

EBS

EASY BUY/ SELL

Team: 6

Vidya Damalacheruvu
Karthik Yanagandula
Gayathri Garikapati
Pujitha Talluri

Story of the project:

We make it possible for millions of people around the country to buy and sell practically anything. We all have items that we never use, haven't used in a long time, or have just outgrown. But, because these items still have worth, why not sell them, or donate them to people in need? We've proposed a project called Easy Buy and Sell, in which a client may sell and buy things. Based on the items mentioned, we will categorize the products. This project makes it much easy to trade stuff than any other. We are also attempting to incorporate split delivery and payment choices in addition to these features.

Technologies used:

We have used MEAN Stack technologies such as MongoDB, Express, Angular, NodeJS for building this Ecommerce website.

Front-end: HTML, CSS, Bootstrap, Angular

Back-end: NodeJS, Express

Database: MongoDB Cloud

APIs Used: Algolia Search, Stripe Payment

Data details:

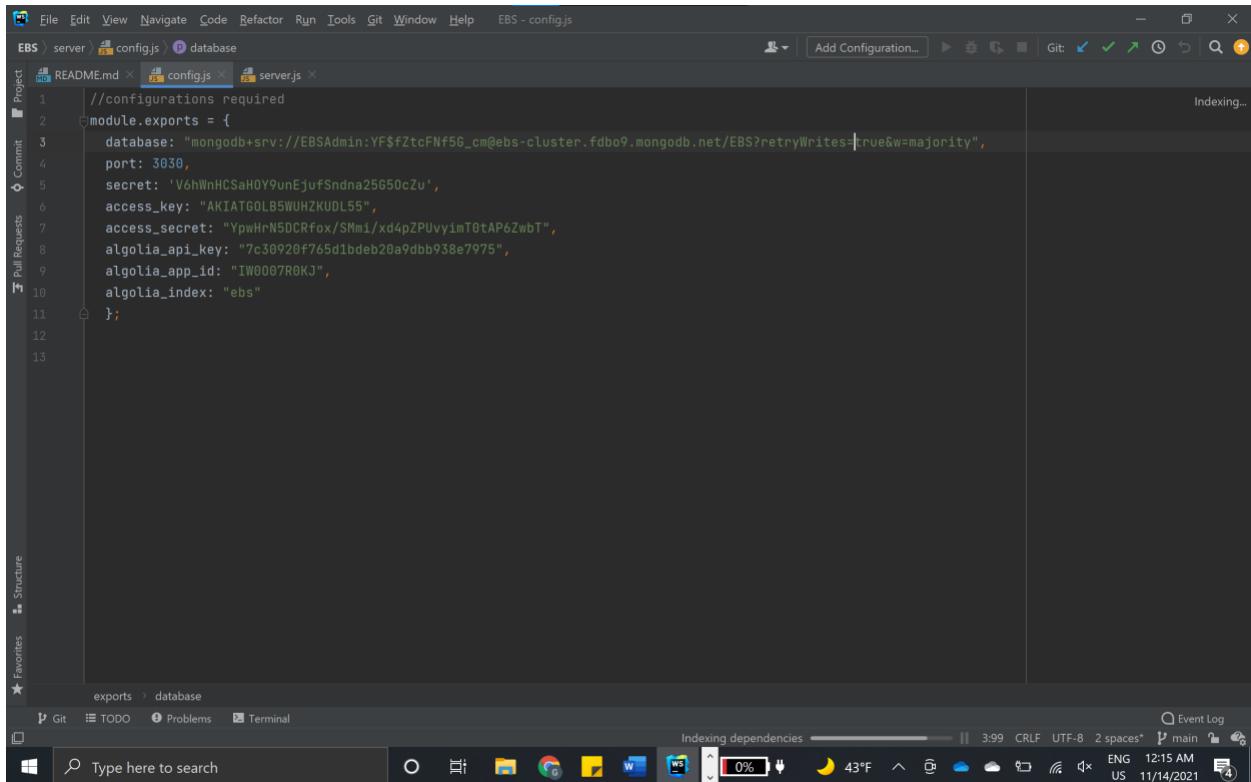
For any Ecommerce website we need,

1. User data
2. Product data
3. Categories data
4. Orders data
5. Reviews data

For this project, we are using MongoDB hosted on cloud and stores data. Stored the product images in AWS S3 bucket.

Algolia Search API is used to incorporate the product's search results into the project. To access this API, we need to have the API key and API id.

The Algolia API key and id, access key and secret for accessing AWS, MongoDB database URL and port are provided in the config.js file as shown below.



```
File Edit View Navigate Code Refactor Run Tools Git Window Help EBS - config.js
EBS > server > config.js database
Project README.md config.js server.js
1 //configurations required
2 module.exports = {
3   database: "mongodb+srv://EBSAdmin:YF$fZtcFNf5G_cm@ebs-cluster.firebaseio.net/EBS?retryWrites=true&w=majority",
4   port: 3030,
5   secret: 'V6hWnHCSaHOY9unEjufSndna25G50cZu',
6   access_key: "AKIATGOLB5WUHZKUDL55",
7   access_secret: "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",
8   algolia_api_key: "7c30920f765d1bdeb20a9dbb938e7975",
9   algolia_app_id: "IW0007R0KJ",
10  algolia_index: "ebs"
11};
12
13
```

The screenshot shows a code editor window with the title "EBS - config.js". The file content is as follows:

```
File Edit View Navigate Code Refactor Run Tools Git Window Help EBS - config.js
EBS > server > config.js database
Project README.md config.js server.js
1 //configurations required
2 module.exports = {
3   database: "mongodb+srv://EBSAdmin:YF$fZtcFNf5G_cm@ebs-cluster.firebaseio.net/EBS?retryWrites=true&w=majority",
4   port: 3030,
5   secret: 'V6hWnHCSaHOY9unEjufSndna25G50cZu',
6   access_key: "AKIATGOLB5WUHZKUDL55",
7   access_secret: "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",
8   algolia_api_key: "7c30920f765d1bdeb20a9dbb938e7975",
9   algolia_app_id: "IW0007R0KJ",
10  algolia_index: "ebs"
11};
12
13
```

The code defines an exports object with properties for database URL, port, secret, access key, access secret, Algolia API key, Algolia app ID, and Algolia index.

We are using Stripe API for payment options which facilitates the option to accept the online payments across the world.

All the data corresponding to categories, products, users, orders, and reviews will be stored in MongoDB cloud.

Categories: With schema containing the details of Id, created date, category name.

The screenshot shows the MongoDB Atlas interface. On the left, the sidebar includes sections for Deployment, Databases (Data Lake), Data Services (Triggers, Data API PREVIEW), and Security (Database Access, Network Access, Advanced). The main area displays the EBS database with the categories collection selected. The collection size is 205B, containing 3 documents with a total index size of 72KB. The query results show two documents:

```
_id: ObjectId("619018dadb303981abb38fe6")
created: 2021-11-13T19:58:18.707+00:00
name: "electronics"
__v: 0

_id: ObjectId("619018e6cb303981abb38fea")
created: 2021-11-13T19:58:30.083+00:00
name: "clothing"
__v: 0
```

Products:

With products details containing schema contents as id, reviews, creates, owner, category, title, price, description, image.

The screenshot shows the MongoDB Atlas interface. The sidebar is identical to the previous one. The main area displays the EBS database with the products collection selected. The collection size is 8.18KB, containing 12 documents with a total index size of 36KB. The query results show one document:

```
_id: ObjectId("6190185dcdb303981abb38ff3")
reviews: Array
created: 2021-11-13T20:00:29.021+00:00
owner: ObjectId("61901814cb303981abb38fe6")
category: ObjectId("619018dadb303981abb38fe6")
title: "MacBook Air 13.3" Laptop - Apple M1 chip - 8GB Memory - 512GB SSD (Lat...)"
price: 1199
description: "Apple's thinnest and lightest notebook gets supercharged with the Appl..."
image: "https://ebsproducts.s3.amazonaws.com/1636833628136"
__v: 2
```

Users: Schema contains id, isSeller, created, name, email, password, picture, address.

The screenshot shows the MongoDB Atlas interface with the 'EBS' project selected. In the left sidebar, under 'Databases', 'EBS' is expanded, and 'users' is selected. The main panel displays the 'EBS.users' collection with the following details:

- Collection Size: 1.9KB
- Total Documents: 5
- Indexes Total Size: 72KB

The 'Find' tab is active, showing a query result for document 1 of 5. The document structure is as follows:

```
_id: ObjectId("61901814cb303981abb38ffd")
isSeller: true
created: 2021-11-13T19:55:00.114+00:00
name: "Karthik Venagandula"
email: "kav@gmail.com"
password: "$2a$10$Uk01GCUKHP0cJT0Y/YGu42RH7CgvdhcVdI02adctqOIp5ZtvyH"
picture: "https://gravatar.com/avatar/eddcf8627fce4c718993b35645cbfba5?size=200&d=r..."
__v: 0
address: Object
```

Orders: Schema contains id, total price, products, owner.

The screenshot shows the MongoDB Atlas interface with the 'EBS' project selected. In the left sidebar, under 'Databases', 'EBS' is expanded, and 'orders' is selected. The main panel displays the 'EBS.orders' collection with the following details:

- Collection Size: 1.86KB
- Total Documents: 13
- Indexes Total Size: 36KB

The 'Find' tab is active, showing a query result for document 1 of 13. The document structure is as follows:

```
_id: ObjectId("61b6da8827b3f39610cb0733")
totalPrice: 749
products: Array
owner: ObjectId("61901814cb303981abb38ffd")
__v: 0
```

Another document is partially visible below it:

```
_id: ObjectId("61b6da8b27b3f39610cb0751")
totalPrice: 1199
products: Array
owner: ObjectId("61901efccb303981abb39022")
```

Reviews: Schema contains id, rating, created, owner, title, description.

The screenshot shows the MongoDB Atlas interface. On the left, the sidebar has sections for Project 0, DEPLOYMENT, Databases (selected), DATA SERVICES, SECURITY, and more. Under Databases, the EBS database is selected, showing namespaces like categories, orders, products, reviews, and users. The main panel displays the EBS.reviews collection with a collection size of 6.31KB, 15 documents, and 36KB indexes. A query results table shows two documents. The first document is:

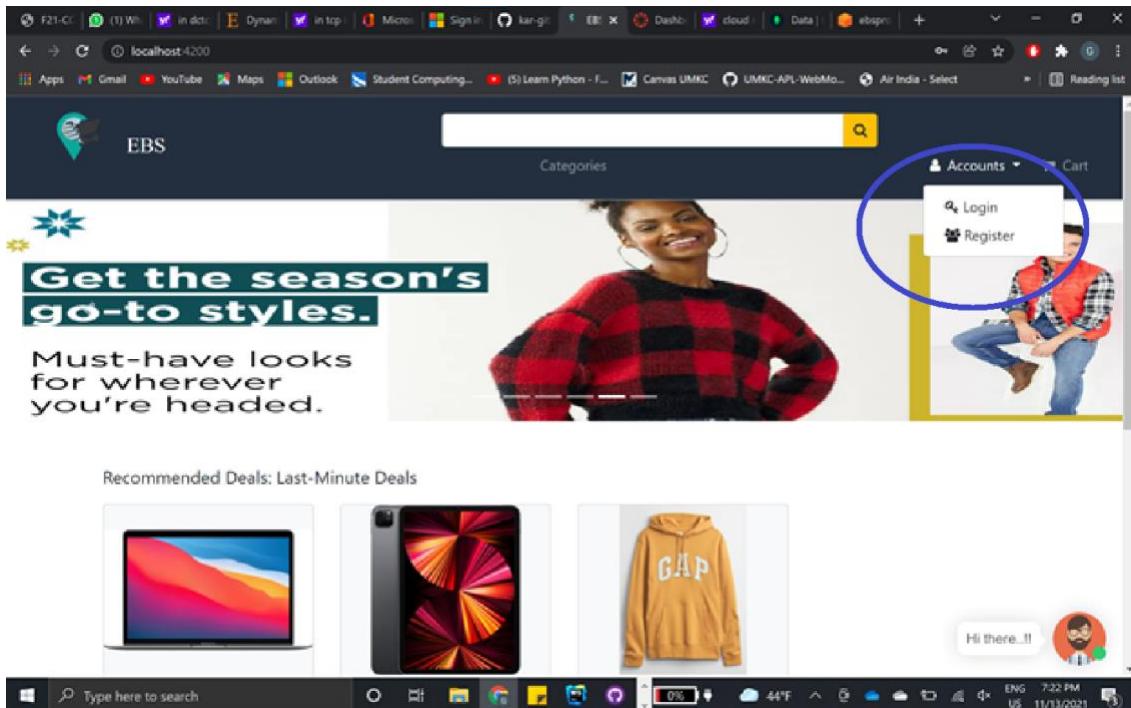
```
_id: ObjectId("61b6ea75ab6b091d84f8a0ab")
rating: 5
created: 2021-12-13T06:38:45.968+00:00
owner: ObjectId("61901e8dcbb30981abb39014")
title: "Great upgrade for a laptop!"
description: "I love how powerful this macbook is. The sound is crisp and the screen..."
__v: 0
```

The second document is similar, with the same schema and values.

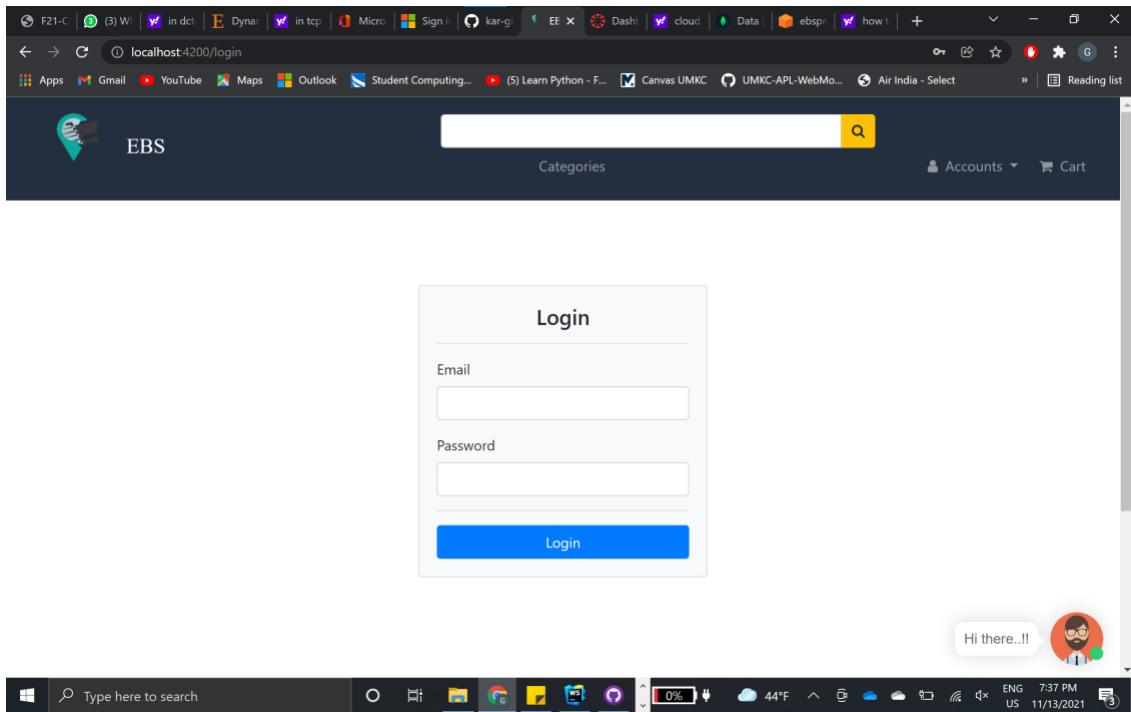
Working screens from project: Home screen of EBS application.

The screenshot shows the EBS application's home screen. At the top, there is a navigation bar with links for F21-CC, (1) Wh, in dict, Dynam, in tcp, Micros, Sign in, kar-git, EBS, Dashb, cloud, Data, ebspro, and a search bar. Below the navigation bar is a header with the EBS logo, a search bar, and account/cart icons. The main content area features a large image of a Ring Stick Up Cam Battery with the text "Smart security, inside or out". Below this, there is a section titled "Recommended Deals: Last-Minute Deals" featuring images of a tablet, a smartphone, and a yellow hoodie. In the bottom right corner, there is a friendly AI bot with the message "Hi there..!!". The taskbar at the bottom shows various open applications and system status.

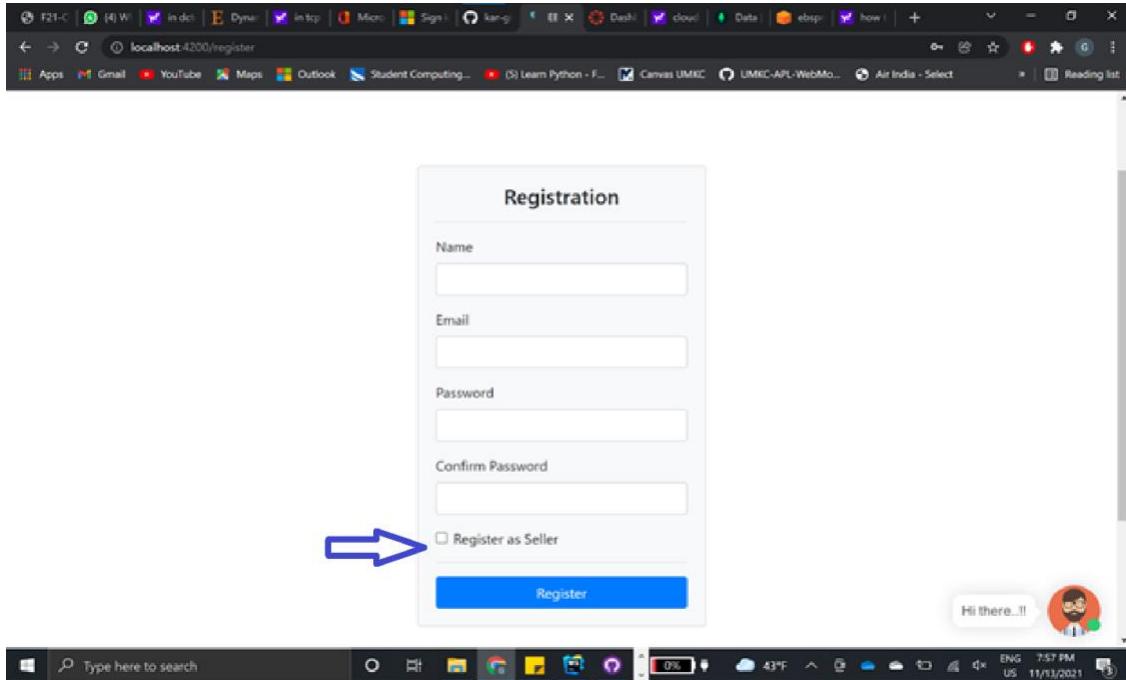
Here, we have two options under accounts i.e., Login and Register.



If it's an existing user, they can just click on login and provide the Email and password to access the application.



If the user is new to the application, they can click on register tab and provide the details such as Name, Email, password. Here, we have an option to register as a seller if the user want's their products under sale.



All this user data is stored in MongoDB database under EBS-Cluster.

Collection Name	Documents	Documents Size	Documents Avg	Indexes	Index Size	Index Avg
categories	3	205B	69B	2	72KB	36KB
products	3	1.93KB	659B	1	36KB	36KB
users	5	1.48KB	304B	2	72KB	36KB

Below screen displays the schema for the user details.

The screenshot shows the MongoDB Atlas interface for the EBS database. On the left, the deployment sidebar is visible with sections for DEPLOYMENT, DATABASES (EBS selected), and SECURITY. The main area displays the EBS.users collection with the following details:

- Collection Size: 1.48KB
- Total Documents: 5
- Indexes Total Size: 72KB

There are tabs for Find, Indexes, Schema Anti-Patterns, Aggregation, and Search Indexes. A large button labeled "INSERT DOCUMENT" is at the top right. Below it is a search bar with a filter condition: { field: 'value' } and buttons for OPTIONS, Apply, and Reset. The results section shows 1-5 of 5 documents, with one document expanded:

```
_id: ObjectId("61901e814cb303981abb38fdd")
isSeller: true
created: 2021-11-13T19:55:00.114+00:00
name: "Karthik Yanagandula"
email: "kaya@gmail.com"
password: "$2a$10$Uk3JGGuXH3Kp0cJT0yY/Gu42RM7CgvdhcvWlI02qdctqDp5ZtvyHC"
picture: "https://gravatar.com/avatar/eddcf8627fce4c710993b35645cbfa57s200&d=re..."
__v: 0
```

Below screen display the options for a user registered as a seller, he/she will have an option such as to post a product for sale and to view the products he has put for sale under my products tab.

The screenshot shows a web application titled "My Profile". At the top, there is a circular profile picture with a green and yellow pixelated pattern. To the right of the picture, the name "Karthik Yanagandula" is displayed. Below the name are three yellow rectangular buttons with white text: "My Orders", "Change Account Settings", and "Change Shipping Address". Underneath these buttons, the section "Seller Actions" is shown with two buttons: "Post Product for Sale" and "My Products". In the bottom right corner, there is a small circular icon with a cartoon character's face and the text "Hi there..!!". The browser taskbar at the bottom shows various open tabs and system status indicators.

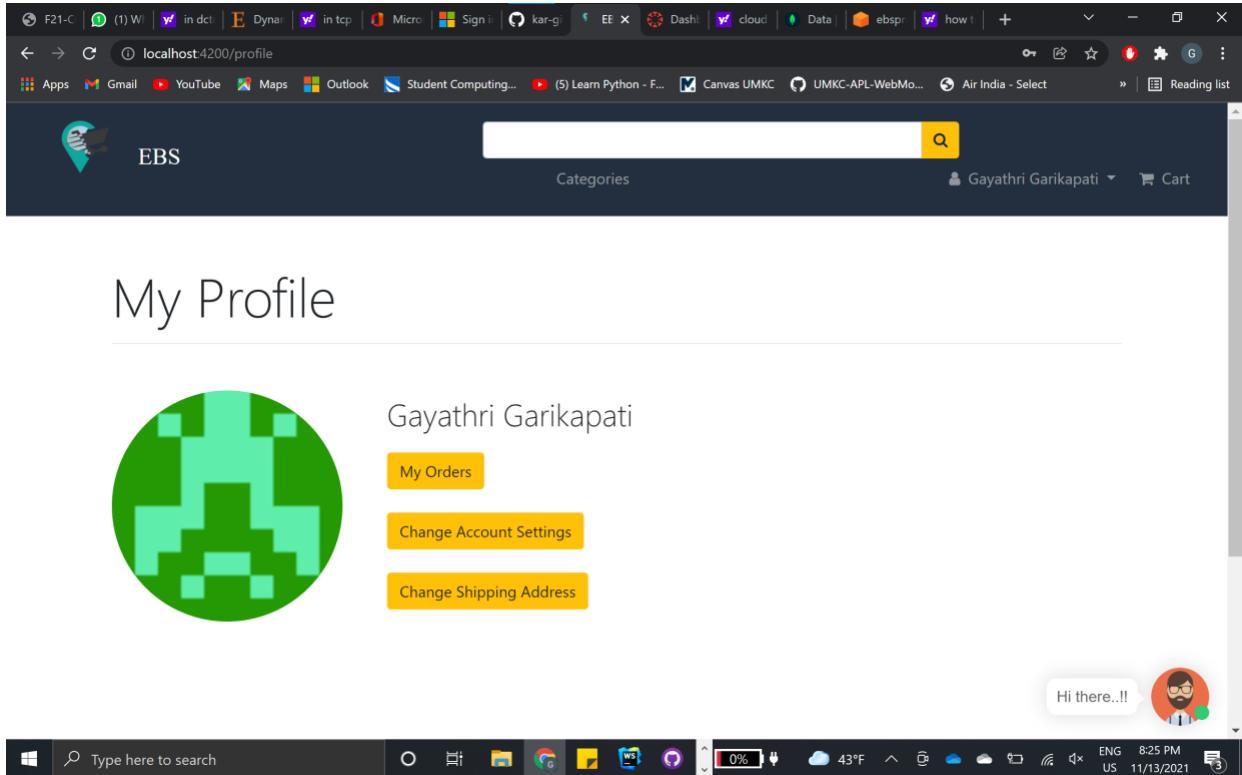
User has an option to change his account settings such as password update.

The screenshot shows a Microsoft Edge browser window with the URL localhost:4200/profile/settings. The page title is "My Account Settings". The form contains fields for Name (Karthik Yanagandula), Email (kaya@gmail.com), and Password (checkbox checked for Is Seller, New Password field containing '*****'). A red "Update" button is at the bottom. The Windows taskbar at the bottom shows various pinned apps and system icons.

There is an option to update the shipping address.

The screenshot shows a Microsoft Edge browser window with the URL localhost:4200/profile/address. The page title is "My Shipping Address". The form includes fields for Address 1 (5054 Baltimore Avenue), Address 2 (NONE), City (64112), State (MO), Postal Code (64112), and Country (united states). A red "Change Address" button is at the bottom. The Windows taskbar at the bottom shows various pinned apps and system icons.

“Post product for sale” and “My products” options are not available for the user who is not registered as a seller.



When the user wants to put a product for sale he can click on Post a product for sale tab and provide the details such as

- Title of the product
- Price
- Category
- Description
- Upload the product image

The screenshot shows a web application interface for posting a product. At the top, the title "Post a Product for Sale" is displayed. Below it is a form with the following fields:

- Title:** An input field.
- Price:** An input field containing "0".
- Category:** A dropdown menu.
- Description:** A large text area.
- Upload Image:** A section with a "Choose File" button and a message "No file chosen".
- Post:** A green button at the bottom of the form.

At the bottom of the screen, the Windows taskbar is visible with various icons and system status information.

These products are stored in the “EBS-Cluster” of a MongoDB database hosted on cloud. All the product images are stored in AWS S3 bucket and we can see that the schema has AWS URL for the image value.

Here we have a product image with id 1636833628136 and this image is stored in the AWS S3 bucket.

The screenshot shows the MongoDB Atlas interface for the "EBS.products" collection. The left sidebar shows the project structure and deployment details. The main panel displays the following document data:

```

{
  "_id": "6190195db303981abb38ff3",
  "reviews": [],
  "created": "2021-11-13T20:00:29.021+00:00",
  "owner": "ObjectID(\"6190195db303981abb38ff3\")",
  "category": "ObjectID(\"6190195db303981abb38ffed\")",
  "title": "MacBook Air 13.3\" Laptop - Apple M1 chip - 8GB Memory - 512GB SSD (Lat...)",
  "price": 1199,
  "description": "Apple's thinnest and lightest notebook gets supercharged with the Appl...",
  "image": "https://ebsproducts.s3.amazonaws.com/1636833628136"
}

```

A green "Get Started" button is located at the bottom left of the interface.

The screenshot shows the AWS S3 console interface. On the left, there's a sidebar with options like Buckets, Storage Lens, and Feature spotlight. The main area is titled 'ebsproducts' and shows 'Objects (3)'. Below this, there's a table with columns for Name, Type, Last modified, Size, and Storage class. The objects listed are:

Name	Type	Last modified	Size	Storage class
1636833628136	-	November 13, 2021, 14:00:29 (UTC-06:00)	17.7 KB	Standard
1636833742245	-	November 13, 2021, 14:02:23 (UTC-06:00)	63.2 KB	Standard
1636835305827	-	November 13, 2021, 14:28:27 (UTC-06:00)	38.0 KB	Standard

Under My Products tab all the products posted for sale by a particular user registered as a seller are visible.

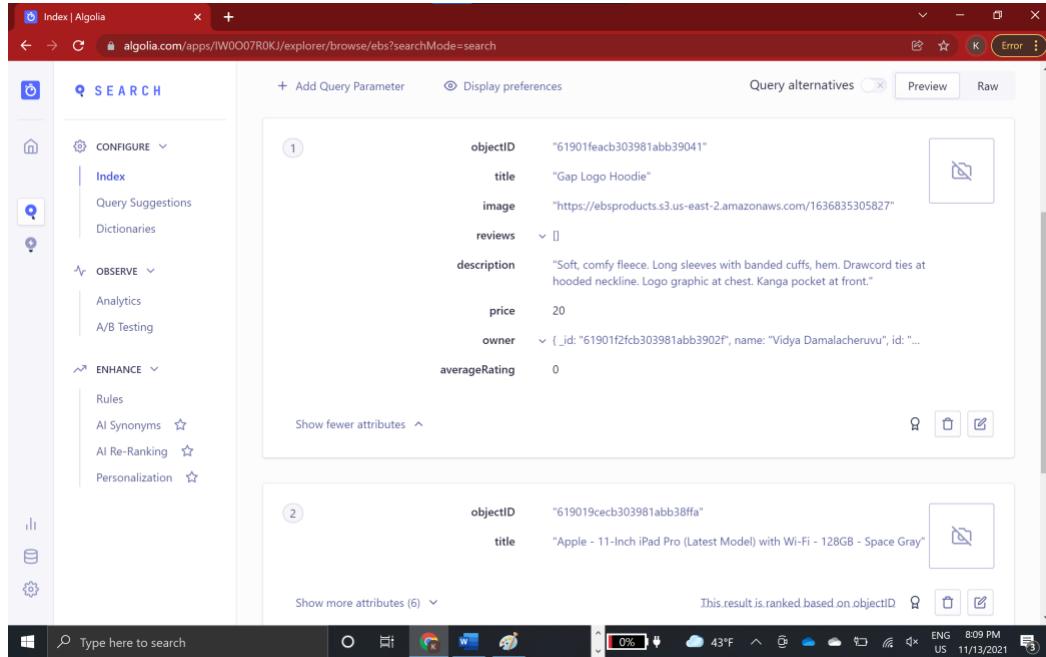
The screenshot shows a custom web application interface. At the top, there's a header with the EBS logo, a search bar, and a user profile for 'Karthik Yanagandula'. Below this is a section titled 'My Products'.

Price

	MacBook Air 13.3" Laptop - Apple M1 chip - 8GB Memory - 512GB SSD (Latest Model) - Space Gray	\$1,199.00
	Apple - 11-Inch iPad Pro (Latest Model) with Wi-Fi - 128GB - Space Gray	\$749.00

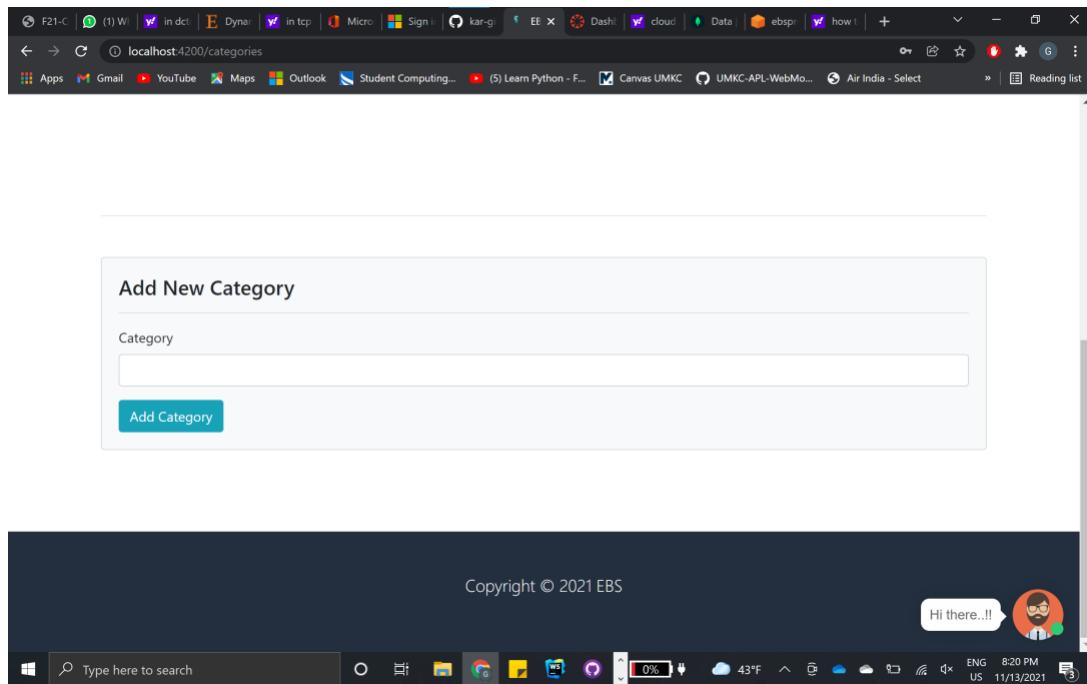
Hi there..!!

All these products are synchronized with Algolia Search API such that the product results can be pulled into the project.



The screenshot shows the Algolia Search API interface. On the left, there's a sidebar with sections like 'CONFIGURE' (Index, Query Suggestions, Dictionaries), 'OBSERVE' (Analytics, A/B Testing), and 'ENHANCE' (Rules, AI Synonyms, AI Re-Ranking, Personalization). The main area displays two product objects. Object 1 has attributes: objectID ("61901feacb303981abb39041"), title ("Gap Logo Hoodie"), image ("https://ebsproducts.s3.us-east-2.amazonaws.com/1636835305827"), reviews (expanded to show 0), description ("Soft, comfy fleece. Long sleeves with banded cuffs, hem. Drawcord ties at hooded neckline. Logo graphic at chest. Kangaroo pocket at front."), price (20), owner (expanded to show {_id: "61901f2fc303981abb3902f", name: "Vidya Damalacheruvu", id: "..."}), and averageRating (0). Object 2 has attributes: objectID ("619019cecb303981abb38ffa") and title ("Apple - 11-Inch iPad Pro (Latest Model) with Wi-Fi - 128GB - Space Gray"). There are buttons for 'Show fewer attributes' and 'Show more attributes (6)'. A note says 'This result is ranked based on objectID'. The bottom of the screen shows a Windows taskbar with various icons and system status.

When the user is registered as a seller, he also has an option to add a new category of the product.



The screenshot shows a web browser window with the URL 'localhost:4200/categories'. The page contains a form titled 'Add New Category' with a 'Category' input field and a 'Add Category' button. Below the form, a dark footer bar displays the text 'Copyright © 2021 EBS' and a speech bubble saying 'Hi there..!!' with a user icon. The bottom of the screen shows a Windows taskbar with various icons and system status.

We have added the below categories as electronics, clothing, and donations for testing the application.

EBS

Categories

electronics

clothing

donations

Hi there..!!

These categories are stored in MongoDB database under EBS-Cluster. Below screen displays the schema for a category.

Karthik's Org - 2021...

Access Manager Billing

All Clusters Get Help Karthik

Project 0

Atlas

Databases

EBS

categories

products

users

COLLECTION SIZE: 205B TOTAL DOCUMENTS: 3 INDEXES TOTAL SIZE: 72KB

Find Indexes Schema Anti-Patterns Aggregation Search Indexes

INSERT DOCUMENT

FILTER { field: 'value' } OPTIONS Apply Reset

QUERY RESULTS 1-3 OF 3

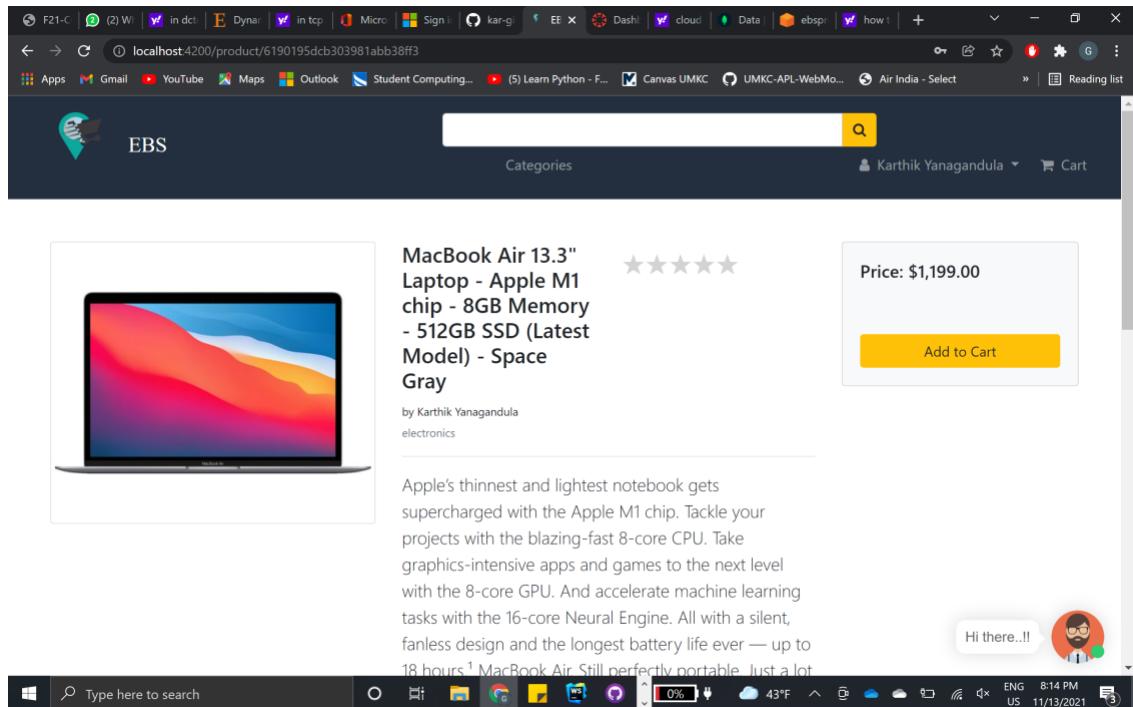
`_id: ObjectId("619018e6cb303981abb38fe6")
created: 2021-11-13T19:58:18.708+00:00
name: "electronics"
__v: 0`

`_id: ObjectId("619018e6cb303981abb38fe6")
created: 2021-11-13T19:58:30.083+00:00
name: "clothing"
__v: 0`

Get Started

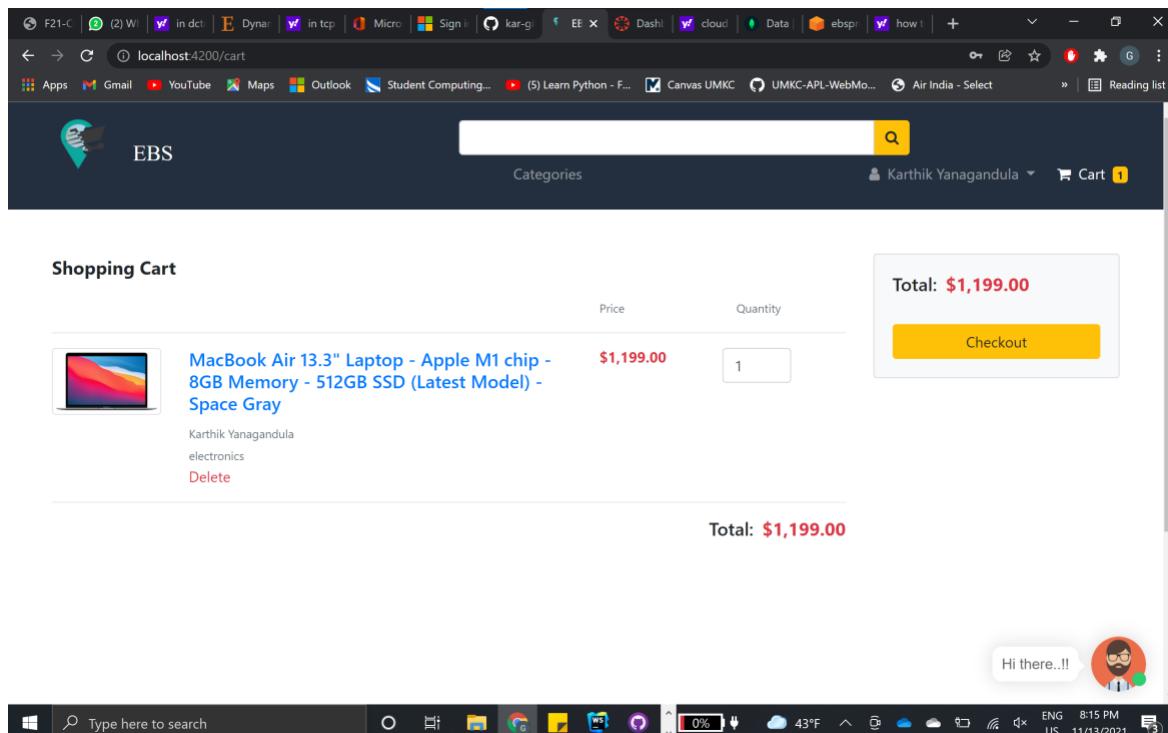
Type here to search

Add to cart option:



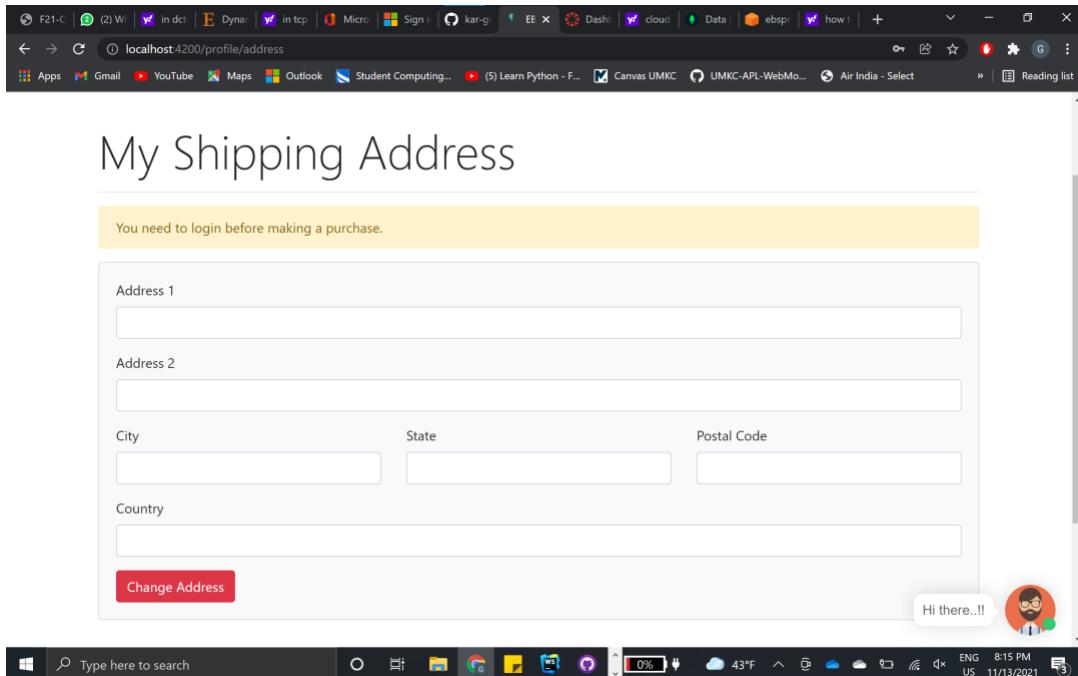
Quantity of the product can be changed as per user requirement.

Product is added to the cart.

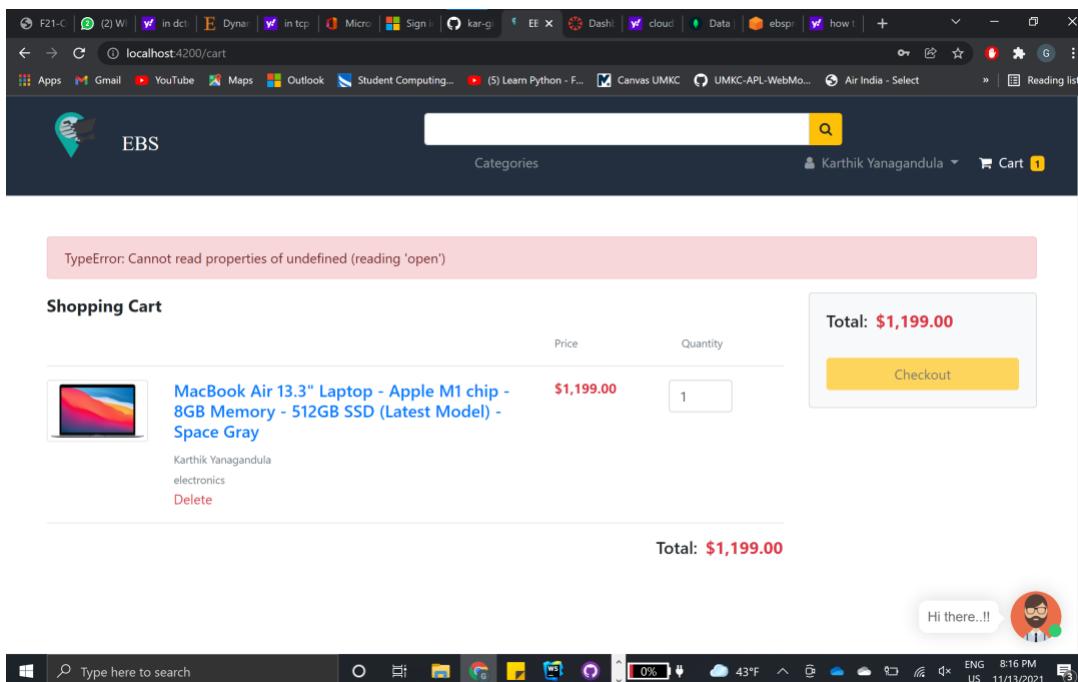


Checkout option:

When user clicks on check out, he must provide the shipping address before making a purchase.



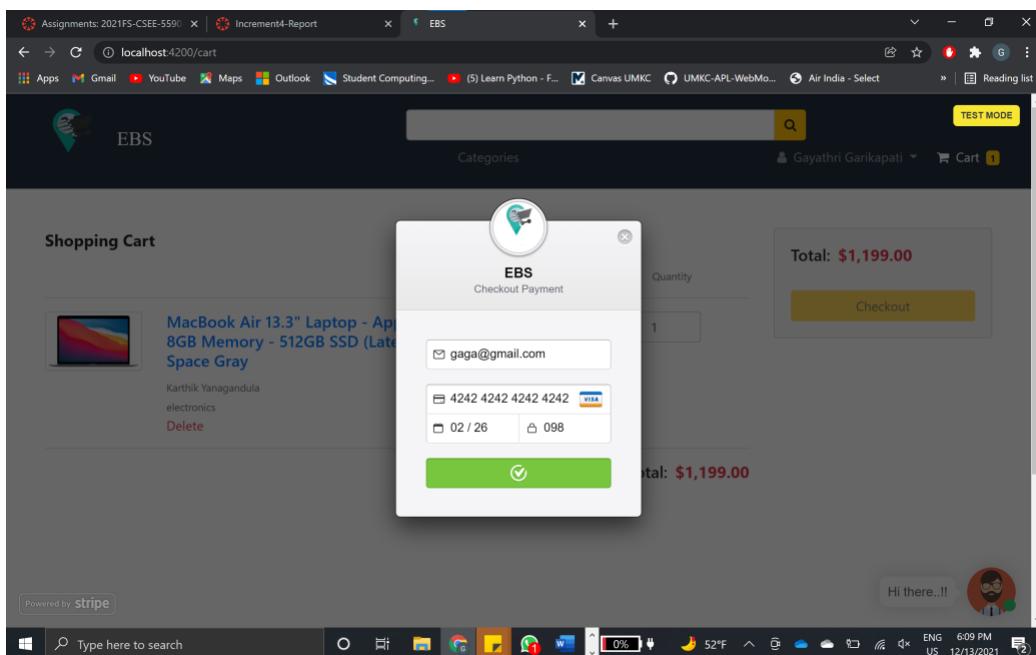
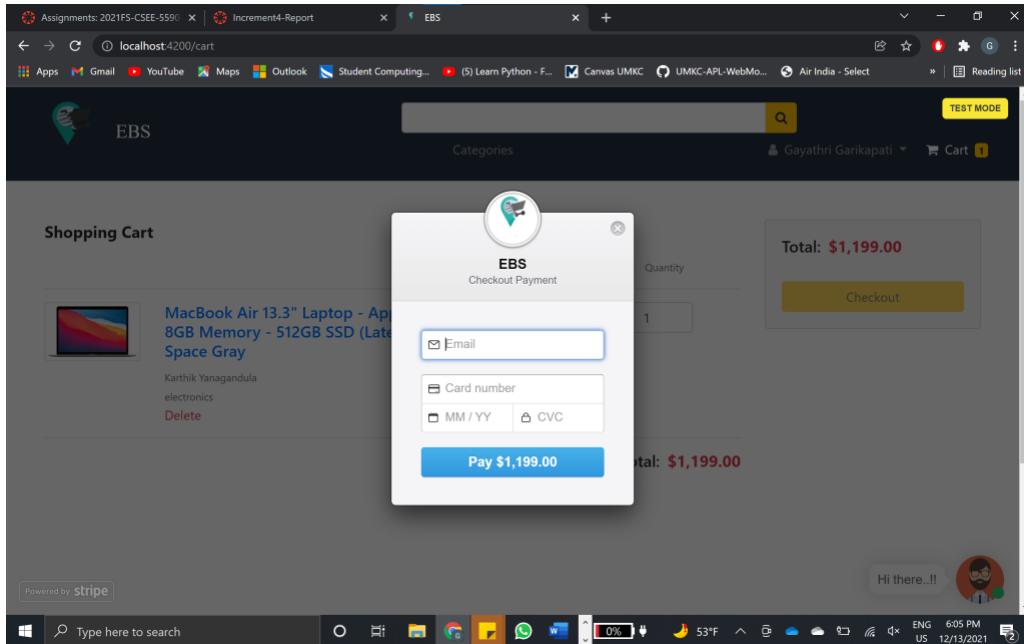
Now when we proceed to checkout for payment, we are receiving error due to Stripe API version update.



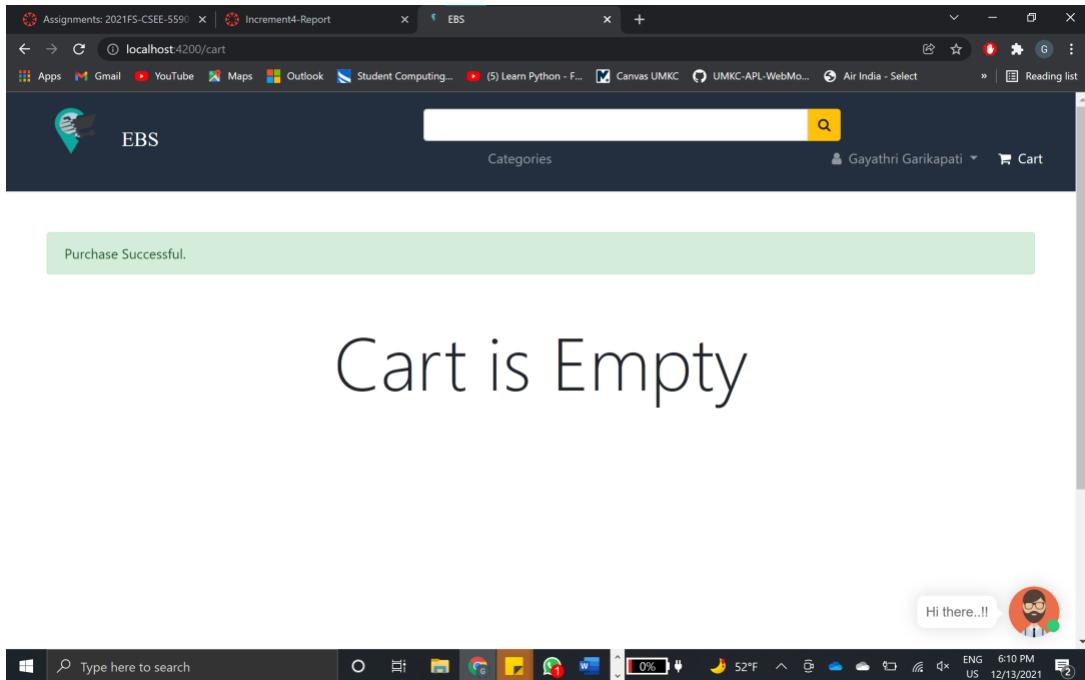
Improvement from the previous increment:

We have resolved the Stripe checkout error and were able to check out the cart and proceed with payment.

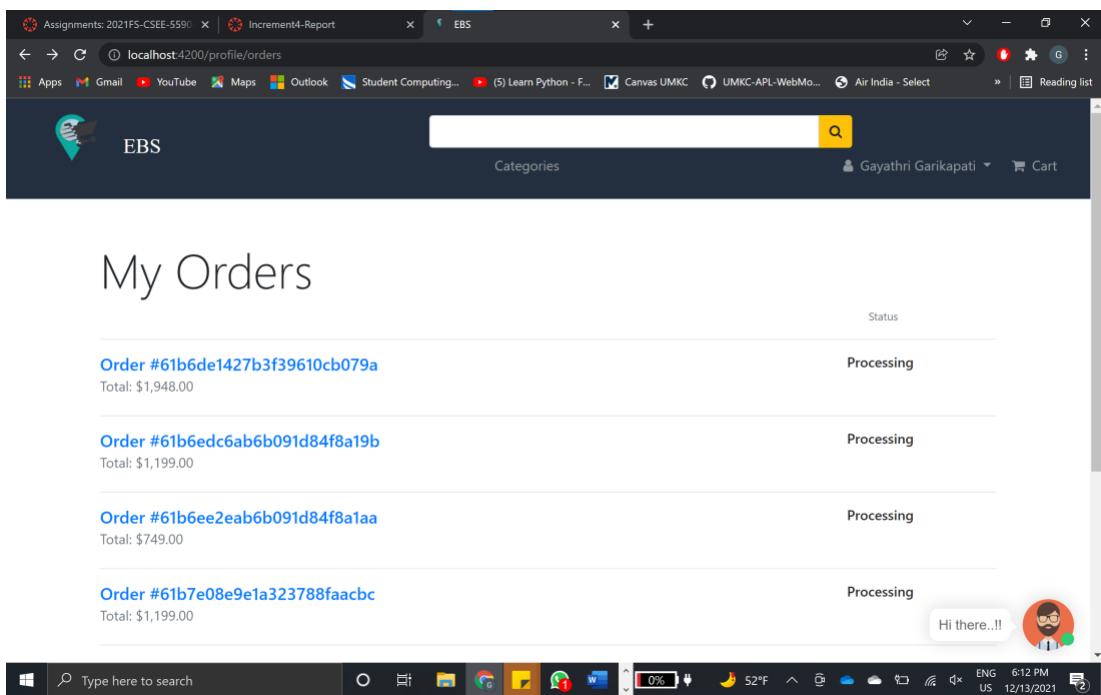
Here when the user clicks on checkout he will be redirected to the checkout payment page where the details like user's email id, card number, expiry date and security code should be provided.



After payment the “payment successful” and the “cart is empty” message will be displayed.



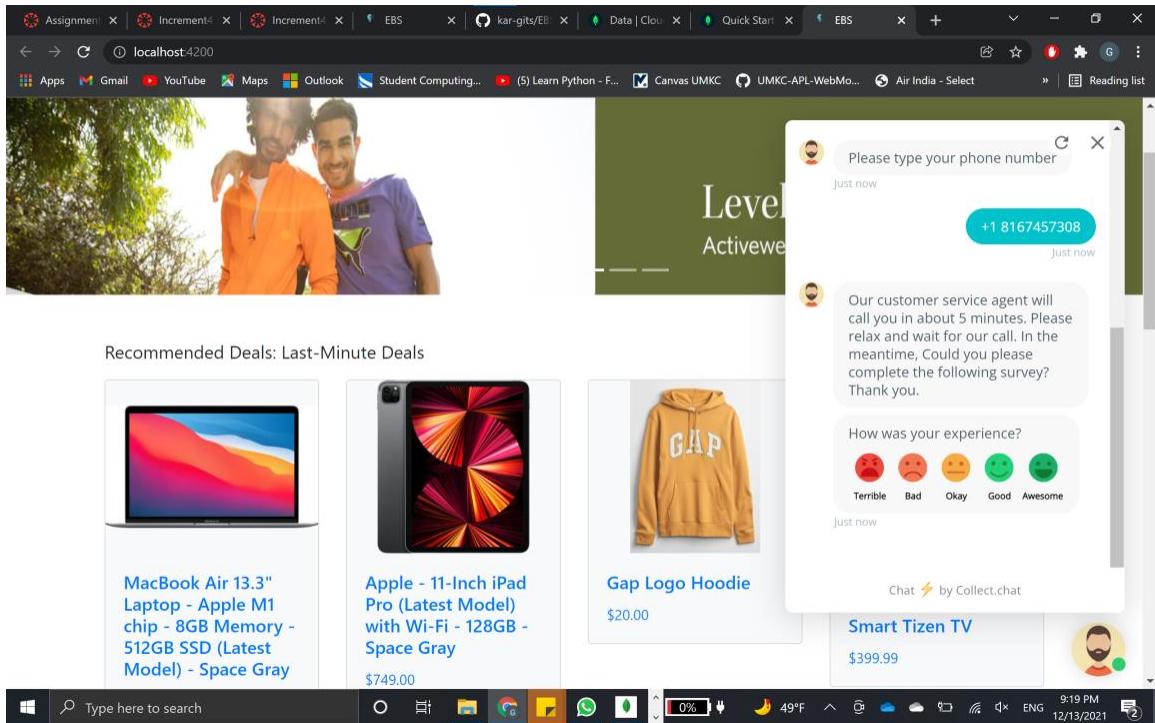
All the purchased items will be displayed under “My Orders”



The screenshot shows a web browser window with the URL localhost:4200/orders/61b7e0d49e1a323788faaccb. The page title is "My Order details". The main content displays a product card for a "MacBook Air 13.3" Laptop - Apple M1 chip - 8GB Memory - 512GB SSD (Latest Model) - Space Gray" at a price of \$1,199.00. A quantity input field shows "1". The EBS logo is visible in the top left corner. The browser's address bar and taskbar are visible at the bottom.

ChatBot:

The screenshot shows a web browser window with the URL localhost:4200. The page features a promotional banner for "Get the season's go-to styles." and a "Must-have looks for wherever you're headed." message. On the right side, a chatbot interface is open, showing a conversation between a user and a bot. The user asks "Hi There, Do you need any help?", and the bot responds "Yes". The user then asks "Please select an option to help you today?", and the bot suggests "Orders". The user is prompted to "Please type your phone number". The browser's address bar and taskbar are visible at the bottom.



Important code snippets from the project:

EBS Server: `npm install`

`node server.js`

```

File Edit View Navigate Code Refactor Run Tools Git Window Help EBS
Project Terminal Local Local (2) Add Configuration... Git: ✓ ✓ ✓
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Try the new cross-platform PowerShell https://aka.ms/pscore6

PS C:\Users\gayat\Documents\GitHub\EBS> cd server
PS C:\Users\gayat\Documents\GitHub\EBS\server> npm install
npm WARN server@1.0.0 No repository field.

audited 179 packages in 0.834s

5 packages are looking for funding
  run `npm fund` for details

found 2 high severity vulnerabilities
  run `npm audit fix` to fix them, or `npm audit` for details
PS C:\Users\gayat\Documents\GitHub\EBS\server> node server.js
Server connected at port: 3030
(node:14216) [MONGODB DRIVER] Warning: Current Server Discovery and Monitoring engine is deprecated, and will be removed in a future version. To use the new Server Discover and Monitoring engine, pass option { useUnifiedTopology: true } to the MongoClient constructor.
(Use 'node --trace-warnings ...' to show where the warning was created)
(node:14216) DeprecationWarning: collection.ensureIndex is deprecated. Use createIndexes instead.
Connected to the database
[6:03:45 PM] [Algolia-sync] -> Cleared Index -> ebs
[6:03:45 PM] [Algolia-sync] -> Updated Settings -> ebs
[6:03:45 PM] [Algolia-sync] -> Synchronized Index -> ebs
★ OPTIONS /api/accounts/profile 204 1.228 ms - 0
  Git  Todo  Problems  Terminal  Event Log  main
  Type here to search  0%  51°F  ENG  6:54 PM  US  12/13/2021

```

```
[0:03:43 PM] [Algolia-sync] -> Updated Settings -> ebs
[0:03:43 PM] [Algolia-sync] -> Synchronized Index -> ebs
OPTIONS /api/accounts/profile 204 1.228 ms - 0
OPTIONS /api/products 204 0.188 ms - 0
(node:14216) DeprecationWarning: collection.count is deprecated, and will be removed in a future version. Use Collection.countDocuments or Collection.estimatedDocumentCount instead
GET /api/accounts/profile 200 49.098 ms - 483
GET /api/products 200 106.668 ms - 12774
OPTIONS /api/categories 204 0.192 ms - 0
GET /api/categories 200 44.251 ms - 349
OPTIONS /api/categories/619018dacb303981abb38fe6?page=0 204 1.030 ms - 0
GET /api/categories/619018dacb303981abb38fe6?page=0 200 324.936 ms - 13432
OPTIONS /api/product/6190195dcb303981abb38ff3 204 0.184 ms - 0
GET /api/product/6190195dcb303981abb38ff3 200 195.606 ms - 3007
OPTIONS /api/categories 204 0.156 ms - 0
GET /api/categories 304 51.638 ms - -
OPTIONS /api/categories/619018e6cb303981abb38fea?page=0 204 0.156 ms - 0
GET /api/categories/619018e6cb303981abb38fea?page=0 200 96.938 ms - 5504
OPTIONS /api/product/61b6e802ab60091d84f8a088 204 0.119 ms - 0
GET /api/product/61b6e802ab60091d84f8a088 200 171.480 ms - 1845
OPTIONS /api/payment 204 0.193 ms - 0
POST /api/payment 200 2334.703 ms - 56
OPTIONS /api/categories 204 0.140 ms - -
GET /api/categories 304 41.252 ms - -
OPTIONS /api/categories/619018dacb303981abb38fe6?page=0 204 0.132 ms - 0
GET /api/categories/619018dacb303981abb38fe6?page=0 304 385.187 ms - -
★ OPTIONS /api/product/6190195dcb303981abb38ff3 204 0.174 ms - 0
```

EBS client: npm install

ng serve -o

```
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Try the new cross-platform PowerShell https://aka.ms/powershell

PS C:\Users\gayat\Documents\GitHub\EBS> cd client/EBS
PS C:\Users\gayat\Documents\GitHub\EBS\client\EBS> npm install
npm WARN @angular/compiler-cli@5.2.11 requires a peer of typescript@>=2.4.2 <2.7 but none is installed. You must install peer dependencies yourself.
npm WARN tsickle@0.27.5 requires a peer of typescript@>=2.4.2 <2.8 but none is installed. You must install peer dependencies yourself.

audited 1684 packages in 4.644s

56 packages are looking for funding
  run 'npm fund' for details

found 80 vulnerabilities (5 low, 32 moderate, 36 high, 7 critical)
  run 'npm audit fix' to fix them, or 'npm audit' for details
PS C:\Users\gayat\Documents\GitHub\EBS\client\EBS> ng serve -o
Your global Angular CLI version (12.2.7) is greater than your local version (1.6.5). The local Angular CLI version is used.

To disable this warning use "ng config -g cli.warnings.versionMismatch false".

@angular/compiler-cli@5.2.11 requires typescript@>=2.4.2 <2.7.0' but 2.8.4 was found instead.
Using this version can result in undefined behaviour and difficult to debug problems.

Please run the following command to install a compatible version of TypeScript.
```

```
PS C:\Users\gayat\Documents\GitHub\EBS\client\EBS> ng serve -o
Your global Angular CLI version (12.2.7) is greater than your local version (1.6.5). The local Angular CLI version is used.

To disable this warning use "ng config -g cli.warnings.versionMismatch false".

@angular/compiler-cli@5.2.11 requires typescript@>=2.4.2 <2.7.0' but 2.8.4 was found instead.
Using this version can result in undefined behaviour and difficult to debug problems.

Please run the following command to install a compatible version of TypeScript.

  npm install typescript@>=2.4.2 <2.7.0'

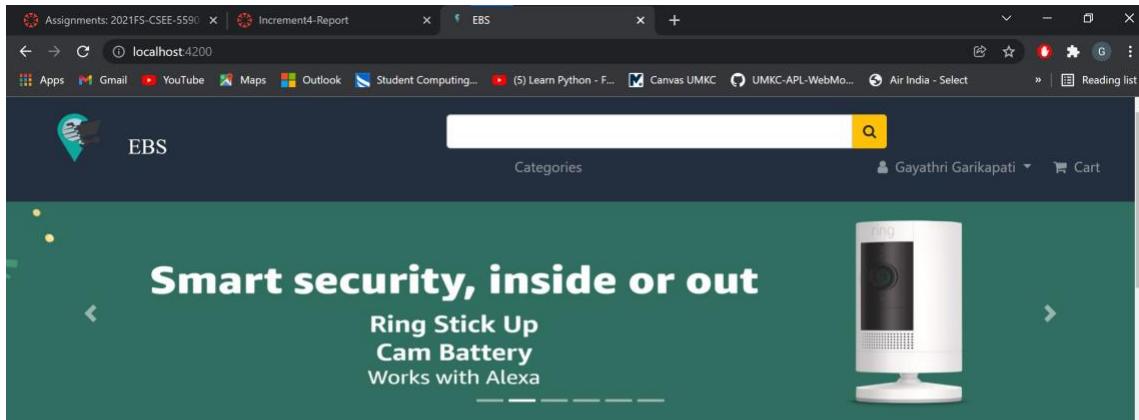
To disable this warning run "ng set warnings.typescriptMismatch=false".

** NG Live Development Server is listening on localhost:4200, open your browser on http://localhost:4200/ **
13 building modules 31/31 modules 0 activewebpack: wait until bundle finished: /
Date: 2021-12-14T00:02:56.665Z
Hash: bb2895dfbc6101d3054
Time: 5380ms
chunk {inline} inline.bundle.js (inline) 5.79 kB [entry] [rendered]
chunk {main} main.bundle.js (main) 393 kB [initial] [rendered]
chunk {polyfills} polyfills.bundle.js (polyfills) 604 kB [initial] [rendered]
chunk {styles} styles.bundle.js (styles) 36.9 kB [initial] [rendered]
chunk {vendor} vendor.bundle.js (vendor) 10.6 MB [initial] [rendered]

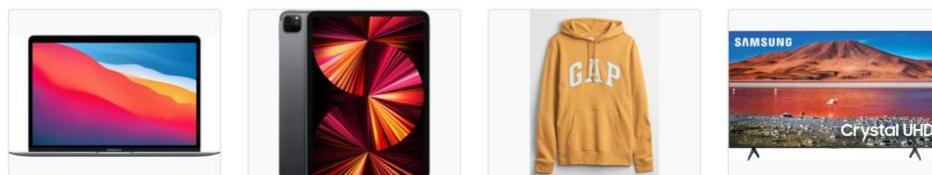
webpack: Compiled successfully.
```

“ng serve -o” will open our application in a browser with web host

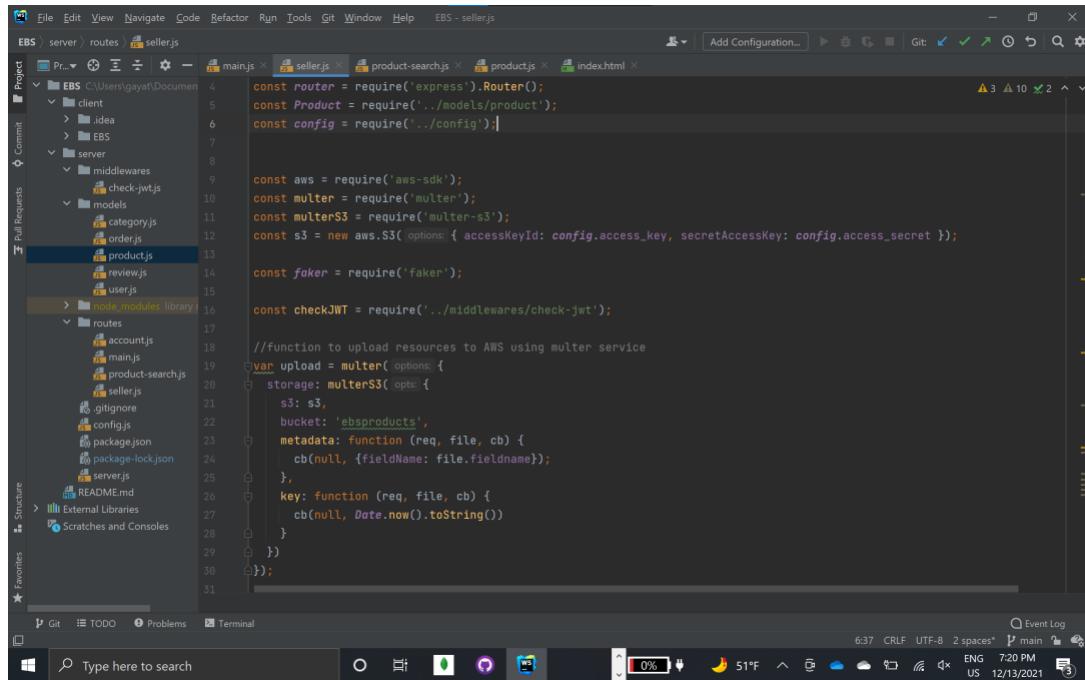
<http://localhost:4200/>



Recommended Deals: Last-Minute Deals



Code to maintain storing of product images on AWS. We are using multer package to connect to AWS S3 bucket using secret keys.



```

File Edit View Navigate Code Refactor Run Tools Git Window Help EBS - seller.js
EBS : server / routes / seller.js
Project C:\Users\gaya\Documents\EB...
  -> client
    -> idea
    -> EBS
      -> server
        -> middlewares
          -> check-jwt.js
        -> models
          -> category.js
          -> order.js
          -> product.js
          -> review.js
          -> user.js
        -> routes
          -> account.js
          -> main.js
          -> product-search.js
          -> seller.js
        -> node_modules
          -> library
            -> aws-sdk
            -> express
            -> multer
            -> mongoose
            -> request
            -> underscore
        -> gitignore
        -> config.js
        -> package.json
        -> package-lock.json
        -> server.js
        -> README.md
      -> External Libraries
      -> Scratches and Consoles
  Favorites
  -> Git
  -> TODO
  -> Problems
  -> Terminal
  Type here to search
  Event Log
  6:37 CRLF UTF-8 2 spaces* P main
  51°F 7:20 PM ENG US 12/13/2021
  
```

```

const router = require('express').Router();
const Product = require('../models/product');
const config = require('../config');

const aws = require('aws-sdk');
const multer = require('multer');
const multerS3 = require('multer-s3');
const s3 = new aws.S3({ options: { accessKeyId: config.access_key, secretAccessKey: config.access_secret } });

const faker = require('faker');

const checkJWT = require('../middlewares/check-jwt');

//function to upload resources to AWS using multer service
var upload = multer({ options: {
  storage: multerS3( opts: {
    s3: s3,
    bucket: 'ebsproducts',
    metadata: function (req, file, cb) {
      cb(null, {fieldName: file.fieldname});
    },
    key: function (req, file, cb) {
      cb(null, Date.now().toString());
    }
  })
}};

main.js
  4 const router = require('express').Router();
  5
  6 const Product = require('../models/product');
  7 const config = require('../config');

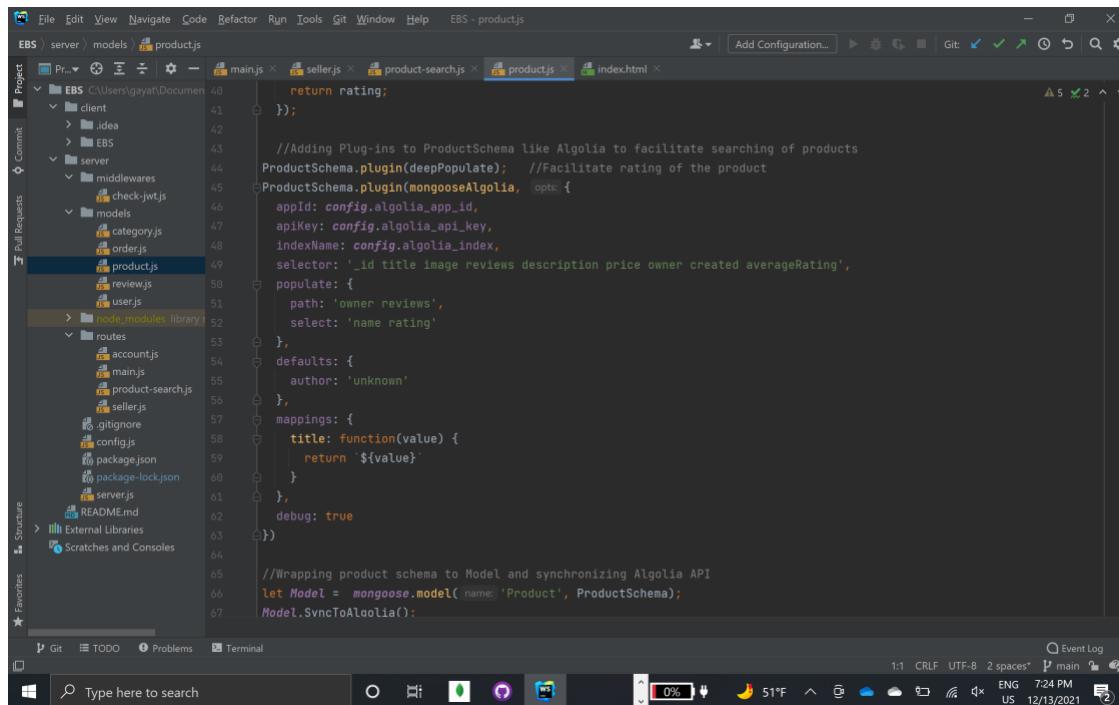
  8
  9 const aws = require('aws-sdk');
 10 const multer = require('multer');
 11 const multerS3 = require('multer-s3');
 12 const s3 = new aws.S3({ options: { accessKeyId: config.access_key, secretAccessKey: config.access_secret } });

 13
 14 const faker = require('faker');

 15
 16 const checkJWT = require('../middlewares/check-jwt');

 17
 18 //function to upload resources to AWS using multer service
 19 var upload = multer({ options: {
 20   storage: multerS3( opts: {
 21     s3: s3,
 22     bucket: 'ebsproducts',
 23     metadata: function (req, file, cb) {
 24       cb(null, {fieldName: file.fieldname});
 25     },
 26     key: function (req, file, cb) {
 27       cb(null, Date.now().toString());
 28     }
 29   })
 30 });
 31
  
```

Using mongoose-algolia, we connected Algolia API to product schema, which loads the product data to Algolia search API. This ensures that all the product details in application are shared with API to perform search operations.



```

File Edit View Navigate Code Refactor Run Tools Git Window Help EBS - product.js
EBS : server / models / product.js
Project C:\Users\gaya\Documents\EB...
  -> client
    -> idea
    -> EBS
      -> server
        -> middlewares
          -> check-jwt.js
        -> models
          -> category.js
          -> order.js
          -> product.js
          -> review.js
          -> user.js
        -> routes
          -> account.js
          -> main.js
          -> product-search.js
          -> seller.js
        -> node_modules
          -> library
            -> algolia
            -> mongoose
            -> request
            -> underscore
        -> gitignore
        -> config.js
        -> package.json
        -> package-lock.json
        -> server.js
        -> README.md
      -> External Libraries
      -> Scratches and Consoles
  Favorites
  -> Git
  -> TODO
  -> Problems
  -> Terminal
  Type here to search
  Event Log
  1:1 CRLF UTF-8 2 spaces* P main
  51°F 7:24 PM ENG US 12/13/2021
  
```

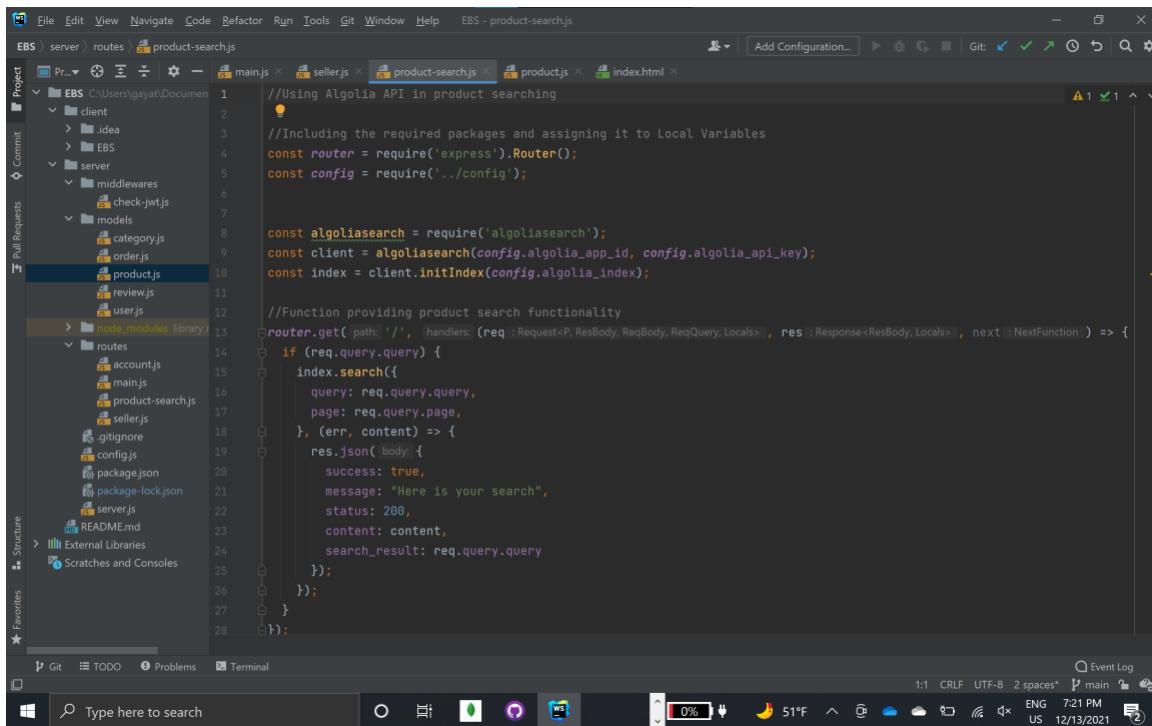
```

main.js
  40
  41   return rating;
  42 };

  43 //Adding Plug-ins to ProductSchema like Algolia to facilitate searching of products
  44 ProductSchema.plugin(deepPopulate); //Facilitate rating of the product
  45 ProductSchema.plugin(mongooseAlgolia, opts: {
  46   appId: config.algolia_app_id,
  47   apiKey: config.algolia_api_key,
  48   indexName: config.algolia_index,
  49   selector: '_id title image reviews description price owner created averageRating',
  50   populate: {
  51     path: 'owner reviews',
  52     select: 'name rating'
  53   },
  54   defaults: {
  55     author: 'unknown'
  56   },
  57   mappings: {
  58     title: function(value) {
  59       return `${value}`
  60     }
  61   },
  62   debug: true
  63 }

  64 //Wrapping product schema to Model and synchronizing Algolia API
  65 let Model = mongoose.model(name: 'Product', ProductSchema);
  66 Model.SyncToAlgolia();
  67
  
```

To facilitate product search using Algolia search API. We are using secret keys and index name of search API to perform the task.

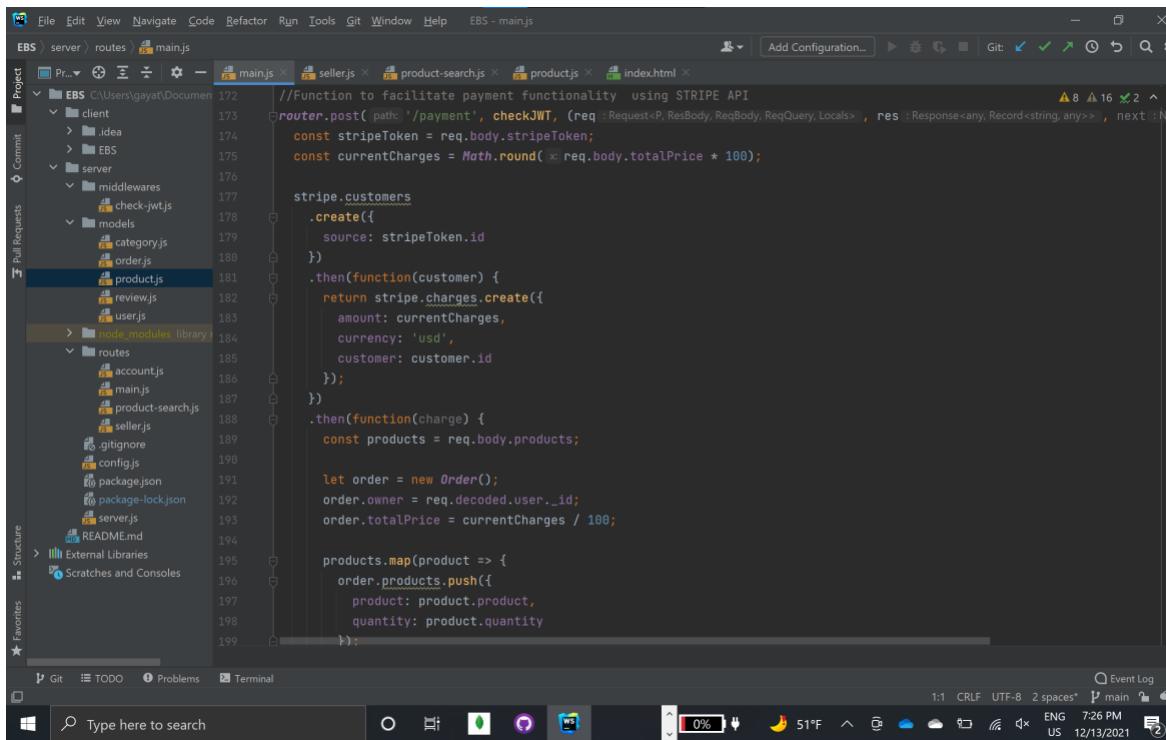


```

EBS > server > routes > product-search.js
File Edit View Navigate Code Refactor Run Tools Git Window Help EBS - product-search.js
EBS C:\Users\gaya\Documents\Project\server\routes\product-search.js
1 //Using Algolia API in product searching
2
3 //Including the required packages and assigning it to Local Variables
4 const router = require('express').Router();
5 const config = require('../config');
6
7
8 const algoliasearch = require('algoliasearch');
9 const client = algoliasearch(config.algolia_app_id, config.algolia_api_key);
10 const index = client.initIndex(config.algolia_index);
11
12 //Function providing product search functionality
13 router.get(path: '/', handlers: (req: Request<P, ResBody, ReqBody, ReqQuery, Locals>, res: Response<ResBody, Locals>, next: NextFunction) => {
14   if (req.query.query) {
15     index.search({
16       query: req.query.query,
17       page: req.query.page,
18     }, (err, content) => {
19       res.json(body: {
20         success: true,
21         message: "Here is your search",
22         status: 200,
23         content: content,
24         search_result: req.query.query
25       });
26     });
27   }
28 });

```

Code to facilitate payment functionality using STRIPE API



```

EBS > server > routes > main.js
File Edit View Navigate Code Refactor Run Tools Git Window Help EBS - main.js
EBS C:\Users\gaya\Documents\Project\server\routes\main.js
172 //Function to facilitate payment functionality using STRIPE API
173 router.post(path: '/payment', checkJWT, (req: Request<P, ResBody, ReqBody, ReqQuery, Locals>, res: Response<any, Record<string, any>>, next: NextFunction) => {
174   const stripeToken = req.body.stripeToken;
175   const currentCharges = Math.round(req.body.totalPrice * 100);
176
177   stripe.customers
178     .create({
179       source: stripeToken.id
180     })
181     .then(function(customer) {
182       return stripe.charges.create({
183         amount: currentCharges,
184         currency: 'usd',
185         customer: customer.id
186       });
187     })
188     .then(function(charge) {
189       const products = req.body.products;
190
191       let order = new Order();
192       order.owner = req.decoded.user._id;
193       order.totalPrice = currentCharges / 100;
194
195       products.map(product => {
196         order.products.push({
197           product: product.product,
198           quantity: product.quantity
199         });
200     });
201   });
202 });

```

The screenshot shows the Stripe Payments dashboard in test mode. The left sidebar has sections for All payments, Fraud & risk, Invoices, Subscriptions, Quotes, and Payment links. The main table lists 16 successful payments. Each row includes columns for Amount, Description, Customer, and Date.

	AMOUNT	DESCRIPTION	CUSTOMER	DATE
<input type="checkbox"/>	\$40.00 USD	Succeeded ✓ ch_3K60sJCRoz77xUcI1mukwl2a	cus_KlypGntsR57Gd	Dec 13, 8:36 PM
<input type="checkbox"/>	\$1,199.00 USD	Succeeded ✓ ch_3K60abCRoz77xUcI1cWkMDCH	cus_KlwTIDIPXZr6t	Dec 13, 6:09 PM
<input type="checkbox"/>	\$1,199.00 USD	Succeeded ✓ ch_3K60ZUCRoz77xUcI0SZLhe6w	cus_KlwSAN6WrEwIE	Dec 13, 6:08 PM
<input type="checkbox"/>	\$1,349.99 USD	Succeeded ✓ ch_3K68TFCRoz77xUcI0kLpGwLP	cus_Klfo5AGXhqHtp5	Dec 13, 12:57 AM
<input type="checkbox"/>	\$59.50 USD	Succeeded ✓ ch_3K68SSCRoz77xUcI0lLaPjju	cus_Klfo1qmoUdTqvD	Dec 13, 12:56 AM
<input type="checkbox"/>	\$749.00 USD	Succeeded ✓ ch_3K68QjCRoz77xUcI1Mcih9IY	cus_KlmfHH0pq3VYTt	Dec 13, 12:54 AM
<input type="checkbox"/>	\$1,199.00 USD	Succeeded ✓ ch_3K68P3CRoz77xUcI1h1e0zy9	cus_KlfkmkAu12mPeT	Dec 13, 12:52 AM
<input type="checkbox"/>	\$20.00 USD	Succeeded ✓ ch_3K67mLCRoz77xUcI1Bx8d5vJ	cus_Klf65CInMxwUic	Dec 13, 12:12 AM
<input type="checkbox"/>	\$1,199.00 USD	Succeeded ✓ ch_3K67htCRoz77xUcI1Uoc5nGd	cus_Klf10bTwFzMhHK	Dec 13, 12:08 AM
<input type="checkbox"/>	\$20.00 USD	Succeeded ✓ ch_3K67gICRoz77xUcI1za1QcTN	cus_Klf0tf0hlpVg1	Dec 13, 12:06 AM
<input type="checkbox"/>	\$11.99 USD	Succeeded ✓ ch_3K67cTCRoz77xUcI0m90yv80	cus_KlewZ6E5jZP0	Dec 13, 12:02 AM
<input type="checkbox"/>	\$11.99 USD	Succeeded ✓ ch_3K679TCRoz77xUcI1zEPgE3n	cus_KleSnFZ4N0xtT8	Dec 12, 11:32 PM

To implement ChatBot functionality, we included the script generated in collect.chat dashboard.

The screenshot shows the collect.chat dashboard. The left sidebar has sections for Build, Design, Script (which is selected), Share, Integrate, Settings, Results, Workspaces, What's New, Help, and Account. The main area displays a conversational script with five steps. Step 1: Hi There, Do you need any help? Step 2: Please select an option to help you today? Step 3: Please type your phone number. Step 4: Our customer service agent will call you in about 5 minutes. Please relax and wait for our call. In the meantime, Could you please complete the following survey? Thank you. Step 5: How was your experience?

The screenshot shows a code editor interface with two tabs open: 'index.html' and 'main.js'. The 'index.html' tab displays the following code:

```
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="utf-8">
<title>EBS</title>
<base href="/">
<meta name="viewport" content="width=device-width, initial-scale=1">
<link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/4.0.0/css/bootstrap.min.css" integrity="sha384-Gn5BaAH4" />
<link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/font-awesome/4.7.0/css/font-awesome.min.css" />
<script src="https://checkout.stripe.com/checkout.js"></script>
<link rel="icon" type="image/png" href="assets/img/logo.png">
<script>
(function(w, d) {
  if (w.collectId = "618d8efd11c7462f21dec6ef"; var h = d.head || d.getElementsByTagName('head')[0];
    var s = d.createElement('script'); s.setAttribute('qualifiedName', 'type', 'text/javascript');
    s.async=true; s.setAttribute('qualifiedName', 'src', value="https://collectcdn.com/launcher.js");
    h.appendChild(s);
})(window, document);
</script>
</head>
<body>
<app-root></app-root>
</body>
</html>
```

The 'main.js' tab shows the following code:

```
1 // Main.js
2
3 <!-- This file is part of the Angular project. It contains the main module
4   entry point, from which Angular bootstraps the application. For
5   more information, see https://angular.io/guide/quickstart -->
6
7 import { platformBrowserDynamic } from '@angular/platform-browser-dynamic';
8 import { AppModule } from './app/app.module';
9
10 platformBrowserDynamic().bootstrapModule(AppModule)
11   .catch(err => console.error(err));
```

Work Sharing:

Vidya D: Worked on profile and review components, tried to implement multiple shipping address feature for placing order.

Karthik Y: Worked on database models, storing database on cloud and backend code for routes. Worked on Stripe Payment API issue and added simple chatbot using collect.chat website.

Gayathri G: Worked on adding Algolia search API for product search, account creation, logging in and product feature implementation.

Pujitha T: Worked on post products, account settings and order details. Completed checking the orders in database and UI after Stripe issue is resolved.

Common Work: Worked on report and presentation for the work said above.

Issues with the project:

Due to a version change for the Stripe API, we had an issue processing payment due to a "Stripcheckout" error during the product checkout. We were able to find a solution to the problem. We didn't have enough time to add the additional capabilities to the application because the end-to-end work took so long, therefore we'll try to incorporate these functionalities (Split payments and multiple address delivery) as part of future work.

Future work:

- Split payment
- Product delivery to multiple address with single checkout

GitHub link:

<https://github.com/kar-gits/EBS>

Video:

<https://youtu.be/bobL7H5Y2ac>

Presentation:

<https://docs.google.com/presentation/d/1xG7kW0mHmBk-elTSNinDcacMdJQOx1r1/edit?usp=sharing&ouid=107184809928603804089&rtpof=true&sd=true>

References:

- <https://www.algolia.com/doc/rest-api/search/>
- <https://stripe.com/docs/api>
- <https://angular.io/docs>
- <https://www.mongodb.com/developer/learn/?content=Articles&text=Node.js>
- <https://nodejs.org/dist/latest-v14.x/docs/api/synopsis.html>
- <https://collect.chat/templates/>