

# International Institute of Information Technology, Bangalore.



SOFTWARE PRODUCTION ENGINEERING - FINAL PROJECT  
**Group - 23**

## Kisan Portal

*A one stop solution to farm produce*

In the supervision of :-

**Prof B.Thangaraju**

Teaching Assistant :-

**Yashasvi**

Team Member Details :-

NAME	ROLL NO
Niraj Gujarathi	MT2021087
Gaurav Tirodkar	MT2021047

08.05.2022

# Index

- Abstract
- Introduction
  - Overview
  - Features
  - Why Devops
- Tools Used -
  - **Github Link** - <https://github.com/NirajGujarathi/Kisanan Portal>
- System Configurations
- SDLC - DevOps Process
  - **Packaging / Building - npm**
  - Create Mongo - DB Atlas server
  - Connect app to database
  - **JWT Authentication**
  - **Version Control - git**
  - **API Documentation**
  - **Postman** - API checking
  - **External API Integration** - Daily Market Prices and OpenWeather
  - **Testing - chai / mocha and react-tests**
  - **Containerization - Docker Hub**
  - **CI / CD Pipeline - Jenkins**
  - **Ansible Deployment**
  - **Docker - Compose**
  - **LOG management - winston**
  - Successful Pipeline Execution
  - **Continuous Monitoring - ELK stack**
- Deployment on **heroku** and **netlify** - using **GitHub Actions**
- **Screenshot of Application**
- Conclusion
- Future work

## **1. ABSTRACT**

In the wake of COVID-19, everything came to a stop. All shops were shut, everything was closed, this affected the farmers too. It became too difficult for them to sell their produce given lockdown and social gathering restrictions.

As all the offices were moving online, we thought why not bring the farmers online too, and sell authentic farm produce directly from the farm to the consumers. This ends the corruption and supremacy of the mediators and the entire amount directly goes to the farmer.

To make this into reality, we built an online portal where the user (Farmer/Consumer) has to first register and then enter the portal. Farmers can upload all their produce and decide at what amount to sell. We provide a dashboard to each farmer to compare his prices with the mandi rates of his locality and set the price accordingly.

The consumer can login and get a list of all the produce, he/she can select as many as needed and add to his/her cart and proceed to checkout once done.

To implement this we have used the MERN stack, React.js is used for the development of the front end. Express library on top of Node.js. was used for the backend and the Database used was MongoDB where the data was stored. Mongoose was used for connecting to MongoDB.

## **2. INTRODUCTION**

### **a. OVERVIEW**

Kisaan Portal is a one stop solution for all farmers to sell their produce directly to consumers from all the different cities of the country. The farmers can access this portal and get the prices of different items of their local markets, and set the selling price accordingly.

Consumers can get a variety of items directly from the website to their homes. All the users need to do is register and set up the payment

## b. FEATURES

Farmer :

- i. Register/LogIn
- ii. Update personal information
- iii. Fetch Market Data
- iv. Get weather information of upcoming five days
- v. Upload produce and fix price

Consumer

- vi. Register/LogIn
- vii. Check prices for different items
- viii. Fetch Market Data
- ix. Add produce to cart
- x. Checkout after payment

## c. WHY DEVOPS ?

DevOps is a set of tools that automate software development and IT operations. It focuses on shortening the systems development life cycle and providing continuous delivery with high software quality.

DevOps is one up on Agile software development; multiple DevOps aspects came from the Agile methodology.

- It optimizes the overall business by increasing efficiency through automation.
- Improves software development and deployment speed and stability.
- Deployment failures, rollbacks, and recovery time are all reduced.
- Improved collaboration and communication; lower costs and IT headcount

### **3. TOOLS USED**

1. Version Control System : Git and GitHub  
[https://github.com/NirajGujarathi/Kisanan\\_Portal](https://github.com/NirajGujarathi/Kisanan_Portal)
2. Continuous Integration/Continuous Delivery : Jenkins
3. Building Tool : NPM (Node Package Manager)
4. Testing : Mocha framework
5. Containerization : Docker
6. Deployment : Ansible
7. Log creation, monitoring and visualization : ELK (Elasticsearch, Logstash, Kibana)
8. Frontend Development : React.js
9. Backend Development : Node.js, Express.js
10. Database : MongoDB

### **4. SYSTEM CONFIGURATION**

1. Operating System - Linux Ubuntu 20
2. CPU & RAM - 4 core processor and 8GB RAM
3. Kernel Version - 5.4.0-89-generic
4. Database - MongoDB 4.4.5

## 5. SOFTWARE DEVELOPMENT LIFECYCLE - SDLC

### i. INSTALLATIONS

The frontend(client) of the project uses React.js and the backend(server) is built using Node.js, Express.js and the database is using MongoDB. To begin with, install nodejs on the system.

### ii. NODEJS INSTALLATION

Node.js is a javascript programming package management. It is the JavaScript runtime environment Node.js' default package manager. It comes pre-installed with Node.js. The package.json file contains the definitions for all npm packages. Package.json's content must be in JSON format. The definition file must have at least two fields. The names and variants are as follows. It is capable of managing dependencies. It installs all of the project's dependencies in a single command line. The package.json file also defines dependencies.

**Command** : npm install

Run npm install command inside the frontend and backend folder to install all the necessary dependencies of the project.

```
gaurav@gaurav-Predator-PH315-51:~/SPE-Project/Kisanan_Portal$ cd frontend/
gaurav@gaurav-Predator-PH315-51:~/SPE-Project/Kisanan_Portal/frontend$ npm install
up to date, audited 1486 packages in 10s
```

**Fig. Frontend npm install**

```
gaurav@gaurav-Predator-PH315-51:~/SPE-Project/Kisan_Portal$ cd backend/  
gaurav@gaurav-Predator-PH315-51:~/SPE-Project/Kisan_Portal/backend$ npm install  
up to date, audited 272 packages in 2s
```

Fig. Backend npm install

```
{ } package.json ×  
  
backend > { } package.json > ...  
1  {  
2    "name": "backend",  
3    "version": "1.0.0",  
4    "main": "index.js",  
5    "scripts": {  
6      "start": "nodemon app.js"  
7    },  
8    "keywords": [],  
9    "author": "",  
10   "license": "ISC",  
11   "dependencies": {  
12     "body-parser": "^1.19.0",  
13     "cookie-parser": "^1.4.5",  
14     "cors": "^2.8.5",  
15     "dotenv": "^8.2.0",  
16     "express": "^4.17.1",  
17     "express-jwt": "^6.1.1",  
18     "express-validator": "^5.3.0",  
19     "formidable": "^1.2.2",  
20     "jsonwebtoken": "^8.5.1",  
21     "lodash": "^4.17.21",  
22     "mongoose": "^6.2.10",  
23     "morgan": "^1.10.0",  
24     "nodemon": "^2.0.7",  
25     "uuid": "^8.3.2",  
26     "uuidv1": "^1.6.14",  
27     "winston": "^3.3.3"  
28   },  
29   "description": ""
```

Fig. package.json server file

```
{ } package.json ×  
frontend > { } package.json > ...  
1  {  
2    "name": "kisaanportal",  
3    "version": "0.1.0",  
4    "private": true,  
5    "dependencies": {  
6      "bootstrap": "^5.0.0",  
7      "mdbreact": "^4.6.1",  
8      "query-string": "^4.3.4",  
9      "react": "^16.8.6",  
10     "react-bootstrap": "^1.5.2",  
11     "react-dom": "^16.14.0",  
12     "react-router-dom": "^5.3.1",  
13     "react-scripts": "^5.0.1"  
14   },  
15   ▷ Debug  
16   "scripts": {  
17     "start": "react-scripts start",  
18     "build": "react-scripts build",  
19     "test": "react-scripts test",  
20     "eject": "react-scripts eject"  
21   },  
22   "eslintConfig": {  
23     "extends": "react-app"  
24   },  
25   "browserslist": {  
26     "production": [  
27       ">0.2%",  
28       "not dead",  
29       "not op_mini all"  
30     ],  
31     "development": [  
32       "last 1 chrome version",  
33       "last 1 firefox version",  
34       "last 1 safari version"  
35     ]  
36   }  
37 }
```

Fig. package.json client file

### iii. MONGODB

We create a mongodb database on atlas to store all the user and farmer information. To set this database with our project use create URI and store it in the .env file

```
.env
backend > .env
1 ONLINE_DATABASE=mongodb+srv://niraj:niraj123@cluster0.xzfpt.mongodb.net/kissanportal?retryWrites=true&w=majority
```

### iv. Create a cluster on MONGO-DB Atlas

The screenshot shows the MongoDB Atlas interface for a deployment named "tinder-clone". The left sidebar has sections for Deployment (Database selected), Data Services, Security, and Network Access. The main area displays "Database Deployments" for "GAURAV'S ORG - 2022-02-10 > TINDER-CLONE". It shows a summary for "Cluster0" with metrics: R 0, W 0, Connections 0, In 0.0 B/s, Out 0.0 B/s, Data Size 180.8 KB. It also shows version 5.0.8, region AWS / Mumbai (ap-south-1), cluster tier M0 Sandbox (General), type Replica Set - 3 nodes, backups Inactive, linked realm app None Linked, and atlas search Create Index. A green "Upgrade" button is visible.

The screenshot shows the MongoDB Atlas interface for a deployment named "tinder-clone". The left sidebar has sections for Deployment (Network Access selected), Data Services, Security, and Network Access. The main area displays "Network Access" for "GAURAV'S ORG - 2022-02-10 > TINDER-CLONE". It shows an "IP Access List" tab with a note: "You will only be able to connect to your cluster from the following list of IP Addresses:". It lists one entry: IP Address 0.0.0.0/0 (includes your current IP address) with Status Active. There are "Edit" and "Delete" buttons for this entry. Other tabs include "Peering" and "Private Endpoint". A green "+ ADD IP ADDRESS" button is visible.

Set Network Access to 0.0.0.0/0 so that the website can be accessed from all IP addresses. This is ideal for real time development project, for development we can set a few specific addresses.

## v. Connect app to database

The screenshot shows the 'Connect to Cluster' wizard. At the top, there are three steps: 'Setup connection security' (completed with a green checkmark), 'Choose a connection method' (completed with a green checkmark), and 'Connect'. Step 2 is currently active, titled 'Select your driver and version'. It shows 'Node.js' selected in the 'DRIVER' dropdown and '4.0 or later' selected in the 'VERSION' dropdown. Step 3, 'Add your connection string into your application code', is shown below. It includes a checkbox for 'Include full driver code example' which is unchecked. A connection string is displayed in a code editor-like area: `mongodb+srv://<username>:<password>@cluster0.gru8m.mongodb.net/myFirstDatabase?retryWrites=true&w=majority`. Below the code editor, a note says: 'Replace <password> with the password for the <username> user. Replace myFirstDatabase with the name of the database that connections will use by default. Ensure any option params are URL encoded.' At the bottom, there are 'Go Back' and 'Close' buttons.

Add the above URL to .env to connect to the database. Replace the username with mongodb's username and set the password for the same.

Change the database name at myFirstDatabase, to your specific database name.

## Database Entries from - Browse Collections

The screenshot shows the MongoDB Compass interface. On the left, the 'Namespaces' sidebar lists 'kissanportal' with collections 'farmers', 'users', and 'vegetables'. The 'vegetables' collection is selected. The main area displays the 'kissanportal.vegetables' collection with the following details:

- Storage Size: 372KB
- Total Documents: 2
- Indexes Total Size: 36KB

Below these details are tabs for 'Find', 'Indexes', 'Schema Anti-Patterns', 'Aggregation', and 'Search Indexes'. A 'FILTER' bar with the query '{ field: 'value' }' is present. The 'QUERY RESULTS 1-2 OF 2' section shows two documents:

```
_id: ObjectId("6261a71b45c5fc7949e4dc53")
name: "Brinjal"
price: 45
quantity: 128
farmer_id: ObjectId("6261739040865c739ed47d12")
photo: Object
  data: Binary('9j/4AAQSkZJRgABAQAAAQABAAAD/2wCEAAoHCBUVFBcVFUYGBcZGhwGhoaGiIIIxwcHRogIBwhJCEgICwjIh0oHSAAjDUkkC0V...', 0)
  contentType: "image/jpeg"
  sold: 0
  createdAt: 2022-04-21T18:48:59.694+00:00
  updatedAt: 2022-04-21T18:48:59.694+00:00
  __v: 0

_id: ObjectId("6276c334fabb30be8b0a92af")
name: "Carrot"
price: 20
quantity: 250
farmer_id: ObjectId("6261739040865c739ed47d12")
photo: Object
  data: Binary('9j/4AAQSkZJRgABAQAAAQABAAAD/2wBDAgGBgcGBQgHBwcJCQgKDBQNDasLDBkSEw8UHwRoTHh0aHbwgJC4nICIsIxwcKDCpLDax...', 0)
  contentType: "image/jpeg"
  sold: 0
  createdAt: 2022-05-07T10:06:28.403+00:00
```

## vi. Store MongoDB URL in .env file

By creating a.env file in the api (backend) subdirectory, you can save the cluster's URL. In that URL, provide all of the MongoDB Atlas' credentials (password). This.env file is part of.gitignore. Furthermore, Mongoose uses the variable used to store this URL to connect to MongoDB.

```
//database
mongoose.connect(process.env.ONLINE_DATABASE, {
  useNewUrlParser:true,
  useCreateIndex:true,
  useUnifiedTopology:true

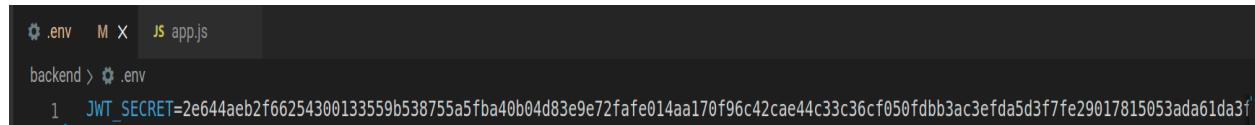
}).then(()=>console.log("DB Connected"));
```

## vii. JWT Authentication

JSON Web Tokens (JWT) are an RFC 7519 open industry standard for representing claims between two parties. For example, you can use jwt.io to decode, verify, and produce JWT.

JWT defines a concise and self-contained way for transmitting information between two parties as a JSON object. This information may be reviewed and trusted because it is signed. A secret (using the HMAC algorithm) or an RSA or ECDSA public/private key pair can be used to sign JWTs.

We create the JWT token and add it to .env file for performing authentication.



```
backend > cat .env
JWT_SECRET=2e644aeb2f66254300133559b538755a5fba40b04d83e9e72fabe014aa170f96c42cae44c33c36cf050fdbb3ac3efda5d3f7fe29017815053ada61da3f
```

## viii. Source Code Management

Source code management (SCM) is used to keep a track of all the modifications done to a source code repository. SCM tracks a current history of changes to a code base and helps resolve conflicts when merging updates from multiple contributors. SCM is very similar to Version control.

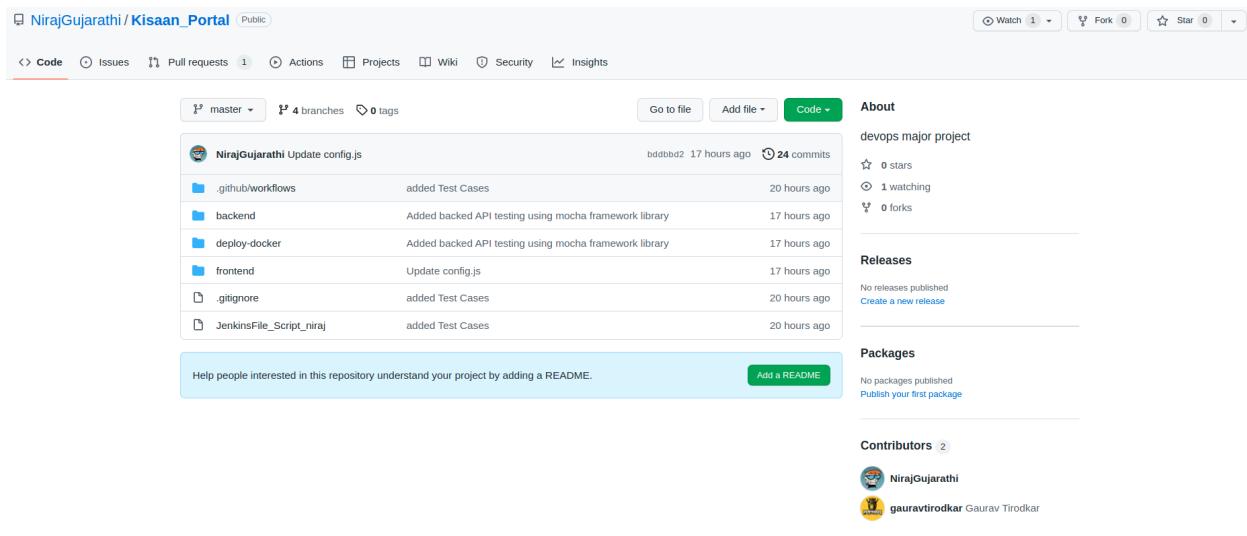


Fig . Create GitHub repository

To integrate project with GitHub we do the following steps :

- **git init**  
→ Initializes the project as a github repository locally.
- **git remote add origin**  
→ Add the details of the remote branch.
- **git add**  
→ This command stages all the changes of local repo.
- **git commit**  
→ Commit command commits the changes to the respective branch in the current repository.
- **git push**  
→ Push command will push the changes to the remote repository on GitHub.

```
TERMINAL PROBLEMS OUTPUT DEBUG CONSOLE

gaurav@gaurav-Predator-PH315-51:~/SPE-Project/Kisanan_Portal$ git init
Reinitialized existing Git repository in /home/gaurav/SPE-Project/Kisanan_Portal/.git/
gaurav@gaurav-Predator-PH315-51:~/SPE-Project/Kisanan_Portal$ 
```

Fig. Initialize git repository from terminal

```

TERMINAL PROBLEMS OUTPUT DEBUG CONSOLE

gaurav@gaurav-Predator-PH315-51:~/SPE-Project/Kisanan_Portals$ git add .
gaurav@gaurav-Predator-PH315-51:~/SPE-Project/Kisanan_Portals$ git commit -m "final commit"
[gaurav-working 2c0e406] final commit
 2 files changed, 526 insertions(+), 422 deletions(-)
 rewrite backend/.env (66%)
 rewrite frontend/src/user/FarmerDashboard.js (91%)
gaurav@gaurav-Predator-PH315-51:~/SPE-Project/Kisanan_Portals$ git push origin master

```

**Fig. Add to staging area and push to GitHub remote**

## ix. API Documentation

API	Details
GET : <a href="http://localhost:8000/api/vegetables">http://localhost:8000/api/vegetables</a> <a href="https://kisanan-portal.herokuapp.com/api/vegetables">https://kisanan-portal.herokuapp.com/api/vegetables</a>	Details all available vegetables by all farmers
GET : <a href="http://localhost:8000/api/vegetables/6261a71b45c5fc794964dc53">http://localhost:8000/api/vegetables/6261a71b45c5fc794964dc53</a> <a href="https://kisanan-portal.herokuapp.com/api/vegetables/6261a71b45c5fc794964dc53">https://kisanan-portal.herokuapp.com/api/vegetables/6261a71b45c5fc794964dc53</a>	Get specific vegetable details
GET : <a href="http://localhost:8000/api/farmers">http://localhost:8000/api/farmers</a> <a href="https://kisanan-portal.herokuapp.com/api/farmers">https://kisanan-portal.herokuapp.com/api/farmers</a>	Details of all farmers
GET : <a href="http://localhost:8000/api/farmer/6261739040865c739ed47d12">http://localhost:8000/api/farmer/6261739040865c739ed47d12</a> <a href="https://kisanan-portal.herokuapp.com/api/farmer/6261739040865c739ed47d12">https://kisanan-portal.herokuapp.com/api/farmer/6261739040865c739ed47d12</a>	Get specific farmer details giving farmer ID
POST : <a href="http://localhost:8000/api/signin">http://localhost:8000/api/signin</a> BODY : { "email": " <a href="mailto:Test2@gmail.com">Test2@gmail.com</a> ", "password": "Test123" }	SignIn API for user
POST : <a href="http://localhost:8000/api/signup">http://localhost:8000/api/signup</a> BODY : { "name": "Tester2" , "email": " <a href="mailto:Test2@gmail.com">Test2@gmail.com</a> " , "password": "Test123" , "location": " <a href="#">Maharashtra</a> " , "role": "0" }	SignUP API for User
DELETE : <a href="http://localhost:8000/api/vegetables/6276c334fabb30be8b0a92af/6261739040865c739ed47d12">http://localhost:8000/api/vegetables/6276c334fabb30be8b0a92af/6261739040865c739ed47d12</a>	Delete vegetable from specific userID

## X. Screenshots from POSTMAN API testing

**GET** http://localhost:8000/api/vegetables/6261a71b45c5fc794964dc53

**Params** Authorization Headers (7) Body Pre-request Script Tests Settings

**Query Params**

KEY	VALUE

**Body** Cookies Headers (8) Test Results

Pretty Raw Preview Visualize JSON

```

1
2   "_id": "6261a71b45c5fc794964dc53",
3   "name": "Brinjal",
4   "price": 45,
5   "quantity": 120,
6   "farmer_id": "6261739040865c739ed47d12",
7   "sold": 0,
8   "createdAt": "2022-04-21T18:48:59.694Z",
9   "updatedAt": "2022-04-21T18:48:59.694Z",
10  "__v": 0
11

```

**GET** http://localhost:8000/api/farmers

**Params** Authorization Headers (7) Body Pre-request Script Tests Settings

**Query Params**

KEY	VALUE

**Body** Cookies Headers (8) Test Results

Pretty Raw Preview Visualize JSON

```

1
2   {
3     "_id": "6261739040865c739ed47d12",
4     "name": "niraj",
5     "createdAt": "2022-04-21T15:09:04.871Z",
6     "updatedAt": "2022-04-21T15:09:04.871Z",
7     "__v": 0
8
9

```

New Collection / New Request

**GET** http://localhost:8000/api/vegetables

**Params** Authorization Headers (8) **Body** Pre-request Script Tests Settings

none form-data x-www-form-urlencoded raw binary GraphQL

This request does not have a body.

Status: 200 OK Time: 119 ms Size: 1.06 KB Save

Pretty Raw Preview Visualize

```

[{"_id": "6261a71b45c5fc794964dc53", "name": "Brinjal", "price": 45, "quantity": 120, "farmer_id": {"_id": "6261739040865c739ed47d12", "name": "niraj", "email": "niraj@gmail.com", "hashed_password": "dd4f10170e50ed3261636a852452f67dc4c063c9c"}, "location": "Karnataka", "salt": "fbfa4a1c0-c184-11ec-8033-bb756792af3f", "role": 1, "__v": 0}, {"sold": 0, "createdAt": "2022-04-21T18:48:59.694Z", "__v": 0}, {"_id": "6276c34fabb30be8b0a92af", "name": "Carrot", "price": 20, "quantity": 250, "farmer_id": {"_id": "6261739040865c739ed47d12", "name": "niraj", "email": "niraj@gmail.com", "hashed_password": "dd4f10170e50ed3261636a852452f67dc4c063c9c"}, "location": "Karnataka", "salt": "fbfa4a1c0-c184-11ec-8033-bb7f6792af3f", "role": 1, "__v": 0}, {"sold": 0, "createdAt": "2022-05-07T19:06:28.403Z", "__v": 0}]

```

**POST** http://localhost:8000/api/signin

**Params** Authorization Headers (10) **Body** Pre-request Script Tests Settings

none form-data x-www-form-urlencoded raw binary GraphQL JSON

Pretty Raw Preview Visualize

```

1
2   {
3     "email": "Test@gmail.com",
4     "password": "Test123"
5

```

**Body** Cookies (1) Headers (9) Test Results

Status: 200 OK Time: 119 ms

Pretty Raw Preview Visualize JSON

```

1
2   {
3     "token": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJfaWQioiI2Mjc3ZDaxNmYzMGV1YT1ZjVhZGNkZTgiLCJpYXQiOjE2NTIwMTkyNzZ9.gTj-jE222NrM703yaWm5yV82fzqS571
4     "user": {
5       "_id": "6277d016f30eba65f5adcde8",
6       "email": "Test@gmail.com",
7       "name": "Tester2",
8       "location": "Maharashtra",
9       "role": 0
10      }

```

The screenshot shows the Postman interface with a POST request to `http://localhost:8000/api/signup`. The request body is set to `JSON` and contains the following JSON data:

```

1  {
2      "name": "Tester2",
3      "email": "Test2@gmail.com",
4      "password": "Test123",
5      "location": "Maharashtra",
6      "role": 0
7  }
8

```

The response tab shows the status as `200 OK` with the following JSON data:

```

1  {
2      "user": {
3          "name": "Tester2",
4          "email": "Test2@gmail.com",
5          "location": "Maharashtra",
6          "role": 0,
7          "_id": "6277d016f30eba65f5adcde8",
8          "__v": 0
9      }
10 }

```

## External API integrations with application

We have used mainly 2 external API to track weather conditions of farmer's location as well as farmer and customer gets access to current vegetable market (mandi) prices at currant day

- Market - mandi prices
  - <https://tn.data.gov.in/resources/current-daily-price-various-commodities-various-markets-mandi/api>
  - <https://api.data.gov.in/resource/9ef84268-d588-465a-a308-a864a43d0070?api-key=579b464db66ec23bdd000001a91006a69a344a2f5750f6bd a75532d8&format=json&offset=0&limit=1000>

```
{
  "created": 1627939168,
  "updated": 1652064920,
  "created_date": "2021-08-02T21:19:28Z",
  "updated_date": "2022-05-09T08:25:20Z",
  "active": "1",
  "index_name": "9ef84268-d588-465a-a308-a864a43d0070",
  "org": [
    "Ministry of Agriculture and Farmers Welfare",
    "Department of Agriculture and Farmers Welfare"
  ],
  "org_type": "Central",
  "source": "data.gov.in",
  "title": "Current Daily Price of Various Commodities from Various Markets (Mandi)",
  "external_ws_url": "",
  "visualizable": "1",
  "field": [
    {
      "name": "State",
      "id": "state",
      "type": "keyword"
    },
    {
      "name": "District",
      "id": "district",
      "type": "keyword"
    },
    {
      "name": "Market",
      "id": "market",
      "type": "keyword"
    },
    {
      "name": "Commodity",
      "id": "commodity",
      "type": "keyword"
    },
    {
      "name": "Variety",
      "id": "variety",
      "type": "keyword"
    },
    {
      "name": "Arrival_Date",
      "id": "arrival_date",
      "type": "date"
    },
    {
      "name": "Min_x0020_Price",
      "id": "min_price",
      "type": "double"
    },
    {
      "name": "Max_x0020_Price",
      "id": "max_price",
      "type": "double"
    },
    {
      "name": "Modal_x0020_Price",
      "id": "modal_price",
      "type": "double"
    }
  ]
}
```

```

    "desc": "Current Daily Price of Various Commodities from Various Markets (Mandi)",
    "message": "Resource lists",
    "version": "2.2.0",
    "status": "ok",
    "total": 277,
    "count": 277,
    "limit": "1000",
    "offset": "0",
    "records": [
        {
            "state": "Karnataka",
            "district": "Bagalkot",
            "market": "Mhalingapur",
            "commodity": "Gur(Jaggery)",
            "variety": "Nizamabad",
            "arrival_date": "08\05\2022",
            "min_price": "2575",
            "max_price": "3750",
            "modal_price": "3455"
        },
        {
            "state": "Karnataka",
            "district": "Bangalore",
            "market": "Ramanagara",
            "commodity": "Beans",
            "variety": "Beans (Whole)",
            "arrival_date": "08\05\2022",
            "min_price": "4000",
            "max_price": "6000",
            "modal_price": "5000"
        },
        {
            "state": "Karnataka",
            "district": "Bangalore",
            "market": "Ramanagara",
            "commodity": "Beetroot",
            "variety": "Beetroot",
            "arrival_date": "08\05\2022",
            "min_price": "1000",
            "max_price": "1500",
            "modal_price": "1200"
        },
        {
            "state": "Karnataka",
            "district": "Bangalore",
            "market": "Ramanagara",
            "commodity": "Bhindi(Ladies Finger)",
            "variety": "Bhindi",
            "arrival_date": "08\05\2022",
            "min_price": "1600",
            "max_price": "2400",
            "modal_price": "2000"
        }
    ]
}

```

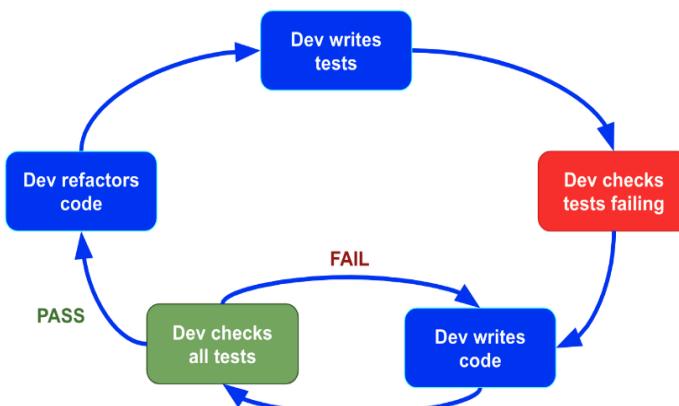
Above API fetches market mandi prices and following fields are present in API which includes max - min prices of vegetable item, its category

- Open weather Api which will track next 5 days weather predictions
  - <https://openweathermap.org/forecast16>
  - <https://api.openweathermap.org/data/2.5/forecast?q=Karnataka&units=metric&cnt=50&appid=44aef6ede898ea08ff2f4d21eea9762c>

```
{
  "cod": "200",
  "message": 0,
  "cnt": 40,
  "list": [
    {
      "dt": 1652086800,
      "main": {
        "temp": 32.2,
        "feels_like": 32.21,
        "temp_min": 32.2,
        "temp_max": 33.45,
        "pressure": 1005,
        "sea_level": 1005,
        "grnd_level": 921,
        "humidity": 38,
        "temp_kf": -1.25,
        "weather": [
          {
            "id": 804,
            "main": "Clouds",
            "description": "overcast clouds",
            "icon": "04d"
          }
        ],
        "clouds": {
          "all": 95
        },
        "wind": {
          "speed": 5.61,
          "deg": 304,
          "gust": 5.6
        },
        "visibility": 10000,
        "pop": 0.24,
        "sys": {
          "pod": "d"
        },
        "dt_txt": "2022-05-09 09:00:00"
      },
      "dt": 1652097600,
      "main": {
        "temp": 27.4,
        "feels_like": 28.87,
        "temp_min": 25.31,
        "temp_max": 27.4,
        "pressure": 1005,
        "sea_level": 1005,
        "grnd_level": 920,
        "humidity": 63,
        "temp_kf": 2.09,
        "weather": [
          {
            "id": 500,
            "main": "Rain",
            "description": "light rain",
            "icon": "10n"
          }
        ],
        "clouds": {
          "all": 94
        },
        "wind": {
          "speed": 8.24,
          "deg": 327,
          "gust": 11
        },
        "visibility": 4190,
        "pop": 0.6,
        "rain": {
          "3h": 2.88
        },
        "sys": {
          "pod": "d"
        },
        "dt_txt": "2022-05-09 12:00:00"
      },
      "dt": 1652108400,
      "main": {
        "temp": 25.31,
        "feels_like": 25.67,
        "temp_min": 25.31,
        "temp_max": 25.31,
        "pressure": 1006,
        "sea_level": 1006,
        "grnd_level": 921,
        "humidity": 68,
        "temp_kf": 0,
        "weather": [
          {
            "id": 500,
            "main": "Rain",
            "description": "light rain",
            "icon": "10n"
          }
        ],
        "clouds": {
          "all": 100
        },
        "wind": {
          "speed": 3.93,
          "deg": 211,
          "gust": 6.4
        },
        "visibility": 10000,
        "pop": 0.68,
        "rain": {
          "3h": 2.88
        },
        "sys": {
          "pod": "n"
        },
        "dt_txt": "2022-05-09 15:00:00"
      },
      "dt": 1652119200,
      "main": {
        "temp": 24.6,
        "feels_like": 25.2,
        "temp_min": 24.6,
        "temp_max": 24.6,
        "pressure": 1006,
        "sea_level": 1006,
        "grnd_level": 921,
        "humidity": 80,
        "temp_kf": 0,
        "weather": [
          {
            "id": 804,
            "main": "Clouds",
            "description": "overcast clouds",
            "icon": "04n"
          }
        ],
        "clouds": {
          "all": 100
        },
        "wind": {
          "speed": 3.14,
          "deg": 251,
          "gust": 4.51
        },
        "visibility": 10000,
        "pop": 0.64,
        "sys": {
          "pod": "n"
        },
        "dt_txt": "2022-05-09 18:00:00"
      },
      "dt": 1652130000,
      "main": ...
    }
  ]
}
```

## x. Testing

To make sure the project is running alright we provide a set of test cases. We use a library called ‘Chai Mocha’. Mocha is a JavaScript test framework running on Node.js and in the browser. Mocha allows asynchronous testing, test coverage reports, and use of any assertion library. Chai is a BDD / TDD assertion library for NodeJS and the browser that can be delightfully paired with any javascript testing framework.



## xii. Backend API testing using Mocha framework

Mocha is a testing framework for Node.js . It is useful for unit and integration testing. Mocha provides the describe functionality which helps in implementing test cases based on the description. Inside each describe section, specific details about each feature can be tested using the it handle.

```
describe("Vegetables API Testing", function(){
    global.id="";
    describe("Testing - Vegetables Details Fetch", function(){
        var url = "https://Kisan-portal.herokuapp.com/api/vegetables";
        it("returns status http-code 200", function(done){
            request({
                url: url,
                method: "GET",
            }, function(error, response, body){
                //console.log(body);
                //console.log(response.statusCode);
                expect(response.statusCode).to.equal(200);
                done();
            })
        })
    })
})
```

```
describe("Testing - SignIn", function(){
    var url = "https://Kisan-portal.herokuapp.com/api/signin";
    it("returns status http-code 200", function(done){
        request({
            url: url,
            method: "POST",
            headers: {'Content-Type': 'application/json'},
            body: {email: "niraj@gmail.com", password: "niraj123"},  
            json: true
        }, function(error, response, body){
            expect(response.statusCode).to.equal(200);
            done();
        })
    })
    it("New Login", function(done){
        request({
            url: url,
            method: "POST",
            headers: {'Content-Type': 'application/json'},
            body: {email: "niraj@gmail.com", password: "niraj123"},  
            json: true
        }, function(error, response, body){
            //console.log(body);
            expect(response.statusCode).to.equal(200);
            done();
        })
    })
})
```

Fig. SignIn authentication and Fetch details from vegetables list

In the above example, the SignIn / SignUp as well as various GET methods for vegetables and farmers are tested by first encapsulating it using **describe**. Then positive and negative test cases are designed to test functionalities based on different parameters.

## xiii. Containerization

Docker is an open source platform for developing, shipping, and running applications. Docker enables users to separate your applications from your infrastructure so you can deliver software quickly. This allows us to deploy products directly to users' computers without installing each software one-by-one.

We need to create an account on DockerHub, which is a public registry.

We then push our created image on this repository, this is publicly available and can be pulled by any user and deployed on a local machine.

The screenshot shows two Docker Hub repository pages for the user gtirodkar1234.

**Repository 1: kisanportal-client**

- General Information:** Docker Image for Major Project client, Last pushed: 10 days ago.
- Docker commands:** docker push gtirodkar1234/kisanportal-client:tagname

**Repository 2: kisanportal-server**

- General Information:** Docker Image for Major Project server, Last pushed: 10 days ago.
- Docker commands:** docker push gtirodkar1234/kisanportal-server:tagname

## xiv. Create repository on docker-hub

Link :

- <https://hub.docker.com/repository/docker/gtirodkar1234/kisanportal-server>
- <https://hub.docker.com/repository/docker/gtirodkar1234/kisanportal-client>
- <https://hub.docker.com/repositories>

## XV. Create DockerFile (Frontend v/s Backend)

```
backend > 📁 Dockerfile > ⚙ FROM
1  FROM node:latest
2
3  WORKDIR /usr/src/app
4
5  COPY ./package.json ./
6  COPY ./package-lock.json ./
7
8  RUN npm install
9  COPY . .
10
11 EXPOSE 8000
12 CMD [ "npm", "start" ]
13
```

```
frontend > 📁 Dockerfile > ⚙ FROM
1  FROM node:latest
2
3  WORKDIR /usr/src/app
4
5  COPY ./package.json ./
6  COPY ./package-lock.json ./
7
8  RUN npm install
9
10 COPY . .
11
12 EXPOSE 3000
13 CMD [ "npm", "start" ]
14
```

Docker holds a set of all the commands a user could call on the command line to assemble an image. For our node project we set to copy the package.json and package-lock.json to the container and then run npm install to install all the packages there.

## xvi. CI/CD Pipeline

**Continuous Integration :** Continuous integration (CI) is the practice of automating the integration of code changes from multiple contributors into a single software project.

**Continuous Delivery :** Continuous Delivery is the ability to get changes of all types such as including new features, configuration changes, bug fixes and experiments into production, or into the hands of users, safely and quickly in a sustainable way.

We use Jenkins to build our CI/CD pipeline. Jenkins is an open source automation server. It helps automate the parts of software development related to building, testing, and deploying, facilitating continuous integration and continuous delivery. It is a server-based system that runs in

servlet containers such as Apache Tomcat.

### a. Manage Plugins

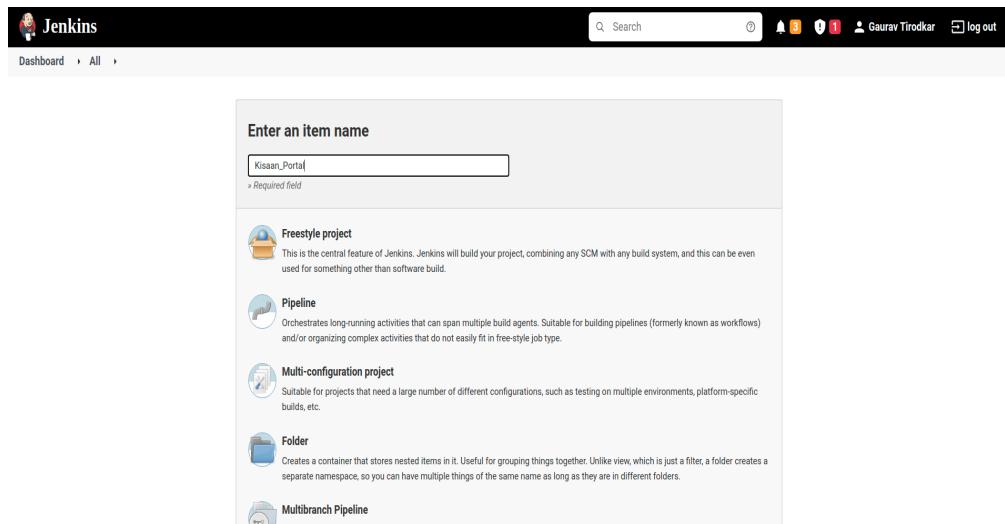
We use Jenkins to automate Building, Testing and Deployment of the entire project with just a single click. To begin with we first need to install various plugins.

We install Docker Pipeline, GitHub, Ansible , Maven Integration, etc. plugins by going to Manage Jenkins -> Manage Plugins. After all the plugins are installed, we need to restart our Jenkins and then add Docker credentials.

To connect our project with the Docker Hub repository we need to add credentials to the Docker Hub repository and also set up an unique id which is equal to docker with Registry credentials id in pipeline script.

### b. Create a new pipeline in Jenkins

To set up our project, we create a project with Kisan Portal as name and configure it. This type of a project is called a pipeline project as opposed to the regular project as we write a pipeline script ,i.e. A list of steps to be followed for project execution.



### c. Pipeline script in Jenkins

Before we set up the pipeline we declare environment variables to use in the pipeline in the future. We add variables for server and client image and docker credentials.

```
JenkinsFile_Script_niraj
1 pipeline{
2     environment{
3         server_image = ""
4         client_image= ""
5         registryCredential='docker'
6     }
7     agent any
8     stages{
9         stage('Stage 1: Git Clone'){
10            steps{
11                git branch: 'niraj-working', url: 'https://github.com/NirajGujarathi/Kisanan_Portal.git'
12            }
13        }
}
```

Stage 1 : Clone the repository from GitHub

```
stage('Stage 2: Build Docker Server'){
    steps{
        dir("backend"){
            script{
                server_image= docker.build "nirajg/kisanan_portal_server:latest"
            }
        }
    }
}
stage('Stage 3: Build Docker Client'){
    steps{
        dir("frontend"){
            script{
                client_image= docker.build "nirajg/kisanan_portal_client:latest"
            }
        }
    }
}
```

Step 2 : Build Docker Image using specified tag

```
33 
34 
35 
36 
37 
38 
39 
40 
41 
42
        stage('Stage : npm testing'){
            steps{
                dir("backend"){
                    sh 'npm i'
                    sh 'npm run test'
                }
            }
        }
```

npm mocha testing

```

stage('Stage 4: Push Server Docker Image to DockerHub') {
    steps {
        script {
            docker.withRegistry('', 'docker') {
                server_image.push()
            }
        }
    }
}
stage('Stage 5: Push Client Docker Image to DockerHub') {
    steps {
        script {
            docker.withRegistry('', 'docker') {
                client_image.push()
            }
        }
    }
}

```

Step 3 : Now we upload these built images to docker hub

To upload this to docker hub we need docker credentials, here the pre defined environment variables come in use.

```

stage('Stage 6: Ansible Deployment to machine'){
    steps{
        ansiblePlaybook becomeUser: 'null', colorized: true, credentialsId: 'docker',
        installation: 'Ansible', inventory: 'deploy-docker/inventory',
        playbook: 'deploy-docker/deploy-app.yml', sudoUser: 'null'
    }
}

```

Step 4 : Ansible Deployment

Ansible is used to pull images from Docker Hub public registry. The name tag in the docker image section defines which image to be pulled. The ansible user is defined in the inventory file.

```

deploy-docker >  ≡ inventory
      1  [ubuntu]
      2  localhost ansible_user=gaurav

```

## xvii. Ansible Deployment

Ansible is an open-source software provisioning, configuration management, and application-deployment tool enabling infrastructure as code. It runs on many Unix-like systems, and can configure both Unix-like systems as well as Microsoft Windows.

To perform configuration management using Ansible we need to create a playbook that has a docker compose file.

Docker compose is used to spin up multiple services together, it is used in multi-container applications. A single command is needed to run this compose file.

```
deploy-docker > 📁 docker-compose.yml
1   version: "3"
2   services:
3     react-app:
4       image: gtirodkar1234/kisanportal-client:latest
5       stdin_open: true
6       ports:
7         - "3000:3000"
8       networks:
9         - mern-app
10      api-server:
11        image: gtirodkar1234/kisanportal-server:latest
12        ports:
13          - "8000:8000"
14        networks:
15          - mern-app
16        depends_on:
17          - mongo
```

To begin with we set a schema version, it's best practice to go with the latest schema version but a fairly recent one will also do.

Next is to set up the services, first is the react-app service which is used for

the frontend of the application and api-server for the backend. The names are custom defined and will become their alias in the future.

Along with this we need a mongo image as well, we pull the `mongo:3.6.23-xenial` version to run.

The “*depends\_on*” option is used here as the backend will not run before the mongo container is built as the app may crash without the help of mongo. “*depends\_on*” ensures that the backend service is executed only after mongoose is ready.

```
mongo:  
  image: mongo:3.6.23-xenial  
  networks:  
    - mern-app
```

We have connected all this on a single network called “*mern-app*”, which allows intra network communication between the containers.

To run docker compose we create a playbook file called docker.yml

```
deploy-docker > ! deploy-app.yml  
1  ---  
2  - name: Pull and Run docker image  
3    hosts: all  
4    tasks:  
5      - name: copy docker compose file from folder to remote host  
6        copy:  
7          src: ./docker-compose.yml  
8          dest: ./  
9      - name: pull images specified in docker-compose  
10        command: docker-compose -f ./docker-compose.yml pull  
11      - name: run the pulled docker images in detached mode  
12        command: docker-compose -f ./docker-compose.yml up -d
```

This will copy the compose file to the remote host and run two commands, first to pull all the specified docker images and second to run all of them.

## Configure Ansible in Jenkins

We need to go to Jenkins -> Dashboard -> Manage Jenkins -> Global Tool Configuration and add Ansible there. Here, we also need to give the correct path to Ansible.

The screenshot shows the Jenkins Global Tool Configuration interface. Under the 'Ansible' section, there is a 'Ansible installations' list with one item named 'Ansible'. The configuration fields for this item include:

- Name: Ansible
- Path to ansible executables directory: /usr/bin/
- A checkbox labeled 'Install automatically' is checked.
- A red 'Delete Ansible' button is visible.

After jenkins configuration, you can run this ansible playbook directly.

## xviii. Log Management

Logging is a process of recording information generated by application activities into log files. Messages saved in the log file are called logs. A log is a single instance recorded in the log file.

To keep a track on the application process we need log management. At every step we add logger comments so that it becomes easier for problem identification while resolving any bugs.

To perform log management we use **winston**, which is a logger for node.js

```

backend > logger > JS index.js > ...
1  const winston = require('winston');
2  const logConfiguration = {
3      'transports': [
4          new winston.transports.File({
5              filename: './logs.log'
6          })
7      ],
8      format: winston.format.combine(
9          // winston.format.label({
10             //   label: `Label``
11            // }),
12            winston.format.timestamp({
13                format: 'YYYY-MM-DD HH:mm:ss'
14            }),
15            // ${info.level}: ${info.label}:
16            winston.format.printf(info => `${[info.timestamp]}, ${info.level}, ${info.message}`),
17        )
18    );
19  const logger = winston.createLogger(logConfiguration);
20  module.exports = logger;

```

To add any logger information we can run a simple command.

```
logger.info("Logger comment");
```

## xix. Building Jenkins Pipeline

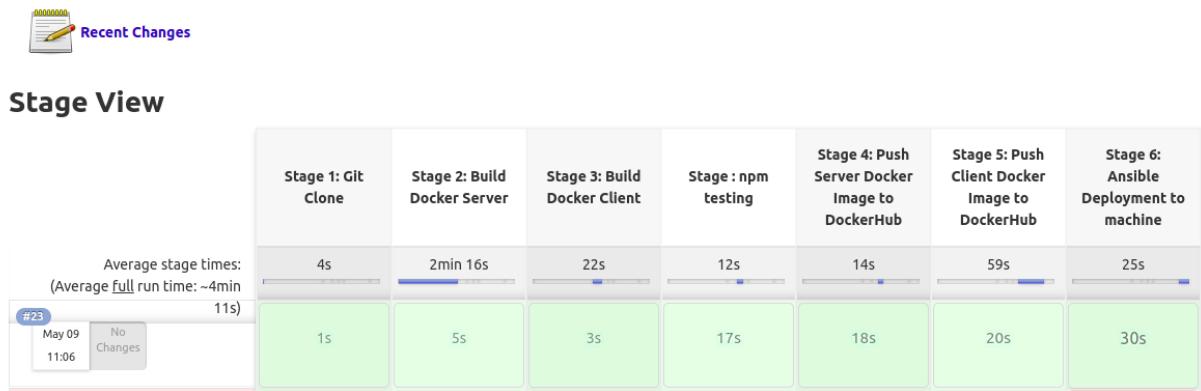
The screenshot shows the Jenkins Pipeline KisanPortal dashboard. On the left, there's a sidebar with various project management and pipeline-related links: Back to Dashboard, Status, Changes, Build Now, Configure, Delete Pipeline, Full Stage View, Rename, and Pipeline Syntax. Below this is the Build History section, which lists recent builds: #11 (Apr 28, 18:04, No Changes), #10 (Apr 28, 18:02, No Changes), #9 (Apr 28, 18:00, No Changes), and #8 (Apr 28, 17:59, No Changes). The main area is titled "Pipeline KisanPortal" and features a "Stage View" grid. The grid has six columns representing stages: Stage 1: Git Clone, Stage 2: Build Docker Server, Stage 3: Build Docker Client, Stage 4: Push Server Docker Image to DockerHub, Stage 5: Push Client Docker Image to DockerHub, and Stage 6: Ansible Deployment to machine. Each stage has a progress bar indicating its duration. Stage 1 takes 1s, Stage 2 takes 5s, Stage 3 takes 15s, Stage 4 takes 20s, Stage 5 takes 23s, and Stage 6 takes 10s. Below the grid, it says "Average stage times: (Average full run time: ~1min 19s)". A "Recent Changes" section is also visible on the left side of the main dashboard area.

The project builds in 6 stages :

1. Git Clone
2. Build Docker Server
3. Build Docker Client
4. npm testing
5. Push Server Docker Image to DockerHub
6. Push Client Docker Image to DockerHub
7. Ansible Deployment to machine

Once everything is done, the image is pulled to the local repository. We need to run this image to create a container. This container will deploy the jar file and the calculator will be executed.

### **Pipeline KisanPortal\_Pipeline**



## xx. Running image on local machine

```
niraj@niraj-iitb:~$ docker images
REPOSITORY          TAG      IMAGE ID      CREATED       SIZE
nirajg/kisan_portal_client  latest   2c681f4ad291  24 minutes ago  1.32GB
nirajg/kisan_portal_server  latest   11ce443ac048  25 minutes ago  1.07GB
```

```
niraj@niraj-iitb:~$ docker run -it nirajg/kisan_portal_server
> backend@1.0.0 start
> nodemon app.js

[nodemon] 2.0.15
[nodemon] to restart at any time, enter `rs`
[nodemon] watching path(s): ***!
[nodemon] watching extensions: js,mjs,json
[nodemon] starting `node app.js`
Server is running on 8000
DB Connected
|
```

Starting docker server container

```
niraj@niraj-iitb:~$ docker run -it nirajg/kisan_portal_client
> kisanportal@0.1.0 start
> react-scripts start
|
```

```
Compiled successfully!

You can now view kisanportal in the browser.

  http://localhost:3000

Note that the development build is not optimized.
To create a production build, use yarn build.

webpack compiled successfully
```

Starting docker client container

```

9d2386bc2653  nirajg/kisaan_portal_server:latest  "docker-entrypoint.s..."  About an hour ago  Exited (1) 17 seconds ago          niraj_api-server
└─f3278d1be8a9  nirajg/kisaan_portal_client:latest  "docker-entrypoint.s..."  About an hour ago  Exited (1) 10 seconds ago      niraj_react-app
1

```

```

OPTIONS /api/vegetables/by/search 204 0.546 ms - 0
GET /api/farmers 304 5413.703 ms - -
GET /api/farmers 304 2177.601 ms - -
POST /api/vegetables/by/search 200 7634.366 ms - 429
POST /api/vegetables/by/search 200 5859.108 ms - 429
GET /api/vegetables/photo/6261a71b45c5fc794964dc53 304 6639.536 ms - -
GET /api/vegetables/photo/6261a71b45c5fc794964dc53 304 708.680 ms - -
GET /api/vegetables/photo/6261a71b45c5fc794964dc53 304 127.292 ms - -
GET /api/vegetables/photo/6261a71b45c5fc794964dc53 304 3171.756 ms - -
GET /api/vegetables/photo/6261a71b45c5fc794964dc53 304 278.748 ms - -
GET /api/vegetables/photo/6261a71b45c5fc794964dc53 304 204.440 ms - -
GET /api/signout 304 4.513 ms - -
GET /api/vegetables?sortBy=createdAt&order=desc&limit6 304 339.700 ms -
GET /api/vegetables?sortBy=sold&order=desc&limit6 304 683.923 ms -
OPTIONS /api/signin 204 0.359 ms - 0
POST /api/signin 200 40.414 ms - 278
OPTIONS /api/vegetables/by/search 204 0.447 ms - 0
POST /api/vegetables/by/search 200 641.747 ms - 429
POST /api/vegetables/by/search 200 3830.270 ms - 429

```

## Backend API logs

The screenshot shows a web application interface for the 'Kisan Portal'. At the top, there's a header bar with navigation links: Home, BuyNow, Dashboard, Cart (with a count of 0), and Signout. Below the header, a large banner displays the text 'Welcome to Kisan Portal gaurav !'.

The main content area features a search interface with two input fields: 'Search district mandi from your state' and 'Search specific vegetable or fruit ( commodity )'. Above these fields, there's a teal-colored box containing the text 'Price in Rupees per Kilogram' and 'Updated On : Mon May 09 2022', along with a yellow 'Fetch Data' button.

To the left, a sidebar titled 'User Links' contains three items: 'My Cart', 'Buy Now', and 'Update Profile'. To the right, another sidebar titled 'User Information' lists the user's details: 'gaurav', 'gaurav@gmail.com', 'Maharashtra', and 'Customer'.

The central part of the page displays a table of vegetable prices:

State	District	Market	Commodity	Price Range	Variety
Maharashtra	Ahmednagar	Parner	Onion	1 - 11.25	Other
Maharashtra	Ahmednagar	Rahata	Bhindi(Ladies Finger)	10 - 25	Other
Maharashtra	Ahmednagar	Rahata	Bitter gourd	15 - 25	Other
Maharashtra	Ahmednagar	Rahata	Bottle gourd	5 - 11	Other
Maharashtra	Ahmednagar	Rahata	Brinjal	5 - 10	Other

## xxi. Continuous Monitoring

"ELK" is the acronym for three open source projects: Elasticsearch, Logstash, and Kibana. ELK stack gives us the ability to aggregate logs from all the systems and applications, analyze these logs, and create visualizations for application and infrastructure monitoring, faster troubleshooting, security analytics, and more.

Logger File -

```
2022-04-21 16:16:54, info, got Farmer list
2022-04-21 16:16:55, info, got Farmer list
2022-04-21 16:16:55, info, Filters applied and vegetables found
2022-04-21 16:17:00, info, Filters applied and vegetables found
2022-04-21 16:17:02, info, Filters applied and vegetables found
2022-04-21 16:18:00, info, got Farmer list
2022-04-21 16:18:00, info, got Farmer list
2022-04-21 16:18:00, info, Filters applied and vegetables found
2022-04-21 16:18:26, info, got Farmer list
2022-04-21 16:18:26, info, Filters applied and vegetables found
2022-04-21 16:18:26, info, got Farmer list
2022-04-21 16:18:30, info, Filters applied and vegetables found
2022-04-21 16:18:32, info, Filters applied and vegetables found
2022-04-21 16:18:41, info, got Farmer list
2022-04-21 16:18:41, info, Filters applied and vegetables found
2022-04-21 16:18:41, info, got Farmer list
2022-04-21 16:19:02, info, Filters applied and vegetables found
2022-04-21 16:19:11, info, Filters applied and vegetables found
2022-04-21 16:19:45, info, Vegetables found
2022-04-21 16:19:45, info, Vegetables found
2022-04-21 16:19:55, info, Signout successfully
2022-04-21 16:24:08, info, successfully login
2022-04-21 16:24:30, info, Signout successfully
2022-04-21 16:24:30, info, Vegetables found
2022-04-21 16:24:30, info, Vegetables found
2022-04-21 16:25:47, info, successfully login
2022-04-21 16:47:37, info, Vegetables found
2022-04-21 16:47:37, info, Vegetables found
2022-04-21 16:47:47, info, Signout successfully
```

**logs.log**

### Import data

**Simple** **Advanced**

**Index name**  
kisan-portal-logs

Create data view  
**Data view name**  
kisan-portal-logs

**Combined fields**  
 Add combined field

**Index settings**

```

1 > {
2   "number_of_shards": 1
3 }

```

**Mappings**

```

2 > {
3   "properties": {
4     "@timestamp": {
5       "type": "date"
6     },
7     "column1": {
8       "type": "date",
9       "format": "yyyy-MM-dd HH:mm:ss"
10    },
11    "column2": {
12      "type": "keyword"
13    },
14    "column3": {
15      "type": "keyword"
16    }
17 }

```

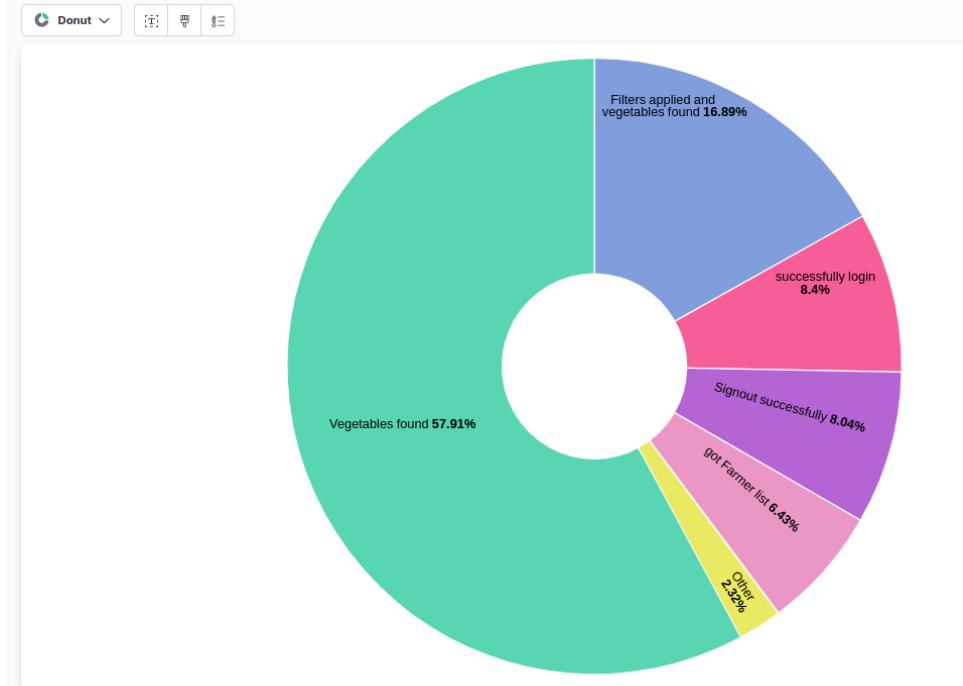
**Ingest pipeline**

```

1 > {
2   "description": "Ingest pipeline created by
3   text structure finder",
4   "processors": [
5     {
6       "csv": {
7         "field": "message",
8         "target_fields": [
9           "column1",
10          "column2",
11          "column3"
12        ],
13        "ignore_missing": false
14      }
15    },
16    {
17      "date": {
18        "field": "column1",
19        "timezone": "[ event.timezone ]",
20        "formats": [
21          "yyyy-MM-dd HH:mm:ss"
22        ]
23      }
24    }
25 }

```

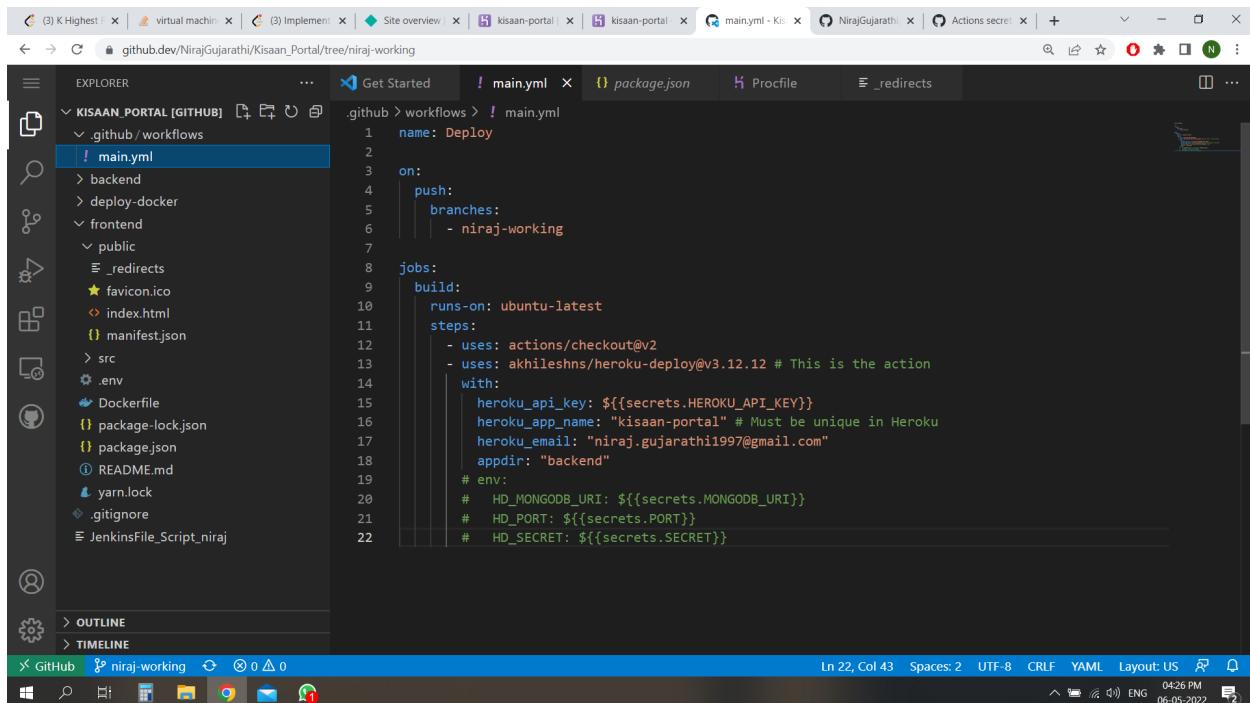
**Import**



Data visualization using ELK

## 6. Deployment - heroku and netlify -

We used **Github actions** as a Continuous Integration tool, to automate the workflow actions. We have the following stages/steps in our workflow. The workflows in Github Actions can be defined as jobs, each of these jobs in turn have multiple steps, which execute sequentially.



The screenshot shows a GitHub repository interface with the main.yml file selected in the code editor. The code defines a workflow named 'Deploy' that triggers on pushes to the 'niraj-working' branch. It uses an Ubuntu-latest runner and performs several steps, including checkout, Heroku deployment, and setting environment variables for MongoDB URI, port, and secret.

```
name: Deploy
on:
  push:
    branches:
      - niraj-working
jobs:
  build:
    runs-on: ubuntu-latest
    steps:
      - uses: actions/checkout@v2
      - uses: akhileshns/heroku-deploy@v3.12.12 # This is the action
        with:
          heroku_api_key: ${{secrets.HEROKU_API_KEY}}
          heroku_app_name: "kisan-portal" # Must be unique in Heroku
          heroku_email: "miraj.gujarathi1997@gmail.com"
          appdir: "backend"
      # env:
      #   HD_MONGODB_URI: ${{secrets.MONGODB_URI}}
      #   HD_PORT: ${{secrets.PORT}}
      #   HD_SECRET: ${{secrets.SECRET}}
```

Main.yml file for backend deployment on heroku - using CI / CD whenever new changes are triggered on niraj-working branch application is automatically deployed to cloud

```
name: Deploy
on:
  push:
    branches:
      - niraj-working

jobs:
  build:
    runs-on: ubuntu-latest
    steps:
```

- uses: actions/checkout@v2
- uses: akhileshns/heroku-deploy@v3.12.12 # This is the action with:
 

```
heroku_api_key: ${secrets.HEROKU_API_KEY}
heroku_app_name: "kisan-portal" # Must be unique in Heroku
heroku_email: "niraj.gujarathi1997@gmail.com"
appdir: "backend"
```

The screenshot shows the GitHub Actions Workflows interface. At the top, there's a navigation bar with 'Workflows' and a 'New workflow' button. Below it is a feedback section with a 'Give feedback' button. The main area is titled 'All workflows' and shows '16 workflow runs'. A search bar labeled 'Filter workflow runs' is present. The workflow runs are listed in a table with columns for name, status, event, status, branch, and actor. Four recent runs are shown:

Workflow Run	Status	Event	Status	Branch	Actor
Update FarmerDashboard.js	Success	Deploy #16	niraj-working	11 hours ago	21s
changed layout	Success	Deploy #15	niraj-working	11 hours ago	21s
Add files via upload	Success	Deploy #14	niraj-working	11 hours ago	20s
updated some css changes styling	Success	Deploy #13	niraj-working	11 hours ago	21s

A specific workflow run, 'Update FarmerDashboard.js Deploy #16', is expanded. It shows a summary card with a checkmark icon, the workflow name, and the deployment ID. The card also includes a 'Summary' link and a 'Jobs' link. The detailed view for this run shows the job named 'build' with a success status. The job details show the following steps:

- > Set up job
- > Run actions/checkout@v2
- > Run akhileshns/heroku-deploy@v3.12.12
- > Post Run actions/checkout@v2
- > Complete job

## 7. App Deployment

The entire website is deployed on two services, Netlify and Heroku. Netlify hosts the front end and heroku hosts the backend of the app.

The screenshot shows the Netlify Team overview page for the 'NG' team. It displays resource usage statistics: Bandwidth used (0 KB/100 GB), Build minutes used (1/300), Concurrent builds (0/1), and Team members (1). Below this, the 'Sites' section lists two sites: 'kisaan-portal' and 'incandescent-sprite-f353bc', both deployed from GitHub. The 'Builds' section shows completed builds for 'kisaan-portal' and 'incandescent-sprite-f353bc'. The browser address bar shows the URL <https://app.netlify.com/teams/niraj-gujarathi1997/overview>.

The screenshot shows the Netlify Deploy details page for the 'kisaan-portal' site. It highlights a 'Published deploy for kisaan-portal' made at 5:06 PM by NirajGujarathi on GitHub. The summary indicates 4 new files uploaded, 1 redirect rule processed, and no header rules or linked resources found. A 'Play game' button is also present. The browser address bar shows the URL <https://app.netlify.com/sites/kisaan-portal/deploy/6275083712cdfe000835023e>.

**Builds**

Usage & insights

**Builds**

Build	Time	Status	Details
kisan-portal: Production: niraj-working@6ce9f5d	Today at 12:21 AM	Completed	By NirajGujarathi: Update FarmerDashboard.js Deployed in 39s
incandescent-sprite-f353bc: Production: niraj-working@6ce9f5d	Today at 12:21 AM	Completed	By NirajGujarathi: Update FarmerDashboard.js Deployed in 40s
kisan-portal: Production: niraj-working@7eafbd4	Today at 12:07 AM	Completed	By NirajGujarathi: changed layout Deployed in 45s
incandescent-sprite-f353bc: Production: niraj-working@7eafbd4	Today at 12:07 AM	Completed	By NirajGujarathi: changed layout Deployed in 44s
kisan-portal: Production: niraj-working@e9e7fe1	Today at 12:06 AM	Completed	By NirajGujarathi: Add files via upload Deployed in 45s
incandescent-sprite-f353bc: Production: niraj-working@e9e7fe1	Today at 12:06 AM	Completed	

The screenshot shows the Heroku dashboard for the 'kisan-portal' app. At the top, there's a navigation bar with tabs like Overview, Resources, Deploy, Metrics, Activity, Access, and Settings. Below that is a breadcrumb trail: Activity Feed > Build Log. The main content area displays the build log output:

```

----> Building on the Heroku-20 stack
----> Determining which buildpack to use for this app
----> Node.js app detected

----> Creating runtime environment
      NPM_CONFIG_LOGLEVEL=error
      NODE_VERBOSE=false
      NODE_ENV=production
      NODE_MODULES_CACHE=true

----> Installing binaries
      engines.node (package.json): unspecified
      engines.npm (package.json): unspecified (use default)

      Resolving node version 16.x...
      Downloading and installing node 16.15.0...
      Using default npm version: 8.5.5

----> Installing dependencies
      Installing node modules
Build finished
  
```

At the bottom of the page, there's a footer with links to heroku.com, Blogs, Careers, Documentation, Support, Terms of Service, Privacy, Cookies, and a copyright notice for © 2022 Salesforce.com. There are also icons for a user profile, notifications, and other platform features.

Heroku deployment logs

Installed add-ons **\$0.00/month**

There are no add-ons for this app  
You can add add-ons to this app and they will show here. [Learn more](#)

Dyno formation **\$0.00/month**

This app has no process types yet  
Add a Procfile to your app in order to define its process types. [Learn more](#)

Collaborator activity

niraj.gujarathi1997@gmail.com 1 deploy

Latest activity

- niraj.gujarathi1997@gmail.com: Deployed b2c032f5 Today at 4:25 PM · v3
- niraj.gujarathi1997@gmail.com: Build succeeded Today at 4:25 PM · [View build log](#)
- niraj.gujarathi1997@gmail.com: Enable Logplex Today at 4:03 PM · v2
- niraj.gujarathi1997@gmail.com: Initial release Today at 4:03 PM · v1

https://dashboard.heroku.com/apps/kisan-portal/settings

Support Terms of Service Privacy Cookies © 2022 Salesforce.com 04:26 PM 06-05-2022

Application Logs

```

2022-05-06T11:44:33.514185+00:00 app[web.1]: ⚡[Om]GET /api/signout [32m200ms] 0.1.266 ms - 330[Om
2022-05-06T11:44:33.982648+00:00 heroku[router]: at=info method=GET path="/api/vegetables?sortBy=createdAt&order=desc&limit=6" host=kisan-portal.herokuapp.com
request_id=d0232cd03-43c0-4563-b1c0-ef5d7118cb2 fwd="103.156.19.229" dyno=web.1 connect=0ms service=38ms status=200 bytes=183 protocol=https
2022-05-06T11:44:33.980188+00:00 app[web.1]: ⚡[Om]GET /api/vegetables?sortBy=createdAt&order=desc&limit=6 [36m304ms] 0m 385.544 ms - 0[Om
2022-05-06T11:44:33.947143+00:00 app[web.1]: ⚡[Om]GET /api/vegetables?sortBy=soldOrderer-desc&limit=6 [36m304ms] 0m 410.395 ms - 0[Om
2022-05-06T11:44:33.947179+00:00 heroku[router]: at=info method=GET path="/api/vegetables?sortBy=soldOrderer-desc&limit=6" host=kisan-portal.herokuapp.com request_id=c602e79d-045d-45d9-9e68-e6777c4d42a fwd="103.156.19.229" dyno=web.1 connect=0ms service=412ms status=304 bytes=183 protocol=https
2022-05-06T11:44:46.468180+00:00 app[web.1]: ⚡[Om]OPTIONS /api/signin [32m204ms] 0m 0.801 ms - 0[Om
2022-05-06T11:44:46.467511+00:00 heroku[router]: at=info method=OPTIONS path="/api/signin" host=kisan-portal.herokuapp.com request_id=49e43622-f2e2-400e-ba11-23dbf32373c4
fwd="103.156.19.229" dyno=web.1 connect=0ms service=10ms status=204 bytes=30 protocol=https
2022-05-06T11:44:46.921267+00:00 heroku[router]: at=info method=POST path="/api/signin" host=kisan-portal.herokuapp.com request_id=193fa662-ea11-46ab-b727-ebca9ac89f04
fwd="103.156.19.229" dyno=web.1 connect=0ms service=210ms status=200 bytes=694 protocol=https
2022-05-06T11:44:46.921881+00:00 app[web.1]: ⚡[Om]POST /api/signin [32m200ms] 0m 205.977 ms - 2800[Om
2022-05-06T11:44:59.692611+00:00 heroku[router]: at=info method=GET path="/api/signout" host=kisan-portal.herokuapp.com request_id=062e3c2b-5101-44ae-af8e-eab52c44817a
fwd="103.156.19.229" dyno=web.1 connect=0ms service=10ms status=304 bytes=24 protocol=https
2022-05-06T11:45:00.076840+00:00 app[web.1]: ⚡[Om]GET /api/vegetables?sortBy=createdAt&order=desc&limit=6 [36m304ms] 0m 377.011 ms - 0[Om
2022-05-06T11:45:00.087959+00:00 app[web.1]: ⚡[Om]GET /api/vegetables?sortBy=soldOrderer-desc&limit=6 [36m304ms] 0m 391.279 ms - 0[Om
2022-05-06T11:45:00.693463+00:00 app[web.1]: ⚡[Om]GET /api/signout [36m304ms] 0m 8.878 ms - 0[Om
2022-05-06T11:45:00.886689+00:00 heroku[router]: at=info method=GET path="/api/vegetables?sortBy=soldOrderer-desc&limit=6" host=kisan-portal.herokuapp.com request_id=cc38cb79-c447-43b1-8e47-9cb81bf5cc fwd="103.156.19.229" dyno=web.1 connect=0ms service=399ms status=304 bytes=183 protocol=https
2022-05-06T11:45:00.976854+00:00 heroku[router]: at=info method=GET path="/api/vegetables?sortBy=createdAt&order=desc&limit=6" host=kisan-portal.herokuapp.com
request_id=j7222ecd-8a36-44f2-88ad-b69d7f98f129 fwd="109.161.98.68" dyno=web.1 connect=0ms service=389ms status=304 bytes=183 protocol=https

```

Save

heroku.com Blogs Careers Documentation Support Terms of Service Privacy Cookies © 2022 Salesforce.com 05:16 PM 05-05-2022

https://kisan-portal.herokuapp.com/api/vegetables/

```
[{"id": "6261a71b45c5fc79496dc53", "name": "Brinjal", "price": 45, "quantity": 120, "farmer_id": 1, "role": 1, "sold": 0, "created_at": "2022-04-21T18:48:59.694Z", "updated_at": "2022-04-21T18:48:59.694Z", "v": 0}, {"id": "6261739940865c739rd47d12", "name": "niraj", "email": "niraj@gmail.com", "hashed_password": "d4f10170e50ed3261636a852452f67dc4c063c9c", "location": "Karnataka", "salt": "fbfb4a1c0-c184-11ec-8033-bb7f6792af3f", "role": 1, "v": 0}, {"id": "6261739940865c739rd47d12", "name": "niraj", "email": "niraj@gmail.com", "hashed_password": "d4f10170e50ed3261636a852452f67dc4c063c9c", "location": "Karnataka", "salt": "fbfb4a1c0-c184-11ec-8033-bb7f6792af3f", "role": 1, "v": 0} ]
```

## Screenshots of application

Kisaan Portal      Home Signin Signup

**Welcome to Kisaan Portal**  
A one stop solution to find the best veggies straight from the farmers.

जय जवान ! जय किसान !



Directly from the Farm

Kisaan Portal      Home Signin Signup

**Sign In**

Email address

Password

**Submit**

Kisan Portal

Home Signin Signup

### Sign Up

Name

Email

Password

Select User Type

Select State location

**Submit**

Kisan Portal

Home Dashboard Signout

## Welcome to Kisan Portal niraj !



**Monday**  
 Temp Range -> 25.29° - 25.3°  
 Wind Speed -> 1.44m/s  
 Humidity -> 72%  
 Weather -> overcast clouds



**Tuesday**  
 Temp Range -> 23.69° - 23.69°  
 Wind Speed -> 1.16m/s  
 Humidity -> 79%  
 Weather -> overcast clouds



**Wednesday**  
 Temp Range -> 23.11° - 23.11°  
 Wind Speed -> 4.19m/s  
 Humidity -> 77%  
 Weather -> overcast clouds



**Thursday**  
 Temp Range -> 22.8° - 22.8°  
 Wind Speed -> 2.12m/s  
 Humidity -> 78%  
 Weather -> overcast clouds



**Friday**  
 Temp Range -> 21.86° - 21.86°  
 Wind Speed -> 3.01m/s  
 Humidity -> 82%  
 Weather -> overcast clouds

Actions
View Vegetable
Add Vegetable

**Price in Rupees per Kilogram**  
Updated On : Mon May 09 2022

Fetch Data
Get Weather

Search district mandi from your state
  Search specific vegetable or fruit ( commodity )

Personal Details
niraj
niraj@gmail.com
Karnataka

Kisan Portal

Home Dashboard Signout

Actions		Price in Rupees per Kilogram				Personal Details	
	View Vegetable	Updated On : Mon May 09 2022				niraj	
	Add Vegetable	<input type="button" value="Fetch Data"/> <input type="button" value="Get Weather"/>				niraj@gmail.com	
	Remove Vegetable					Karnataka	
	Update Profile					Farmer	
		State	District	Market	Commodity	Price Range	Variety
		Karnataka	Bagalkot	Mhalingapur	Gur(Jaggery)	25.75 - 37.5	Nizamabad
		Karnataka	Bangalore	Ramanagara	Beans	40 - 60	Beans (Whole)
		Karnataka	Bangalore	Ramanagara	Beetroot	10 - 15	Beetroot
		Karnataka	Bangalore	Ramanagara	Bhindi(Ladies Finger)	16 - 24	Bhindi
		Karnataka	Bangalore	Ramanagara	Bitter gourd	20 - 28	Bitter Gourd
		Karnataka	Bangalore	Ramanagara	Brinjal	16 - 35	Brinjal

Search district mandi from your state

onion

State	District	Market	Commodity	Price Range	Variety
Madhya Pradesh	Alirajpur	Alirajpur(F&V)	Onion	10 - 16	Onion
Madhya Pradesh	Dewas	Dewas(F&V)	Onion	2 - 7	Other
Madhya Pradesh	Dhar	Kukshi	Onion	4 - 9	Onion
Madhya Pradesh	Dhar	Manawar	Onion	7 - 9	Other
Madhya Pradesh	Harda	Harda(F&V)	Onion	6.5 - 8	Medium
Madhya Pradesh	Hoshangabad	Itarsi	Onion	7 - 9	Other
Madhya Pradesh	Indore	Indore(F&V)	Onion	1 - 11	Onion
Madhya Pradesh	Khargone	Khargone	Onion	5 - 10	1st Sort
Madhya Pradesh	Morena	Porsa(F&V)	Onion	10 - 10	Onion

Search by vegetable

State	District	Market	Commodity	Price Range	Variety
Madhya Pradesh	Indore	Indore(F&V)	Banana	6 - 14	Other
Madhya Pradesh	Indore	Indore(F&V)	Bhindi(Ladies Finger)	5 - 20	Bhindi
Madhya Pradesh	Indore	Indore(F&V)	Bitter gourd	6 - 15	Bitter Gourd
Madhya Pradesh	Indore	Indore(F&V)	Bottle gourd	10 - 20	Bottle Gourd
Madhya Pradesh	Indore	Indore(F&V)	Brinjal	4 - 8	Arkasheela Mattigulla
Madhya Pradesh	Indore	Indore(F&V)	Cabbage	2 - 6	Cabbage

Search by district

Kisaan Portal
Home
Dashboard
Signout

## Your Vegetables

**Click here**

[View Items](#)

**Filter (Price Range)**

- Any
- Rs. 0 to Rs. 9
- Rs. 10 to Rs. 19
- Rs. 20 to Rs. 29
- Rs. 30 to Rs. 39
- More than Rs. 40

**Carrot**



Rs. 20 per kg

Added on a day ago

0 kgs sold

[In Stock](#)

---

250 kgs Left

**Brinjal**



Rs. 45 per kg

Added on 17 days ago

0 kgs sold

[In Stock](#)

---

120 kgs Left

Kisaan Portal

Home Dashboard Signout

## Add a new vegetable

Hello niraj, ready to add a new vegetable?

Item name  
Cucumber

Price  
60

Quantity  
100

Upload Photo  
Choose file far3.jpg

Confirm  
Add Vegetable

Kisaan Portal

Home Dashboard Signout

Click here  
[View Items](#)

Filter by price range

- Any
- Rs. 0 to Rs. 9
- Rs. 10 to Rs. 19
- Rs. 20 to Rs. 29
- Rs. 30 to Rs. 39
- More than Rs. 40

**Vegetables**

**Carrot**



Rs. 20 per kg

Added on a day ago

0 kgs sold

In Stock

---

250 kgs Left

[Remove Vegetable](#)

**Brinjal**



Rs. 45 per kg

Added on 17 days ago

0 kgs sold

In Stock

---

120 kgs Left

[Remove Vegetable](#)

Kisaan Portal Home Dashboard Signout

## ***Profile Update***

**Update Details**

Name

Email

Password

Select State location

**Submit**

Kisaan Portal Home BuyNow Dashboard Cart<sup>0</sup> Signout

## ***Welcome to Kisan Portal gaurav !***

**User Links**

- [My Cart](#)
- [Buy Now](#)
- [Update Profile](#)

Price in Rupees per Kilogram  
Updated On : Mon May 09 2022

**Fetch Data**

**User Information**

- gaurav
- gaurav@gmail.com
- Maharashtra
- Customer

State	District	Market	Commodity	Price Range	Variety
Maharashtra	Ahmednagar	Parner	Onion	1 - 11.25	Other
Maharashtra	Ahmednagar	Rahata	Bhindi(Ladies Finger)	10 - 25	Other
Maharashtra	Ahmednagar	Rahata	Bitter gourd	15 - 25	Other
Maharashtra	Ahmednagar	Rahata	Bottle gourd	5 - 11	Other

Kisaan Portal

Home BuyNow Dashboard Cart 0 Signout

## Shopping Cart

Manage your cart items. Add remove checkout or continue shopping.

Your cart has 1 items

**Carrot**



Rs. 20 per kg

Added on a day ago

In Stock

[Remove Vegetable](#)

Quantity: 1

Your cart summary

**Total: Rs. 20**

[Checkout](#)

Kisaan Portal

Home BuyNow Dashboard Cart 0 Signout

## Buy now

Fresh Veggies and Fruits at your Door-Step

Filter by farmers

Filter by price range

- Any
- Rs. 0 to Rs. 9
- Rs. 10 to Rs. 19
- Rs. 20 to Rs. 29
- Rs. 30 to Rs. 39
- More than Rs. 40

**Vegetables**

**Carrot**



Rs. 20 per kg

Added on a day ago

In Stock

[Add to cart](#)

**Brinjal**



Rs. 45 per kg

Added on 17 days ago

In Stock

[Add to cart](#)

## **8. Conclusion**

The webapp for buying and selling vegetables directly from farmers to consumers is built using MERN stack and integrated with Jenkins,Docker,Ansible,GitHub,etc. to perform CI/CD development pipeline.

The complete Jenkins pipeline is automated through Jenkins.

## **9. Future Work -**

- We can add payment portal at the time of checkout
- Also can predict yield predictions in locations based on soil data and weather conditions in the locality

---

**THANK YOU**