



ONLINE GROCERY STORE

CSE3002 – INTERNET & WEB PROGRAMMING

J COMPONENT PROJECT DOCUMENT

FALL 2019 - 2020

INDIVIDUAL PROJECT
1. 17BCE0980 NAMAN L UPADHYAY

UNDER THE GUIDENCE OF
PROF. NAVEEN KUMAR N
SCOPE



School of Computer Science and Engineering

DECLARATION

I hereby declare that the J Component project entitled "**Online Grocery Store**" submitted by me to the School of Computer Science and Engineering, VIT University, Vellore-14 in partial fulfillment of the requirements for the **Internet and Web Programming (CSE3002)** course, is a record of bonafide work carried out by me/us under the supervision of **Naveen Kumar N, Assistant Professor (Sr)**. I further declare that the work reported in this project has not been submitted and will not be submitted, either in part or in full, for the award of any other degree or diploma of this institute or of any other institute or university.

Reg.No: 17BCE0980

Name: Naman L Upadhyay

Reg.No: 17BCE0980

Name: Naman L Upadhyay

Chapter Title	Page
Title Page	i
Declaration	ii
Table of Contents	iii
List of Figures	iv
Abstract	vii
1. Introduction	1
1.1. Aim	1
1.2. Objective	1
2. System Requirements	3
3. Solution for Existing problems	3
3.1. Solution For Existing Problem	3
3.1.1. Problem Description And Its Solution	3
3.1.2. Sample Code	3
3.1.3. Sample Screenshots	4
3.2. Solution For Existing Problem	5
3.2.1. Problem Description And Its Solution	5
3.2.2. Sample Code	6
3.2.3. Sample Screenshots	7
4. Recent Technologies	10
4.1. Technology GOOGLE API	10
4.1.1. Importance of Technology	10
4.1.2. Sample Code	10
4.1.3. Sample Screenshots	11
4.2. Technology Node JS	11
4.2.1. Importance of Technology	12
4.2.2. Sample Code	12
4.2.3. Sample Screenshots	16
4.3. Technology MongoDB	18
4.3.1. Importance of Technology	18

4.3.2. Sample Code	19
4.3.3. Sample Screenshots	20
5. Screenshots	21
5.1. Customer Module	21
5.2. Cart	26
5.3. Vendor Module	28
6. Conclusions	30

	Page
1. Input Image 1	4
2. Input Image 2	5
3. Input Image 3	7
4. Input Image 4	7
5. Input Image 5	8
6. Input Image 6	8
7. Input Image 7	9
8. Input Image 8	11
9. Input Image 9	11
10. Input Image 10	17
11. Input Image 11	17
12. Input Image 12	18
13. Show dbs	20
14. Use iwp	21
15. Show collections	21
16. Register As New User(Customer)	21
17. Existing User	22
18. Splash Screen after login	23
19. Figure 19	23
20. Offers and products visible to user	24
21. Figure 21	24
22. CATEGORIES OF PRODUCTS	25
23. CONTACT INFORMATION	25
24. ORDER PLACING PAGE	26
25. Adding items to Cart	26
26. Items in Cart	27
27. Delivery Options	27
28. ORDER PLACED DESCRIPTION	28
29. Vendor orders	28
30. Vendor Deliveries	29
31. Customers Ordering	29
32. Delivery Boys	29

ABSTRACT

Systems and methods provided for the management of products sold/bought. This system includes the usage of both back-ended and front-ended programming languages with the knowledge of system architecture. To be a leading market company, one needs a good Database Management system for its management.

For this we need to have a role-based access control system which gives different rights to different users. There are various entities such as supplier, product, price list, quantity, vendors etc. and attributes such as id of every entity, email, phone number also their types such as primary key, alternate key etc. which are needed to be maintained.

The purpose of this website is to provide with an online market for vendors to sell their groceries online. Also the system aims to provide users with variety of groceries easily available online. The website is well managed with independent modules which are tightly coupled together to provide a good solution for vendors to keep track of their stock and also an easy stop and go place for the user.

1. INTRODUCTION

Nowadays we mostly see, or as we have been seeing through our childhood, that many or most of the distributors are using pen format to keep records but the young ones or perhaps the shopkeepers/distributors of this era wants to use computer technology, which is indeed a trend now, to keep a track of all its transactions and day to day operation to achieve its business goal.

We also see that there is no place present for buying day to day groceries online. The market still relies on small vendors for daily groceries. Our aim is to provide users access to groceries online and order them online. We will help them link with small vendors who sell these items. The modern technology is bringing everything closer and easily accessible through online services.

Our project includes the database with user friendly interface with user authentication derived from the previously proposed data model as designed.

Currently online shopping portals like Flipkart and Amazon don't sell these types of items which are required daily, they sell other kinds of products. Our projects main focus is to sell all these day to day items online and make it accessible to the user more easily. Our goal is to link small scale vendors and help them reach out to a large people who need day to day groceries easily.

1.1. AIM

Objective of proposed project, is to provide Online Grocery Shopping solution to consumers and vendors. It will automate some of the basic operations of an online store. Scope would be to provide basic functionalities using a web application so that those manual process can be automated. It will include to provide administration access to vendors and admins and user specific access to customers.

1.2. OBJECTIVE

The objective of the project is to make an application in web platform to purchase items in an existing shop. In order to build such an application complete web support need to be provided. A complete and efficient web application which can provide the online

shopping experience is the basic objective of the project. The web application can be implemented in the form of a web application with web view.

2. SYSTEM REQUIREMENTS

The system required for the project is not a high-tech system. The basic requirements of the projects are a

- Web browser
- Php server
- Text editor
- Mongodb server
- Node application.

3. SOLUTION FOR EXISTING PROBLEM

3.1. SOLUTION 1: DIFFERENT LANGUAGES

3.1.1. PROBLEM DESCRIPTION AND ITS SOLUTION

In today's world if we want to expand to every location of the world, we need to provide the facility in their own native or local language. If we have one common language, then we may tend to lose our customers. Hence to solve this problem I have come up with the solution of translating the page into the customer chosen language using Google API. Using this Google API, we can translate any webpage into the customer selected one. This API provides the luxury of 157 world recognized language and we can select any one of them. For e.g. If a person from north India does not understand English he can translate the entire webpage in Hindi and benefit himself and will also help us in increasing our customer base.

3.1.2. SAMPLE CODE

The below sample code displays the option of changing the language from English to any language from the dropdown list.

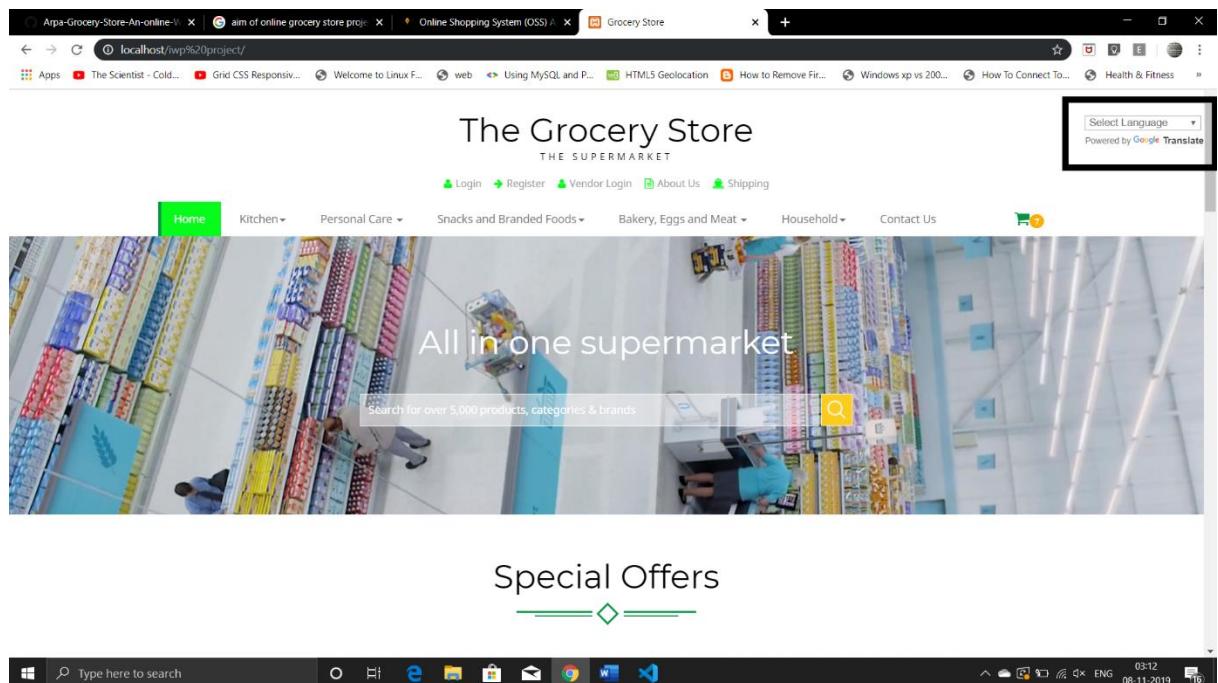
```
<!DOCTYPE html>
<html lang="en-US">
<head>
</head>
<body>
    <div style="float: right;" id="google_translate_element"></div>
    <script type="text/javascript">
```

```

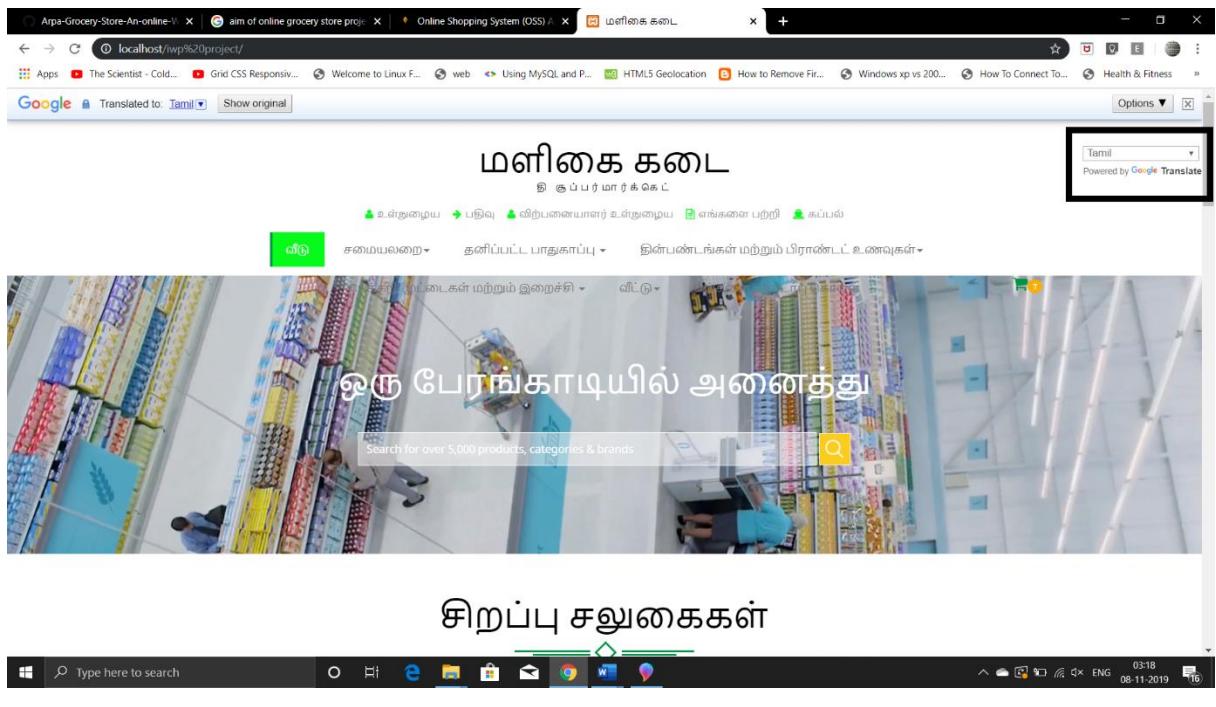
function googleTranslateElementInit() {
    new google.translate.TranslateElement(
        {pageLanguage: 'en'}, 'google_translate_element'
    );
}
</script>
<script type="text/javascript" src= "https://translate.google.com/translate_a/element.js?cb=googleTranslateElementInit">
</script>
</body>
</html>

```

3.1.3. SAMPLE SCREENSHOTS



Input Image 1



Input Image 2

3.2. SOLUTION 2: SENDING MAIL FROM DIFFERENT CHECKPOINTS DURING DELIVERY OF THE PRODUCT

3.2.1. PROBLEM DESCRIPTION AND ITS SOLUTION

In recent times we have seen many online grocery stores are coming in the market. But there are few issues with them and important among them is that they don't provide the tracking the order and they do not receive the conformation mail from the company of successful placement of the order. In order to receive an mail from the website it needs a proper hosting and the server. Without the hosting we cannot send the email to our customer. There are 2 ways the customer can pay for the order. One is online mode and the other one is offline mode. As soon as the order is received, we generate the order id and the confirmation mail is sent through a repository of the GitHub. Currently since we don't have a hosting, we cannot send the mail and there will be an error from the GitHub repository.

3.2.2. SAMPLE CODE

The below sample code displays the php code to send the mail from website to the registered customer once the order number is received.

```
<?php
```

```

/**
 * PHPMailer SPL autoloader.
 * PHP Version 5
 * @package PHPMailer
 * @link https://github.com/PHPMailer/PHPMailer/ The PHPMailer GitHub project
 * @author Marcus Bointon (Synchro/coolbru) <phpmailer@synchromedia.co.uk>
 * @author Jim Jagielski (jimjag) <jimjag@gmail.com>
 * @author Andy Prevost (codeworxtech) <codeworxtech@users.sourceforge.net>
 * @author Brent R. Matzelle (original founder)
 * @copyright 2012 - 2014 Marcus Bointon
 * @copyright 2010 - 2012 Jim Jagielski
 * @copyright 2004 - 2009 Andy Prevost
 * @license http://www.gnu.org/copyleft/lesser.html GNU Lesser General Public
License
 * @note This program is distributed in the hope that it will be useful -
WITHOUT
 * ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or
 * FITNESS FOR A PARTICULAR PURPOSE.
 */

/**
 * PHPMailer SPL autoloader.
 * @param string $classname The name of the class to load
 */
function PHPMailerAutoload($classname)
{
    //Can't use __DIR__ as it's only in PHP 5.3+
    $filename = dirname(__FILE__).DIRECTORY_SEPARATOR.'class.'.strtolower($classname).'.php';
    if (is_readable($filename)) {
        require $filename;
    }
}

if (version_compare(PHP_VERSION, '5.1.2', '>=')) {
    //SPL autoloading was introduced in PHP 5.1.2
    if (version_compare(PHP_VERSION, '5.3.0', '>=')) {
        spl_autoload_register('PHPMailerAutoload', true, true);
    } else {
        spl_autoload_register('PHPMailerAutoload');
    }
} else {
    /**
     * Fall back to traditional autoload for old PHP versions
     * @param string $classname The name of the class to load
     */
    function __autoload($classname)
    {
}
}

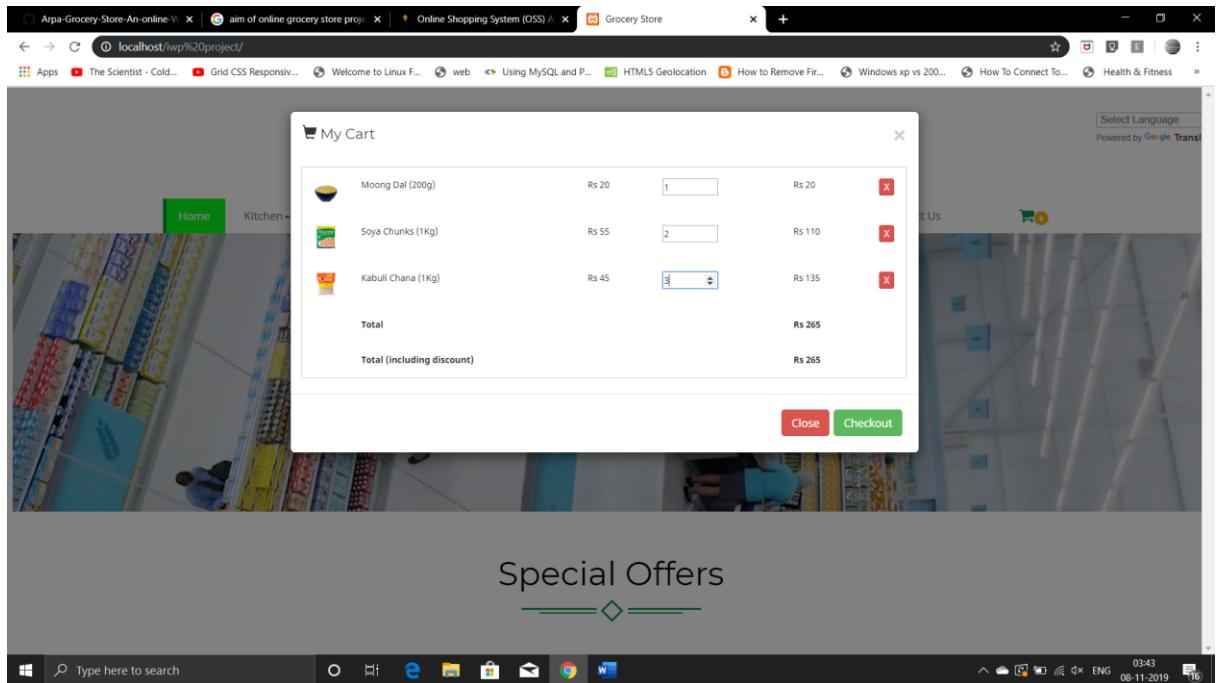
```

```

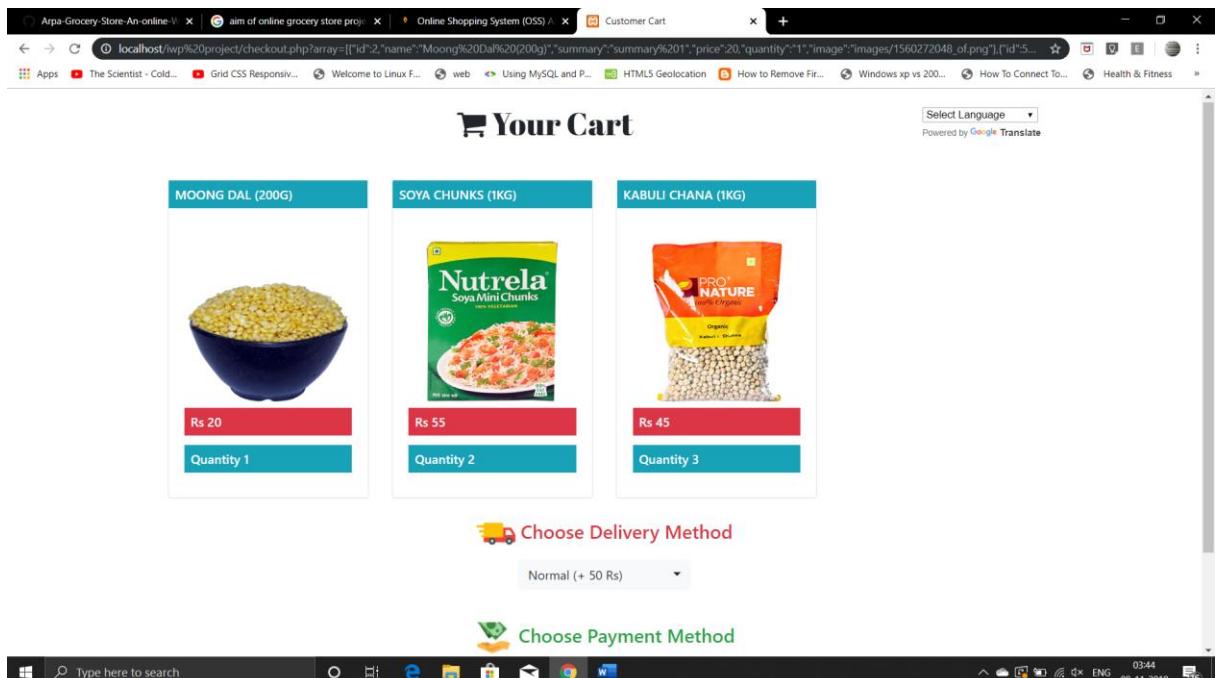
        PHPMailerAutoload($classname);
    }
}

```

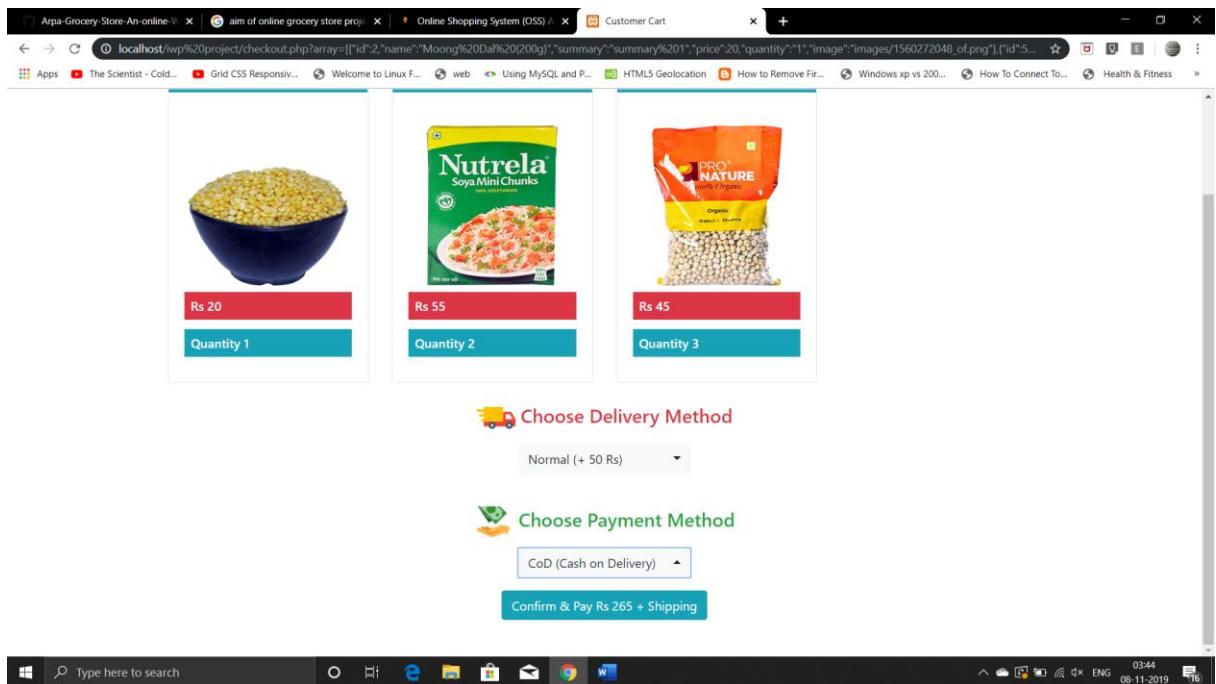
3.2.3. SAMPLE SCREENSHOTS



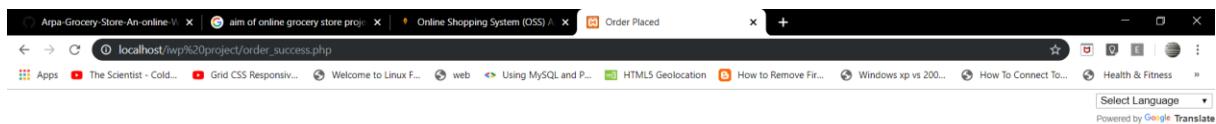
Input Image 3



Input Image 4



Input Image 5



Input Image 6

The screenshot shows a web browser window with multiple tabs open. The active tab is titled "Order Placed" and has the URL "localhost/wp%20project/order_success.php". The page content is as follows:

Order Placed

Purchased from	Grocery Store
Payment ID	COD251988632
Buyer Name	naman
Buyer Phone No.	9876543210
Buyer Email	naman@gmail.com
Buyer Address	D6, vit, 632014
Payment Status	Pending
Order Date	08/11/2019 03:46:59

[Track your Order](#) [Continue Shopping](#)

Message could not be sent.Mailer Error: SMTP connect() failed. <https://github.com/PHPMailer/PHPMailer/wiki/Troubleshooting>Message could not be sent.Mailer Error: SMTP connect() failed. <https://github.com/PHPMailer/PHPMailer/wiki/Troubleshooting>Message could not be sent.Mailer Error: SMTP connect() failed.
https://github.com/PHPMailer/PHPMailer/wiki/TroubleshootingMessage could not be sent.Mailer Error: SMTP connect() failed.
https://github.com/PHPMailer/PHPMailer/wiki/TroubleshootingMessage could not be sent.Mailer Error: SMTP connect() failed.
https://github.com/PHPMailer/PHPMailer/wiki/TroubleshootingMessage could not be sent.Mailer Error: SMTP connect() failed.
https://github.com/PHPMailer/PHPMailer/wiki/TroubleshootingMessage could not be sent.Mailer Error: SMTP connect() failed.
https://github.com/PHPMailer/PHPMailer/wiki/TroubleshootingMessage could not be sent.Mailer Error: SMTP connect() failed.
https://github.com/PHPMailer/PHPMailer/wiki/TroubleshootingMessage could not be sent.Mailer Error: SMTP connect() failed.

Input Image 7

4. RECENT TECHNOLOGY

4.1. TECHNOLOGY 1: GOOGLE API

4.1.1. IMPORTANCE OF TECHNOLOGY

Everybody who needs to translate a document or any other piece of text will come across a dilemma of whether to find a translator or use a machine to cope with this task. There are a lot of online translation tools you can use and it seems pretty reasonable to save your money and avoid turning to a professional translator. There are 103 of them now. The service translates more than 100 billion words a day and is one of the most popular tools for people worldwide to have a conversation with each other without any barriers. It's convenient. All you have to do is just paste your text to get its immediate translation. There's no need for communicating with people and discuss terms of cooperation. It's fast. It takes seconds to receive a ready translation of your source text. It's free. You don't have to pay anyone for the translation services. It recognizes almost any language. It's very difficult to find a human translator who knows all the languages in the world.

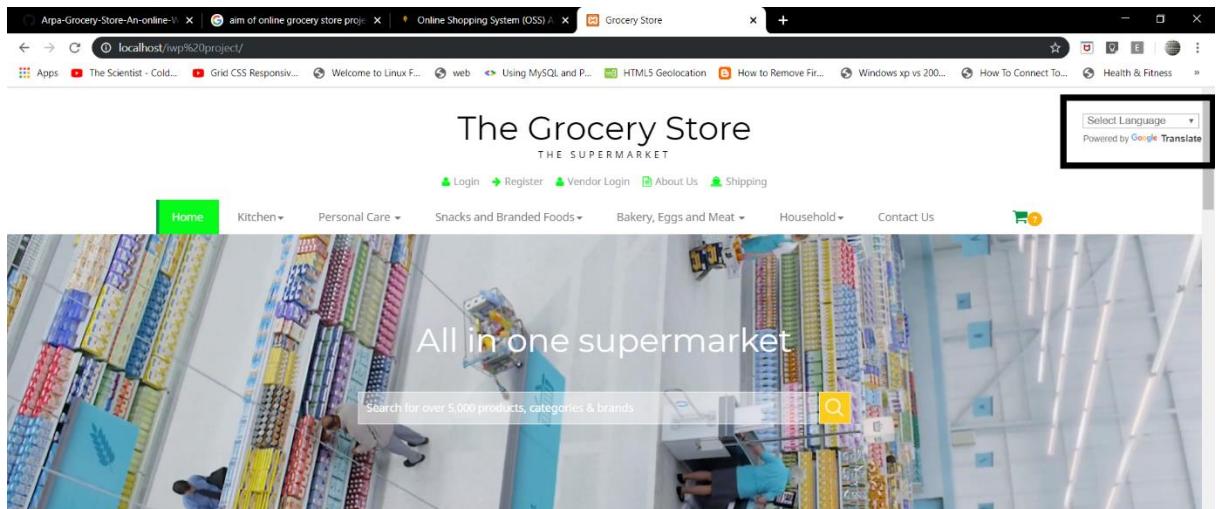
4.1.2. SAMPLE CODE

The below sample code displays the option of changing the language from English to any language from the dropdown list.

```
<!DOCTYPE html>
<html lang="en-US">
<head>
</head>
<body>
    <div style="float: right;" id="google_translate_element"></div>
    <script type="text/javascript">
        function googleTranslateElementInit() {
            new google.translate.TranslateElement(
                {pageLanguage: 'en'}, 'google_translate_element'
            );
        }
    </script>
    <script type="text/javascript" src= "https://translate.google.com/translate_a/element.js?cb=googleTranslateElementInit">
    </script>
</body>
```

</html>

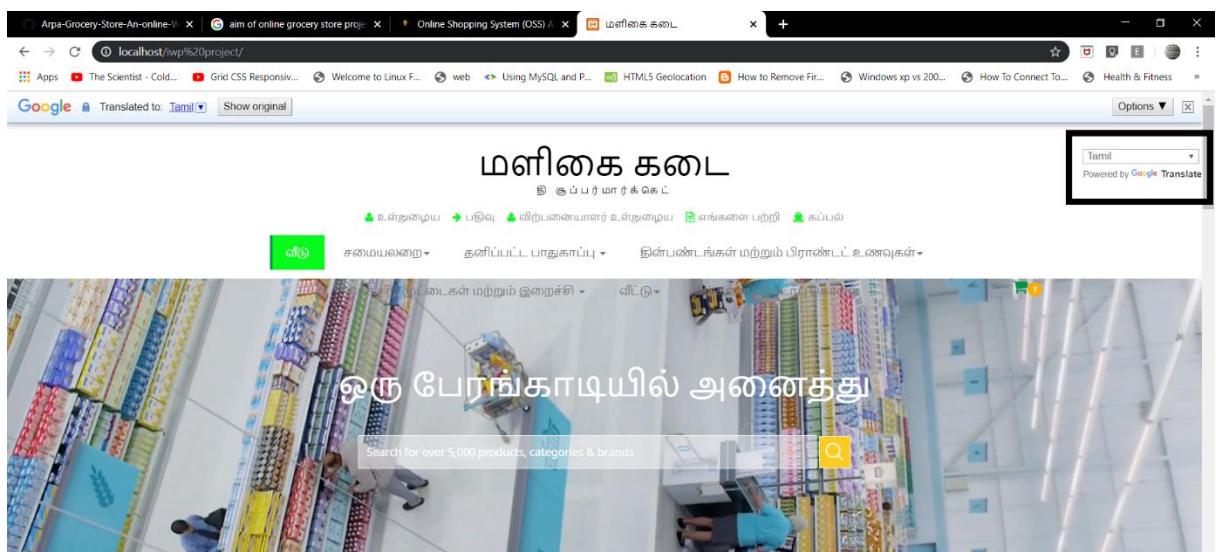
4.1.3. SAMPLE SCREENSHOTS



Special Offers



Input Image 8



சிறப்பு சலுகைகள்



Input Image 9

4.2. TECHNOLOGY 2: NODE JS

4.2.1. IMPORTANCE OF TECHNOLOGY

As a cross-platform runtime environment, Node.js enables web programmers to build a variety of server-side applications using JavaScript. Node.js executes JavaScript code using Google V8 engine. Unlike other JavaScript interpreters, V8 engine compiles the JavaScript code into native machine code. Thus, it enables the runtime environment to boost the performance of web server applications by executing JavaScript code in a faster and more efficient way. Node.js further performs all I/O operations asynchronously through a single-threaded event loop. The modern approach makes the Node.js application perform I/O operations by sending the asynchronous task to an event loop along with a call-back function. After sending the asynchronous task to the event loop, the application continues executing the remaining code. After completing the asynchronous operation, the event loop returns to the task, and executes the call-back function. In addition to consuming less memory, the feature enables Node.js to handle a large number of concurrent connections efficiently.

4.2.2. SAMPLE CODE

APP.JS

```
/* importing node module files */
const bodyParser = require('body-parser');
const cookieParser = require('cookie-parser');
const cookieSession = require('cookie-session');
const exphbs = require('express-handlebars');
const express = require('express');
const handlebars = require('handlebars');
const handlebarsIntl = require('handlebars-intl');
const handlebarsPaginate = require('handlebars-paginate');
const flash = require('connect-flash');
const passport = require('passport');
/* express server configuration */
const app = express();
/* session configuration */
app.use(require('express-session')({
    secret: 'keyboard cat',
    resave: false,
    saveUninitialized: true,
    cookie: { secure: false }
}));
app.use(flash());
app.use(cookieParser());
/* passport authenticator initialization */
app.use(passport.initialize());
```

```

app.use(passport.session());
/* body parser configuration */
app.use(bodyParser.json());
app.use(bodyParser.urlencoded({ extended: true }));
/* static pages configuration */
const static = express.static(__dirname + '/public');
app.use("/public", static);
/* view or handlebars configuration */
handlebarsIntl.registerWith(handlebars);      // handlebars formatting
handlebars.registerHelper('paginate', handlebarsPaginate); // paging
app.engine('handlebars', exphbs({ defaultLayout:'main' }));
app.set('view engine', 'handlebars');
/* routing configuration */
const configRoutes = require("./routes");
configRoutes(app);
/* running server on port 3000 */
app.listen(3000, () => {
  console.log("We've now got a server");
  console.log("Your routes will be running on http://localhost:3000");
});

```

INDEX.JS

```

const contactsData = require('./contacts');
const credentialsData = require('./credentials');
const productsData = require('./products');
const subscriptionsData = require('./subscriptions');
const transactionOrderData= require('./transactionOrder');
const transactionWalletData = require('./transactionWallet');
const usersData = require('./users');
const usersCardData = require('./usersCard');
const usersCartData = require('./usersCart');
const usersWalletData = require('./usersWallet');

let dataMethod = (app) => {
  app.use("/contacts-us", contactsData);
  app.use("/credentials", credentialsData);
  app.use("/products", productsData);
  app.use("/subscription", subscriptionsData);
  app.use("/user", usersData);
};

module.exports = {
  contacts: require('./contacts'),
  credentials: require('./credentials'),
  products: require('./products'),
  subscriptions: require('./subscriptions'),
  transactionOrder: require('./transactionOrder'),
}

```

```

transactionWallet: require('./transactionWallet'),
users: require('./users'),
usersCard: require('./usersCard'),
usersCart: require('./usersCart'),
usersWallet: require('./usersWallet')
};

```

PRODUCTS.JS

```

/* importing required files and packages */
const uuid = require('uuid');
const xss = require('xss');
const mongoDbCollection = require('../config/mongodb-collection');
const products = mongoDbCollection.products;

/* exporting controllers apis */
module.exports = productsControllers = {

    //----- fetch a product information by email id
    getProductId: (id) => {
        return products().then((productsCollection) => { // returning a found
            json document else returning null
            return productsCollection.findOne({ _id:id });
        })
        .catch(() => { // returning a reject promise
            return Promise.reject("Server issue with 'products' collection.");
        });
    },

    //----- fetch a product information by email id
    getProductByCategory: (category) => {
        return products().then((productsCollection) => { // returning a found
            json document else returning null
            return productsCollection.find({ category: { $regex : `.*${category}.*` }, $options : 'i' }).toArray();
        })
        .catch(() => { // returning a reject promise
            return Promise.reject("Server issue with 'products' collection.");
        });
    },

    //----- fetch a product information by search string
    getProductBySearch: (keyword) => {
        return products().then((productsCollection) => { // returning a found
            json document else returning null

```

```

let query = [
    { _id:keyword },
    { title: { $regex : `.*${keyword}.*`, $options : 'i' } },
    { category: { $regex : `.*${keyword}.*`, $options : 'i' } },
    { description: { $regex : `.*${keyword}.*`, $options : 'i' } }
];

return productsCollection.find({ $or: query }).toArray();
})
.catch(() => { // returning a reject promise
    return Promise.reject("Server issue with 'products' collection.");
});
},
//----- fetch a product information by search string
getProductBySearchFilter: (keyword, startRange, endRange) => {
    return products().then((productsCollection) => { // returning a found
json document else returning null

    let query = [
        { _id:keyword },
        { title: { $regex : `.*${keyword}.*`, $options : 'i' } },
        { category: { $regex : `.*${keyword}.*`, $options : 'i' } },
        { description: { $regex : `.*${keyword}.*`, $options : 'i' } }
    ];

    return productsCollection.find({ $or: query, price: { $gte: startR
ange, $lte: endRange } }, { _id:1, title:1, category:1, price:1, images:1 }).t
oArray();
})
.catch(() => { // returning a reject promise
    return Promise.reject("Server issue with 'products' collection.");
});
},
//----- fetch a product information by filter string
getProductByFilter: (category, startRange, endRange) => {
    return products().then((productsCollection) => {
        return productsCollection.find({ category: { $regex : `.*${category}.*` , $options : 'i' } , price: { $gte: startRange, $lte: endRange } }, { _id:1, title:1, category:1, price:1, images:1 }).toArray();
    })
    .catch(() => { // returning a reject promise
        return Promise.reject("Server issue with 'products' collection.");
    });
}

```

```

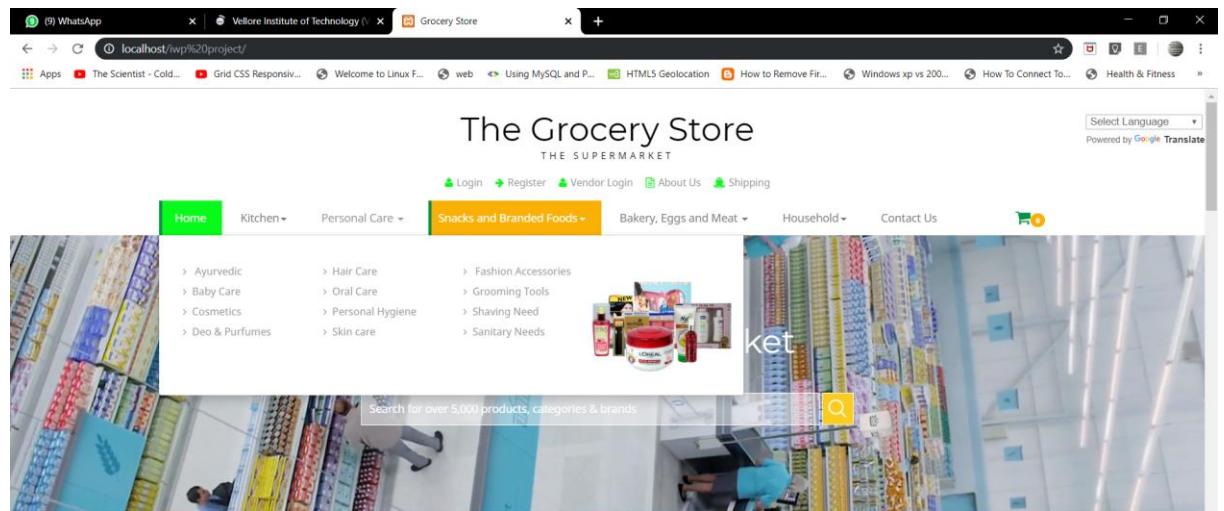
    },
// -----
//----- fetch all product information
getAllProducts: () => {
    return products().then((productsCollection) => { // returning a found
json document else returning null
        return productsCollection.find({ }).toArray();
    })
    .catch(() => { // returning a reject promise
        return Promise.reject("Server issue with 'products' collection.");
    });
},
//-----
//----- insert/create a new product record
addNewProduct: (title, description, category, brand, expDate, mfdDate, size, price, stock, images, suggestion, allegations) => {
    return products().then((productsCollection) => {

        // new product object
        let newProduct = {
            _id: uuid.v4(),
            title: xss(title),
            description: xss(description),
            category: xss(category),
            brand: brand,
            expDate: xss(expDate),
            mfdDate: xss(mfdDate),
            size: xss(size),
            price: parseFloat(xss(price)),
            stock: xss(stock),
            images: images,
            suggestion: suggestion,
            allegations: allegations
        }

        // adding a record in to the collection
        return productsCollection.insertOne(newProduct)
            .then((newProductInformation) => {
                return newProductInformation.insertedId;
            });
    })
    .catch(() => { // returning a reject promise
        return Promise.reject("Server issue with 'products' collection.");
    });
},
};

```

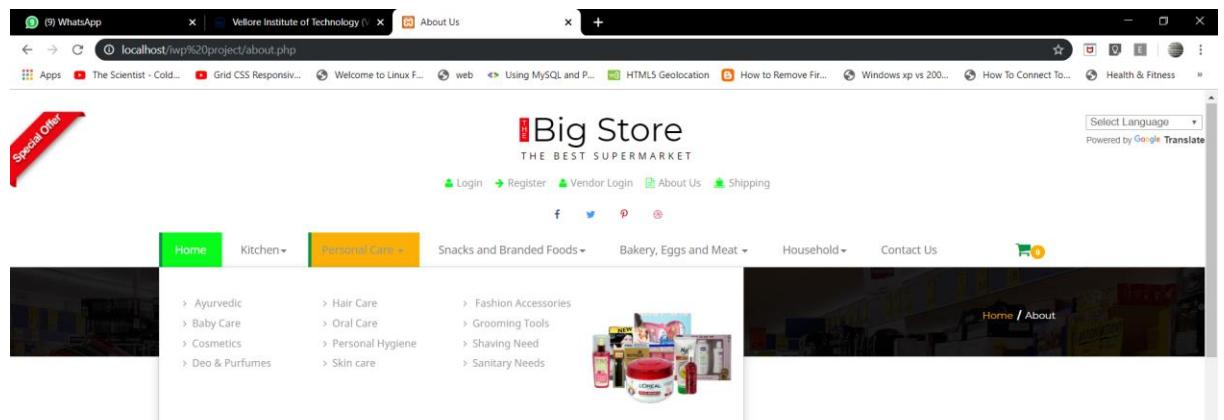
4.2.3. SAMPLE SCREENSHOTS



Special Offers



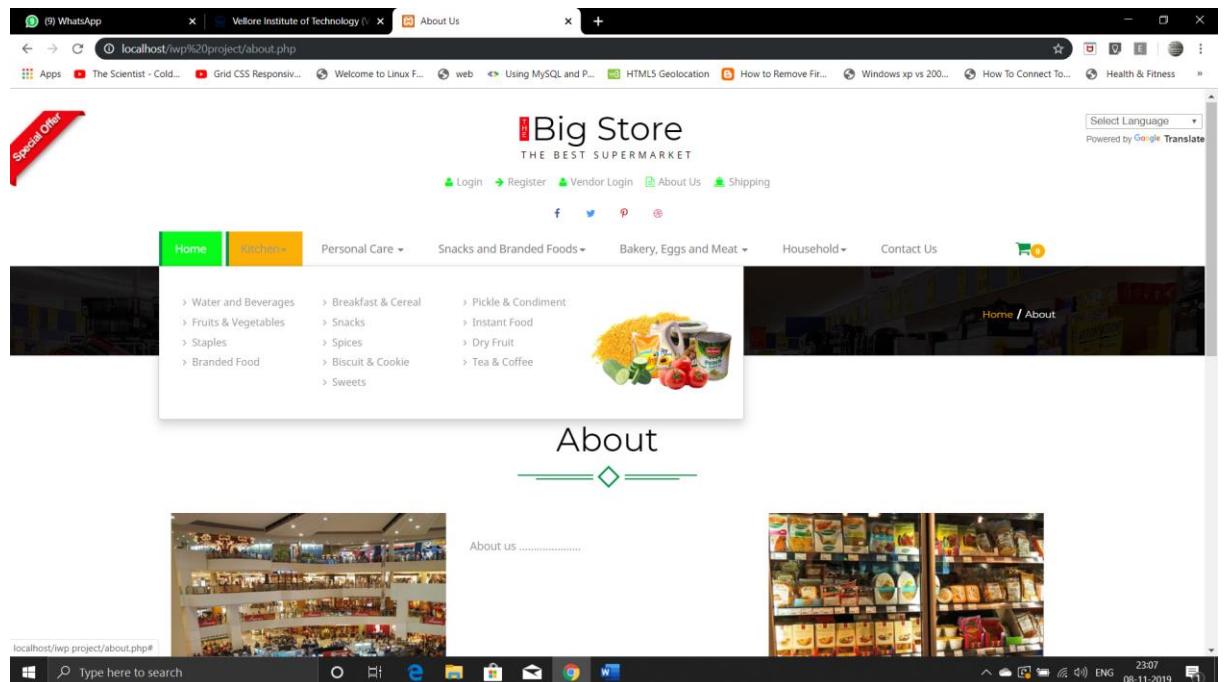
Input Image 10



About



Input Image 11



Input Image 12

4.3. TECHNOLOGY 3: MONGO DB

4.3.1. IMPORTANCE OF TECHNOLOGY

MongoDB is one of the most popular NoSQL databases around and is gaining popularity exponentially. Many more developers have started using it and increasing numbers of applications are built using MongoDB. As organisations adopt MongoDB as an integral architectural component of their solutions, it is important that they build it on the solid foundation of data Modelling. The very basic feature of MongoDB is that it is a schema-less database. No schema migrations anymore. Since MongoDB is schema-free, your code defines your schema. The ability to derive a document-based data model is one of the most attractive advantages of MongoDB. Because, the way it stores the data in the form of BSON (Binary JSON), ruby hashes etc., helps to store the data in a very rich way while being capable of holding arrays and other documents. The document query language supported by MongoDB plays a vital role in supporting dynamic queries. Very easy to scale. Due to the structuring (BSON format - key value pair) way of the data in MongoDB, no complex joins are needed. Performance tuning is absolutely easy compared to any relational databases. No need of mapping the application objects to the data objects. Enables faster access of the data due to its nature of using the internal memory for the storage. Since, it is a NOSQL database, then it is obviously secure because no sql injection can be made.

MongoDB can also be used as a file system, which helps in easier way of load balancing. MongoDB supports, the search by regex and fields as well. MongoDB can be run as windows service as well. Good amount of documentation is available. MongoDB does not require a VM to be run. MongoDB follows regular release cycle of its newer versions. The support for Sharding is one of its key feature. Sharding is the process of storing the data in different machines and MongoDB's ability to process the data, as and when the size of the data grows. This results in the horizontal scaling. With sharding, more amount of data can be written and read back as and when there is an increase in the data growth.

4.3.2. SAMPLE CODE

COLLECTION FILE

```
/* database collection configuration */
const dbConnection = require('./mongodb-connection');
/* This will allow to have one reference to each collection per app */
let getCollectionFn = (collection) => {
    let _col = undefined;
    return () => {
        if (!(_col)) {
            _col = dbConnection().then((db) => {
                return db.collection(collection);
            });
        }
        return _col;
    }
}

/* listing collections here: */
module.exports = {
    contacts: getCollectionFn("contacts"),
    credentials: getCollectionFn("credentials"),
    products: getCollectionFn("products"),
    subscriptions: getCollectionFn("subscriptions"),
    transactionOrder: getCollectionFn("orderTransaction"),
    transactionWallet: getCollectionFn("walletTransaction"),
    users: getCollectionFn("users")
};
```

CONNECTION FILE

```
/* database configuration */
const MongoClient = require('mongodb').MongoClient;;
```

```

const settings = {
  mongoConfig: {
    serverUrl: "mongodb://localhost:27017/",
    database: "db-grocery"
  }
};

let fullMongoUrl = settings.mongoConfig.serverUrl + settings.mongoConfig.database;
let _connection = undefined;

let connectDb = () => {
  if (!_connection) {
    _connection = MongoClient.connect(fullMongoUrl)
      .then((db) => {
        return db;
      });
  }

  return _connection;
};

module.exports = connectDb;

```

4.3.3. SAMPLE SCREENSHOTS

```

C:\Users\ymanan>mongo
MongoDB shell version v4.2.1
connecting to: mongodb://127.0.0.1:27017/?compressors=disabled&gssapiServiceName=mongodb
Implicit session: session { "id" : UUID("b86ee545-8216-4fd7-ab20-57365e7f470c") }
MongoDB server version: 4.2.1
Server has startup warnings:
2019-11-02T21:38:44.522+0530 I CONTROL [initandlisten]
2019-11-02T21:38:44.522+0530 I CONTROL [initandlisten] ** WARNING: Access control is not enabled for the database.
2019-11-02T21:38:44.523+0530 I CONTROL [initandlisten] ** Read and write access to data and configuration is unrestricted.
2019-11-02T21:38:44.523+0530 I CONTROL [initandlisten]

Enable MongoDB's free cloud-based monitoring service, which will then receive and display
metrics about your deployment (disk utilization, CPU, operation statistics, etc).
The monitoring data will be available on a MongoDB website with a unique URL accessible to you
and anyone you share the URL with. MongoDB may use this information to make product
improvements and to suggest MongoDB products and deployment options to you.

To enable free monitoring, run the following command: db.enableFreeMonitoring()
To permanently disable this reminder, run the following command: db.disableFreeMonitoring()
> --
> show dbs
admin 0.000GB
config 0.000GB
local 0.000GB
test 0.000GB
>

```

Show dbs

```

C:\WINDOWS\system32\cmd.exe - mongo
(c) 2019 Microsoft Corporation. All rights reserved.

C:\Users\naman>mongo
MongoDB shell version v4.2.1
connecting to: mongodb://127.0.0.1:27017/?compressors=disabled&gssapiServiceName=mongodb
Implicit session: session { "_id" : UUID("08d8fe:a-4d2b-41c0-9ded-a70dadff0176") }
MongoDB server version: 4.2.1
Server has startup warnings:
2019-11-02T21:38:44.522+0530 I CONTROL [initandlisten]
2019-11-02T21:38:44.522+0530 I CONTROL [initandlisten] ** WARNING: Access control is not enabled for the database.
2019-11-02T21:38:44.523+0530 I CONTROL [initandlisten] ** Read and write access to data and configuration is unrestricted.
2019-11-02T21:38:44.523+0530 I CONTROL [initandlisten]
...
Enable MongoDB's free cloud-based monitoring service, which will then receive and display
metrics about your deployment (disk utilization, CPU, operation statistics, etc).

The monitoring data will be available on a MongoDB website with a unique URL accessible to you
and anyone you share the URL with. MongoDB may use this information to make product
improvements and to suggest MongoDB products and deployment options to you.

To enable free monitoring, run the following command: db.enableFreeMonitoring()
To permanently disable this reminder, run the following command: db.disableFreeMonitoring()
---

> show dbs
admin 0.000GB
config 0.000GB
local 0.000GB
local 0.000GB
> use db imp
2019-11-08T23:00:51.873+0530 E QUERY [js] Error: [db imp] is not a valid database name :
Mongo.prototype.getDB@src/mongo/shell/mongo.js:51:12
getDatabase@src/mongo/shell/session.js:913:28
DB.prototype.getSibling@src/mongo/shell/db.js:22:12
shellHelper_use@src/mongo/shell/utils.js:803:10
shellHelper_use@src/mongo/shell/utils.js:790:15
#(shellHelp2):1:1
> use imp
switched to db imp
>

```

Use iwp

```

C:\WINDOWS\system32\cmd.exe - mongo
MongoDB shell version v4.2.1
connecting to: mongodb://127.0.0.1:27017/?compressors=disabled&gssapiServiceName=mongodb
Implicit session: session { "_id" : UUID("08d8fe:a-4d2b-41c0-9ded-a70dadff0176") }
MongoDB server version: 4.2.1
Server has startup warnings:
2019-11-02T21:38:44.522+0530 I CONTROL [initandlisten]
2019-11-02T21:38:44.522+0530 I CONTROL [initandlisten] ** WARNING: Access control is not enabled for the database.
2019-11-02T21:38:44.523+0530 I CONTROL [initandlisten] ** Read and write access to data and configuration is unrestricted.
2019-11-02T21:38:44.523+0530 I CONTROL [initandlisten]
...
Enable MongoDB's free cloud-based monitoring service, which will then receive and display
metrics about your deployment (disk utilization, CPU, operation statistics, etc).

The monitoring data will be available on a MongoDB website with a unique URL accessible to you
and anyone you share the URL with. MongoDB may use this information to make product
improvements and to suggest MongoDB products and deployment options to you.

To enable free monitoring, run the following command: db.enableFreeMonitoring()
To permanently disable this reminder, run the following command: db.disableFreeMonitoring()
---

> show dbs
admin 0.000GB
config 0.000GB
iwp 0.000GB
local 0.000GB
local 0.000GB
> use db iwp
2019-11-08T23:00:51.873+0530 E QUERY [js] Error: [db iwp] is not a valid database name :
Mongo.prototype.getDB@src/mongo/shell/mongo.js:51:12
getDatabase@src/mongo/shell/session.js:913:28
DB.prototype.getSibling@src/mongo/shell/db.js:22:12
shellHelper_use@src/mongo/shell/utils.js:803:10
shellHelper_use@src/mongo/shell/utils.js:790:15
#(shellHelp2):1:1
> use iwp
switched to db iwp
> show collections
brands
cart
categories
customers
delivery
orders
products
vendors
>

```

Show collections

5. SCREENSHOTS

5.1. CUSTOMER MODULE

localhost / Register

Register

Username

Email

Phone

Street Address

City

Pincode/ Zipcode

Submit

Register As New User(Customer)

localhost / Login

Grocery Store

THE BEST SUPERMARKET

Special Offer

Login Register Vendor Login About Us Shipping

f t p g

Home Kitchen Personal Care Snacks and Branded Foods Bakery, Eggs and Meat Household Contact Us

Cart (1)

Type here to search

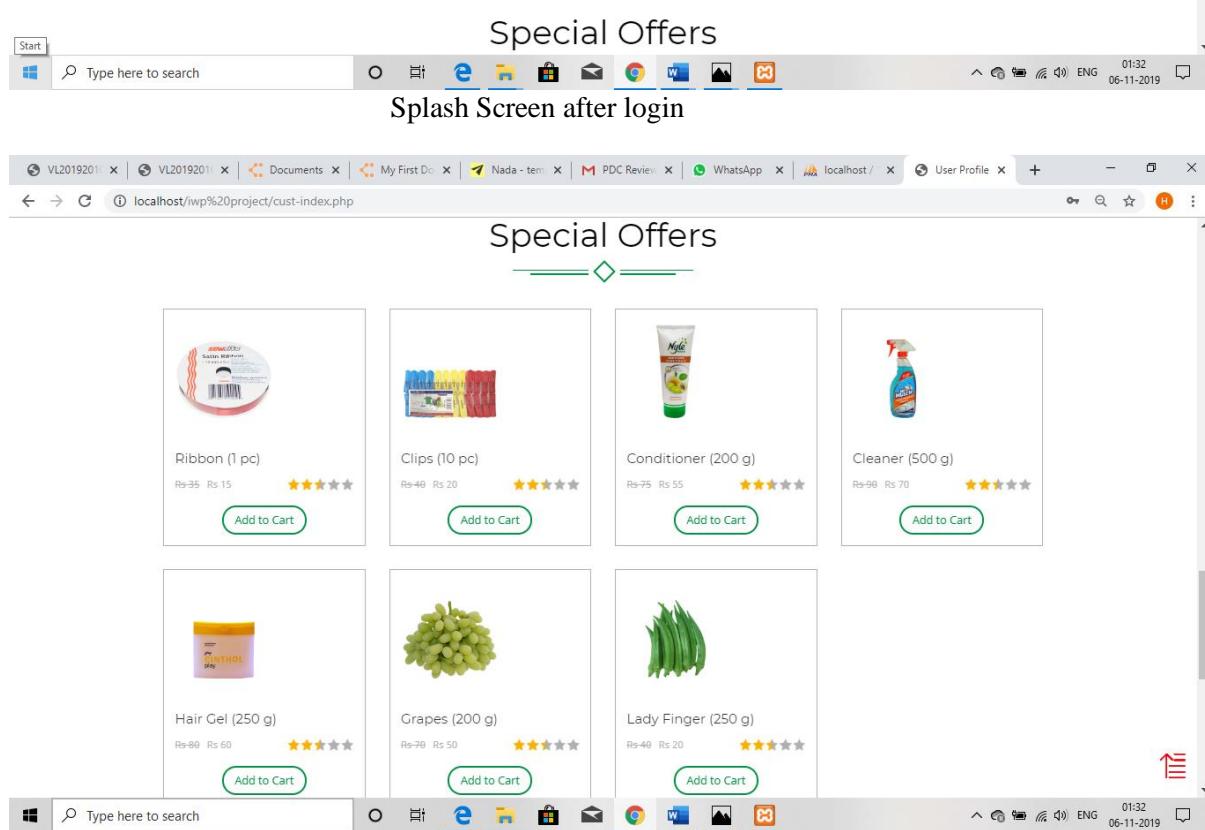
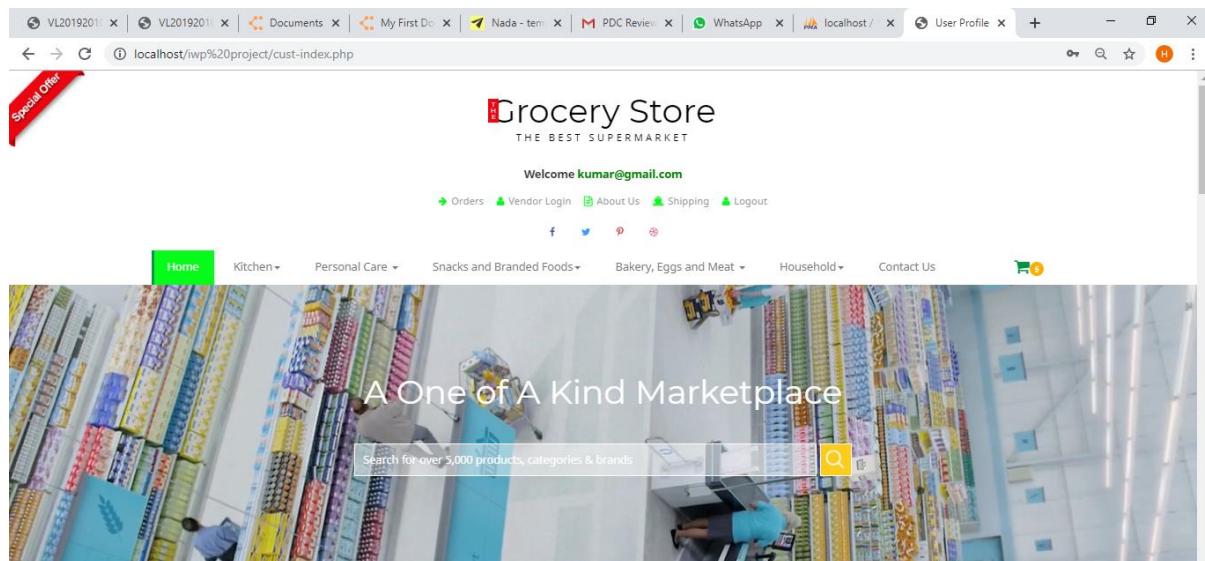
Login

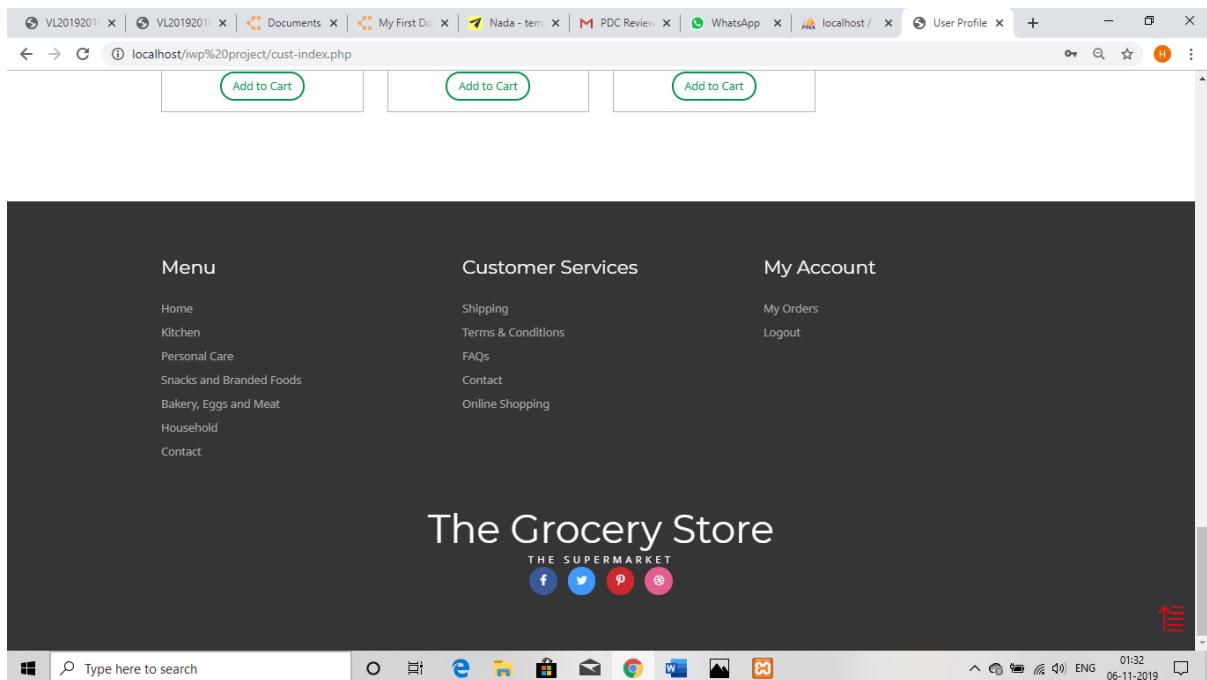
Email

Login

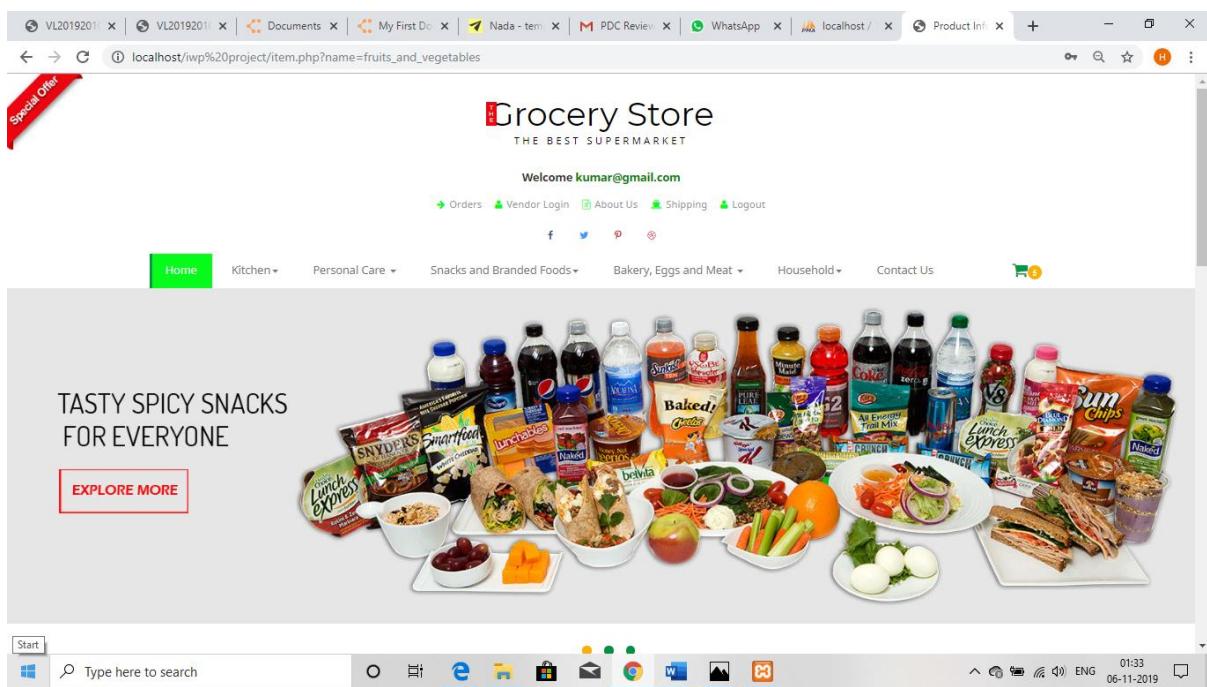
Forgot Password Register

Existing User





Offers and products visible to user



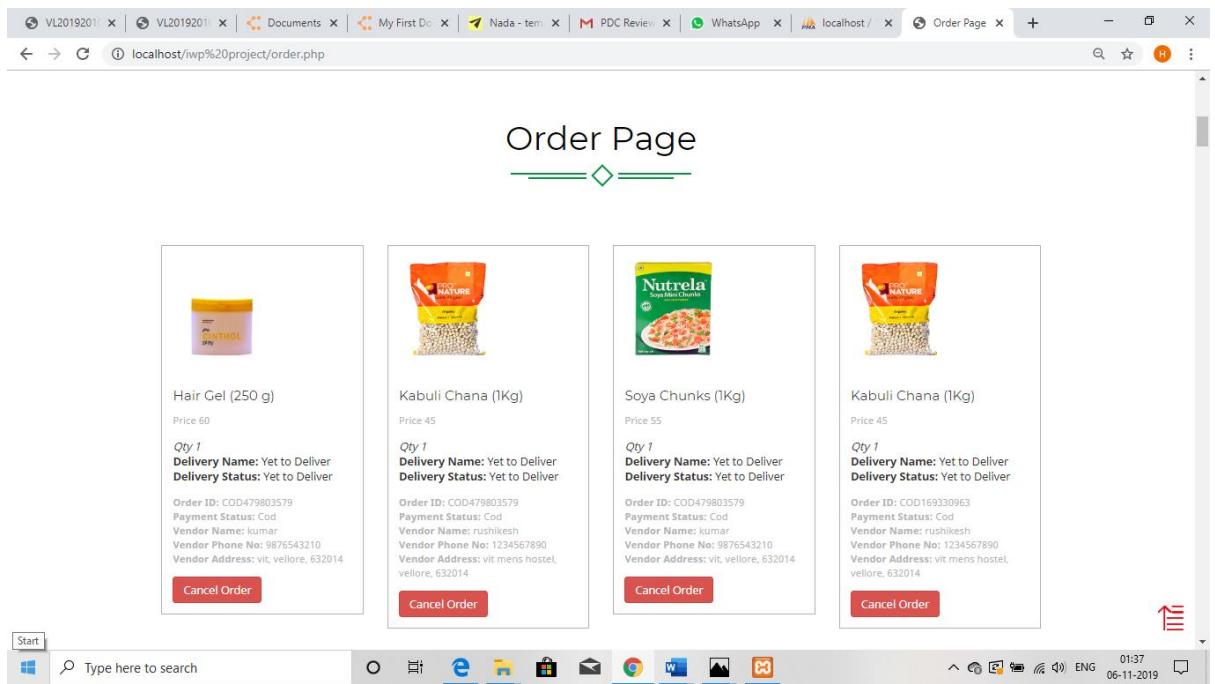
The screenshot shows a web browser window with multiple tabs open at the top. The active tab displays a grocery store website titled "Grocery Store - THE BEST SUPERMARKET". A red ribbon banner on the left says "Special Offer". The header includes a welcome message "Welcome kumar@gmail.com" and links for Orders, Vendor Login, About Us, Shipping, and Logout. Below the header is a navigation menu with categories like Home, Personal Care, Snacks and Branded Foods, Bakery, Eggs and Meat, Household, and Contact Us. A search bar is present, and the background features images of grocery store aisles.

This screenshot shows a Windows desktop environment. The taskbar at the bottom includes icons for File Explorer, My Computer, Task View, Edge, File Explorer, Mail, Photos, Google Chrome, Word, and Pictures. A search bar is visible on the left of the taskbar. The system tray shows battery status, signal strength, and the date/time (01:33 06-11-2019). The desktop background is a light blue color.

CATEGORIES OF PRODUCTS

The screenshot shows a contact page from a website. The title "Contact" is at the top. Below it is a section titled "Contact Information" with fields for Name, Email, and a large text area for a message. A "Submit" button is at the bottom of this form. To the left of the form is a photograph of a person on a motorcycle carrying boxes. Below the photo is a Google Map showing a street view of a neighborhood with various landmarks labeled. The taskbar at the bottom is identical to the one in the previous screenshot, showing the same set of application icons and system status.

CONTACT INFORMATION



5.2.CART

ORDER PLACING PAGE

My Cart

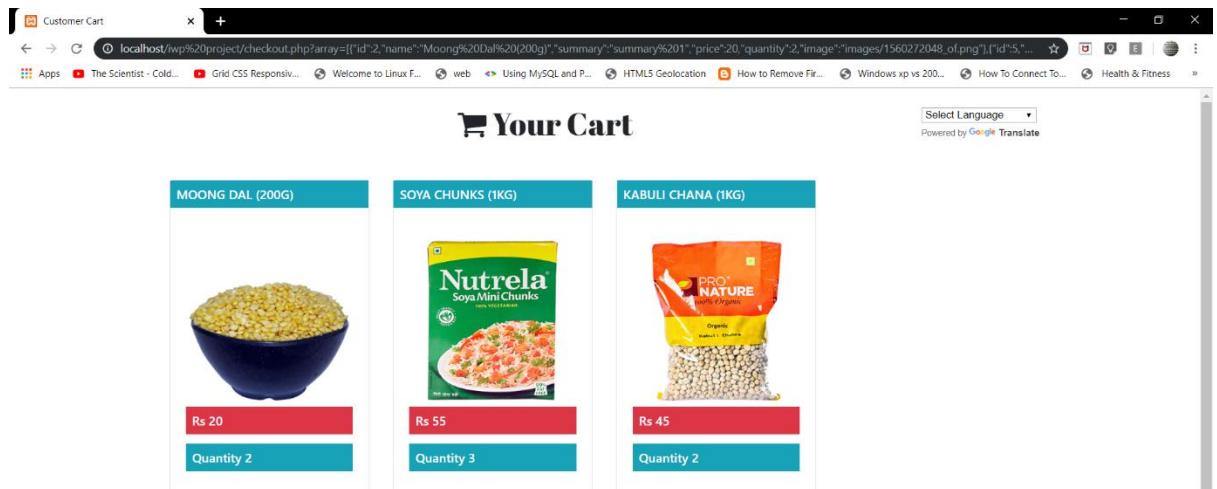
Moong Dal (200g)	Rs 20	2	Rs 40	<input type="button" value="X"/>
Soya Chunks (1kg)	Rs 55	3	Rs 165	<input type="button" value="X"/>
Kabuli Chana (1kg)	Rs 45	2	Rs 90	<input type="button" value="X"/>
Total			Rs 295	
Total (including discount)			Rs 295	

Close **Checkout**

Special Offers

Staples Snacks Fruits & Vegetables Breakfast & Cereal

Adding items to Cart



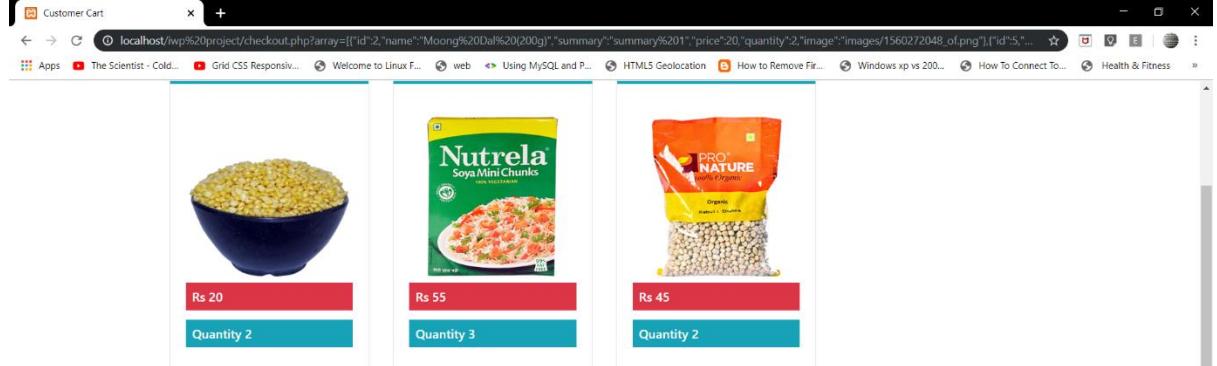
Choose Delivery Method

Normal (+ 50 Rs)

Choose Payment Method

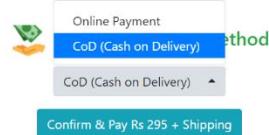
Normal (+ 50 Rs)

Items in Cart



Choose Delivery Method

Normal (+ 50 Rs)



Delivery Options

Order Placed

Purchased from	Grocery Store
Payment ID	COD768078202
Buyer Name	naman
Buyer Phone No.	9876543210
Buyer Email	naman@gmail.com
Buyer Address	D6, vit, 632014
Payment Status	Pending
Order Date	06/11/2019 01:33:18

Track your Order | Continue Shopping

ORDER PLACED DESCRIPTION

5.3. Vendor Module

Hello rushikesh@gmail.com

Orders Total 1875

#	Image	Product Name	Quantity	Price	Order Id	Payment Status	Order Date	Vendor Name	Buyer Name	Buyer Phone	Shipping Method	Delivery Status
#		Clips (10 pc)	1	20	COD666012840	Cod	16/10/2019 00:26:33	rushikesh@gmail.com	Express	ND		
#		Clips (10 pc)	1	20	COD356572164	Cod	16/10/2019 00:27:53	rushikesh@gmail.com	Express	ND		
#		Sunflower Oil (5Kg)	1	100	COD356572164	Cod	16/10/2019 00:27:53	rushikesh@gmail.com	Express	ND		
#		Kabuli Chana (1Kg)	1	45	COD356572164	Cod	16/10/2019 00:27:53	rushikesh@gmail.com	Express	ND		
#		Clips (10 pc)	1	20	COD782700800	Cod	16/10/2019 00:28:15	rushikesh@gmail.com	Express	ND		

Vendor orders

localhost/wp%20project/delivery.php

Sign out Change Password

[Dashboard](#) [Orders](#) [Products](#) [Brands](#) [Categories](#) [Deliveries](#) [Customers](#)

Hello rushikesh@gmail.com

Deliveries

Add Delivery Contact

Name	Email	Phone	Address
delivery_boy	delivery@gmail.com	1234567890	vit vellore 632014
delivery_boy_2	delivery1@gmail.com	1221111122	vit mens hostel vellore 632014
delivery_boy_3	something@gmail.com	0123456789	some_street some city 632014
del_4	del_4_4	0987667890	vit vellore 632014

localhost/wp%20project/customers.php

Sign out Change Password

[Dashboard](#) [Orders](#) [Products](#) [Brands](#) [Categories](#) [Deliveries](#) [Customers](#)

Hello kumar@gmail.com

Customers

#	Name	Email	Mobile	Address
#	kumar	kumar@gmail.com	9876543210	D6 vit 632014

Customers Ordering

localhost/wp%20project/customer_orders.php

[Dashboard](#) [Orders](#) [Products](#) [Brands](#) [Categories](#) [Deliveries](#) [Customers](#)

Update Delivery

Select Order ID: COD417530873

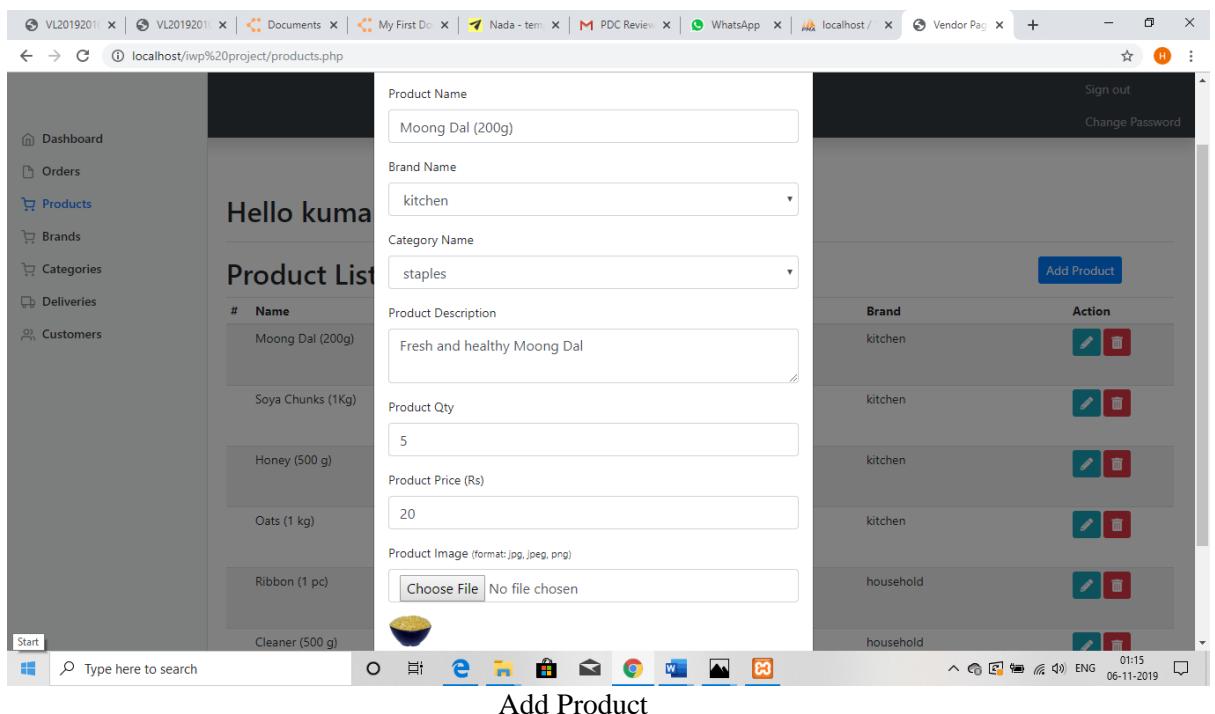
Select Delivery Boy/Company: delivery_boy

Update Delivery

Soya Chunks (1Kg)	1	55	COD169330963	Cod	26/10/2019	kumar@gmail.com	Normal	ND
Soya Chunks (1Kg)	1	55	COD479803579	Cod	26/10/2019	kumar@gmail.com	Normal	ND
Hair Gel (250 g)	1	60	COD479803579	Cod	26/10/2019	kumar@gmail.com	Normal	ND

Request Delivery **Update Delivery**

Delivery Boys



6. CONCLUSIONS

This is a small prototype of a sales management application for a grocery business. The limitation of the application is that it lacks some features that can be removed and some to be added so that it can be implemented in a real-life business situation.

The system is such that it requires the user (customer/vendor or the admins) to login before access other function on the system which are the most important credentials provided by the company or as the user(customer) as filled in their details, without this no proper authentication or validation of the customer is possible.

This project aim is to simplify the daily business activity of the grocer and reduce the time consumed for processing business activity. This project will allow the wholesales man to have a better understanding of the distribution of product and amount stocks.

This project will contribute to improve the efficiency and performance of the business activity of the grocer man. This system will increase the performance of recording all the information related with business activity of grocer salesman like supplier, customer, staff and product and payment information orderly.

Also as we see the younger retailers are taking over the shops and thus they would like this method of online ordering and doesn't have to travel themselves to a number of suppliers/whole salesmen and do bargaining and then order and then wait to receive the

requested goods, also there will be some discount or if possible there can made be a feature such that retailer and whole salesman can bargain over phone or something like that.

Therefore, we would love to see the more proper implementation of our project which would help the grocer salesman or a company to acquire this method of online ordering which is in fact similar to e-commerce but in that the amount of products ordered by a customer is not in bulk. This method of management is for supplying goods in bulk either by small or sometimes big retailers, whoever finds it convenient.