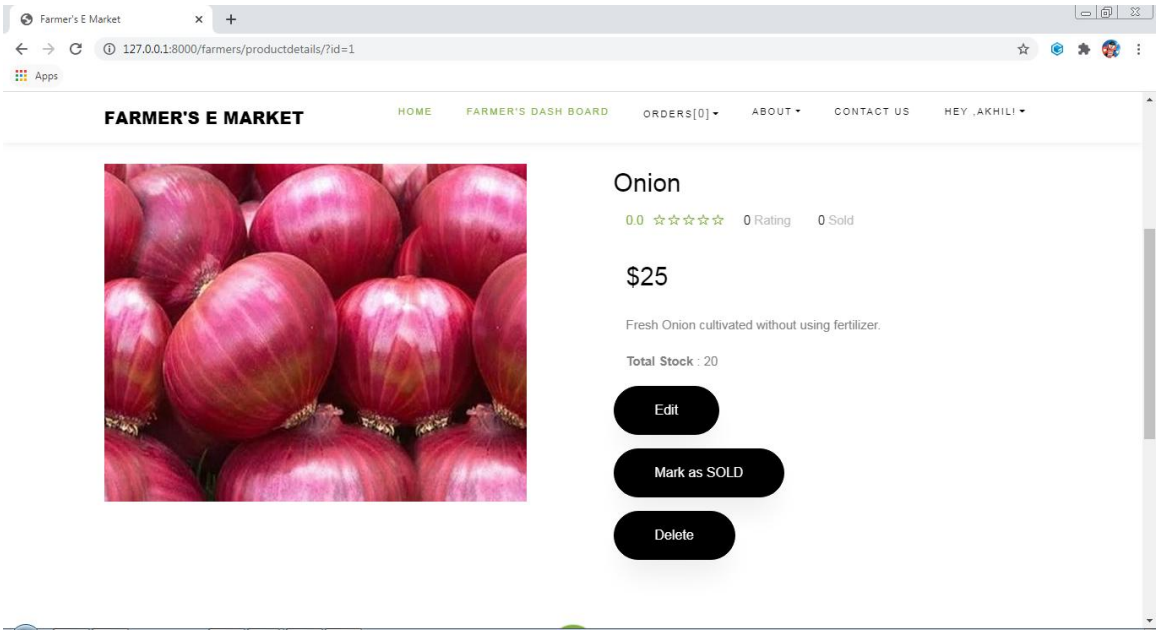


Fig 3: Farmers product Page

BIBLIOGRAPHY

Internet sites

1. https://docs.djangoproject.com/en/3.0/topics/auth/default/#built-in-auth-forms
2. https://www.geeksforgeeks.org/django-url-patterns-python/
3. https://docs.djangoproject.com/en/3.0/ref/contrib/admin/
4. https://docs.djangoproject.com/en/3.1/topics/templates/
5. https://docs.djangoproject.com/en/3.1/topics/settings/
6. https://docs.djangoproject.com/en/3.1/howto/deployment/
7. https://docs.djangoproject.com/en/3.1/ref/contrib/staticfiles/
8. https://docs.djangoproject.com/en/3.1/topics/migrations/
9. https://docs.djangoproject.com/en/3.1/intro/