

Smart Web Application for Efficient Management of Agricultural Products

Digvijay Babar, Zaid Patwekar, Manojkumar Patil, Ravikumar Pawar, Akanksha Jadhav

Peth, CSE, NMCOE, Peth, India

Abstract: Web Application for Agricultural Product helps farmers to sell their product at the best possible price. It has features like online payment integration, searching and sorting items for finding products easily. This will makes website very useful for the farmers and the local vendors. Based on the ratings which are included with the product, buyers can differentiate the items.

Keywords: APMC - Agricultural product market committee, smart web application, confidentiality, integrity.

I. INTRODUCTION

A web based platform to provide a single reliable platform for all kind of users to buy and sell their products. Existing systems promises to provide similar kind of services but lack of user friendly GUI and lack of services users need a reliable platform which will support this kind of services.

II. EXISTING SYSTEM

Instacart is a one of the biggest online grocery delivery service that partners with grocery store chains to provide delivery from a variety of stores available in nearby cities. As such, they offer a wide variety of products to suit all dietary needs. Drawbacks: 1. Consumers need to take the membership in order to use the platform and some hidden charges can cost lot of money to farmers. 2. This system focuses only on delivery of product from grocery stores to the consumers.

III. PROPOSED SYSTEM

The proposed system will create a platform where farmers and consumers have better deals. The farmer can sell their product with good cost and without any broker. Farmers can trade their product on a platform at the price above the MSP (Minimum Selling Price) provided by the Government of India. The main advantageous of a platform is farmers could not face any delivery charge. Project aims to provide a desirable price for farmers with multiple selling options. Users from tier 1 and tier 2 cities will able to buy good quality of groceries with minimum price. This farmer-centric application gives options like:

1. Sell product to APMC.
2. Sell product to consumers

IV. PROBLEM STATEMENT

To design and develop a secure web based application for trading agricultural products for farmers, customers and traders.

Following are the specific objectives set for our work.

1. To design an efficient database and secure database for efficient storage and retrieval of data.
2. To design and implement a friendly UI for easy access and browsing for farmer and consumers.
3. To facilitate the users with searching and sorting of the products with respect to name, quantity and price.
4. To generate various weekly, monthly and annual reports for all the users.
5. To deploy secure web application on reliable and efficient web server.

V.SYSTEM DESIGN

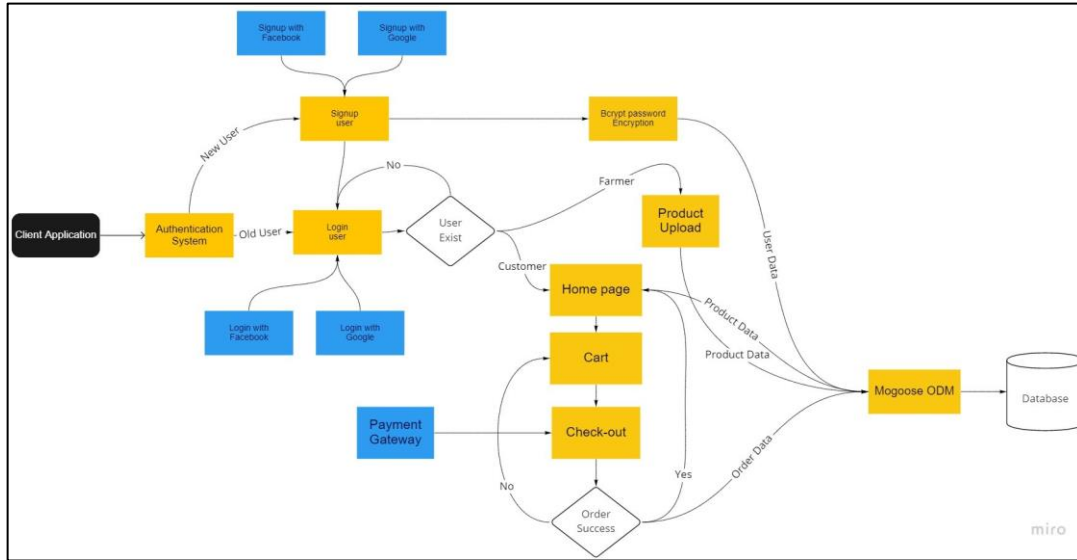


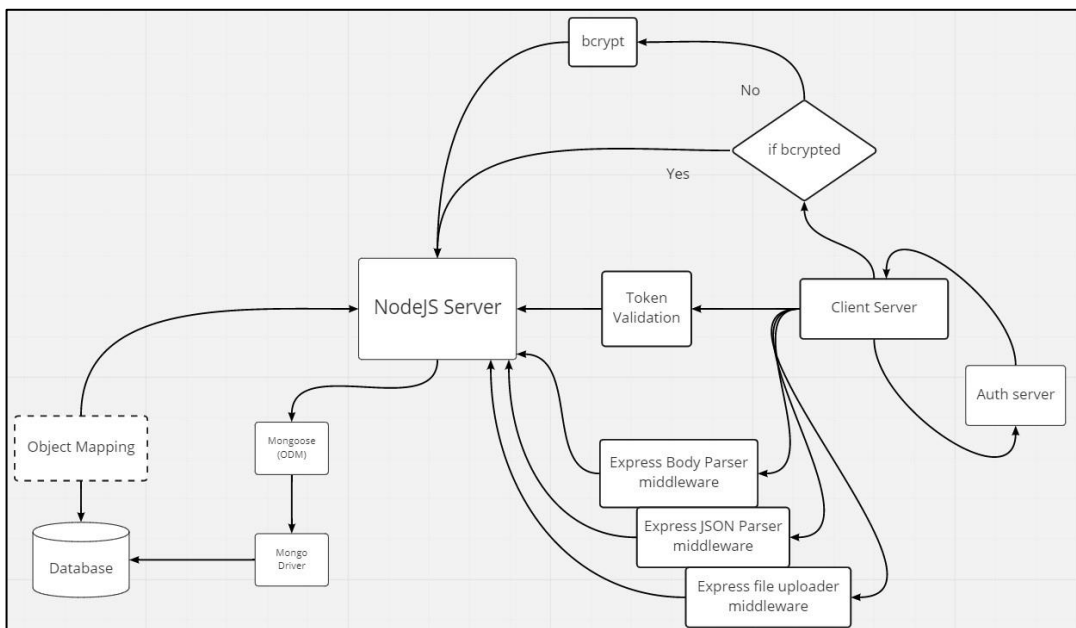
Fig. User Interface flow diagram

Client connects with the authentication system and makes an authentication request. If the user does not exist he can signup and if he already has an account then he can login. There are three ways to login with google, facebook and manual login.

After successful login he gets access to the homepage where all products are listed to sell. Customer can add products to card and make payment through razorpay payment gateway at the time of checkout. If user is farmer then he can get the access to the dashboard for uploading product and reports section to access the various stocks and selling reports.

V. METHODOLOGY

The Following Figure is the project flow diagram of our project.



• Project Flow Diagram

Figure : Project Flow Diagram

The flow diagram for the system is as shown in Figure 4.5, where we have a NodeJS run-time at the center. All the requests and responses are handled by the by back-end server which are made by the front-end client.

- **Efficient and secured database**

This section explains the different methodologies used in the application for designing the secure and efficient database for the easy uploading and retrieval of data.

- **Database design**

The database design of the proposed work is depicted in Figure 4.6.

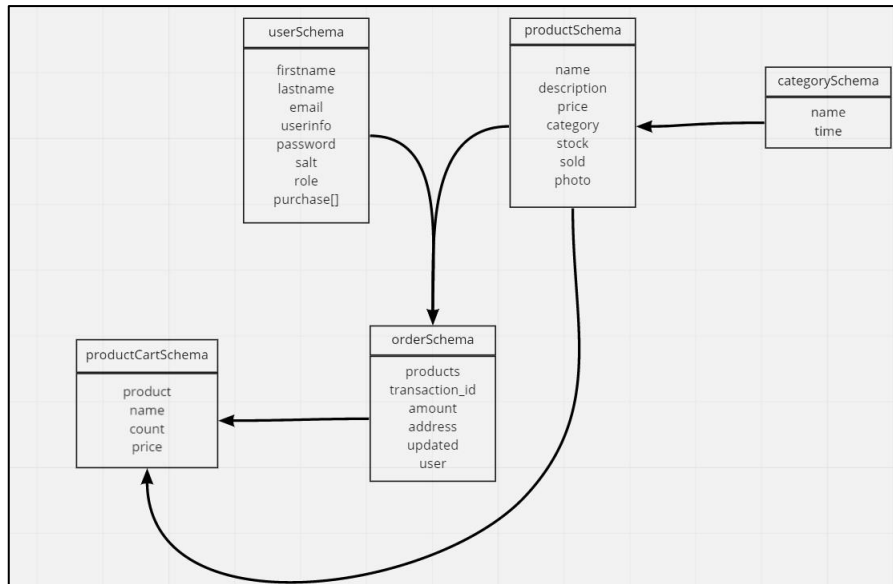


Figure : Database Schema

The userSchema which are for the users information. productSchema for products information. categorySchema will help us to categorise the products into multiple categories. productCartSchema is for keeping the products selected to order. order- Schema is for maintaining the record of orders made by customers.

- **Object Document Modeler (ODM)**

Database ODM and schema diagram as shown in below Figure 4.7.

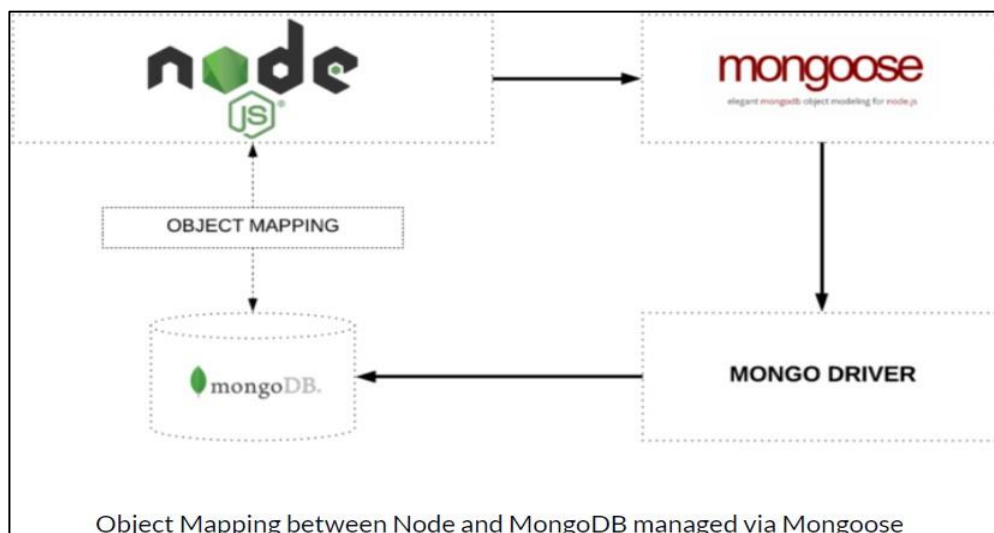


Figure :Mongoose ODM flow diagram

MongoDB is installed on AWS elastic cloud computers where the server is hosted so that we can reduce the time required for database applications. The Mongoose ODM i.e. Object Documentation Modelling is used with the mongoDB to make it easy to perform CRUD operations with JavaScript only. Mongoose is known for its elegance and ease of use; it will avoid the complexity of Mongo drivers.

- Security and hosting of the application
- Communication of micro-services

System overview diagram as shown in Figure

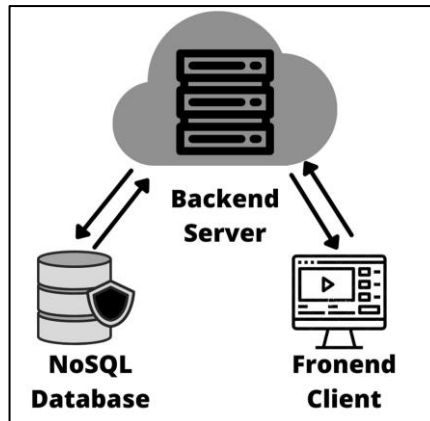


Figure : Communication overview

NodeJS is a runtime environment for JavaScript which is the basis for the server. Server uses ExpressJS library to handle the web requests to perform the CRUD. The web requests are as follows.

- GET - To get the data from the server
- POST - To send the data to the server
- PUT - To update specific data in the database
- DELETE - To delete specific data in the database

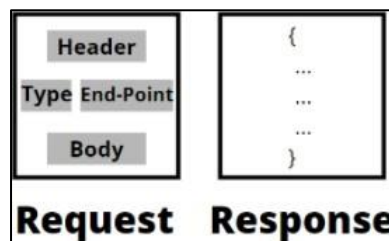


Figure : API request and response

For specific requests we get a certain response back. Each request has its specific components which helps for various things such as to perform authentication, send the endpoint, specify the request type and send some data in body. The type of request made to the particular endpoint will be defined to return the appropriate response back from the server to the frontend client.

JSON(JavaScript Object Notation) is used to carry the data throughout the application which is one the standard and modern way to send and receive the data. Express Json parser middleware is used in between to parse the JSON objects.

- Password Encryption

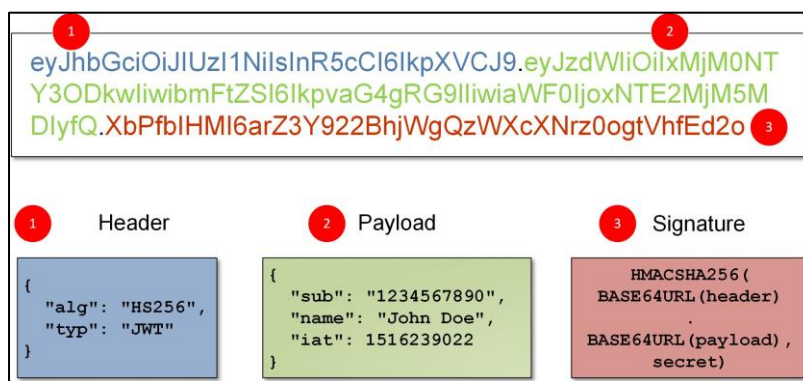
Example of encrypted password is as shown in Figure



Figure : Password Encryption

Privacy and the security of users and their information is the core of any web based application. BcryptJS is the library used to encrypt the user password with the help of blowfish cipher. It is one of the safest modern day algorithm to encrypt the passwords.

- **JSON web tokens**



• Figure : JSON web token

JWTs can be used at the time of authorization as well as at the time of data transfer.

JWT in its compact form, JSON Web Tokens consist of three parts separated by dots (.), which are:

- Header - contains type of algorithm and other information
- Payload - data to be carried in encrypted format
- Signature - to verify the data at other end Therefore, a JWT typically looks like the Figure 4.12.

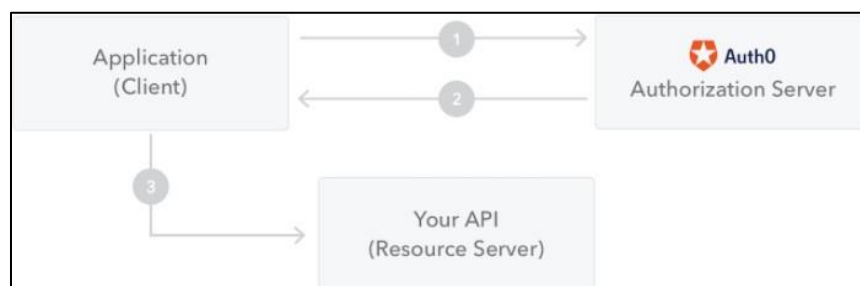


Figure : JWT flow diagram

When a user is authenticated, a Token based approach is used to keep him logged into the application. JSON Web Token (JWT) is an open standard (RFC 7519) that defines a compact and self-contained way for securely transmitting information between parties as a JSON object. The information comes from JWT can be verified and trusted because it is digitally signed. JWTs can be signed using a secret (with the HMAC algorithm) or a public/private key pair using RSA or ECDSA. Working of this tokens is shown in the figure 4.13 as a flow diagram.

VI.IMPLEMENTATION

- **Database Implementation**

As MongoDB is secure and reliable platform it helps our web application in better performance. For our web application we created few collections like categories, products, users, and orders. Each collection has several

documents each document referred as individual entry of data.

To communicate with database we need to include the mongoose object data modeling library it provides methods to directly connect the back-end with the MongoDB drivers. *const mongoose = require(mongoose)*

To create the collection we need to use Schema method from mongoose which creates the new collection schema. Inside the schema function we can add various fields and the data-types of that fields given in the database design.

```
1  const mongoose = require("mongoose");
2
3  const categorySchema = new mongoose.Schema(
4    {
5      name: {
6        type: String,
7        trim: true,
8        required: true,
9        maxlength: 32,
10       unique: true
11     }
12   },
13   { timestamps: true }
14 );
15
16 module.exports = mongoose.model("Category", categorySchema);
```

Figure : Mongoose category model example

Here in the figure we have the name field which describes how user should enter the data. This example will help us to create the collection of category in our database. We used the same methods for the other fields in the database.

- **User Interface implementation**

For implementing the user interface we used the ReactJS it is an open-source JavaScript library that is used for building user interfaces it also allows us to create reusable UI components. There are multiple reusable components used in the entire application like buttons, drop-downs, forms, data-fields, etc.

- **Bootstrap and Material UI**

By using react bootstrap we can achieve a interface where the user can easily understand and navigate through the application in an efficient way. Material UI provides quick access to common features and components.

- **ReactDOM**

ReactDOM is a package that provides DOM specific methods by using we manage the DOM elements of the web page. By using render method we can render a single React Component or several Components wrapped together in a Component or a div element.

Method for rendering the component :

ReactDOM.render(element, container, callback)

By using findDOMNode() method user can get the DOM node where a particular React component was rendered.

Method to find DOM node :

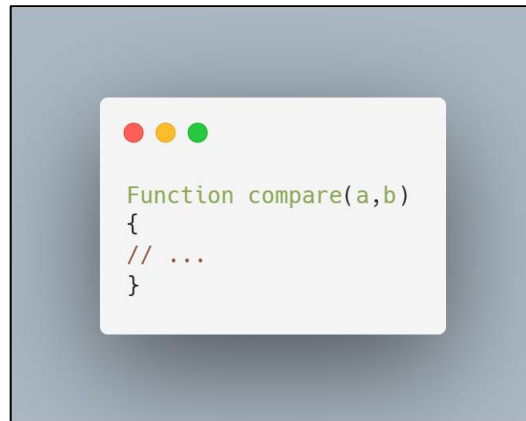
ReactDOM.findDOMNode(component)

- **Internal algorithms implementation**

- **Sorting algorithm implementation**

sort() method required two different separate arguments and returns a value that decides the sort order

Following is the syntax of the compare function



```
Function compare(a,b)
{
  // ...
}
```

Figure : Compare function in JavaScript sort() method

The value of function compare(a,b) is less than 0 then, sort a to an index lower than b. If the function returns 0 leave a and b as it is and proceed to the next step If compareFunction(a,b) is greater than 0, sort the a and b such that b comes first

- **Data visualization implementation**

Javascript library chart.js is open source and freely available on GitHub which helps us to create different types of charts with the help of HTML5.

It supports graphs like -

1. Line
2. Bar
3. Doughnut
4. Pie

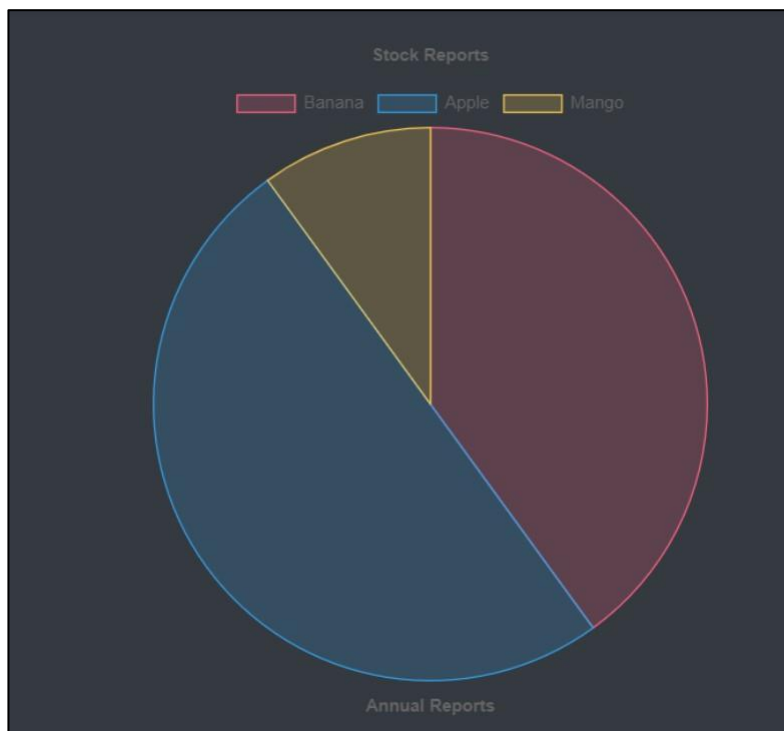


Figure : Chart.js implementation for report

By using chart.js we can implement different types of graphs Here we have used a pie graph. The most useful property of a pie graph is visualize data in circular pieces with various colors.

A pie graph for the stock report can be implemented like figure with various colors representing different products.

```
data = {  
  Datasets: [{data: [10,20,30]}],  
  Labels: [  
    'Red',  
    'Yellow',  
    'Blue'  
  ]  
};
```

Figure : Pie chart data example

In the figure we provided the data-set for the pie chart where it contains the data of stock and labels representing various colors. Similarly we can send data for the bar graph or any other type of graph.

- **Security and deployment implementation**
 - **JWT implemenation in authentication**

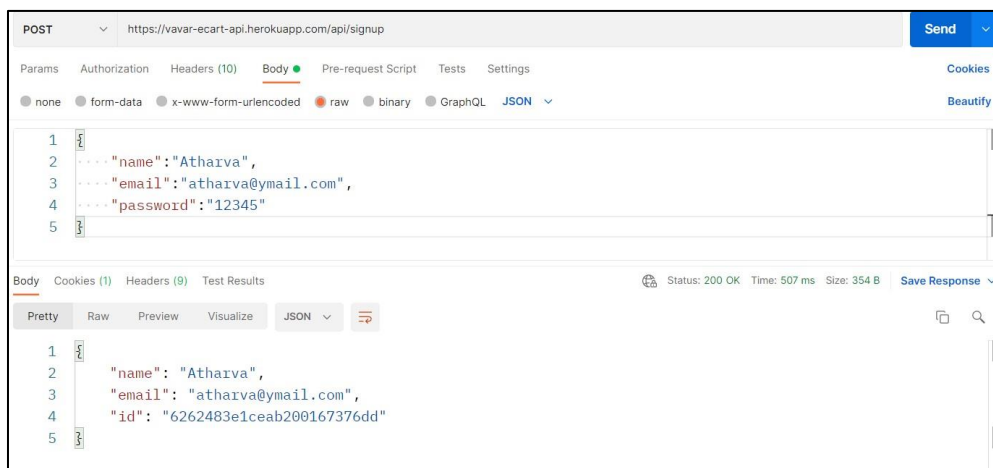


Figure: Pie chart data

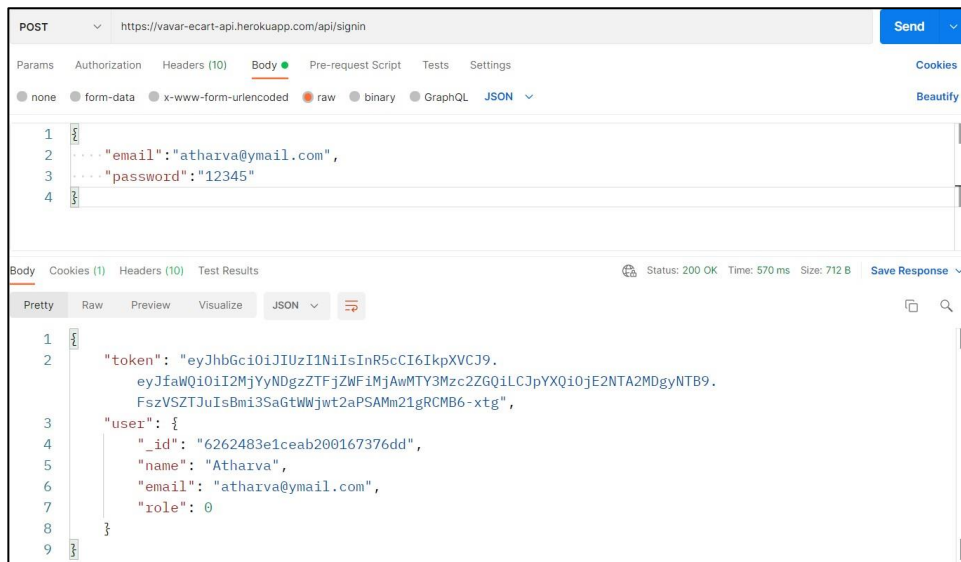


Figure : Pie chart data

▪ Heroku Deployment

Heroku is a container-based platform as a service it helps developers to host the entire application on a cloud platform, because it is a container-based platform developer can test each module of an application differently or they can test the entire application without using lot of resources.

The application can be deployed on the Heroku server by using Heroku Command Line Interface(CLI). Following is the process of deploying the application by using git.

Step 1 :

Change the directory to the Final project using *cd filename*

Step 2 :

Add the file to the staging area for the next commit. Step 3 :

Commit the changes by using *git commit -m "Initial file upload"*

Step 4 :

To create new empty-applicaiton use *heroku create -a vavar*

Step 5 :

You can use the git remote command to confirm that a remote named Heroku had been set for your app.

Step 6 :

To deploy the app to heroku use *git push* and specify the branch with *git push heroku main* Now it will automatically get build and deployed after every commit on the main branch of the git.

▪ MongoDB Atlas Deployment

As database is the one of the most important part in the web application it needs to be secure and faster to maintain the reliability of the web application to achieve this MongoDB database is hosted on the MongoDB atlas server which is cluster of distributed instances of multiple data points to ensure that their should not be any single point of failure.

Step 1:

Creating an account and adding the name for the database.

Step 2:

To maintain the security of an application we need to give the access of database to single server that is our backend server which is hosted and live we can do it by adding ip address of our back-end application to the database.

Step 3:

Now one more thing is to connect the database with specific URL and password. For the connection we can use the secured URL given by the MongoDB atlas.

VII.CONCLUSION

The final conclusion obtained from the proposed system and also describes the future work that can be done in the proposed system.

- We have successfully designed an efficient and secure database for efficient storage and retrieval of data. For storing information of farmers, customers and products we use MongoDB database MongoDB stores data in

BSON format and also handle high volume and can scale both vertically or horizontally to accommodate large data loads.

- We have successfully designed friendly UI for sellers and customers so that seller can easily add their product in efficient manner and customer can purchase products easily.
- We have successfully provided the searching and sorting of the products with respect to name, quantity and price so that customer can easily search and sort the products.
- We have successfully provided the facility to generate reports so that seller can see their available stocks and sells reports.
- We have successfully deployed the secure web application on Heroku, MongoDB atlas and Varcel which are reliable and efficient web servers.

VIII. REFERENCES

- [1]. Kroger: Bernard Kroger, kroger.com, The kroger Co, Jan 2012, Accessed on - jan 2022, Available: <https://www.kroger.com/stores/search>
- [2]. BigBasket: V. S. Sudhakar, bigbasket.com, The bigbasket Co, may 2011, Accessed on - jan 2022 Available : <https://en.wikipedia.org/wiki/Bigbasket>
- [3]. All India crop situation, progress of area coverage under rabi crops, Agricoop nic, Dec 2021, Accessed on - Jan 2022, Available: <https://agricoop.nic.in/sites/default/files/Crop>
- [4]. Instacart: Apoorva Mehta, instacart corporate blog, Google LLC, jan 2012 Accessed on- jan 2022, Available: <https://www.instacart.com/company/blog>
- [5]. shipt: Bill smith, shipt shopping, Shipt LLC, March 2019, Accessed on - Jan 2022 Avail- able: <https://www.shipt.com/>
- [6]. Harjinder Kaur and Rakesh K. Shukla, Consumer's Attitude for Acceptance of Online Grocery Shopping in India, Social Sciences and Humanities, Accessed on- May, 2017 Avail- able: <https://www.journalcra.com/sites/default/files/issue-pdf/17371.pdf>
- [7]. Robert Kulick, Ph.D., February 2020 Available: <https://www.nera.com/content/dam/nera/publications/20>
- [8]. Target: George Dayton, Target groceries, Target Brands, Inc, june 1999, Accessed on - Jan 2022, Available: <https://www.target.com/c/grocery/-/N-5xt1a>