Dharmsinh Desai University, Nadiad

Faculty of Technology, Department of Computer Engineering

B.Tech. CE Semester – V

Subject: (CE-515) Advanced Technologies

Project Title:

# Books And Mobile Phone  Reselling System

[ Reventa.in ]

By:

Pathak Vyom A. (Roll No: CE100 ID: 17CEUON038)

Raval Deepang M. (Roll No: CE114 ID: 17CEUON076)

Guided By:

Prof. Ankit P. Vaishnav

Prof. Parth R. Dave

# Dharmsinh Desai University, Nadiad

Faculty of Technology, Department of Computer Engineering

## CERTIFICATE

This is to certify that Advanced Technologies project entitled "Books And Mobile Phone Reselling System" is the bonafied report of work carried out by

1) **Pathak Vyom A. (17CEUON038)**
2) **Raval Deepang M. (17CEUON076)**

Of Department of Computer Engineering, Semester V, academic year 2019-20, under our supervision and guidance.

| Guide | Guide | HOD |
|---|---|---|
| **Prof. Ankit P. Vaishnav** Assistant Professor of Department of Computer Engineering, Dharmsinh Desai University, Nadiad. | **Prof. Parth R. Dave** Assistant Professor of Department of Computer Engineering, Dharmsinh Desai University, Nadiad. | **Dr. C. K. Bhensdadia** Head of the Department of Department of Computer Engineering, Dharmsinh Desai University, Nadiad. |

## Overview

This website is used for reselling mobile phones and books. It gives users option from various books and various phones. After user selects the desired product, it shows the sellers information so that the user can directly contact the seller to discuss about product. Apart from this we give angular authentication with auth0 API which guarantees user security. For any query about the product or website we give an integrated bot to interact with the customers to fulfill their needs. For any further query we have provided our email addresses and social media tags so that they can contact us any time. We thrive for user interaction and user satisfaction.

# Introduction

Reventa.in is a website which gives users a chance to sell and buy books and phones.
The user login is designed on the auth0 API given by angular 5. It is an authentication API provided for Angular 6. Users can login through their emails or through Google or facebook according to their convenience. It also provides users to easily change password if any user has forgotten their password.

Users can see and select their desired product. They can filter products according to category and according to price range. They can also search a desired product they want to buy.
Users can also add products to their favorite's list. This helps us to recommend that user with products similar to that one. After they have selected their product and they wish to buy this product the sellers contact formation is provided with the description of the product. With this information user can directly contact with the seller and get the product.

If the user cannot find the desired product, they can always contact us to check the status of their desired product. For user's convenience we have provided a chat bot which will answer all the frequently asked questions, asked by different users. If the user has any type of query they can also contact us through the email provided by us. They can also contact us through social media platforms like instagram, facebook.  They can also call us for any type of query.

## Technology Used:

- Cascading Style Sheet (CSS 3) for styling the HTML pages.
- JavaScript for providing dynamic content for the HTML pages.
- Jquery to provide dynamic data to the request pages.
- Bootstrap4 for styling the HTML pages using predefined classes of bootstrap.
- Angular for frontend purposes.
- Express.js for routing purposes.
- Node.js for backend purposes.
- Auth0, Auth0's very own authentication and authorization API for Login purpose.
- Mongo DB as a database manager to add, update and fetch data from the database using Node.js mongoose module.

## Platform Used:

- localhost/3000 for Angular
- localhost/8080 for Node js
- mongodb://127.0.0.1:27017

## Tools Used:

- We used Github as a version control method and to manage the project.
- We used Visual Studio Code for development of the code.

# Software Requirement Specification

**INDEX:**

### 1.5 MANAGE CONTACT INFO
#### 1.5.1 ABOUT US
#### 1.5.2 CONTACT US
#### 1.5.3 MANAGE FORGET PASSWORD

## 2. NON FUNCTIONAL REQUIREMENTS

## FUNCTIONAL REQUIREMENTS:

### R1. MANAGE_CUSTOMERS_INFORMATION:

DESCRIPTION: This function allows user to create his/her account while he/she is using website for first time. This function also allows the user to login through his/her account and proceed further according to the needs.

**R1.1 Add_Customer_Info:**

INPUT: User enters the Name, age, email, account-type, username and password.
OUTPUT: Details will be forwarded to system and system will assign the account accordingly.
PROCESS: The entered details will be authenticated internally.
NEXT: Login page will be displayed.

**R1.2 Customer_Login:**

INPUT: The login page will be available for customer with his/her Customer-id and Password he/she can Login.
OUTPUT: The product page will be displayed.
PROCESS: The id and password will be authenticated and then output will be provided if successful.
NEXT: The display page will be shown.

**R1.3 Delete_Customer_Info:**

INPUT: Customer Selection.
OUTPUT: Confirmation Message.

## R2. MANAGE_SELLING:

DESCRIPTION: We can add, remove, update product by seller using this function and seller can mark it as sold.

**R2.1 Add_Product_Details:**

INPUT: Product Data.
OUTPUT: Confirmation message.

**R2.2 Update_Product_Details:**

INPUT: Product Data.
OUTPUT: Confirmation message.

**R2.3 Delete_Product_Details:**

INPUT: Seller/Admin Selection.
OUTPUT: Confirmation message.

**R2.4 Sales_Confirmation:**

INPUT: Seller Selection
OUTPUT: That particular product will be shifted to sell list.

## R3. MANAGE_BUYING:

DESCRIPTION: The list of mobiles and books will be viewed to the customer, customer can search for the product, share its details and can add the product to favorites list also.

**R3.1 Display_Product_Details:**

INPUT: Customer Selection.
OUTPUT: The page will be displayed which will have all details about the selected product and suggestions of the similar kind of mobiles and books.

**R3.2 Display_Searched_Product_Details:**

INPUT: Customer searches for the product as per the name of the mobiles and books.
OUTPUT: The display page will be redirected with new request.

PROCESS: The list of mobiles or books will be checked internally and then output will be generated accordingly.

**R3.3 Display_Seller_Info:**

INPUT: User Selection.
OUTPUT: Displays the information about the seller like address and mobile number.

**R3.4 Share_Product_Details:**

INPUT: User Selection.
OUTPUT: A mail will be sent to the email entered by user.

**R3.5 Manage_Favourite_Products:**

INPUT: User Selection.
OUTPUT: The selected product will be displayed in the favorite's list.

## R4. MANAGE_SELLER:

DESCRIPTION: The seller's account can be managed through this option.

**R4.1 Add_Seller_Info:**

INPUT: Seller details.
OUTPUT: Confirmation message.

**R4.2 Seller_Login:**

INPUT: The login page will be available for seller with his/her Seller-id and Password he/she can Login.
OUTPUT: The product page will be displayed.
PROCESS: The id and password will be authenticated and then output will be provided if successful.
NEXT: The display page will be shown.

**R4.3 Delete_Seller_Info:**

INPUT: User Selection.
OUTPUT: Confirmation Message.

## R5. MANAGE_CONTACT_INFO:

DESCRIPTION: The user can connect with admin through this option.

**R5.1 About_Us:**

INPUT: User Selection.
OUTPUT: About us page.

**R5.2 Contact_Us:**

INPUT: User Selection.
OUTPUT: Contact us page.

**R5.3 Manage_Forget_Password:**

INPUT: User Selection.
OUTPUT: An email will be sent to user to change his/her password.

## NON FUNCTIONAL REQUIREMENTS:

SOFTWARE: Visual Studio Code, Microsoft Azure, Github Desktop

OS: Windows, Linux, ChromeOS

BROWSERS: Google Chrome, Microsoft Edge, Firefox

# Design Documents

1. XML:

**Seller:**

```
<sellers>
        <seller id="s_01">
                <name>ABCXYZ</name>
                <phone_no>9999999999</phone_no>
                <email>ABC12345@gmail.com</email>
                <address>603, SDFG</address>
                <img_link>DFGH.jpg</img_link>
                <latitude>19.200</latitude>
                <longitude>20.009</longitude>
        </seller>
</sellers>
```

**Customer:**

```
<customers>
        <customer id="c_01">
                <name>JHKASD</name>
                <phone_no>9898989698</phone_no>
                <email>gehfajd12345@gmail.com</email>
                <address>615, adfjg</address>
                <img_link>dfsghjg.jpg</img_link>
                <latitude>19.200</latitude>
                <longitude>20.009</longitude>

        </customer>
</customers>
```

**Product:**

```
<products>

    <product p_id="p_01">
            <s_id>01</s_id>
            <name>node.js</name>
            <price>5000</price>
            <type>Book</type>
            <description>
<![CDATA[This book is based on learning of node.js]]></description>
            <img_link>book.jpg</img_link>
    </product>

    <product p_id="p_02">
            <s_id>02</s_id>
            <name>Redmi Note 4</name>
            <price>5000</price>
            <type>Refurbished</type>
            <description>
            <![CDATA[This is the phone provided by MI]]></description>
            <img_link>redmi.jpg</img_link>
    </product>

</products>
```

2. DTD:

**Seller:**

```
<!DOCTYPE sellers

[

   <!ELEMENT sellers (seller+)>

   <!ELEMENT seller
(name,phone_no,email,address,img_link,latitude,longitude)>

   <!ELEMENT name (#PCDATA)>

   <!ELEMENT phone_no (#PCDATA)>

   <!ELEMENT email (#PCDATA)>

   <!ELEMENT address (#PCDATA)>

   <!ELEMENT img_link (#PCDATA)>

   <!ELEMENT latitude (#PCDATA)>

   <!ELEMENT longitude (#PCDATA)>

   <!ATTLIST seller id ID #REQUIRED>

]

>
```

**Customer:**

```
<!DOCTYPE customers

[

   <!ELEMENT customers (customer+)>

   <!ELEMENT customer (name,phone_no,email,address,img_link)>

   <!ELEMENT name (#PCDATA)>

   <!ELEMENT phone_no (#PCDATA)>

   <!ELEMENT email (#PCDATA)>

   <!ELEMENT address (#PCDATA)>

   <!ELEMENT img_link (#PCDATA)>

   <!ATTLIST customer id ID #REQUIRED>

]

>
```

**Product:**

```
<!DOCTYPE products

[

   <!ELEMENT products (product+)>

   <!ELEMENT product (s_id,name,price,type,description,img_link)>

   <!ELEMENT s_id (#PCDATA)>

   <!ELEMENT name (#PCDATA)>

   <!ELEMENT price (#PCDATA)>

   <!ELEMENT type (#PCDATA)>

   <!ELEMENT dscription (#PCDATA)>

   <!ELEMENT img_link (#PCDATA)>

   <!ATTLIST product p_id ID #REQUIRED>

]

>
```

3. XSD:

**Product:**

```
<?xml version="1.0" encoding="utf-8"?>
<xs:schema id="Products"
    elementFormDefault="qualified"
    xmlns:xs="http://www.w3.org/2001/XMLSchema"
>
  <xs:element name="products">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="product" type="ProductType"
minOccurs="0" maxOccurs="unbounded"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
  <xs:complexType name="ProductType">
    <xs:all>
      <xs:element name="s_id" type="SIDType"/>
      <xs:element name="name" type="NameType"/>
      <xs:element name="price" type="PriceType"/>

      <xs:element name="type" type="TType"/>

      <xs:element name="description" type="DType"/>

      <xs:element name="img_link" type="IMGType"/>

    </xs:all>

    <xs:attribute name="p_id" type="xs:ID" use="required"/>

  </xs:complexType>

  <xs:simpleType name="SIDType">

    <xs:restriction base="xs:int"/>

  </xs:simpleType>

  <xs:simpleType name="NameType">

    <xs:restriction base="xs:string">

      <xs:minLength value="2"/>

      <xs:maxLength value="50"/>

    </xs:restriction>

  </xs:simpleType>

  <xs:simpleType name="PriceType">

    <xs:restriction base="xs:double">

      <xs:minExclusive value="50"/>

      <xs:maxInclusive value="500000"/>

    </xs:restriction>

  </xs:simpleType>
```

```
  <xs:simpleType name="TType">
    <xs:restriction base="xs:string"/>
  </xs:simpleType>
  <xs:simpleType name="DType">
    <xs:restriction base="xs:string">
      <xs:minLength value="30"/>
      <xs:maxLength value="2000"/>
    </xs:restriction>
  </xs:simpleType>
  <xs:simpleType name="IMGType">
    <xs:restriction base="xs:string">
      <xs:maxLength value="500"/>
    </xs:restriction>
  </xs:simpleType>
</xs:schema>
```

## Seller:

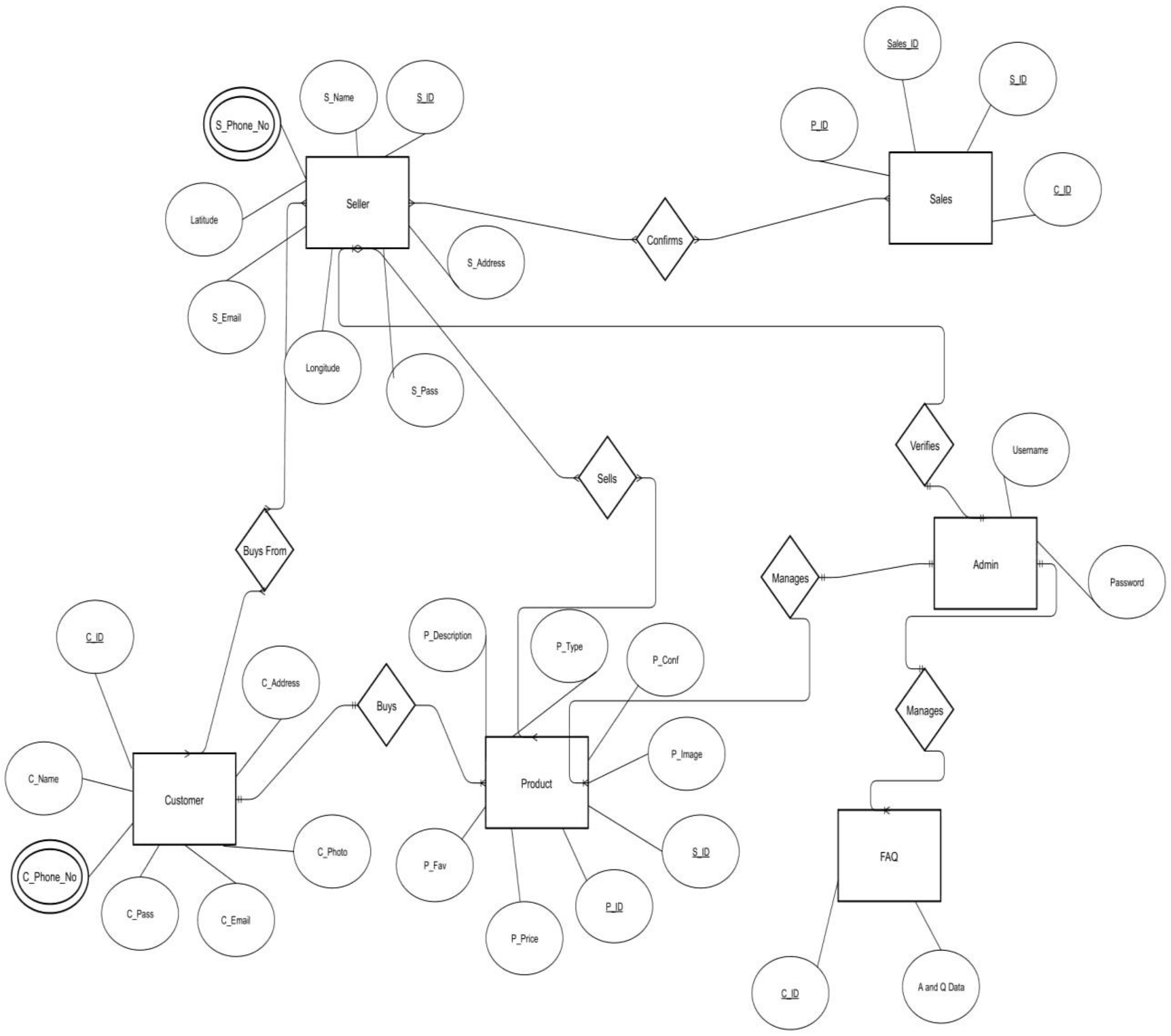
```
<xs:schema attributeFormDefault="unqualified" elementFormDefault="qualified"
xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <xs:element name="sellers">
    <xs:complexType>
      <xs:sequence>
        <xs:element minOccurs="0" name="seller">
          <xs:complexType>
            <xs:sequence>
              <xs:element minOccurs="0" name="name" type="xs:string" />
              <xs:element minOccurs="0" name="phone_no" type="xs:unsignedLong"/>
              <xs:element minOccurs="0" name="email" type="xs:string" />
              <xs:element minOccurs="0" name="address" type="xs:string" />
              <xs:element minOccurs="0" name="img_link" type="xs:string" />
              <xs:element minOccurs="0" name="latitude" type="xs:decimal" />
              <xs:element minOccurs="0" name="longitude" type="xs:decimal" />
            </xs:sequence>
            <xs:attribute name="id" type="xs:string" use="optional" />
          </xs:complexType>
        </xs:element>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
</xs:schema>
```

**Customers:**

```
<?xml version="1.0" encoding="utf-8"?>

<!-- Created with Liquid Technologies Online Tools 1.0
(https://www.liquid-technologies.com) -->

<xs:schema attributeFormDefault="unqualified"
elementFormDefault="qualified"
xmlns:xs="http://www.w3.org/2001/XMLSchema">

  <xs:element name="customers">

    <xs:complexType>

      <xs:sequence>

        <xs:element minOccurs="0" name="customer">

          <xs:complexType>

            <xs:sequence>

              <xs:element minOccurs="0" name="name" type="xs:string" />

              <xs:element minOccurs="0" name="phone_no"
type="xs:unsignedLong" />

              <xs:element minOccurs="0" name="email" type="xs:string" />

              <xs:element minOccurs="0" name="address" type="xs:string"/>

              <xs:element minOccurs="0" name="img_link"
type="xs:string"/>

            </xs:sequence>

            <xs:attribute name="id" type="xs:string" use="optional" />

          </xs:complexType>
        </xs:element>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
</xs:schema>
```

## 4. E-R Diagram :

7/16/2019 Untitled Diagram.drawio



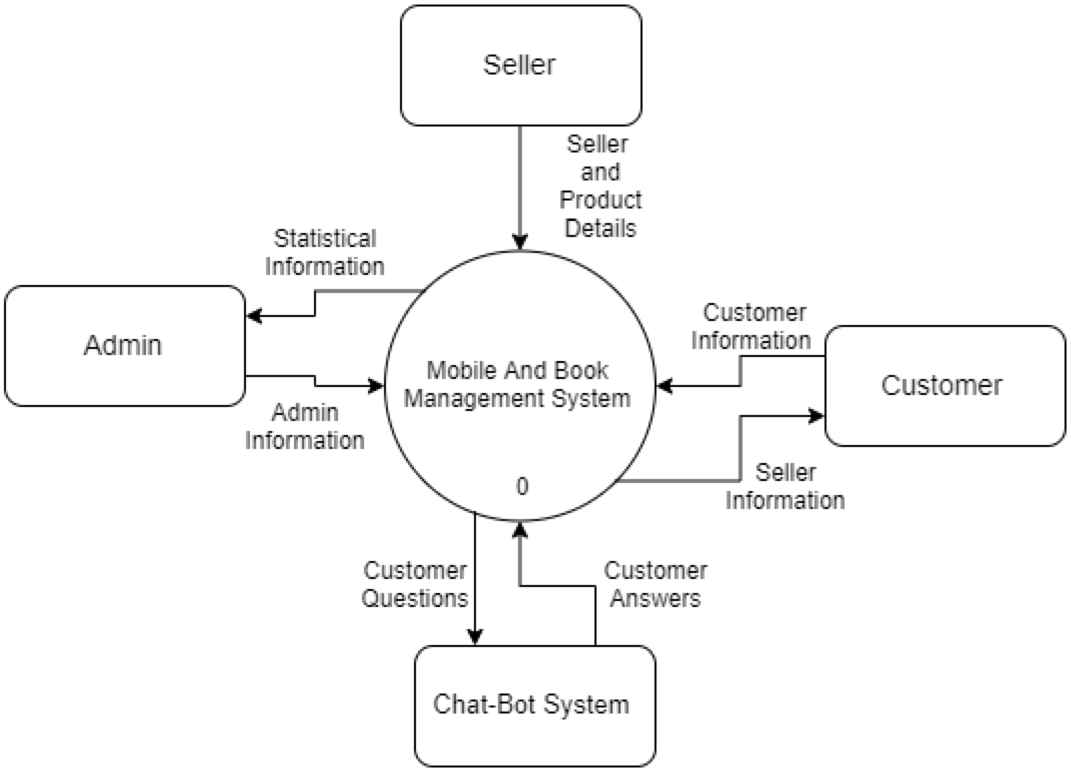
1/1

## 5. Data Dictionary:

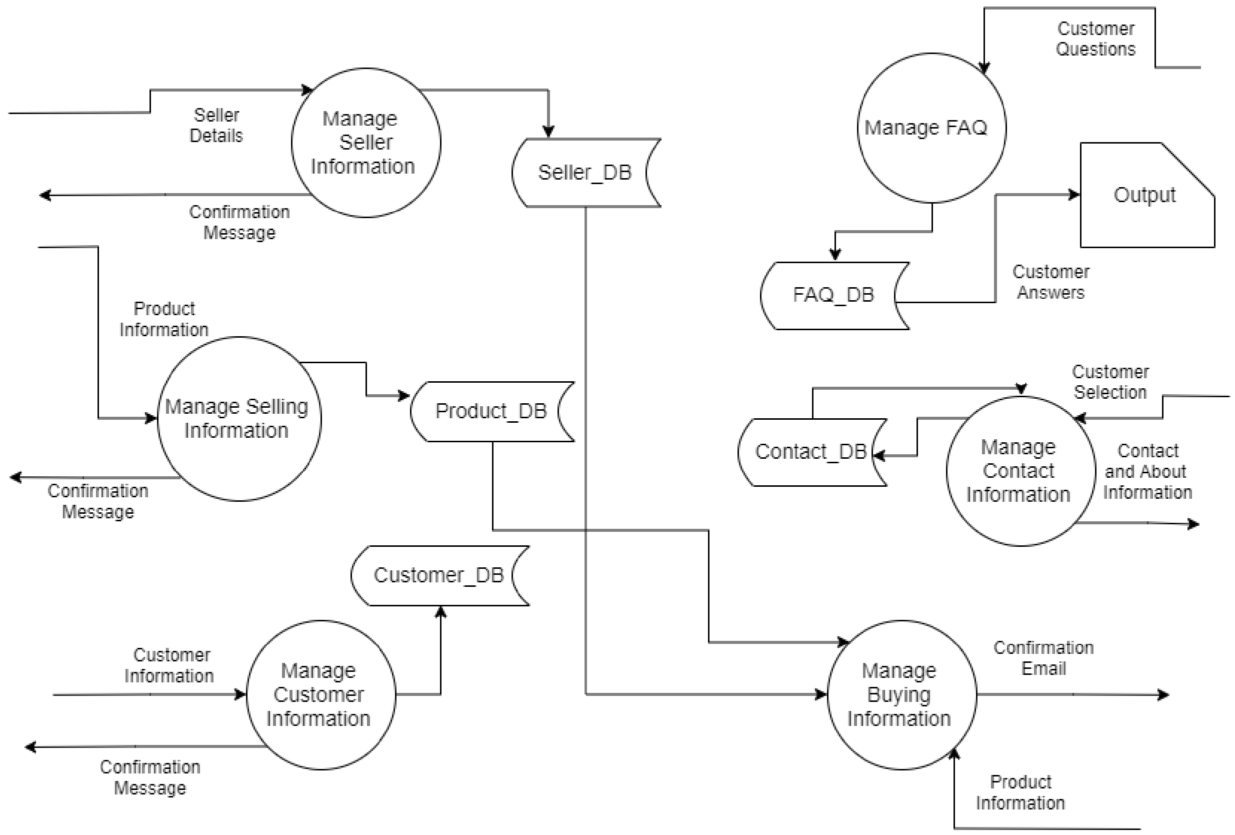
| Customer | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| Sr. No. | Filed Name | Data Type | Width | Required | Unique | PK/FK | Referenced Table | Description |
| | | | | | | | | |
| 1 | C_id | integer | 200 | Yes | Yes | PK | | AUTO INCREMENT |
| 2 | C_name | varchar | 200 | Yes | Yes | | | |
| 3 | C_phone_no | integer | 10 | Yes | Yes | | | |
| 4 | C_pass | varchar | 200 | Yes | Yes | | | |
| 5 | C_email | varchar | 50 | Yes | Yes | | | |
| 6 | C_photo | varchar | 2000 | No | No | | | |
| 7 | C_address | varchar | 1000 | Yes | Yes | | | |
| Product | | | | | | | | |
| Sr. No. | Filed Name | Data Type | Width | Required | Unique | PK/FK | Referenced Table | Description |
| | | | | | | | | |
| 1 | P_id | integer | 200 | Yes | Yes | PK | | AUTO INCREMENT |
| 2 | S_id | integer | 200 | Yes | No | FK | Seller | |
| 3 | P_price | float | 10 | Yes | No | | | |
| 4 | P_type | varchar | 200 | Yes | No | | | |
| 5 | P_image | varchar | 2000 | No | No | | | |
| 6 | P_description | varchar | 5000 | Yes | No | | | |
| 7 | P_fav | integer | 400 | No | No | | | |
| 8 | P_Name | Varchar | 50 | Yes | No | | | |
| Seller | | | | | | | | |
| Sr. No. | Filed Name | Data Type | Width | Required | Unique | PK/FK | Referenced Table | Description |
| | | | | | | | | |
| 1 | S_id | integer | 20 | Yes | Yes | PK | | AUTO INCREMENT |
| 2 | S_name | varchar | 200 | Yes | No | | | |
| 3 | S_Phone_No | integer | 10 | Yes | Yes | | | |
| | | | | | | | | |
| 4 | S_address | varchar | 1000 | Yes | Yes | | | |
| 5 | S_email | varchar | 200 | Yes | Yes | | | |
| 6 | S_pass | varchar | 200 | Yes | Yes | | | |

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| 7 | Latitude | double | 100 | No | No | | | |
| 8 | Longitude | double | 100 | No | No | | | |
| **Sales** | | | | | | | | |
| **Sr. No.** | **Filed Name** | **Data Type** | **Width** | **Required** | **Unique** | **PK/FK** | **Referenced Table** | **Description** |
| | | | | | | | | |
| 1 | Sales_id | integer | 200 | Yes | Yes | PK | | AUTO INCREMENT |
| 2 | C_id | integer | 200 | Yes | No | FK | Customer | |
| 3 | S_id | integer | 200 | Yes | No | FK | Seller | |
| 4 | P_id | integer | 200 | Yes | No | FK | Product | |
| **FAQ** | | | | | | | | |
| **Sr. No.** | **Filed Name** | **Data Type** | **Width** | **Required** | **Unique** | **PK/FK** | **Referenced Table** | **Description** |
| | | | | | | | | |
| 1 | C_id | integer | 200 | Yes | No | FK | Customer | |
| 2 | A and Q Data | | | No | No | | | Unstructured Data |

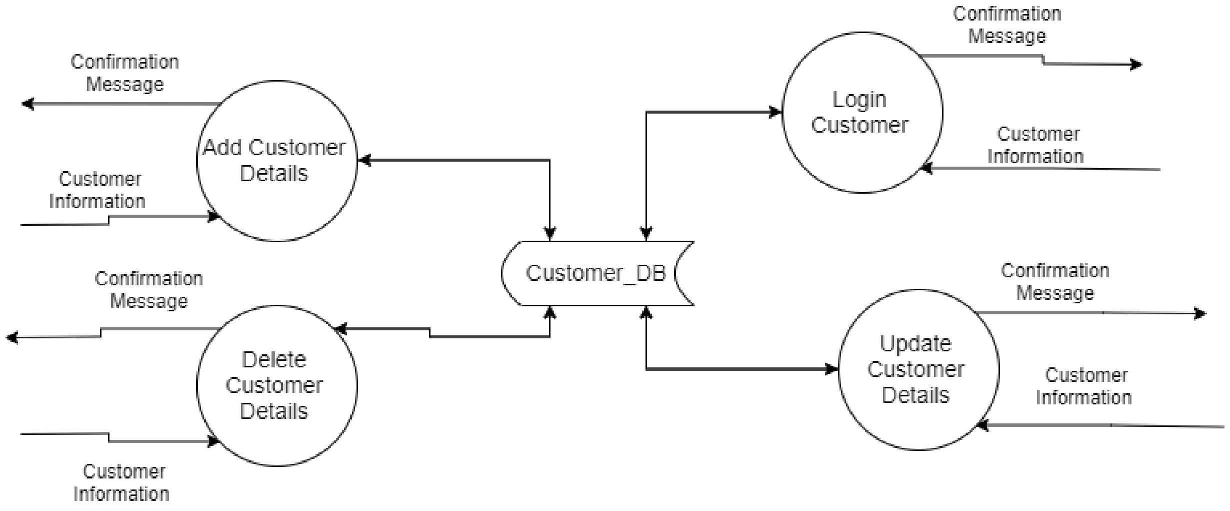
## 6. DFD:

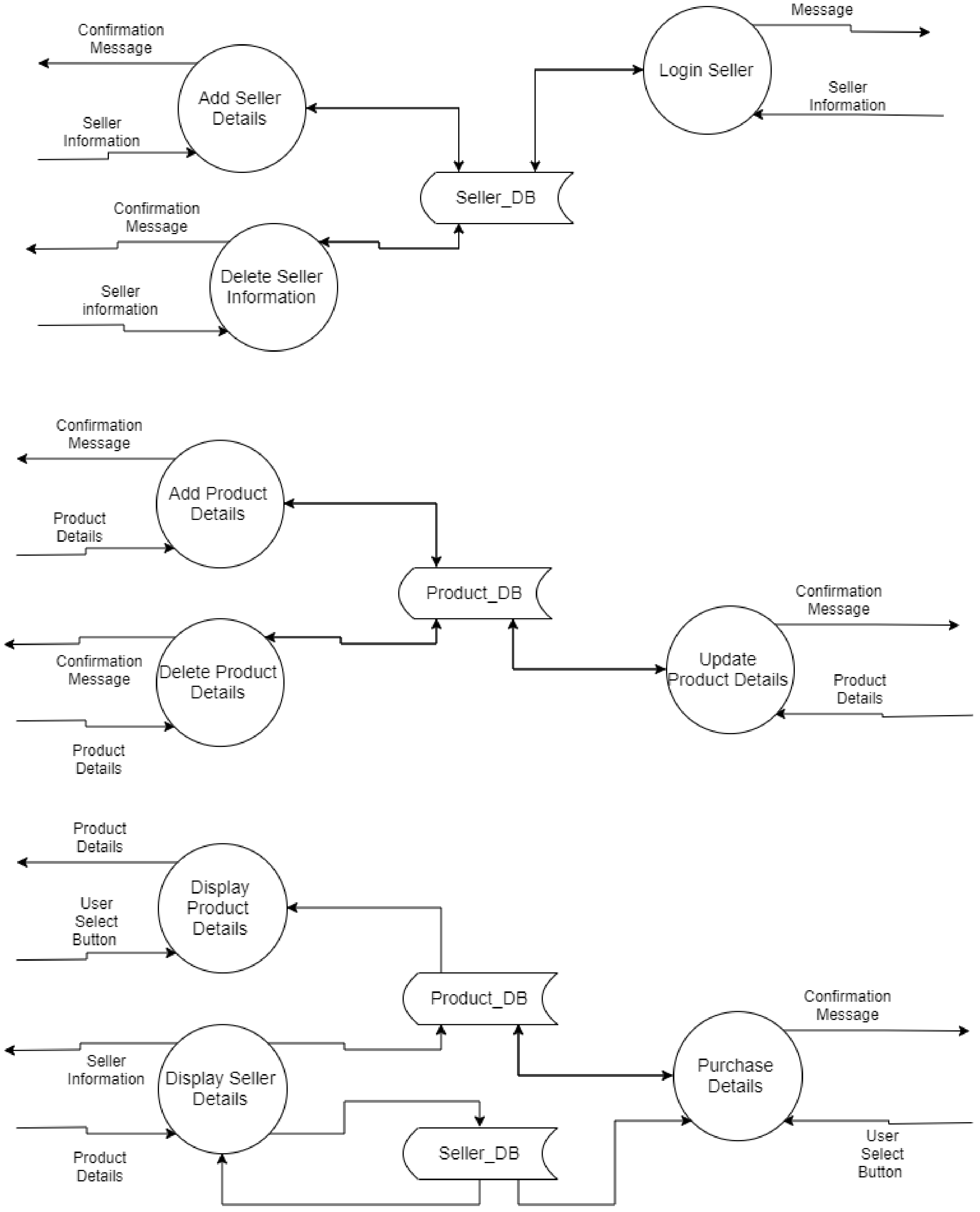


**Context Level Diagram**


Seller Details
Manage Seller Information
Confirmation Message
Seller_DB
Customer Questions
Manage FAQ
Output
FAQ_DB
Customer Answers
Product Information
Manage Selling Information
Product_DB
Confirmation Message
Customer Selection
Contact_DB
Manage Contact Information
Contact and About Information
Customer_DB
Customer Information
Manage Customer Information
Confirmation Message
Manage Buying Information
Confirmation Email
Product Information

**Level 1**


Confirmation Message
Add Customer Details
Customer Information
Login Customer
Confirmation Message
Customer Information
Customer_DB
Confirmation Message
Delete Customer Details
Customer Information
Update Customer Details
Confirmation Message
Customer Information


Confirmation Message
Add Seller Details
Seller Information
Login Seller
Confirmation Message
Seller Information
Seller_DB
Confirmation Message
Delete Seller Information
Seller information
Confirmation Message
Add Product Details
Product Details
Product_DB
Confirmation Message
Delete Product Details
Product Details
Update Product Details
Confirmation Message
Product Details
Product Details
Display Product Details
User Select Button
Product_DB
Seller Information
Display Seller Details
Product Details
Seller_DB
Purchase Details
Confirmation Message
User Select Button

**Level 2**

# Implementation Details

## Sign In And Sign Up Module:

We used Auth0 API for authentication and authorization and to maintain the session for the whole project. User can add his/her other information like location profile picture using live camera.

## Product Search Module:

We used pagination and filter searching for finding the right product. Filter like Price Range using range slider of Jquery-UI Slider, type of products like Books and Mobile phones. Search bar to search for the desired product name. Displaying all the products using bootstrap card. We used font awesome star icon for simulating the favorite's icon for a particular product. For pagination we used Jquery to change the product details to display on the

## Product Detail Module:

After user clicks on his/her desired product , we displayed product details along with the details about the seller along with his/her location using Google map and also provided sellers phone number so that customer can directly call the seller to discuss about the product and to purchase the same. We fetched every detail about the customer and the product details using node.js and mongoose module of node.js for fetching data from the Mongo DB Database. For fetching data the angular sends a request to the node and express server which processes the request and fetches the details from the database and send the data in form of json object back to the angular component.

## Index Module:

As the home page is unveiled, it starts off with a loading animation and after the loading is complete the animation of particles is visible along with navbar to go the desired location of the current page or any other page like login page or search page. The about us and contact us pages are in the single page in the index page. As user goes to then contact us section, he/she can contact us through any of the given contact methods like email, phone … etc. We have also provided the portfolio to give user the type of products we sell and user can go to the product detail page using this portfolio also.

## User Profile Module:

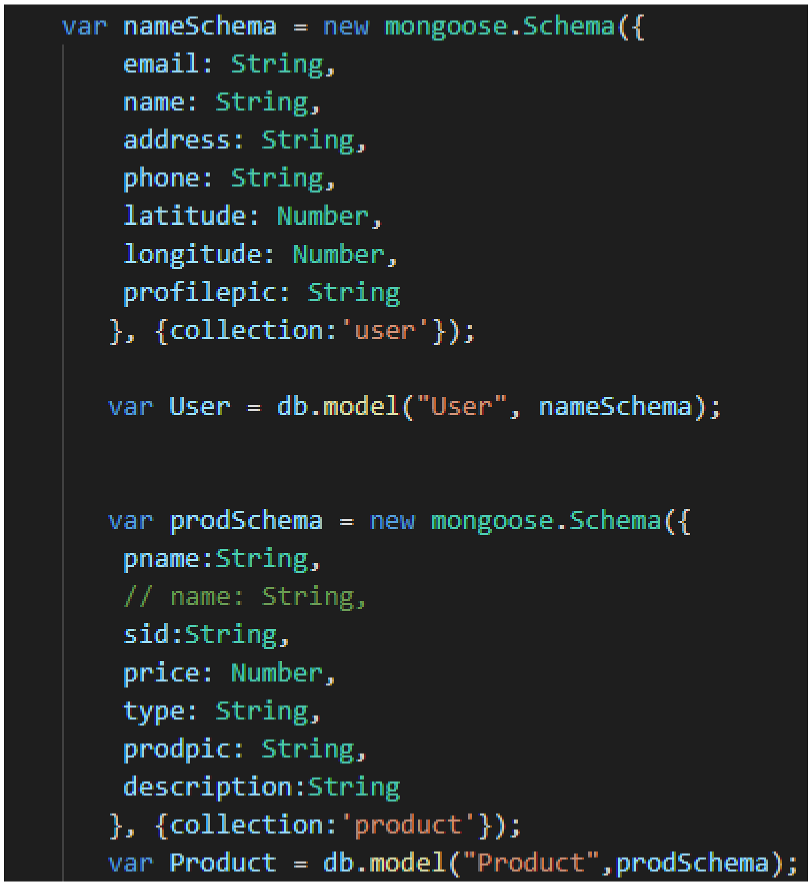
As soon as the user successfully creates his or her own profile he/she is able to go to the user profile page where the user can change his/her details like name, email and can also add his/her own image as a profile image using camera or from his his/her own local storage. Here user can add product so that seller can buy that product. User can also see all the product that he/she has added to the database.

## Admin Module:

We use auth0's admin page to manage users and assign roles to the users. We get all the statistics and data for all the users which have signed up to our website.

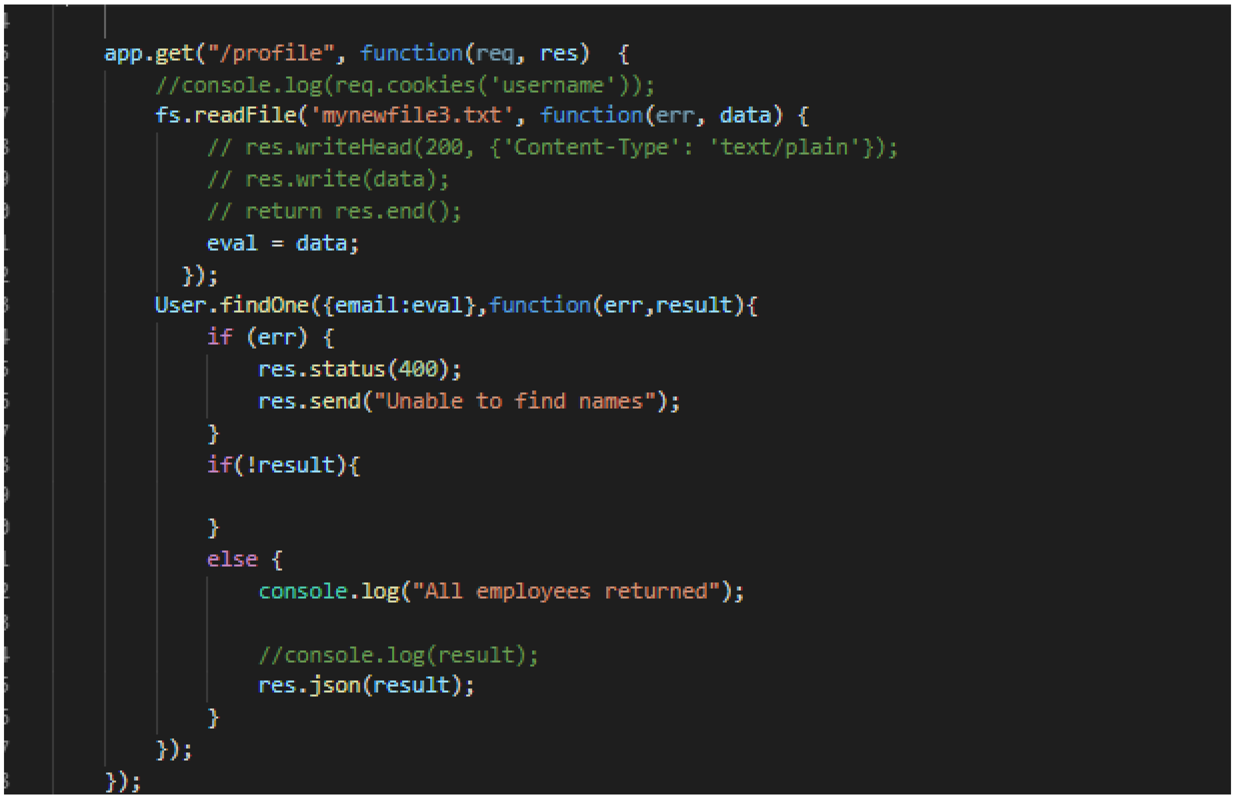
## Function Prototype:

### Database Schemas For MongoDB:



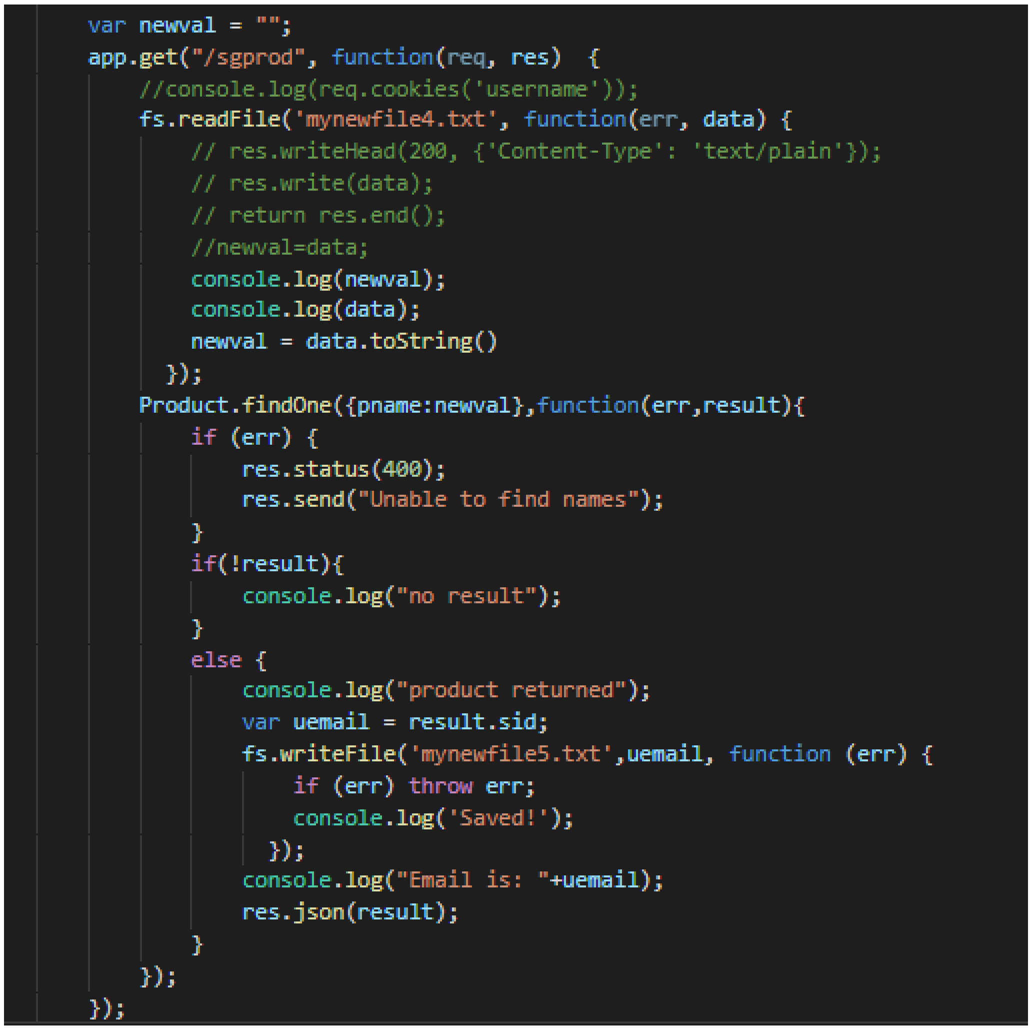
```
    var nameSchema = new mongoose.Schema({
        email: String,
        name: String,
        address: String,
        phone: String,
        latitude: Number,
        longitude: Number,
        profilepic: String
       }, {collection:'user'});

       var User = db.model("User", nameSchema);


       var prodSchema = new mongoose.Schema({
        pname:String,
        // name: String,
        sid:String,
        price: Number,
        type: String,
        prodpic: String,
        description:String
       }, {collection:'product'});
       var Product = db.model("Product",prodSchema);
```

## Getting User Profile Data:



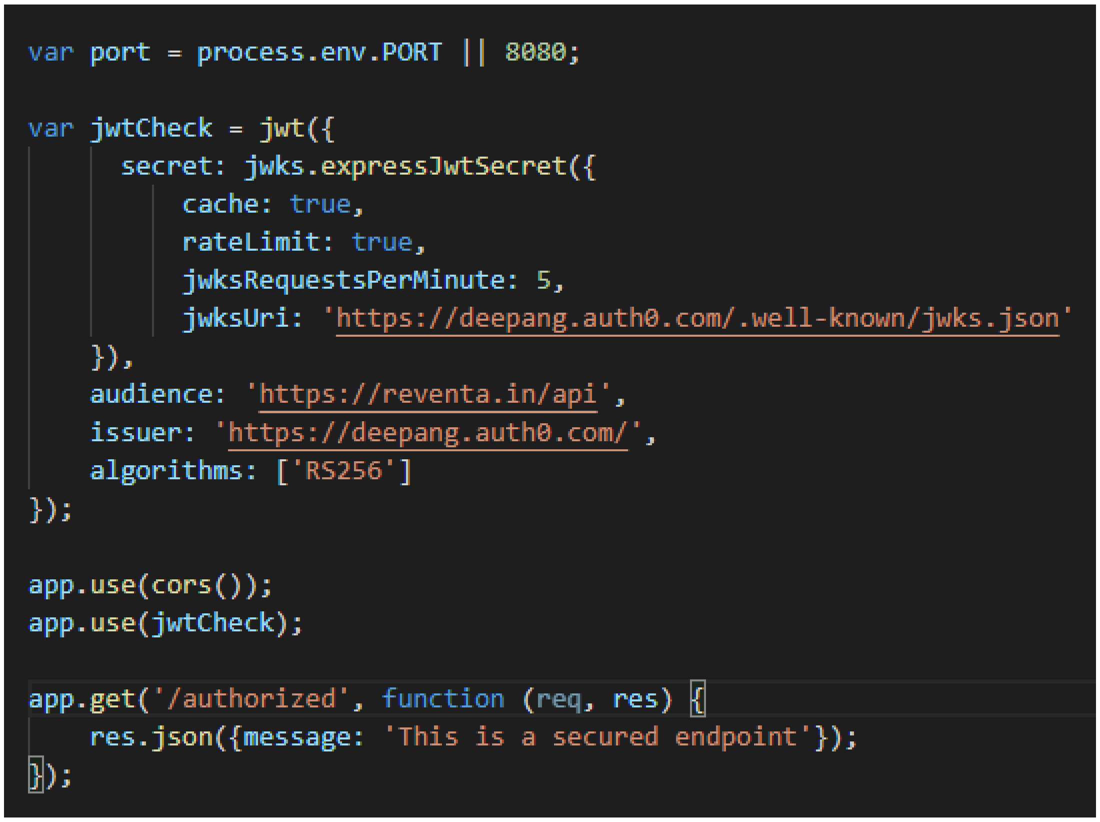
```
app.get("/profile", function(req, res)  {
    //console.log(req.cookies('username'));
    fs.readFile('mynewfile3.txt', function(err, data) {
        // res.writeHead(200, {'Content-Type': 'text/plain'});
        // res.write(data);
        // return res.end();
        eval = data;
      });
    User.findOne({email:eval},function(err,result){
        if (err) {
            res.status(400);
            res.send("Unable to find names");
        }
        if(!result){

        }
        else {
            console.log("All employees returned");

            //console.log(result);
            res.json(result);
        }
    });
});
```

## Getting Single Product Detail:



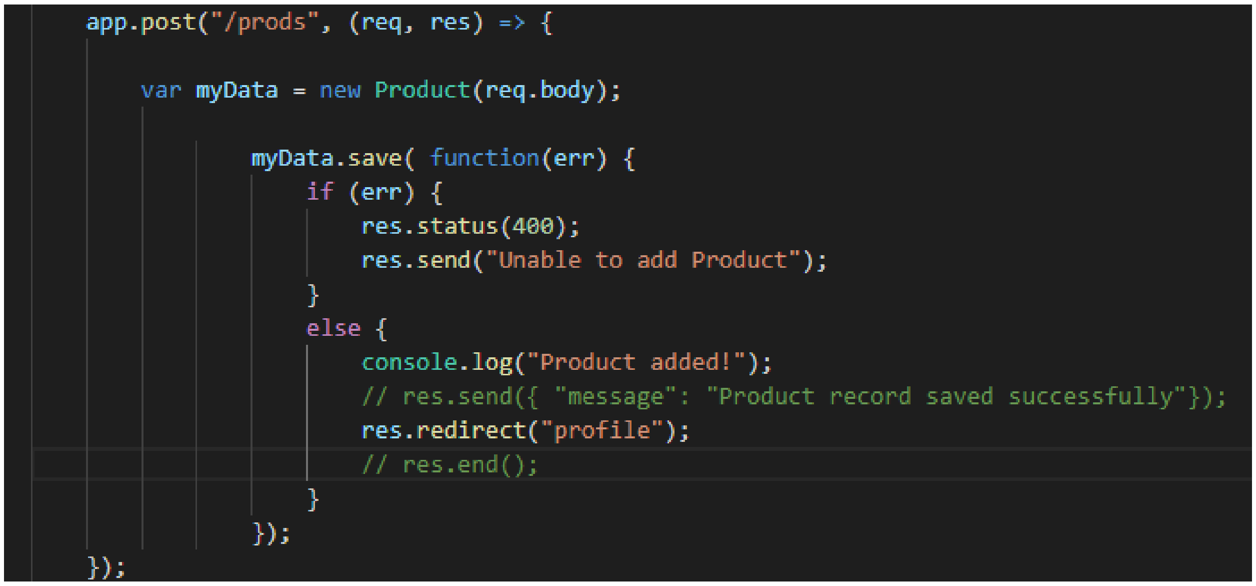
```
var newval = "";
app.get("/sgprod", function(req, res)  {
    //console.log(req.cookies('username'));
    fs.readFile('mynewfile4.txt', function(err, data) {
        // res.writeHead(200, {'Content-Type': 'text/plain'});
        // res.write(data);
        // return res.end();
        //newval=data;
        console.log(newval);
        console.log(data);
        newval = data.toString()
      });
    Product.findOne({pname:newval},function(err,result){
        if (err) {
            res.status(400);
            res.send("Unable to find names");
        }
        if(!result){
            console.log("no result");
        }
        else {
            console.log("product returned");
            var uemail = result.sid;
            fs.writeFile('mynewfile5.txt',uemail, function (err) {
                if (err) throw err;
                console.log('Saved!');
              });
            console.log("Email is: "+uemail);
            res.json(result);
        }
    });
});
```

## Authentication Cookie Generation:



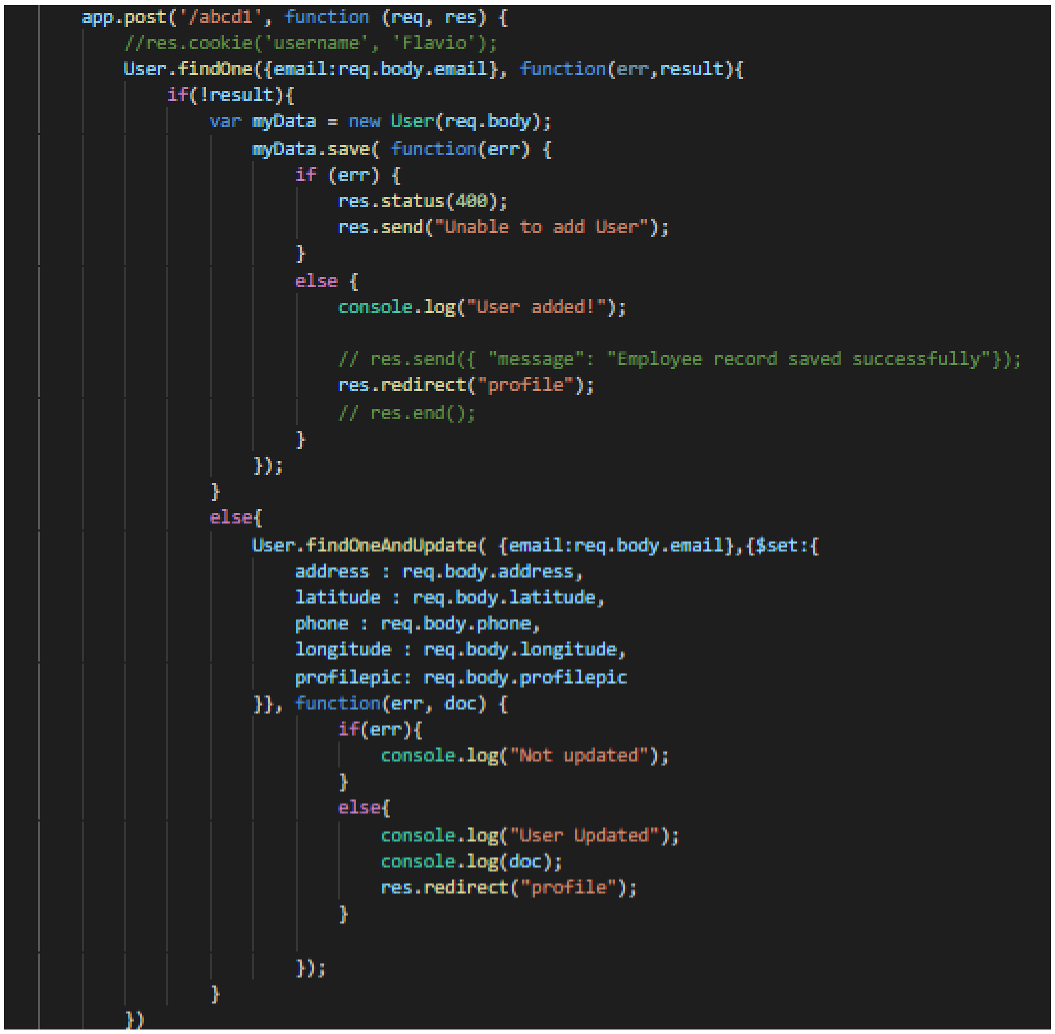
```
var port = process.env.PORT || 8080;

var jwtCheck = jwt({
      secret: jwks.expressJwtSecret({
          cache: true,
          rateLimit: true,
          jwksRequestsPerMinute: 5,
          jwksUri: 'https://deepang.auth0.com/.well-known/jwks.json'
    }),
    audience: 'https://reventa.in/api',
    issuer: 'https://deepang.auth0.com/',
    algorithms: ['RS256']
});

app.use(cors());
app.use(jwtCheck);

app.get('/authorized', function (req, res) {
    res.json({message: 'This is a secured endpoint'});
});
```

## Posting Product Details:



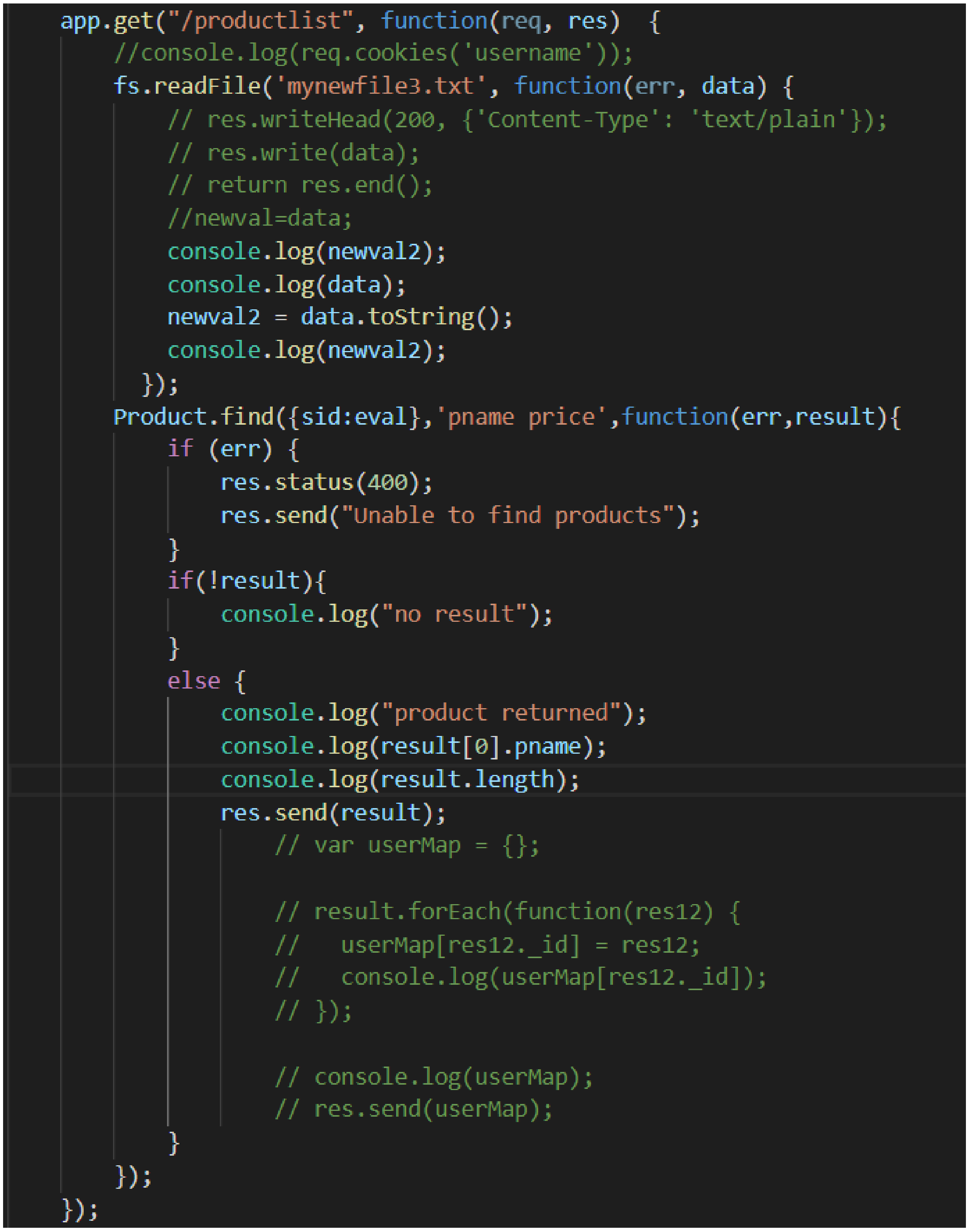
```
    app.post("/prods", (req, res) => {

        var myData = new Product(req.body);

                myData.save( function(err) {
                    if (err) {
                        res.status(400);
                        res.send("Unable to add Product");
                    }
                    else {
                        console.log("Product added!");
                        // res.send({ "message": "Product record saved successfully"});
                        res.redirect("profile");
                        // res.end();
                    }
                });
    });
```

## Updating User Profile:



```
app.post('/abcd1', function (req, res) {
    //res.cookie('username', 'Flavio');
    User.findOne({email:req.body.email}, function(err,result){
        if(!result){
            var myData = new User(req.body);
                myData.save( function(err) {
                    if (err) {
                        res.status(400);
                        res.send("Unable to add User");
                    }
                    else {
                        console.log("User added!");

                        // res.send({ "message": "Employee record saved successfully"});
                        res.redirect("profile");
                        // res.end();
                    }
                });
        }
        else{
            User.findOneAndUpdate( {email:req.body.email},{$set:{
                address : req.body.address,
                latitude : req.body.latitude,
                phone : req.body.phone,
                longitude : req.body.longitude,
                profilepic: req.body.profilepic
            }}, function(err, doc) {
                    if(err){
                        console.log("Not updated");
                    }
                    else{
                        console.log("User Updated");
                        console.log(doc);
                        res.redirect("profile");
                    }

                });
        }
    })
```

## Getting Seller Products:



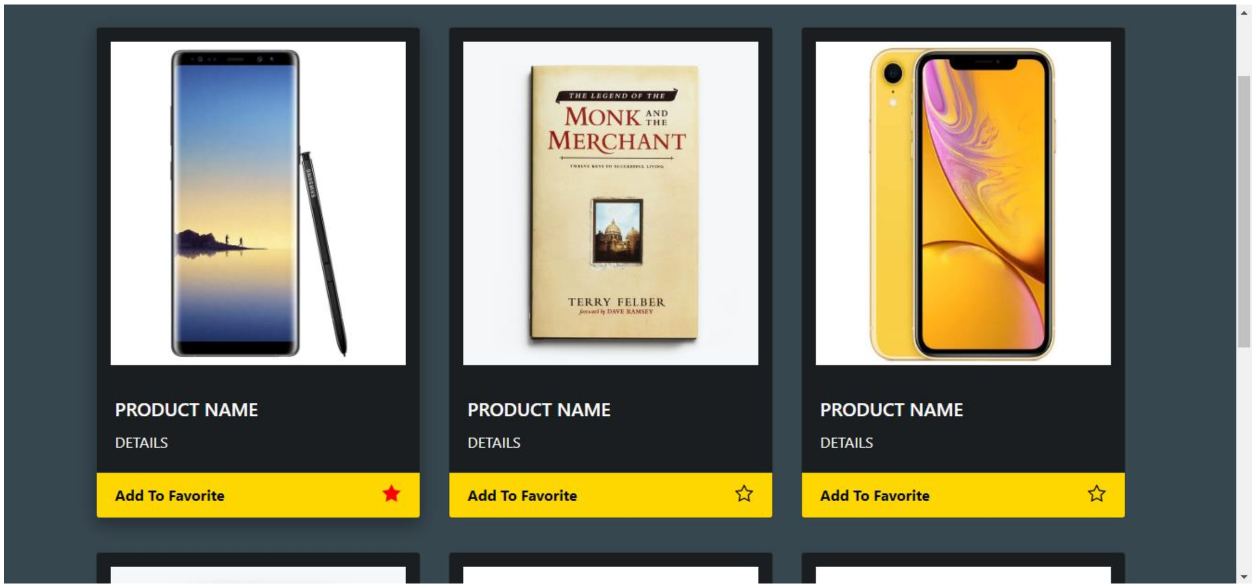
```
app.get("/productlist", function(req, res)  {
    //console.log(req.cookies('username'));
    fs.readFile('mynewfile3.txt', function(err, data) {
        // res.writeHead(200, {'Content-Type': 'text/plain'});
        // res.write(data);
        // return res.end();
        //newval=data;
        console.log(newval2);
        console.log(data);
        newval2 = data.toString();
        console.log(newval2);
      });
    Product.find({sid:eval},'pname price',function(err,result){
        if (err) {
            res.status(400);
            res.send("Unable to find products");
        }
        if(!result){
            console.log("no result");
        }
        else {
            console.log("product returned");
            console.log(result[0].pname);
            console.log(result.length);
            res.send(result);
                // var userMap = {};

                // result.forEach(function(res12) {
                //   userMap[res12._id] = res12;
                //   console.log(userMap[res12._id]);
                // });

                // console.log(userMap);
                // res.send(userMap);
        }
    });
});
```

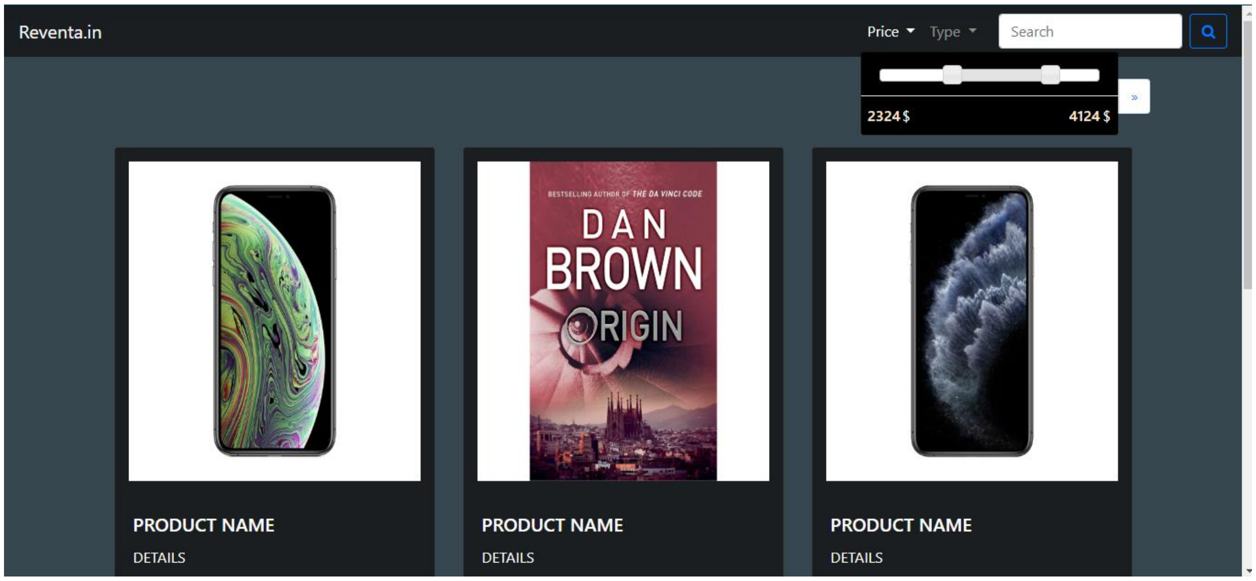
# Testing

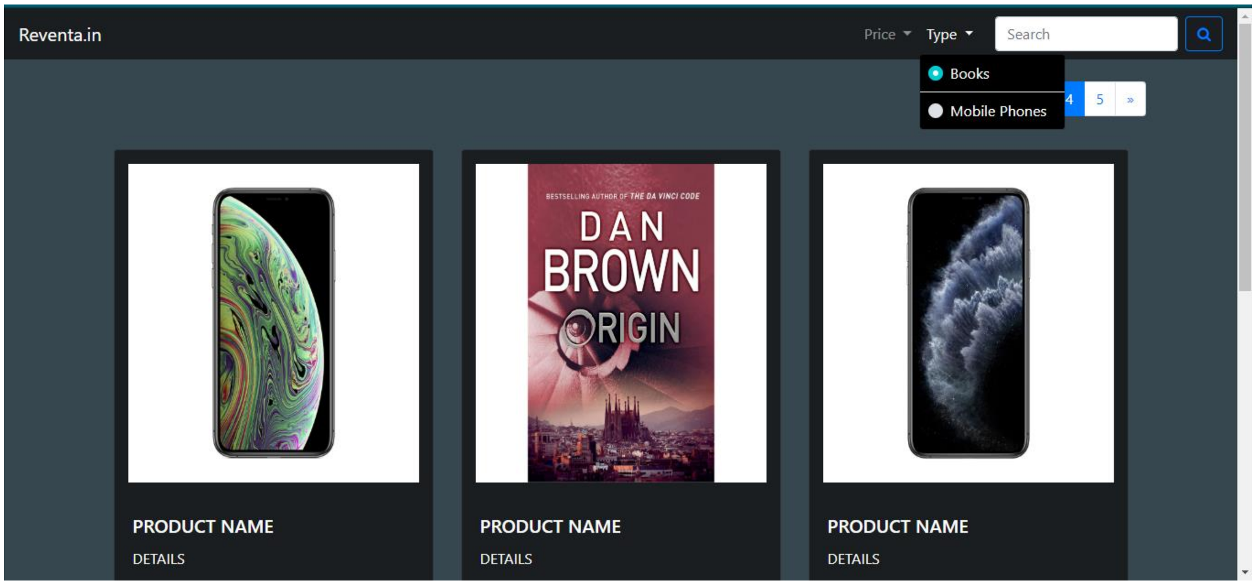
## Testing Methods:

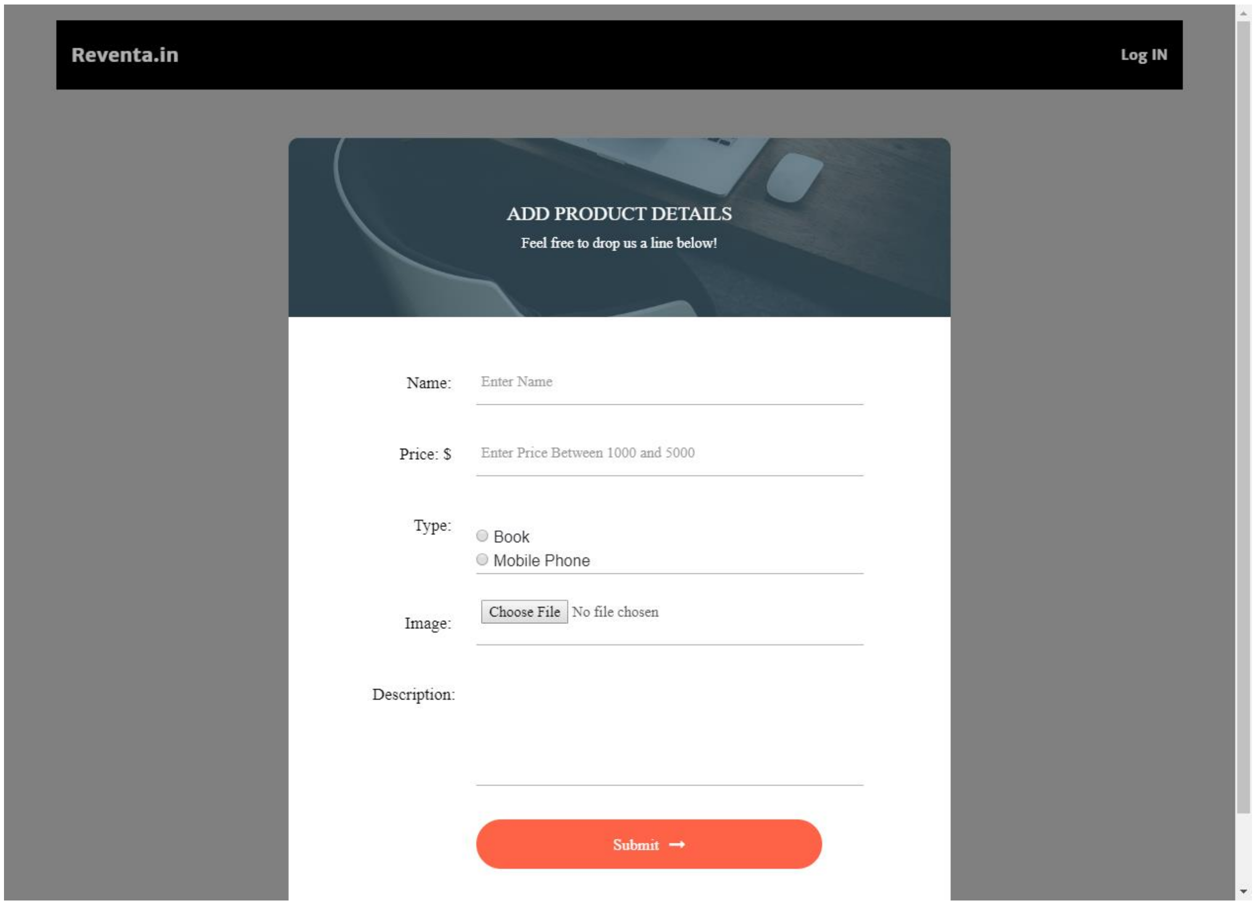
- We used curl “url” for node.js server for testing node server functionalities.

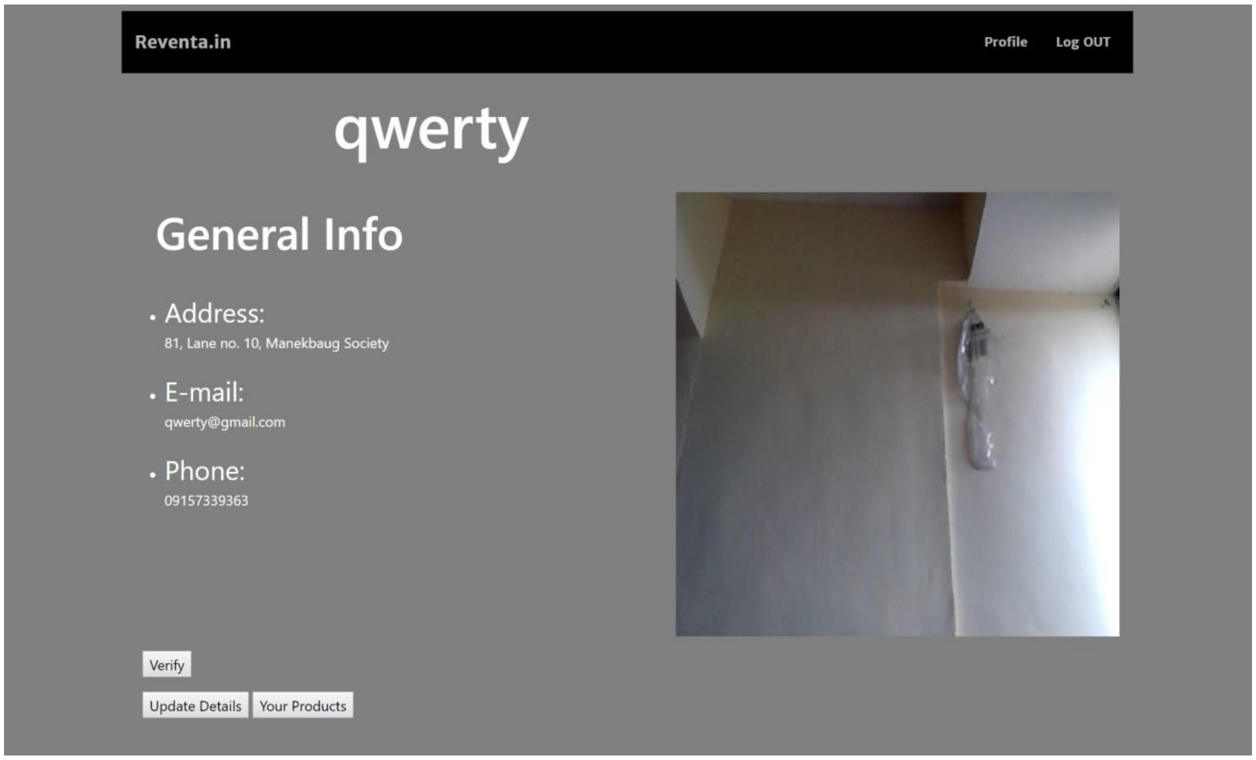
# Screen-Shots






Reventa.in
Price
Type
Search
Books
Mobile Phones
4
5
»
PRODUCT NAME
DETAILS
PRODUCT NAME
DETAILS
PRODUCT NAME
DETAILS

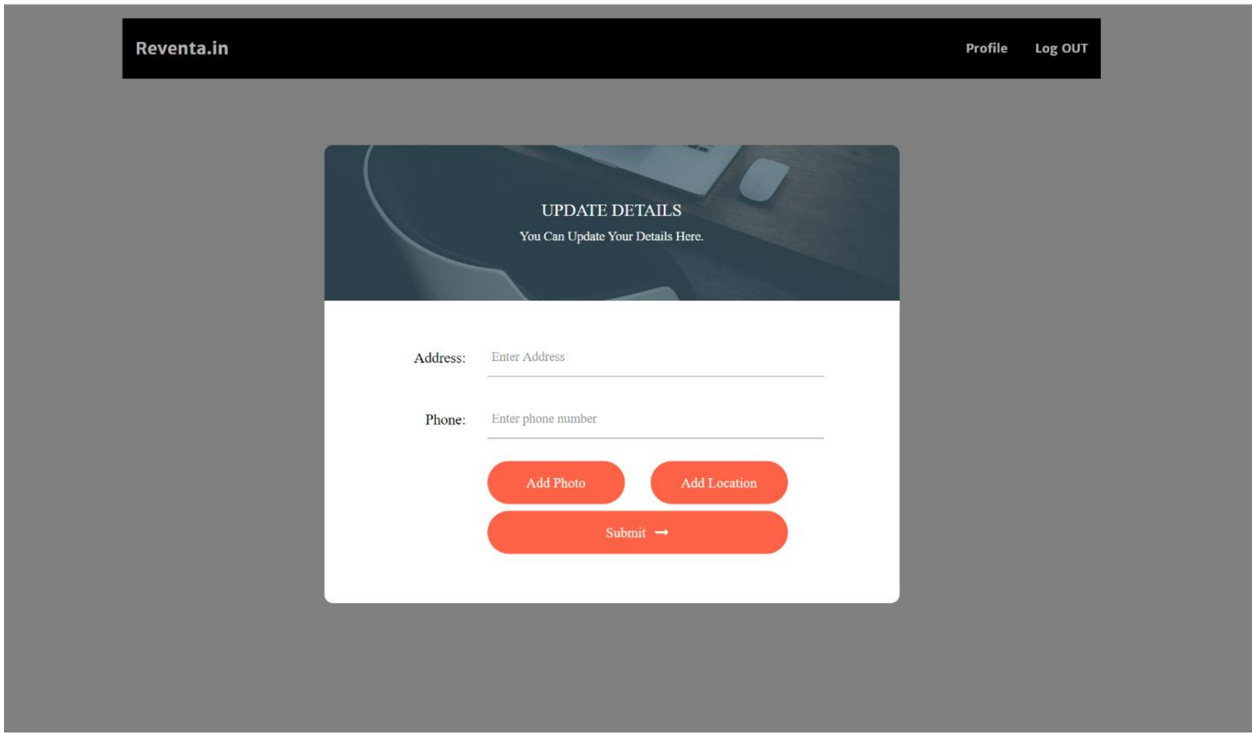

Reventa.in
Log IN
ADD PRODUCT DETAILS
Feel free to drop us a line below!
Name:
Enter Name
Price: $
Enter Price Between 1000 and 5000
Type:
Book
Mobile Phone
Image:
Choose File
No file chosen
Description:
Submit →



qwerty

General Info

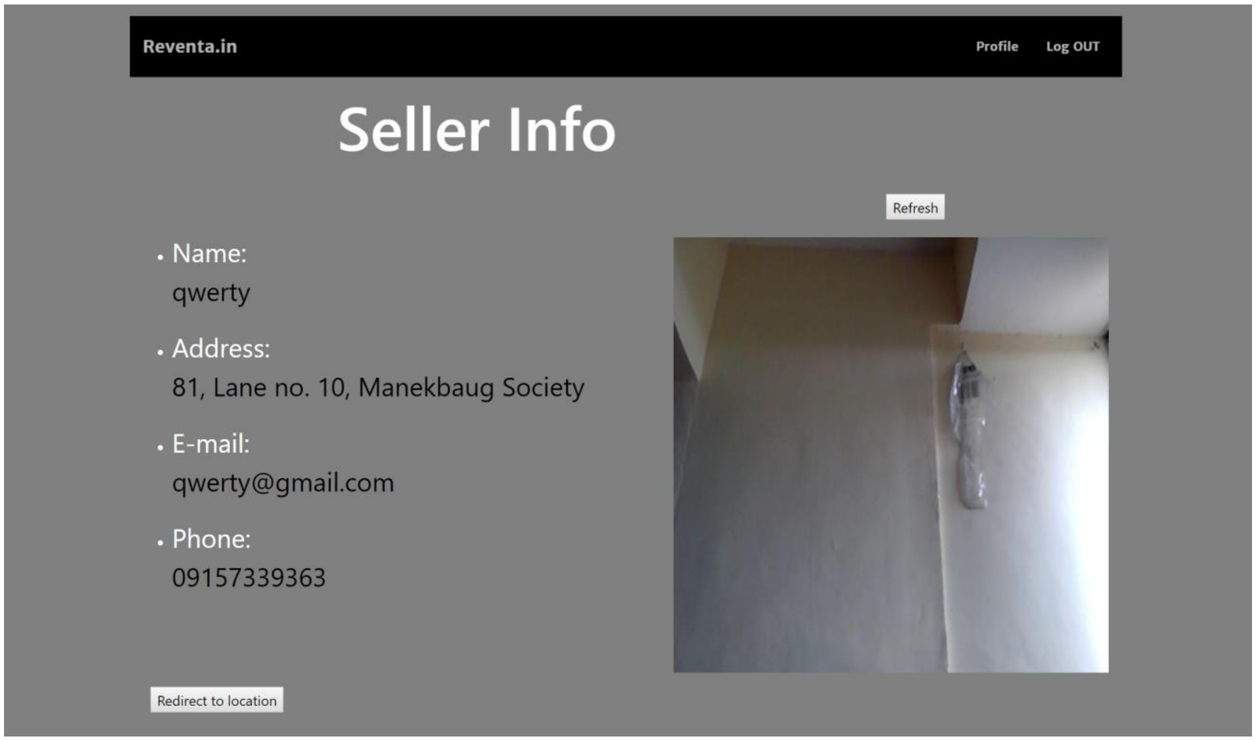
- Address:
  81, Lane no. 10, Manekbaug Society
- E-mail:
  qwerty@gmail.com
- Phone:
  09157339363



UPDATE DETAILS

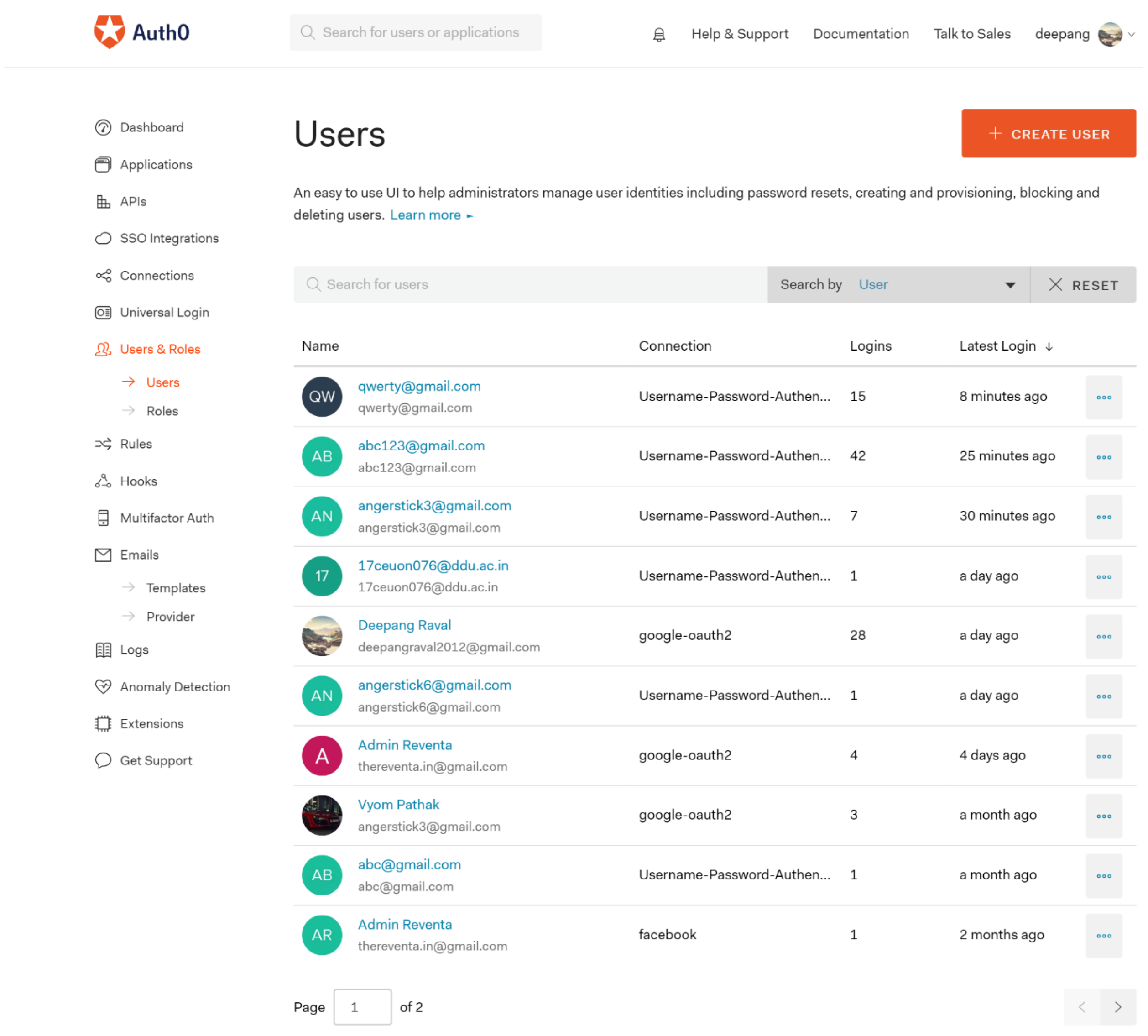
You Can Update Your Details Here.

| | |
|---|---|
| Address: | Enter Address |
| Phone: | Enter phone number |



Seller Info

- Name:
  qwerty
- Address:
  81, Lane no. 10, Manekbaug Society
- E-mail:
  qwerty@gmail.com
- Phone:
  09157339363

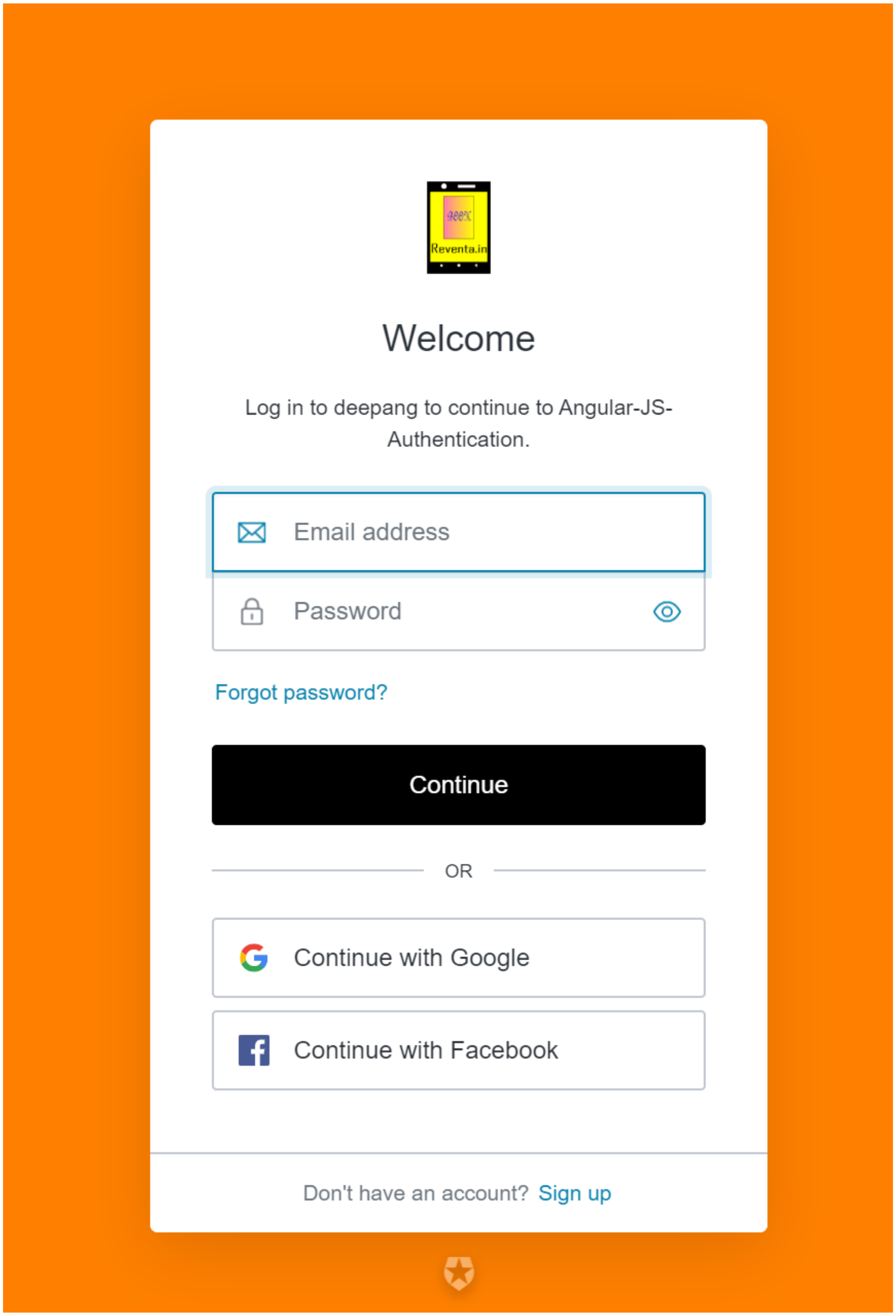


Users

An easy to use UI to help administrators manage user identities including password resets, creating and provisioning, blocking and deleting users. Learn more ▸

| Name | Connection | Logins | Latest Login ↓ |
|---|---|---|---|
| qwerty@gmail.com qwerty@gmail.com | Username-Password-Authen... | 15 | 8 minutes ago |
| abc123@gmail.com abc123@gmail.com | Username-Password-Authen... | 42 | 25 minutes ago |
| angerstick3@gmail.com angerstick3@gmail.com | Username-Password-Authen... | 7 | 30 minutes ago |
| 17ceuon076@ddu.ac.in 17ceuon076@ddu.ac.in | Username-Password-Authen... | 1 | a day ago |
| Deepang Raval deepangraval2012@gmail.com | google-oauth2 | 28 | a day ago |
| angerstick6@gmail.com angerstick6@gmail.com | Username-Password-Authen... | 1 | a day ago |
| Admin Reventa thereventa.in@gmail.com | google-oauth2 | 4 | 4 days ago |
| Vyom Pathak angerstick3@gmail.com | google-oauth2 | 3 | a month ago |
| abc@gmail.com abc@gmail.com | Username-Password-Authen... | 1 | a month ago |
| Admin Reventa thereventa.in@gmail.com | facebook | 1 | 2 months ago |

Page 1 of 2



Welcome

Log in to deepang to continue to Angular-JS-Authentication.

# Conclusion

Login functionality was successfully implemented by the use of Angular6’s Auth0 API. Home page animation, contact details was successfully implemented using css and Javascript. Product Search page’s pagination was successfully implemented using Jquery. Product filter UI was successfully implemented using Jquery slider for price filtering. Product details page was successfully implemented by fetching the details from the database and displayed on the browser using CSS and javascript. User’s location was successfully displayed using the Google Map. User Edit Profile page was successfully implemented for updating user’s details as well as adding user’s photo using the camera and canvas provided by css. Update User Profile details, Add Product Details was successfully implemented using post requests and mongoose RestAPI.

# Limitation and Future Extension

## Limitations:

- User needs to verify on the profile page to display his/her information.
- Product page is having static images so search functionality is not working.
- We have specified only 2 type of filters for search.

## Future Extension:

- Using the favorites tag we wish to extend this application to recommend users with the type of product which are similar to their favorite products using advanced algorithms.
- We would like to make this website more robust so that user can search for sellers in their vicinity by providing advanced map functionality.

# Bibliography

**W3School:**

- **HTML5:** https://www.w3schools.com/html/default.asp
- **CSS:** https://www.w3schools.com/css/default.adp
- **JavaScript:** https://www.w3schools.com/js/default.asp
- **AJAX:** https://www.w3schools.com/js/js_ajax_intro.asp
- **Jquery:** https://www.w3schools.com/jquery/default.asp
- **XML:** https://www.w3schools.com/xml/default.asp
- **XSLT:** https://www.w3schools.com/xml/xsl_intro.asp
- **XSD:** https://www.w3schools.com/xml/schema_intro.asp
- **JSON:** https://www.w3schools.com/js/js_json_intro.asp

**Bootstrap4:**

https://getbootstrap.com/docs/4.3/getting-started/introduction/

**Icons:**

https://fontawesome.com/icons?d=gallery

**Fonts:**

https://fonts.google.com/

**Express.js:**

https://expressjs.com/en/api.html

**Node.js:**

https://nodejs.org/en/

**Mongo DB:**

https://www.mongodb.com/

**Auth0 API:**

https://auth0.com/

**Other References:**

https://stackoverflow.com/
https://www.youtube.com/
https://codepen.io/