**Final Review Document**

**For**

**EasyShop**

**Version 1.0 approved**

**Prepared by Deep Golani, Laksh Gupta, Nimish Shah**

**VIT Vellore**

**22nd October 2020**

**Table of Contents**

**Table of Contents ............................................................................................................................................................................. 2**

**1. Introduction ............................................................................................................................................................. 4**

1.1 Purpose .................................................................................................................................................................... 4

1.2 Document Conventions ........................................................................................................................................... 4

1.3 Intended Audience and Reading Suggestions.......................................................................................................... 4

1.4 Problem Definition …............................................................................................................................................. 4

1.5 Motivation ............................................................................................................................................................... 5

1.6 Project Scope ........................................................................................................................................................... 5

1.7 Product Scope ..........................................................................................................................................................5

1.8 Stakeholders............................................................................................................................................................. 6

1.9 Budget Estimations...................................................................................................................................................6

**2. Overall Description.................................................................................................................................................. 7**

2.1 Product Perspective ................................................................................................................................................. 7

2.2 Product Functions.................................................................................................................................................... 8

2.3 User Classes and Characteristics ...................................................................................................................,......... 9

2.4 Operating Environment ....................................................................................................................................,...... 9

2.5 Design and Implementation Constraints ............................................................................................................,.... 9

2.6 User Documentation...............................................................................................................................................10

2.7 Assumptions and Dependencies. ............................................................................................................................10

2.8 Gant Diagram …………………………………………………………………………………………………….10

2.9 UML Diagrams……...………................................................................................................................................11

2.10 Work Breakdown Structure……………………………………………………………………………………...17

**3. External Interface Requirements ........................................................................................................................ 19**

3.1 User Interfaces ...................................................................................................................................................... 19

3.2 Hardware Interfaces............................................................................................................................................... 22

3.3 Software Interfaces.................................................................................................................................................22

3.4 Communications Interfaces ................................................................................................................................... 23

**4. System Features ..................................................................................................................................................... 24**

4.1 Description and priority ........................................................................................................................................ 24

4.2 Stimulus/Response Sequence ................................................................................................................................ 24

4.3 Functional Requirement........................................................................................................................................ .24

4.3.1 Admin ................................................................................................................................................................ 24

4.3.2 Customer……….. .............................................................................................................................................. 24

**5. Other Non-functional Requirements.................................................................................................................... 25**

5.1 Performance Requirements ................................................................................................................................... 25

5.2 Safety Requirements ............................................................................................................................................. 25

5.3 Security Requirements........................................................................................................................................... 25

5.4 Software Quality Attributes .................................................................................................................................. 26

5.5 Business Rules ...................................................................................................................................................... 26

**6. Final Output………………………………………………………………………………………………………27**

6.1 Website .................................................................................................................................................................. 27

6.2 Security…….... ......................................................................................................................................................30

**7. Code Optimization……………………………………………………………………………………....………..32**

**8. Testing………………………………………...………………....………………………………………………...35**

5.5 Website Testing...................................................................................................................................................... 35

5.5 Security Testing...................................................................................................................................................... 37

**9. Future Work and Conclusions…………………………………………………………………………………...39**

**10. References………………………………………………………………………………………………………..40**

**1. Introduction**

**1.1 Purpose**

The project allows users to check the inventories of nearby shops. Easyshop is the place where customers can view their day to day products and order them online instead of going there physically. In this project, operators enter the Website and buy some product, we then look at their cart and compare it to all nearby shops available, we then generate the bill for every shop and allow the customer to choose which shop they want to order from. Supervisors and admin will have separate login ID and pin through which they can access their account to modify their inventory, which can be done in real-time giving the customer an accurate view of their inventory.

**1.2 Document Conventions**

*• DB- DATABASE*

*• ER- ENTITY RELATIONSHIP DIAGRAM*

*• UML- UNIFIED MODELING LANGUAGE*

*• DFD- DATA FLOW DIAGRAM*

**1.3 Intended Audience and Reading Suggestions**

*This project is a prototype for the Easyshop groceries website and it is restricted within the college premises. This has been implemented under the guidance of our college faculty of Software Engineering. This project is useful for the Website development team and as well as for future users of the website.*

**1.4 Problem definition**

Due to the coronavirus pandemic we all have been confined to our homes. People are not heading out un-necessarily, but getting groceries is important and going to multiple shops can be dangerous.

Also due to the pandemic there is a disruption in the supply of groceries to the shops, and the customer has no way of confirming whether the thing they want will be in stock or not without calling the shop or going there.

Also there are discrepancies in prices, currently there is no feasible way for a customer to compare the prices of different shops for all the items in their shopping list.

**1.5 Motivation**

we went into lockdown from 25th march 2020, so people were forced to get their groceries from local shops. The ones they could walk to, so we thought that if we could make a web-based application that can help them know the inventory of these shops before they set out of their homes, it would help them shop easily and also save some money and time. This will also help in reducing contact between the shopkeepers and the customers, hence reducing the chance of the spread of covid-19.

**1.6 Project scope**

First of all the development team will analyse tasks and produce a document of the functionalities minimum of the project and possible features that can be added later.

Then they will discuss and finalize the technologies that are going to be used. A milestone chart will be prepared based on which the final scope and dates(deadlines) will be agreed.

* **Project deliverables**

A web based application that satisfies the points mentioned in the problem statement in a satisfactory manner.

* **Project acceptance criteria**

A website that is able to meet the set criteria.

* **Project constraints**

Application should be ready by end of semester for final review.

**1.7 Product scope**

We wish to show the users all the shops close to them and allow them to search for the products that they want through the options provided on our website. They should be able to create a cart which contains all the items they wish to buy. Then the website should show the user a comparative analysis of the prices at each shop and the approximate final price for the whole cart for every shop available nearby. The website should also notify the user if a product is not available.

There is also a feature for the customer to send their list to the shop so that the shopkeeper can keep the items ready so that the customer can only pick the groceries quickly.

**1.8 Stakeholders**

**Primary stakeholder-**

* Users of the website
* Shops enrolled in the website
* Development team and employees
* stockholders
* Vendors
* partners

**Secondary Stakeholder**

* Government regulators
* Communities
* General public
* Media
* Creditors

**1.9 Budget estimation**

Google Firebase database 10 GB @14rs per GB -140

Amazon hosting services - 30000rs

Advertisement (google adware)- 500

Total- 30640 Rs/month

Fixed costs

Github premium for collaboration -750rs

Development costs-10000rs

Logistics -15000rs

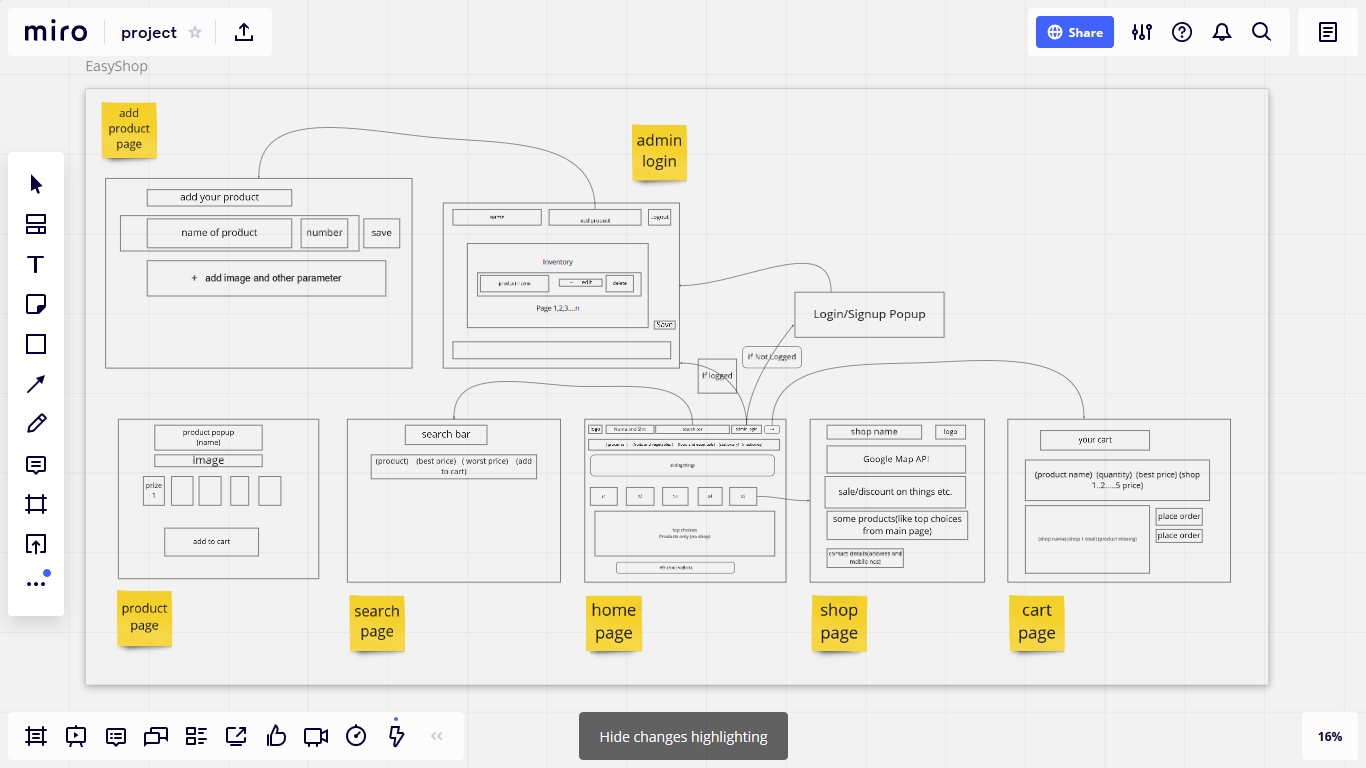
Total- 25750 Rs

**2. Overall Description**

**2.1 Product Perspective**

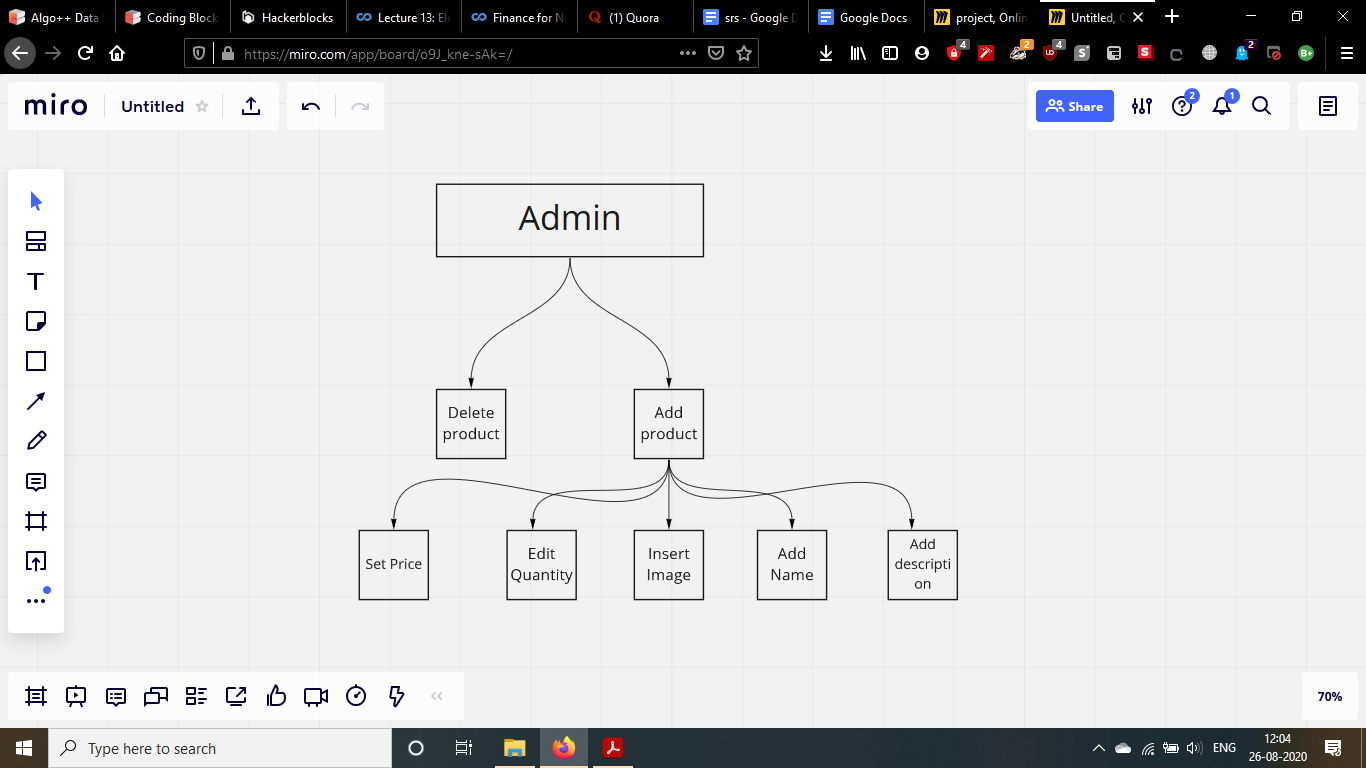
*This is a completely new project and is not a part of any other ongoing product.*

*A functional overview of the proposed website:*

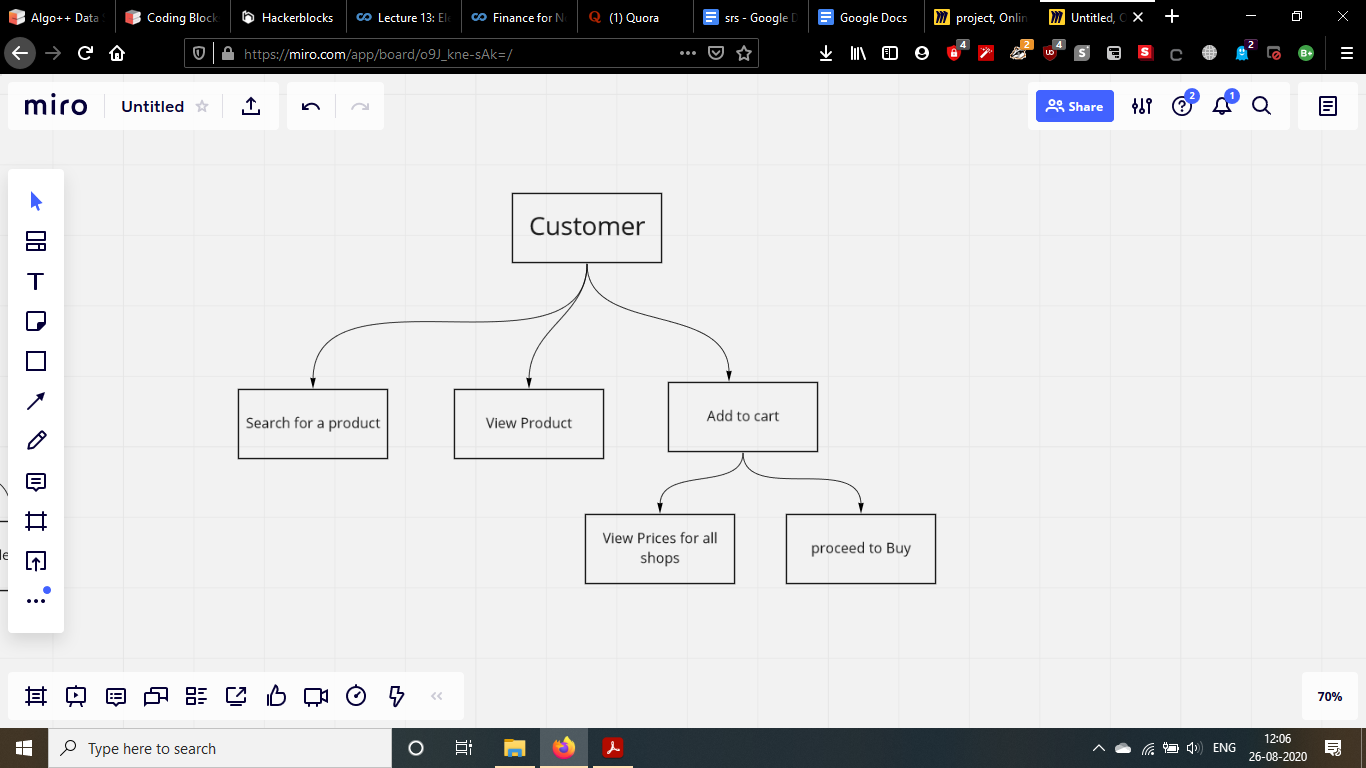


**2.2 Product Functions**

*Admin profile*

**

*Customer profile*

**

**2.3 User Classes and Characteristics**

*Various user levels that are involved in the whole system are;*

* *ADMIN*
* *CUSTOMER*

*The ADMIN should be able to do the following things:*

* *Access the database for their respective shops.*
* *Perform basic CRUD operation on the database*
* *Perform queries on database if required*
* *Edit their inventories.*

*The CUSTOMER should be able to do the following things:*

* *To view the various items available.*
* *To add any item into their carts.*
* *Get the information of any item like its price and shop available etc.*
* *Get a pdf of the items they’ve chosen with price analysis.*

**2.4 Operating Environment**

*1. The database used in the system is Firebase - Cloud Firestore by Google.*

*2. You need a node environment to run the server on any machine.*

*3. Nginx to load balance on the hosted website.*

**2.5 Design and Implementation Constraints**

*1. System constraints like memory, processor, server connectivity and hardware capability.*

*2. Costs associated with hosting the website.*

*3. Costs associated with hosting the database on a cloud platform.*

*4. Size of the database will be ever growing so maintaining an efficient database would be a challenge.*

**2.6 User Documentation**

* *A customer care number will be provided in case end customers or the shop owners face any kind of difficulties.*
* *Website design will be user friendly allowing users to easily navigate it.*
* *Various examples showing the user on how to operate the website.*

**2.7 Assumptions and Dependencies**

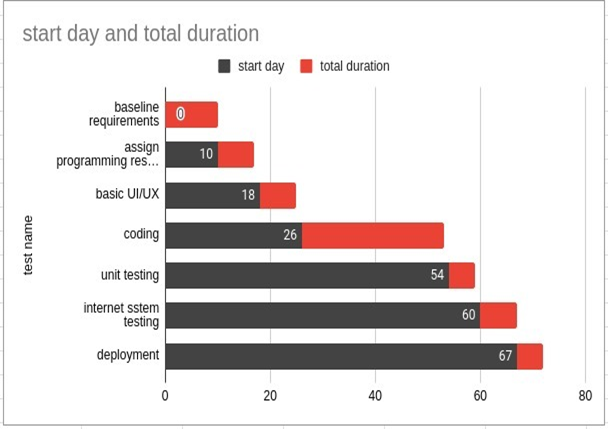
We are using blakeJS,cors,express,node-rsa for modifying the transmission between client-server architecture.

We are using body-parser,cookie-parser,dotenv,jsonwebtoken,path for back-end development.

We are using reactJS, material-UI, axios, mdb icons, react-router-dom for managing front-end .

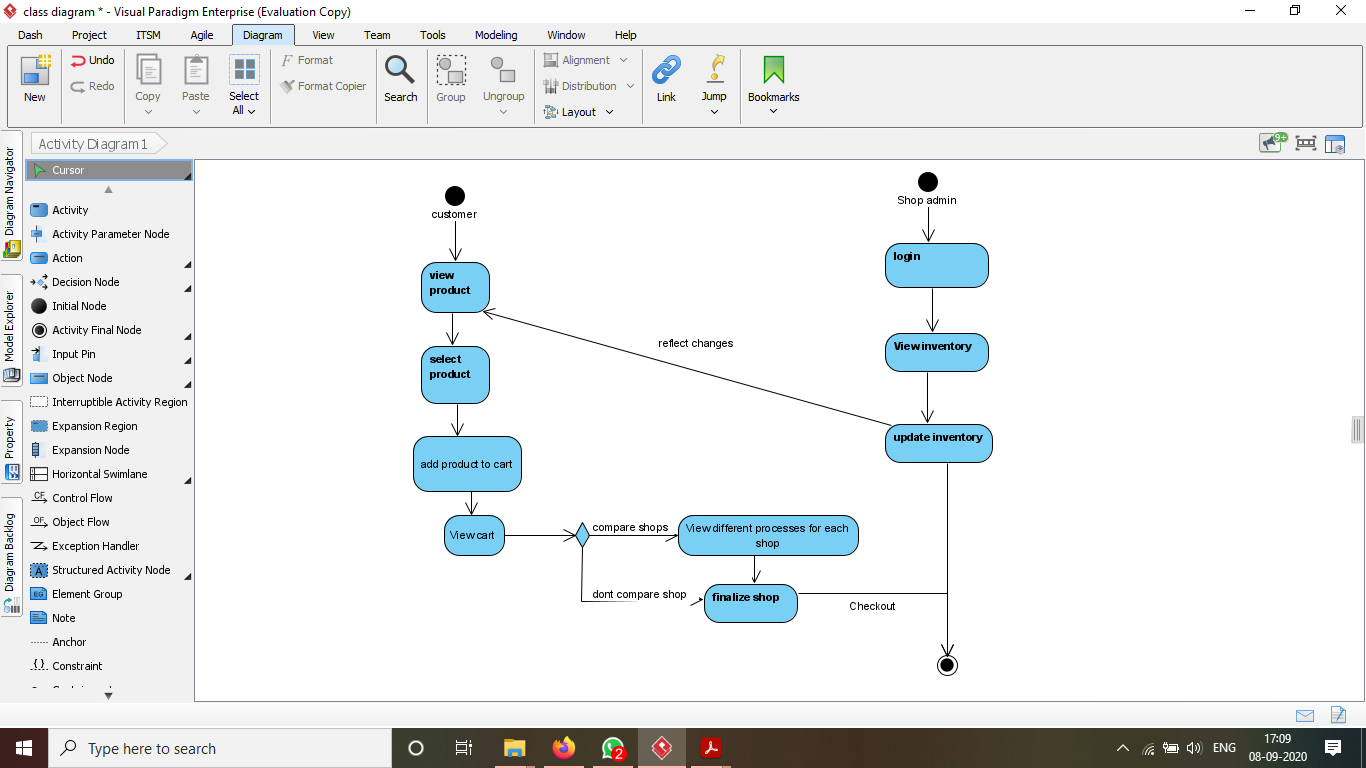
*We will be using google maps embedded API (free one) to locate the shops natively on the website.*

**2.8 Gantt Diagram**

****

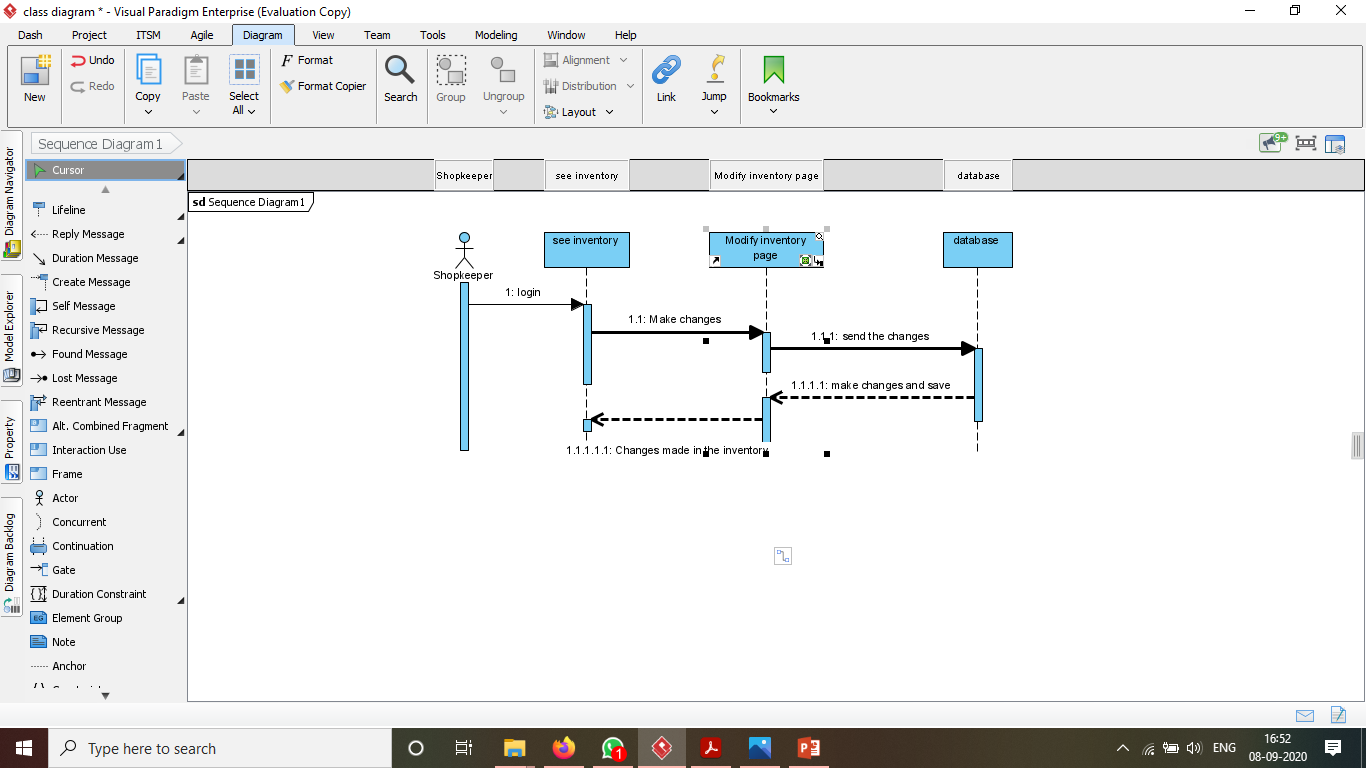
**2.9 UML Diagrams**

* Activity diagram

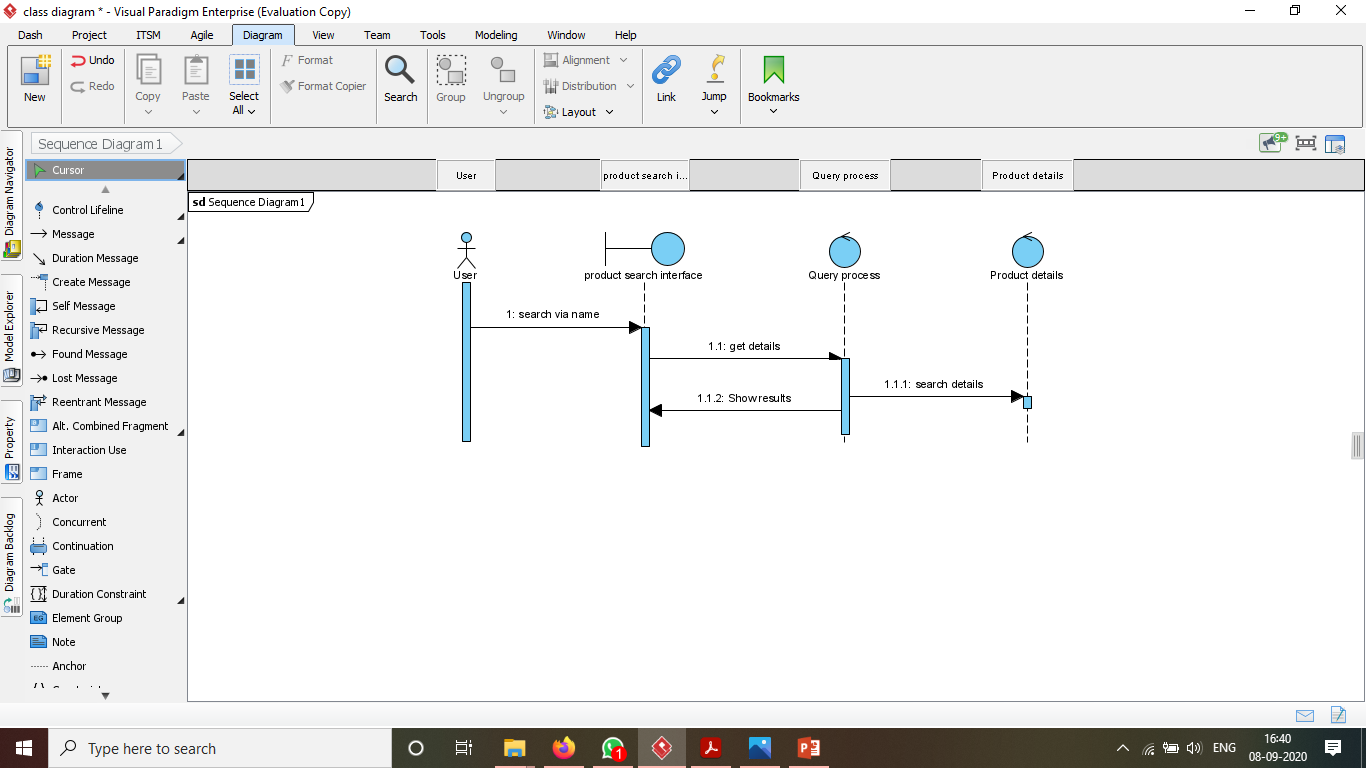


* Sequence Diagram

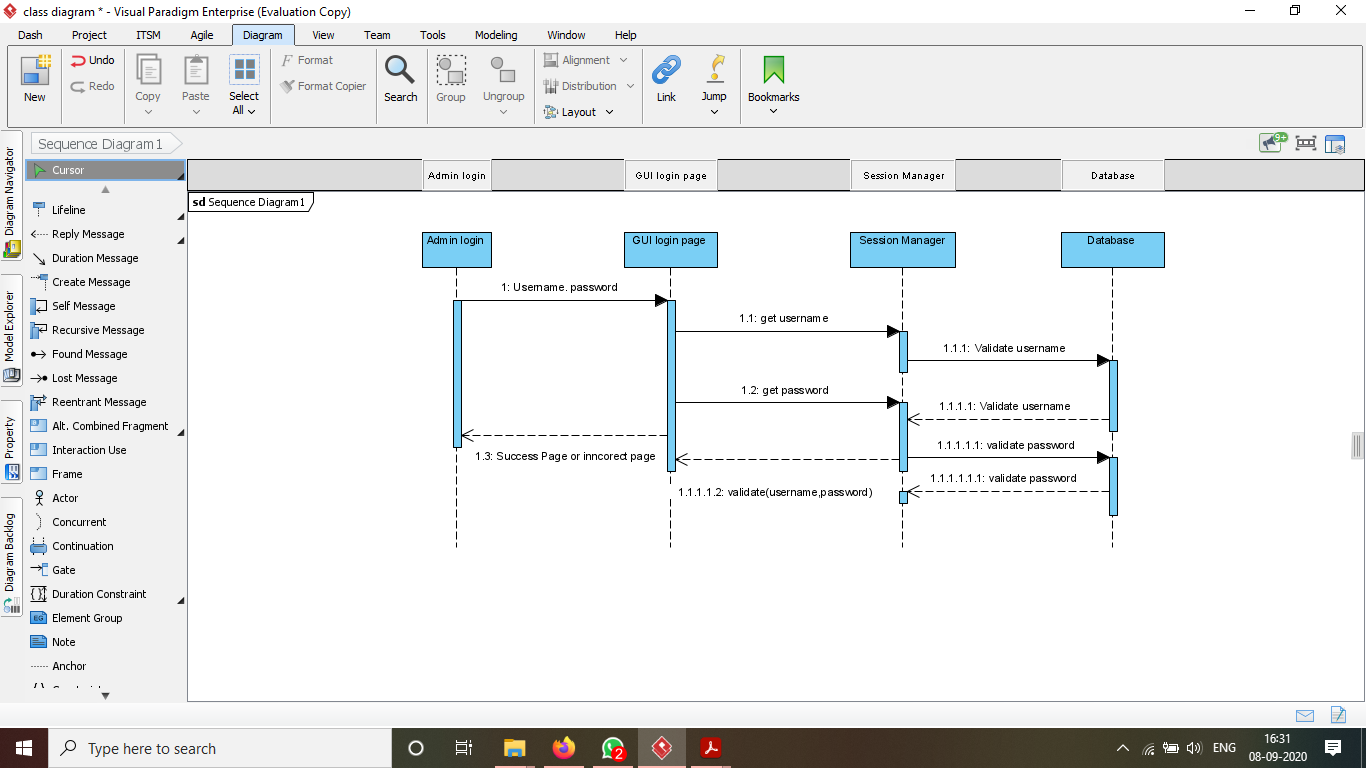
Shopkeeper profile



User Profile

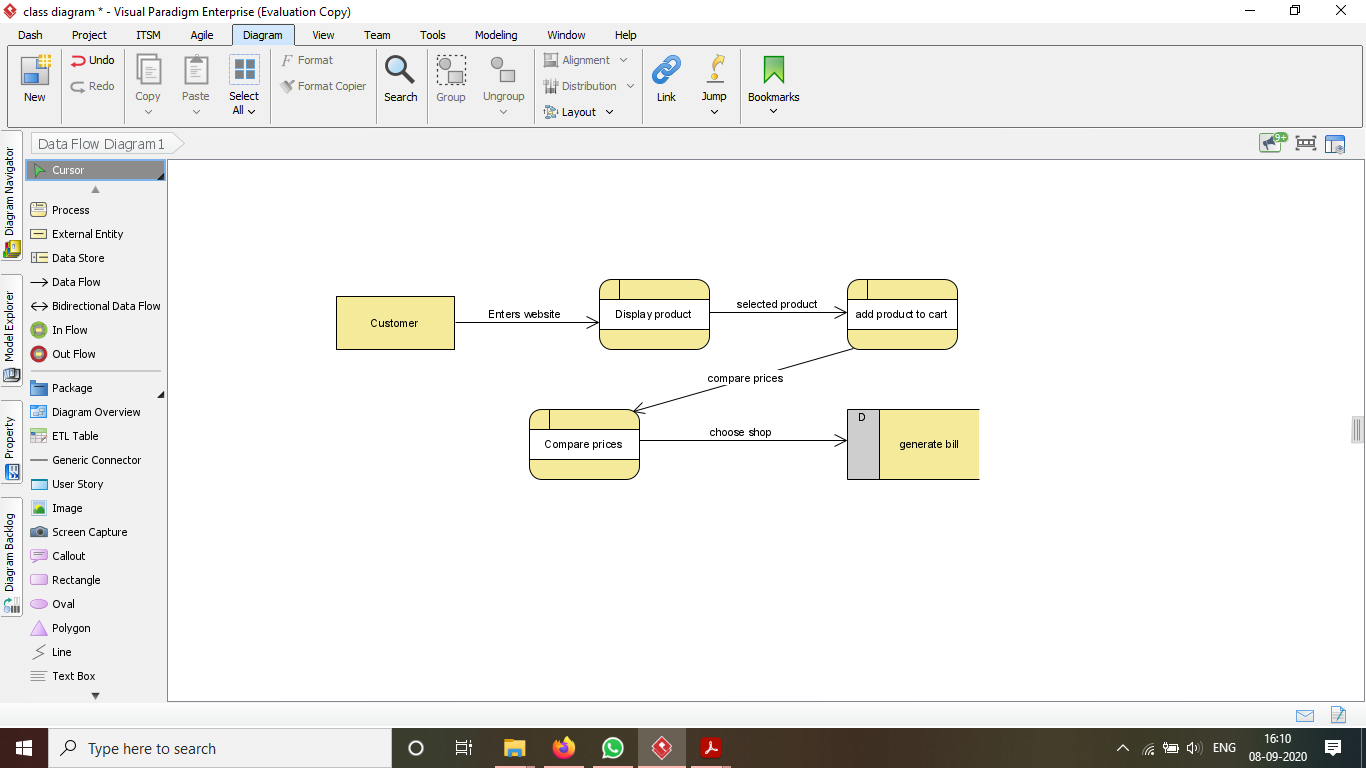


Admin login page

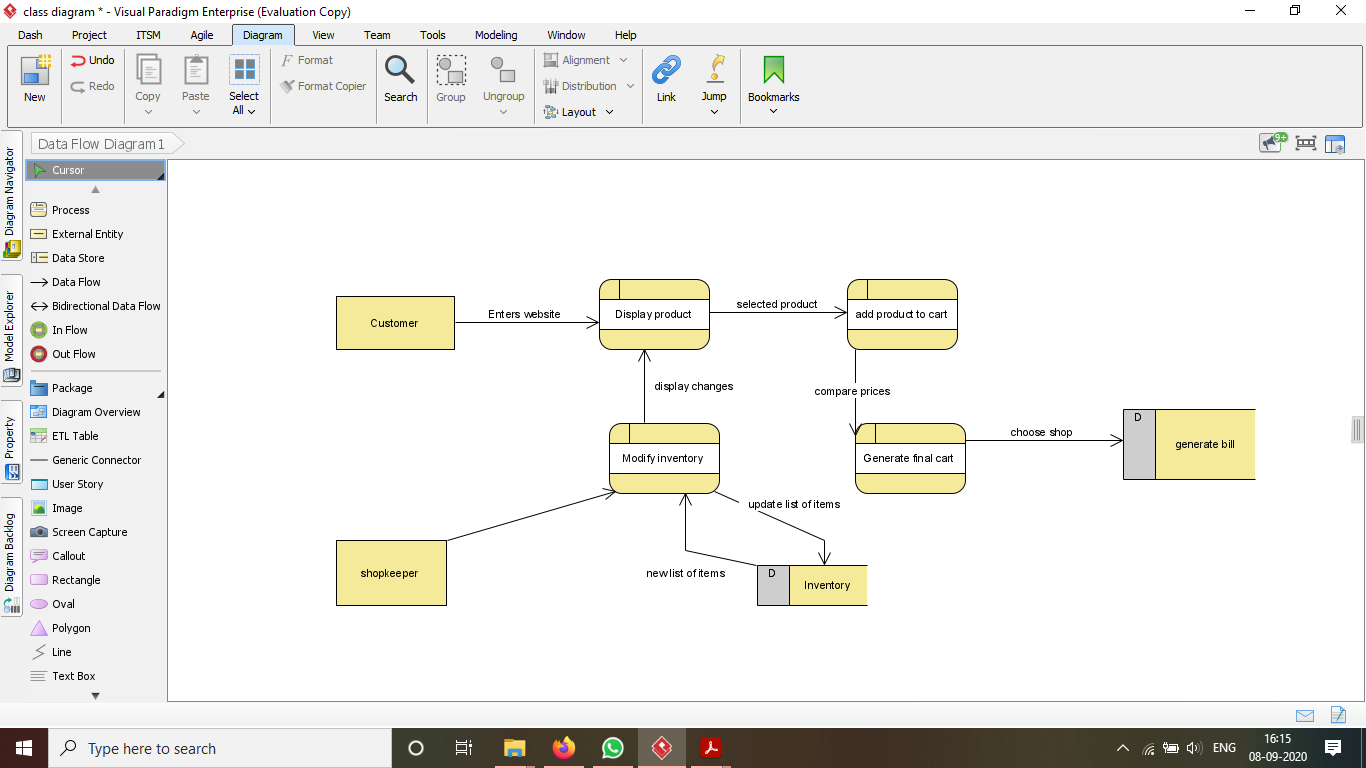


* Data Flow Diagram

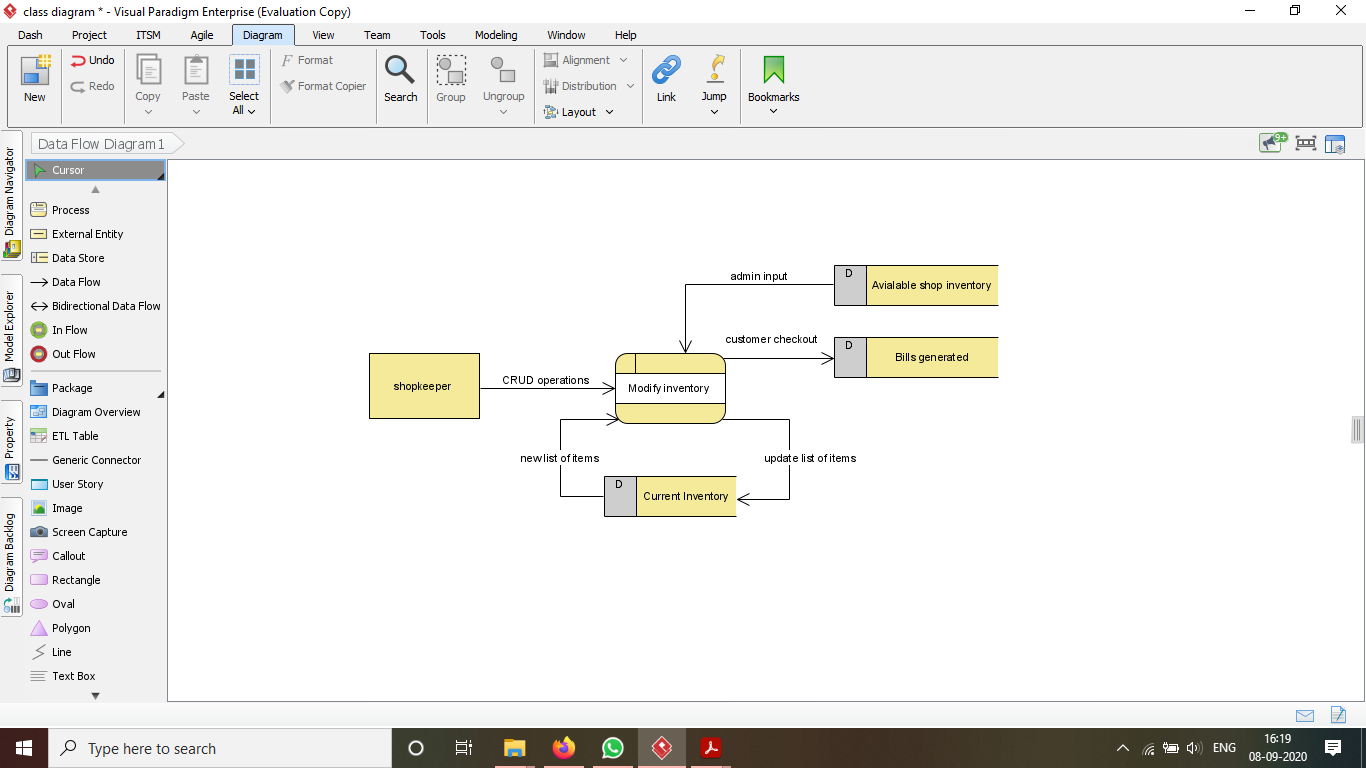
Level 0



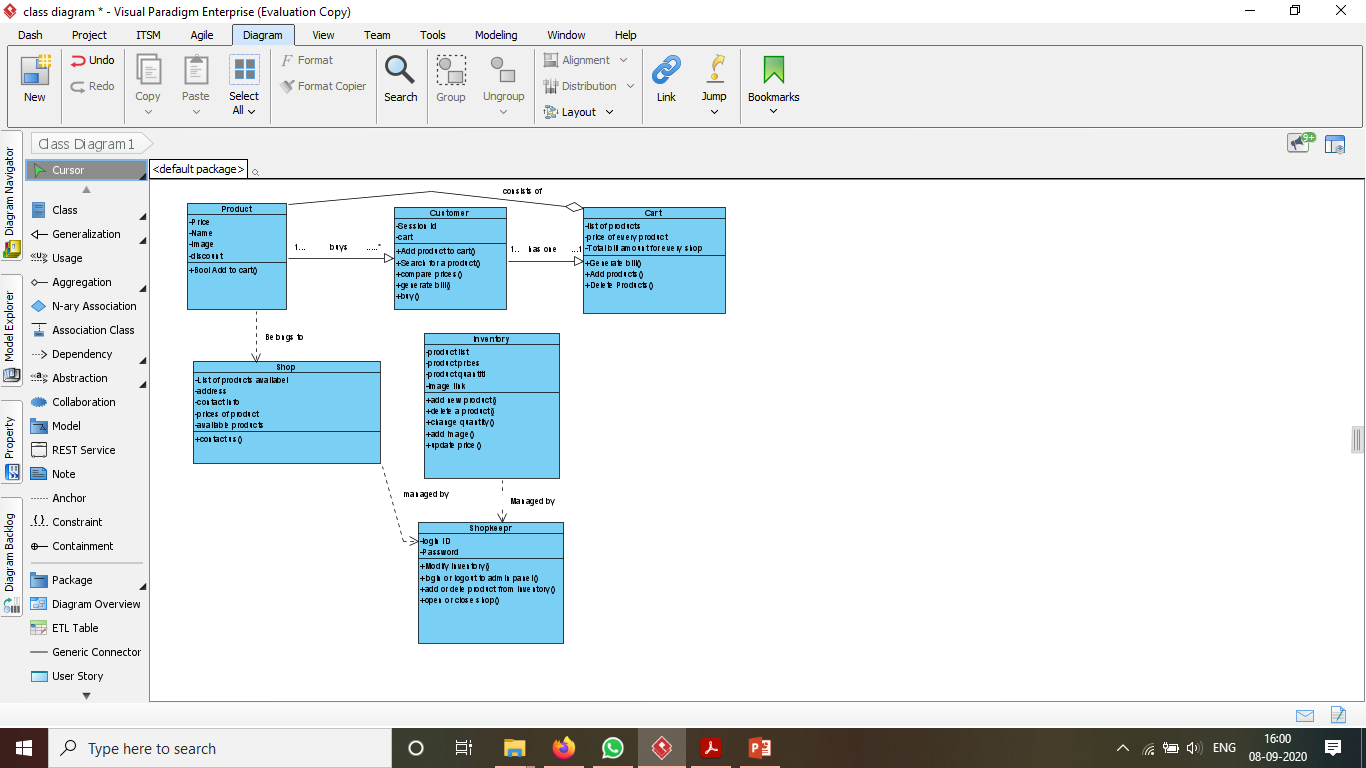
Level 1



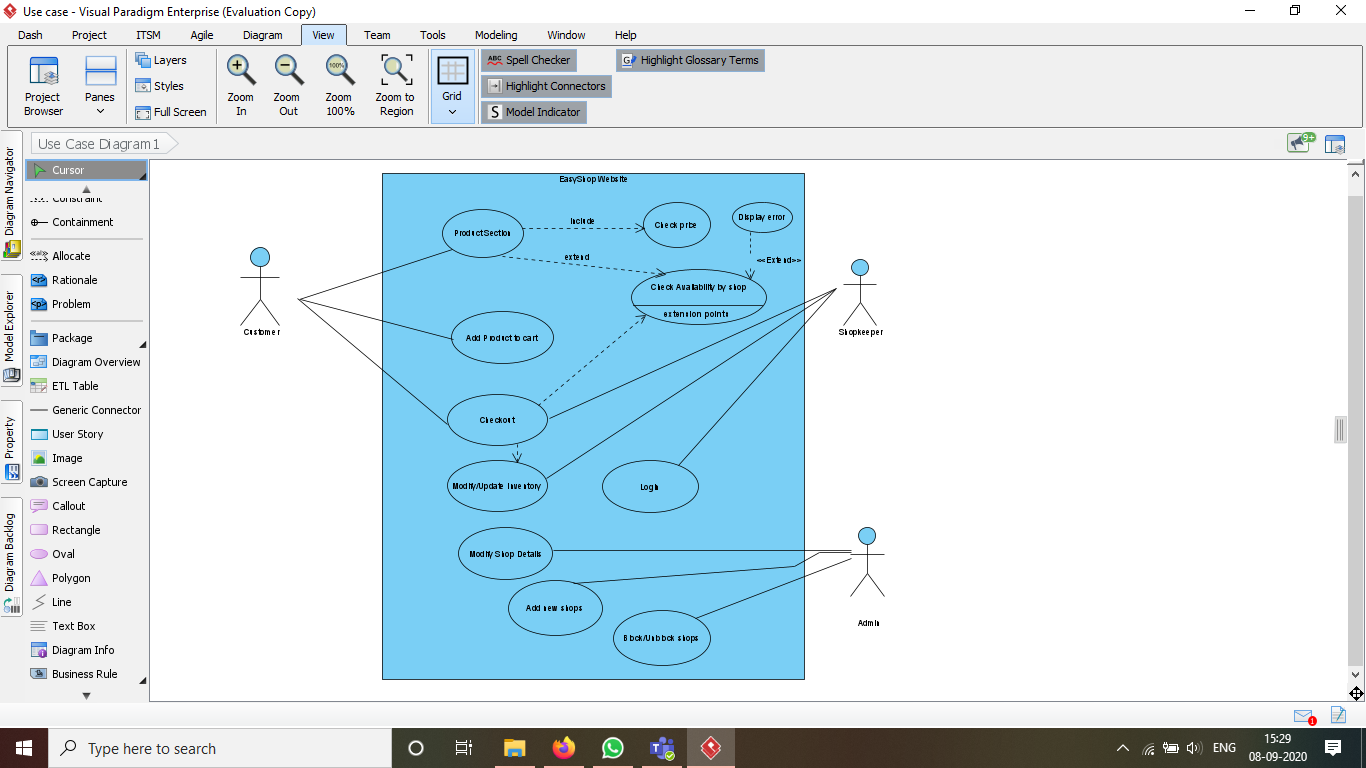
Level 2



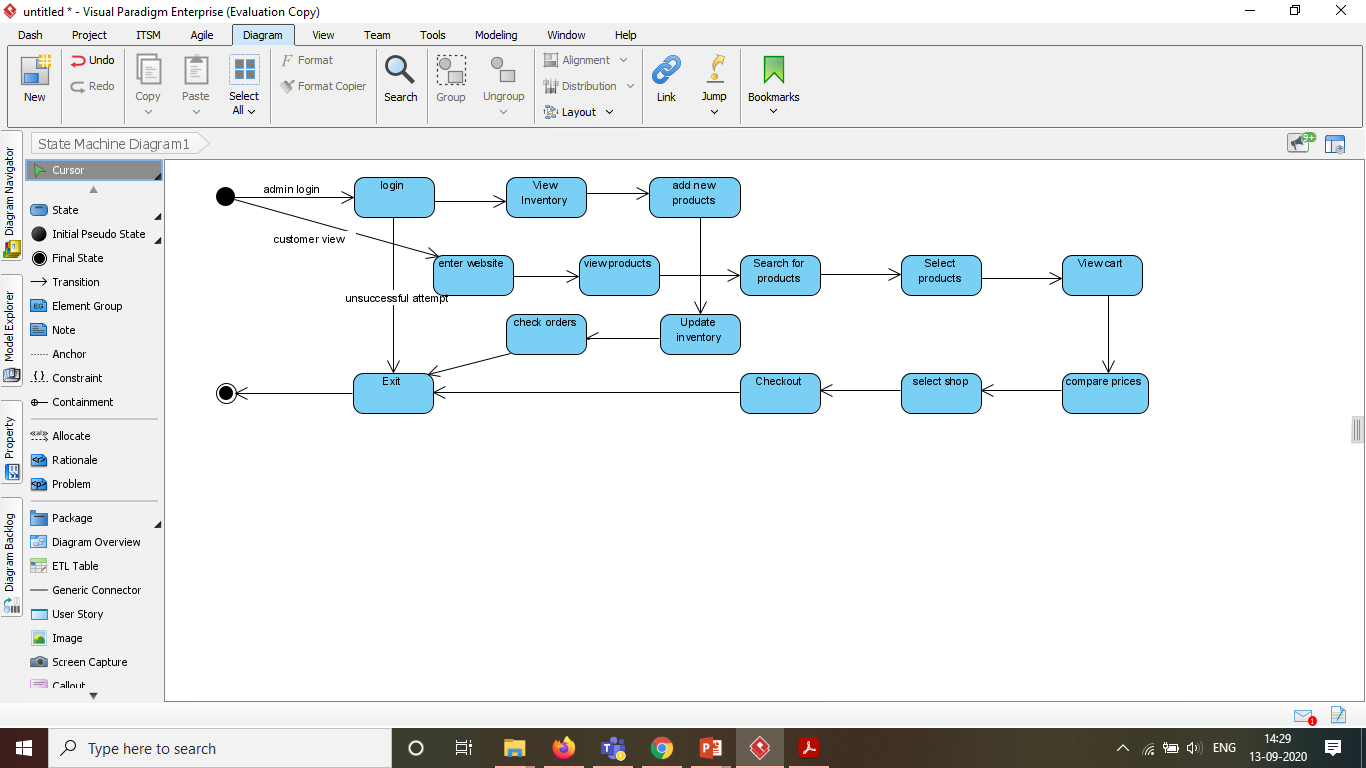
* Class Diagram



* Use case diagram

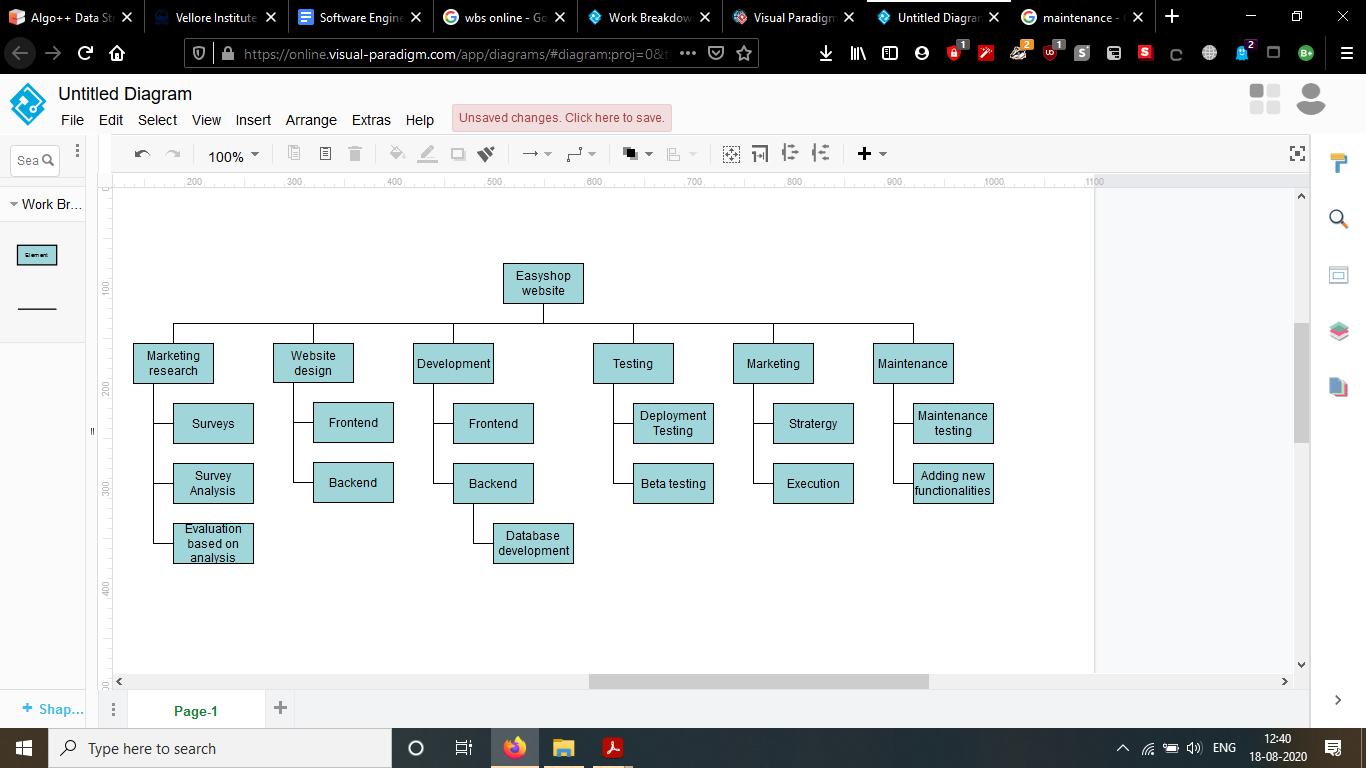


* **State diagram**

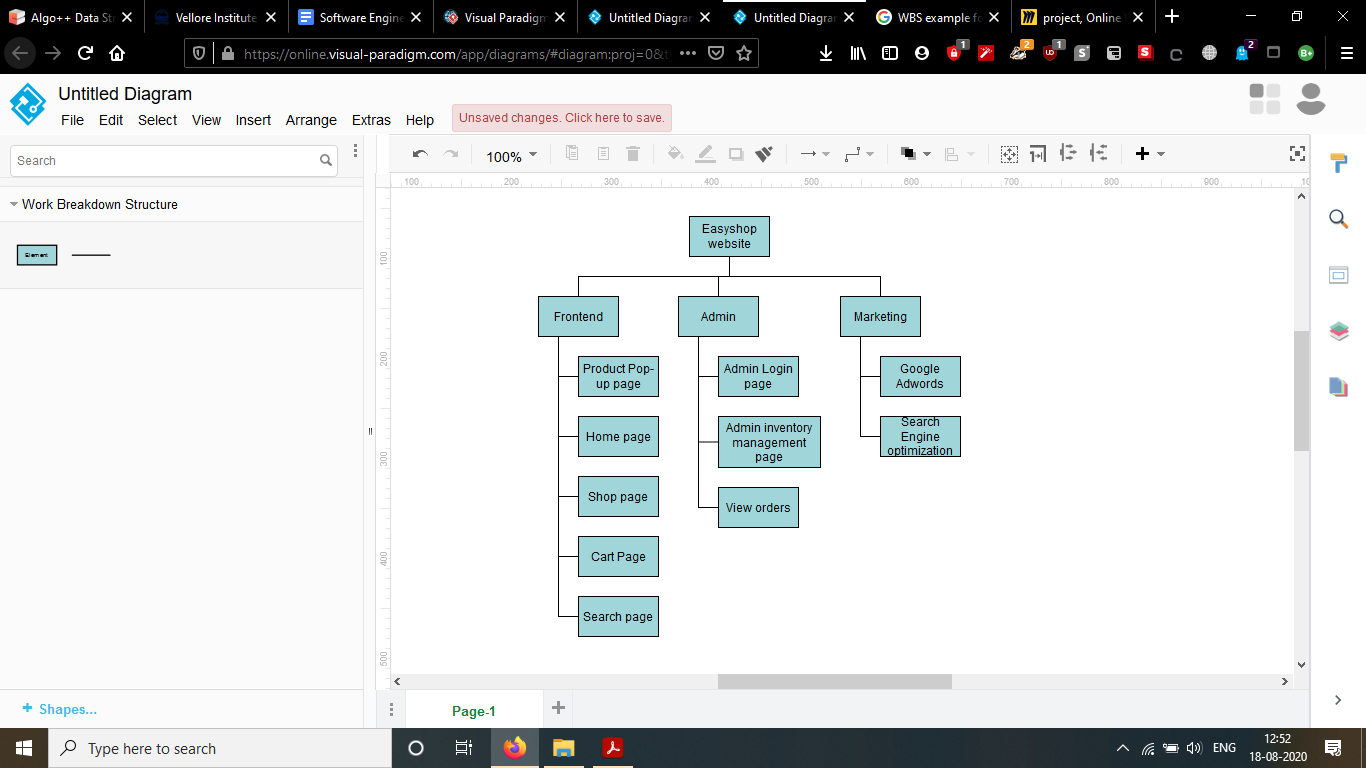
****

**2.10 Work Breakdown Structures**

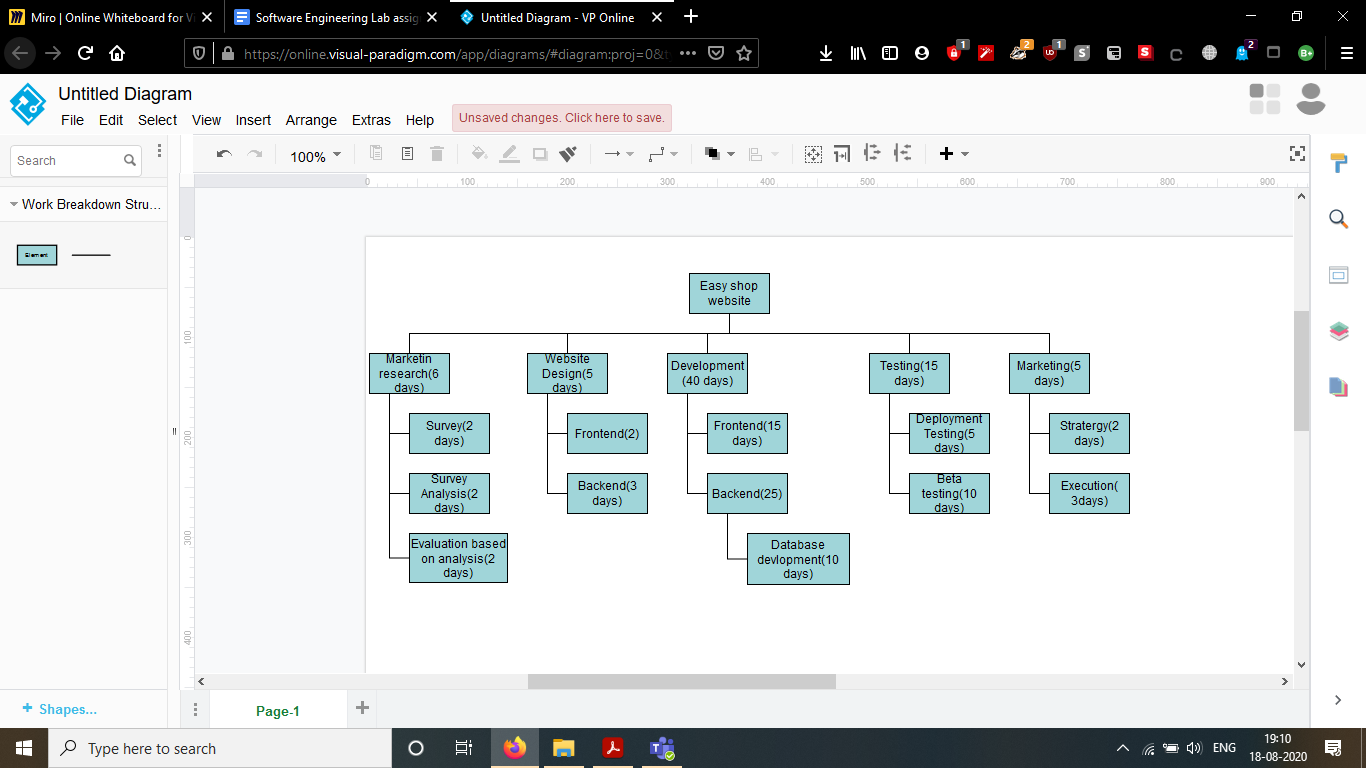
### Verb based



### Noun based



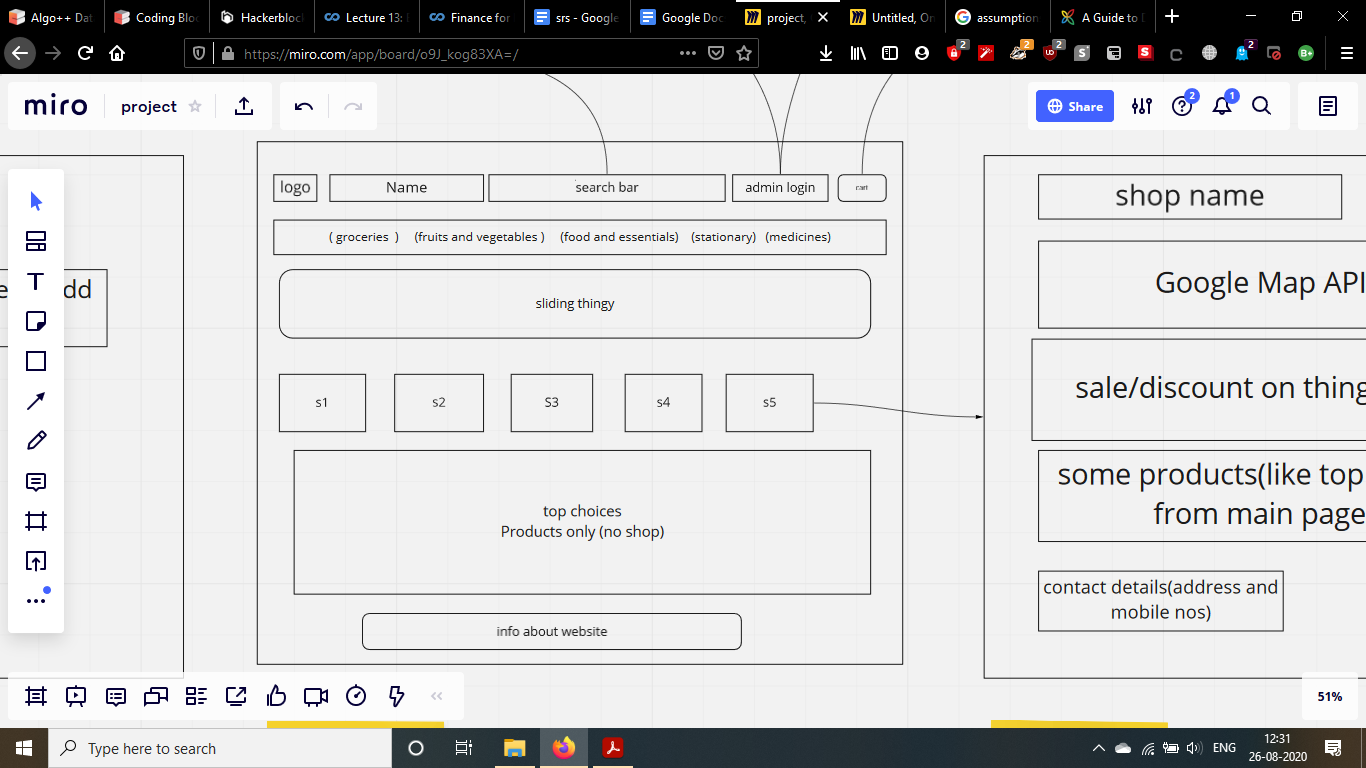
### Time based



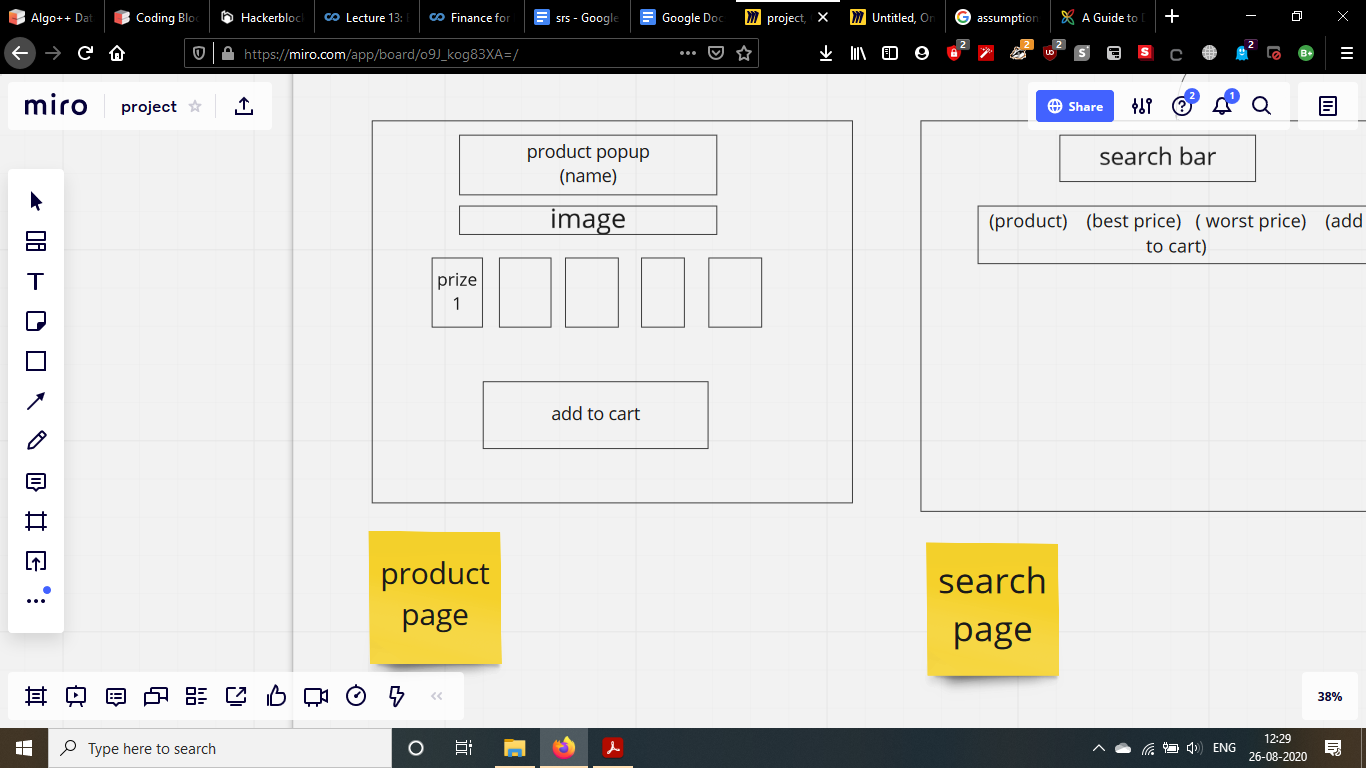
**3. External Interface Requirements**

**3.1 User Interfaces**

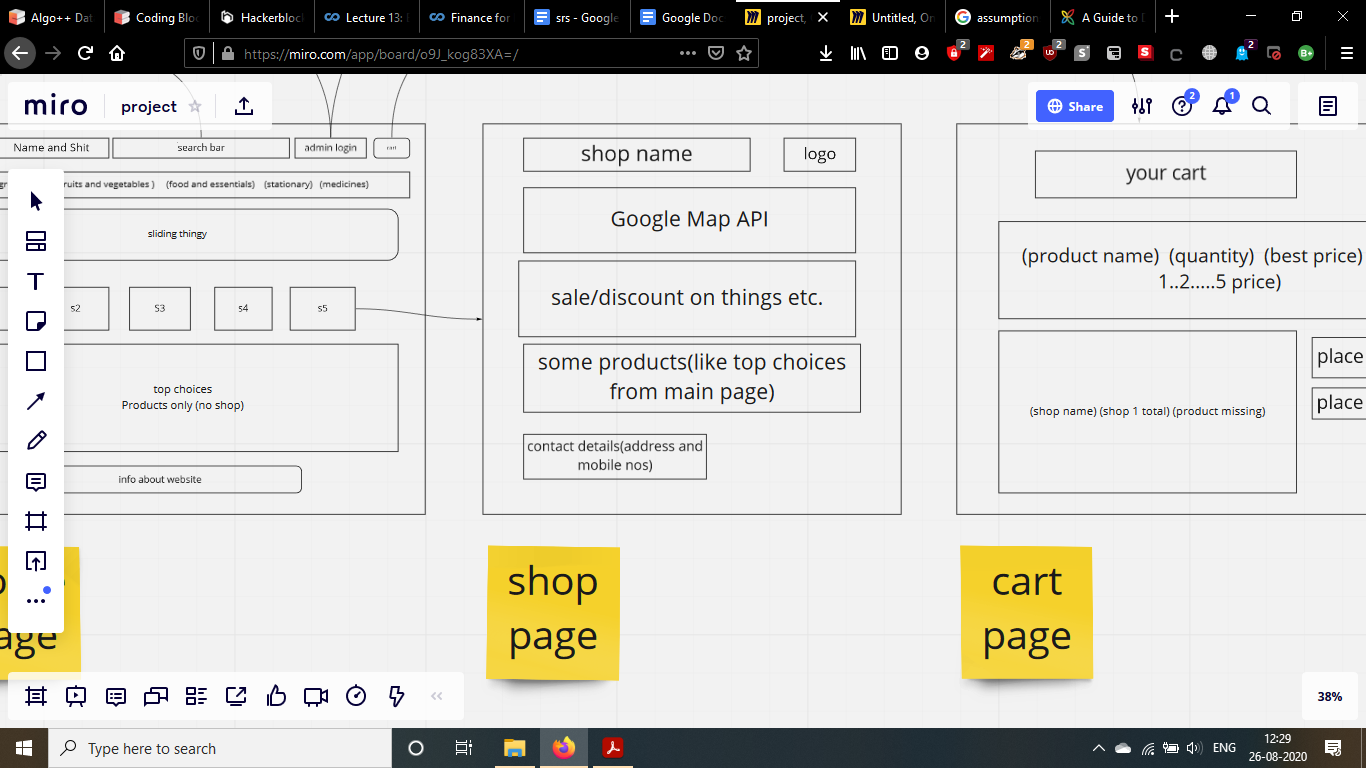
*Home Page of the website:*

**

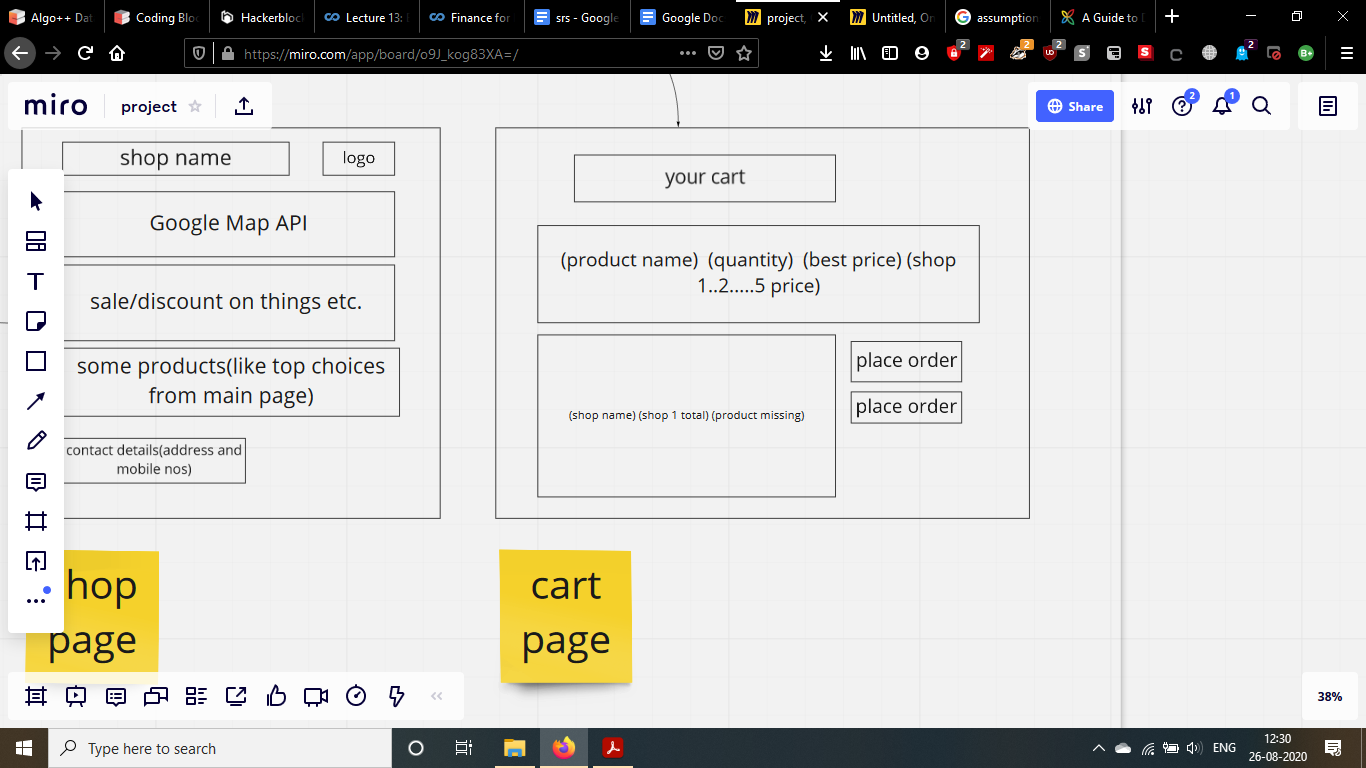
*Product popup page:*

**

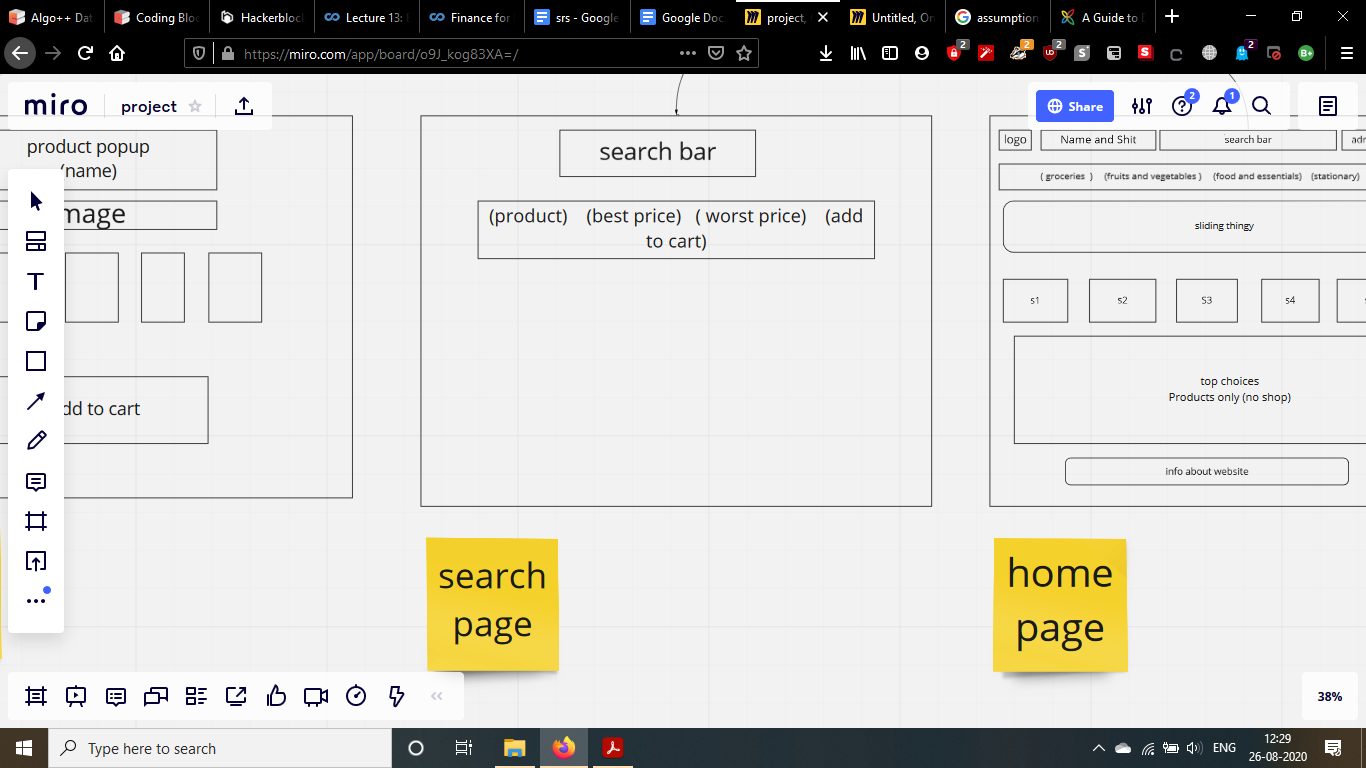
*Shop page:*

**

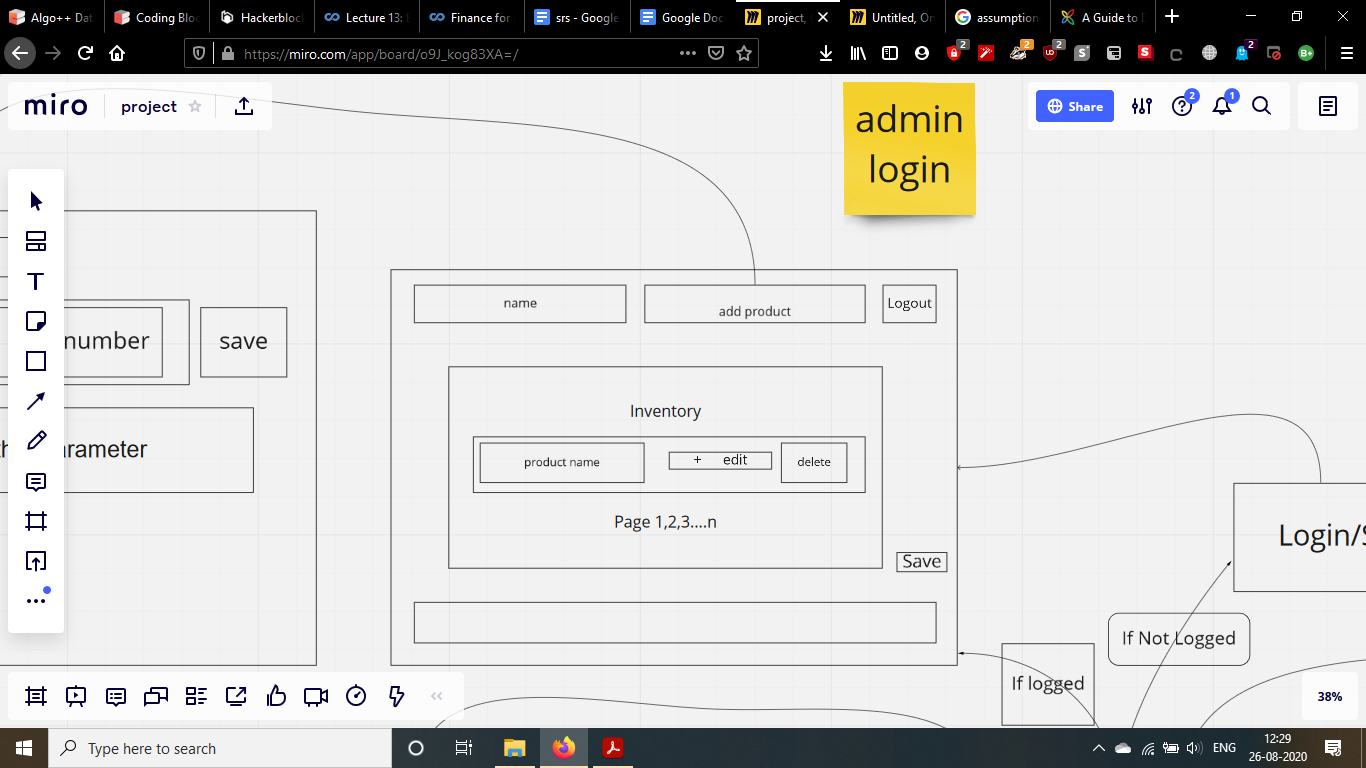
*Cart page:*

**

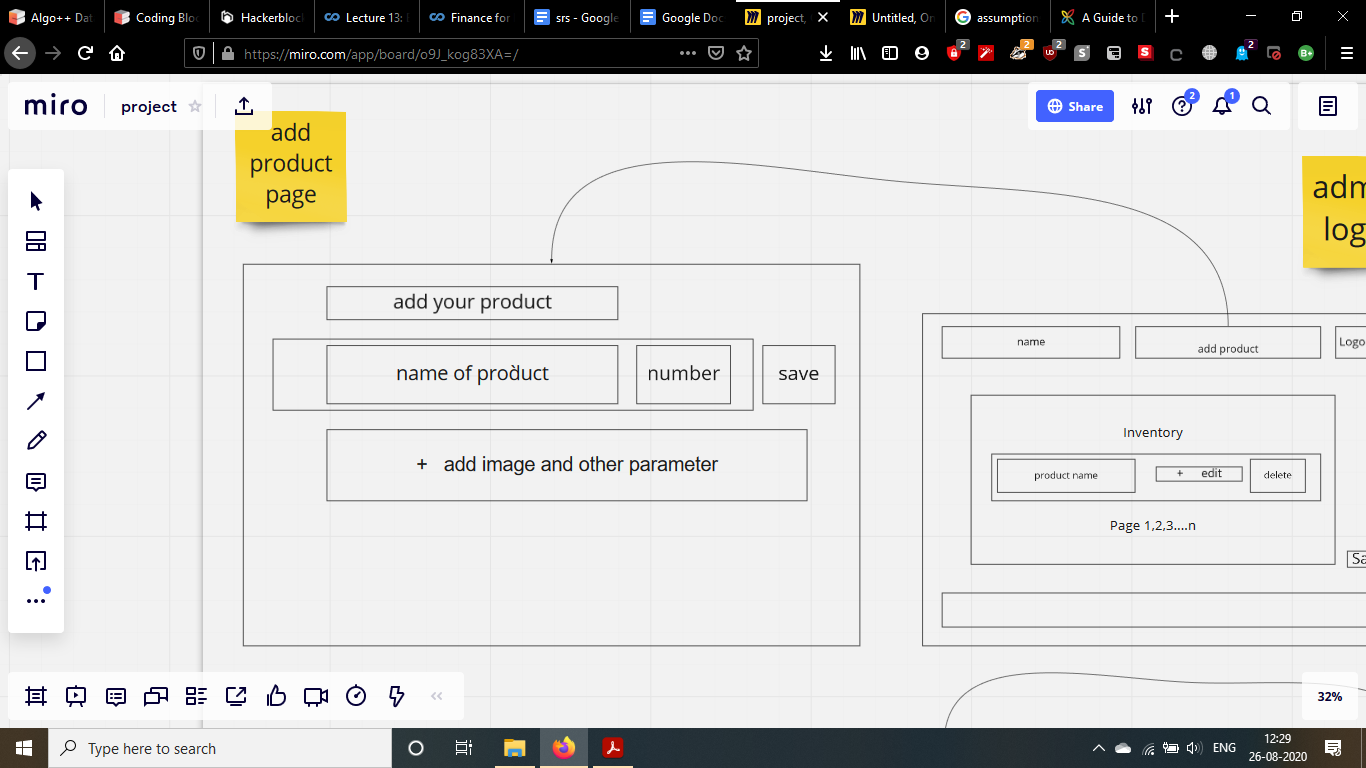
*Search Page:*

**

*Admin Login:*

**

*Admin edit inventory page*

**

**3.2 Hardware Interfaces**

*To host the website, min. HW requirement is that you need to run minimal linux flavour.*

*As the product is a website it requires no special hardware interface except for the device the customer/admin is using to access the website.*

**3.3 Software Interfaces**

Database : Firebase - Cloud Firestore by Google.

Operating System : Any OS that supports Modern Web Browsers.

Tools :

"blakejs": "^1.1.0", "body-parser": "^1.19.0", "cookie-parser": "^1.4.5", "cors": "^2.8.5", "crypto-js": "^4.0.0", "dotenv": "^8.2.0", express": "^4.17.1", "firebase-admin": "^9.0.0", "jsonwebtoken": "^8.5.1", "mongoose": "^5.9.9", "node-rsa": "^1.1.1", "path": "^0.12.7", "@material-ui/core": "4.10.0", "@material-ui/icons": "4.9.1", "axios": "^0.19.2", "classnames": "2.2.6", "clsx": "^1.1.1", "mdb": "^0.1.0", "mdbootstrap": "^4.19.1", "mdbreact": "^4.27.0", "moment": "2.26.0", "node-rsa": "^1.1.1", "node-sass": "4.14.1", "nouislider": "14.5.0", "prop-types": "^15.7.2", "react": "16.13.1", "react-datetime": "2.16.3", "react-dom": "16.13.1", "react-feather": "^2.0.8", "react-helmet": "^6.1.0", "react-router-dom": "5.2.0", "react-scripts": "3.4.1", "react-slick": "0.26.1",

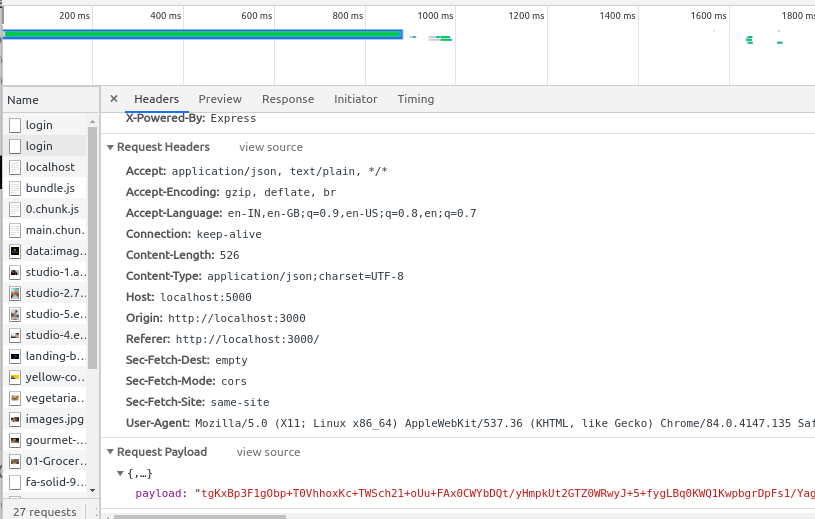
There will be no integrated commercial components

**3.4 Communications Interfaces**

*The communication between client and server is done with basic three-tier architecture.*

*To communicate we are using http/1.1 protocol which is the standard for web technologies now.*

*The request body is changed as opposed to conventional methods. We are encrypting the body and sending it as a payload to the backend. Backend decrypts it and does the normal operations.*

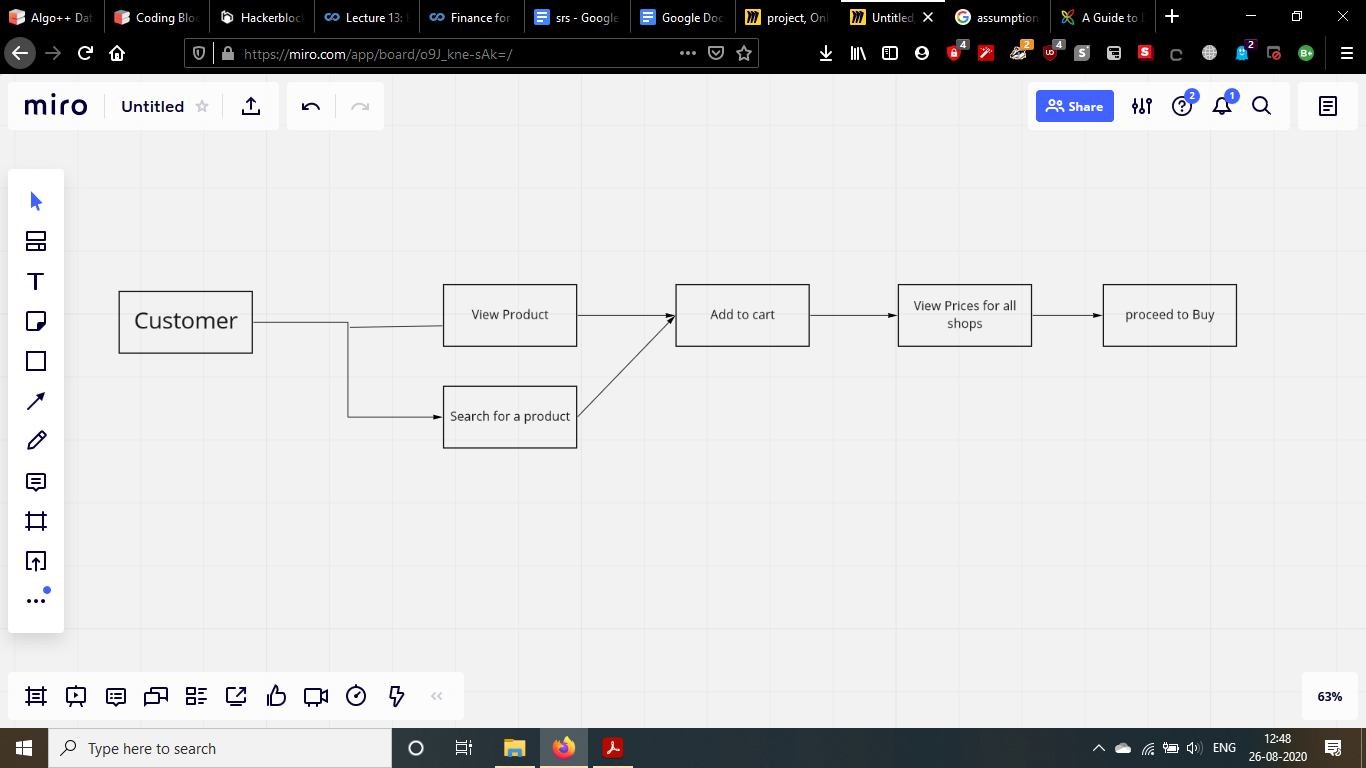
**

**4. System Features**

***4.1 Description and Priority***

*Customers enter the market and buy some products, we show them the various products that our partner shops offer, allowing them to search and add products in a cart. Then we generate the comparative bill for that customer. Admins will have a separate login ID and pin through which they can access their account and manage their shops inventory in real time.*

***4.2 Stimulus/Response Sequences***

**

***4.3 Functional requirement***

***4.31 ADMIN***

*An admin is a high priority who holds the authority to manage the working of their entire*

*Shop . Also, the admin is in-charge of managing the inventory of their establishments.*

***4.3.2 CUSTOMER***

*A customer is beneficially a high priority which can be any person who accesses the website and makes the purchase. The website should then connect the customer to the admin of the shop of their choice.*

**5. Other Nonfunctional Requirements**

**5.1 Performance Requirements**

*The website should be easily accessible via both laptop/desktops and mobile phones as most customers will prefer to order via their phones.*

*Also the database should be distributed, and NOSQL based so that it can be easily accessed by multiple users simultaneously.  
The website should also feel quick and responsive to the user to give them a convenient shopping experience.*

**5.2 Safety Requirements**

*The password of admins i.e the shopkeepers are being stored in hashed format instead of plain text format in the database. For this, we are using the blake2b hashing algorithm.*

*The hashing should be done in such a way that it should be hashed in controller logic itself...not after entering in the database. For that a function is defined which needs to be called while doing CRUD operations with the database.*

*We need to use IP filtering for preventing DDOS attacks on the server to ensure availability from the CIA triad.*

*Use of SSL certificate for establishing secure links between networked computers.*

**5.3 Security Requirements**

*Requires the use of RSA to encrypt the form data before sending it on the server. Passwords are protected in the database with the help of Blake2B hashing algorithm.*

*An SSL certificate to ensure secure links between networked nodes.*

**5.4 Software Quality Attributes**

*Availability and correctness of the information displayed is very important to the project as the prices/availability of the goods shown should match their real world status.*

*So reliability is also very important for the product.*

**5.5 Business Rules**

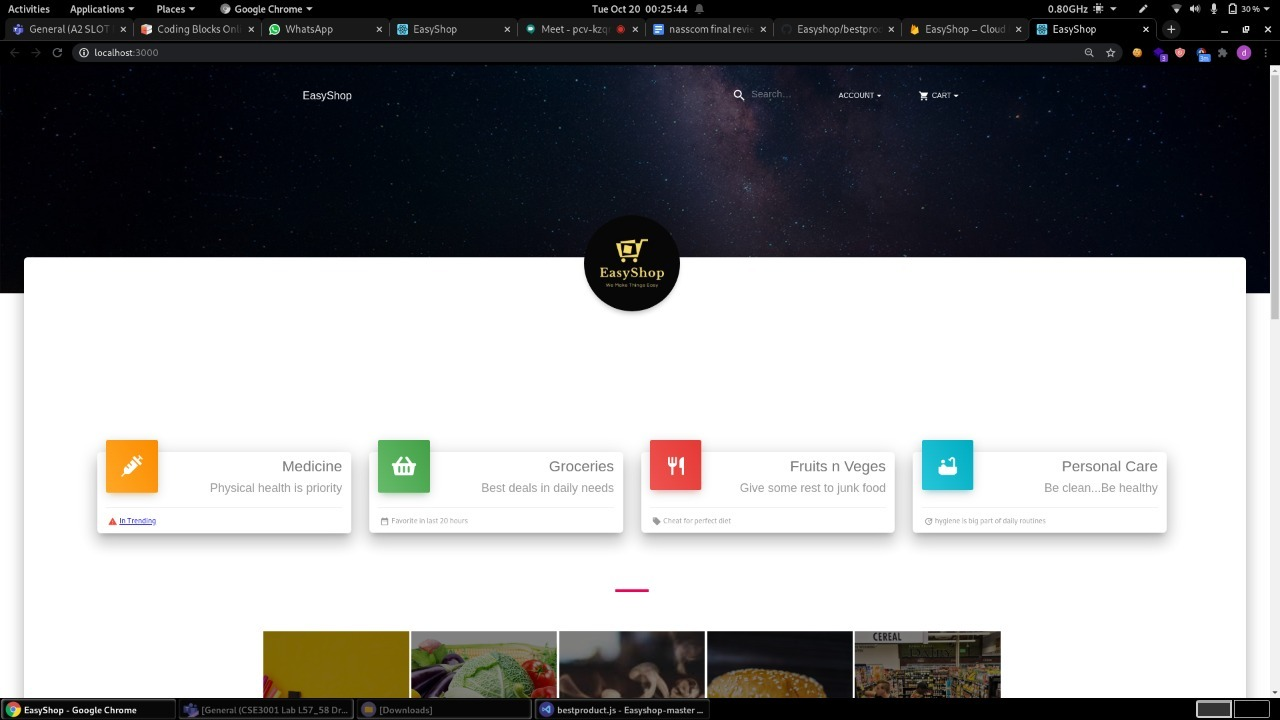
*The shop administrators are very important for the proper functioning of the website as they need to continuously update their inventory and sales onto the database so the customer can get correct information.*

**6. Final Output**

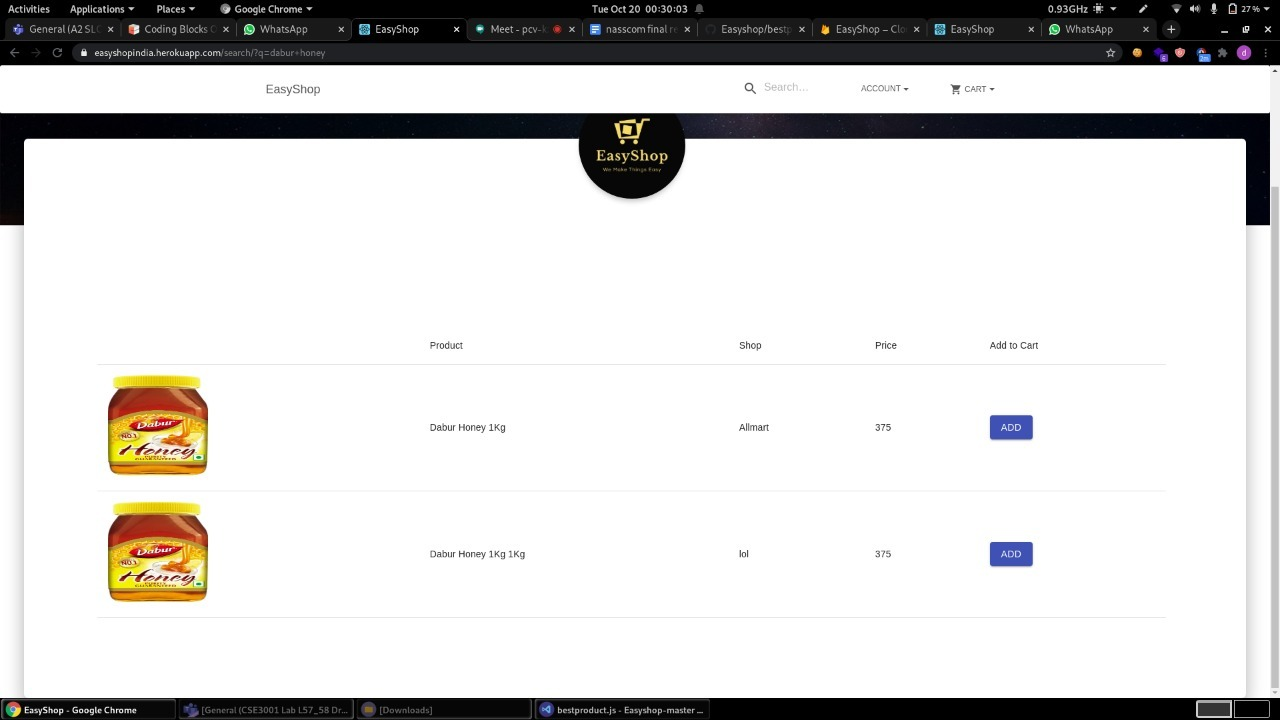
**6.1 website**

This is the final website that was developed for this project it can be accessed using the given link

Link: [http://easyshopindia.herokuapp.com**/**](http://easyshopindia.herokuapp.com/)

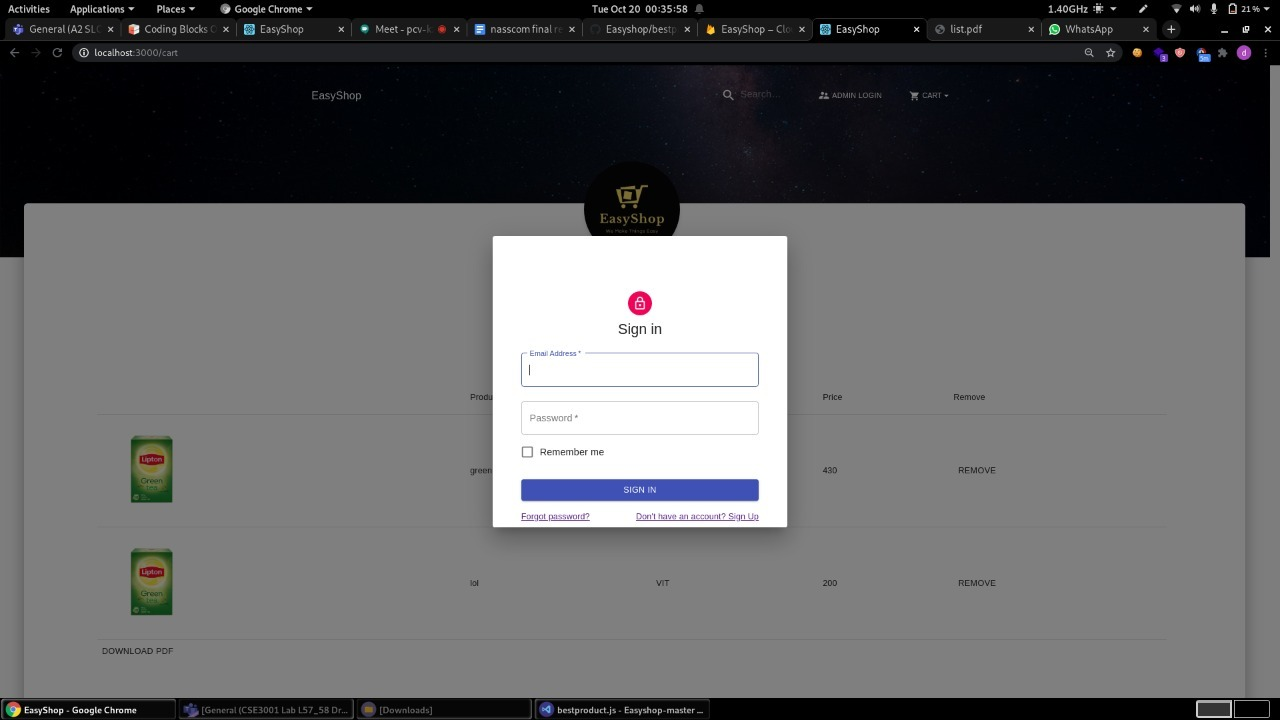
****

*Home page*

**

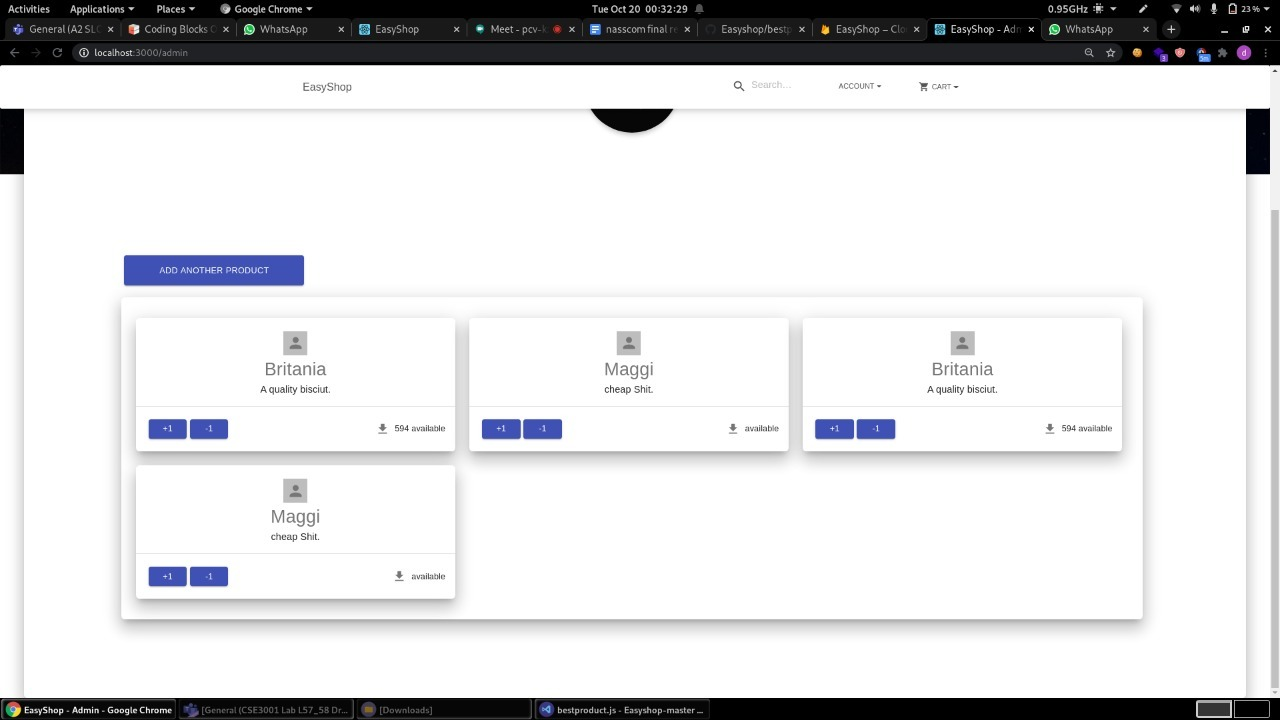
*Search page*

This is the search page. Upon searching for the product, this page will show up with the matched products.

**

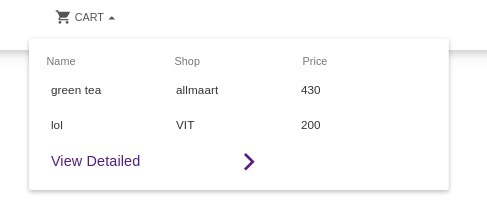
*Admin Login*

The login page where shopkeepers can login to the website to manage their inventory.

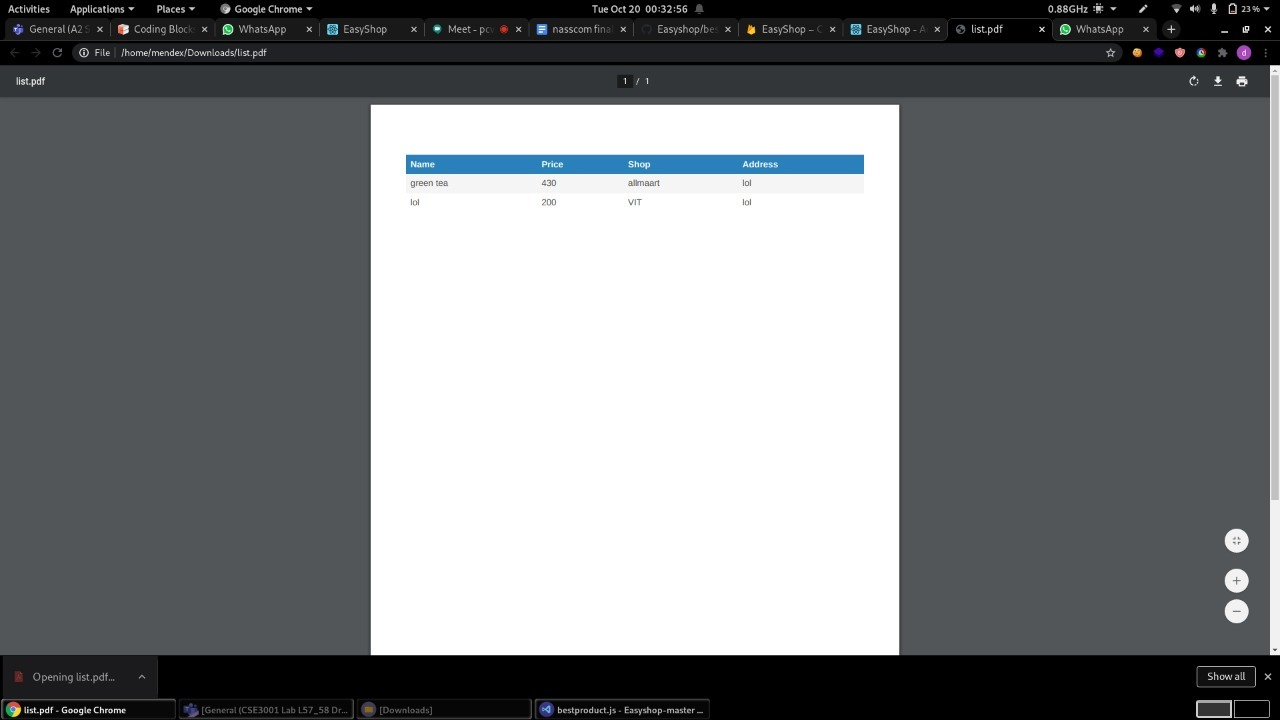
**

*Admin Inventory panel*

After the shopkeeper logs in, he will see this page. He can update his inventory with this page.

**** *Fig. 4.5 cart*

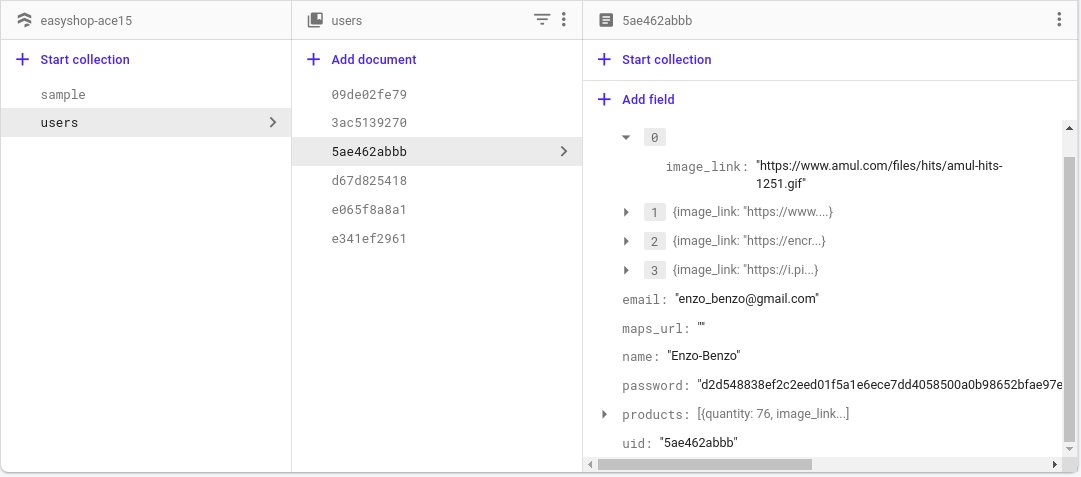
This is a quick cart, the user can quickly checkout his cart here instead of having to go to a different page to see the cart.

**

*Checkout*

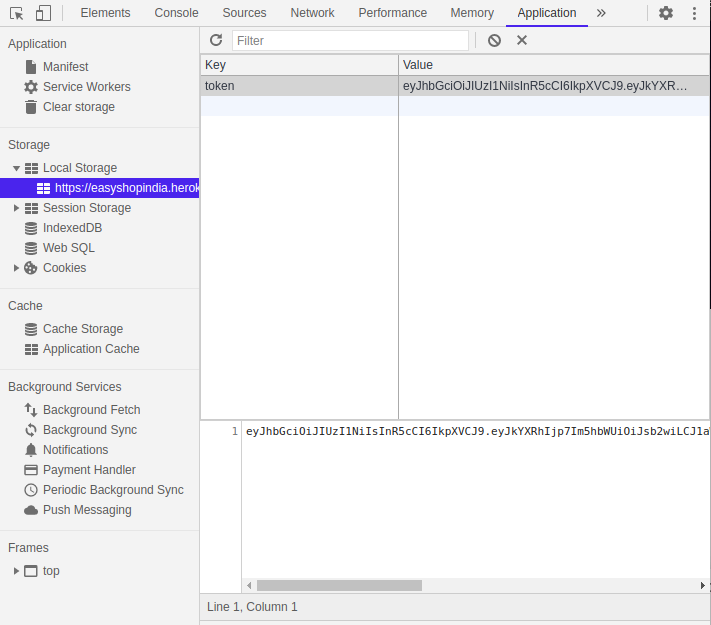
After the user “checks out”, a pdf will be downloaded giving the approx amount needed for the purchase and other things.

**6.2 Security Aspects**

****

*Hashed Password Stored in Database*

We store our users’ data in hashed format. So if in any case there is database breach, hacker can’t see the details of our user

****

*Encrypted JSON web Token*

We also take care of the request that our website makes. Our website will all request in just one line of “payload”. So it will be very secure in case like the hacker will try to use a burp suite then the hacker can see the only “payload” that is our request to the server.

**7. Code Optimizations**

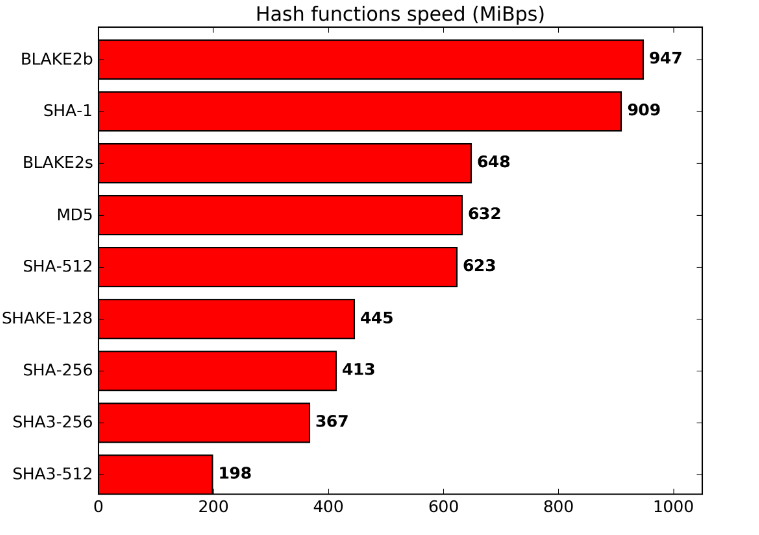
We updated our hashing algorithm from sha256 to a faster and secure blake2b hashing algorithm. While both have the same algorithmic time complexity, runtime complexity of blake2b is far more better than sha256. You can see that in picture below,

**Old Code:**

|  |
| --- |
| const db = require('../models/firebaseSDK').db; const sha256 = require('sha256');  module.exports = {  addUser : async (req,res) => {  const { name, email, password } = req.body;   const uid = sha256(email).slice(0,10);  await db.collection('users').doc(uid).set({  uid,  name,  email,  password:sha256(password)  }).then((data)=>{  console.log(data);  res.status(200).send('success');  }).catch((err)=>{  console.log(err);  res.status(503).send('error');  })  },   deleteUser : async (req,res) => {  const { uid } = req.body;   await db.collection('users').doc(uid).delete().then((data)=>{  res.status(200).send('success');  }).catch(err=>{  console.log(err);  res.status(503).send('error');  });   },   updateUser : async (req,res) => {  const { uid, updateFields } = req.body;  if(updateFields.password)  updateFields.password = sha256(updateFields.password)    await db.collection('users').doc(uid).update(updateFields).then((data)=>{  res.status(200).send('success');  }).catch(err=>{  console.log(err);  res.status(503).send('err');  })  } } |

**New Code:**

|  |
| --- |
| Const db = require('../models/firebaseSDK').db; const blake2b = require('blakejs').blake2bHex;  module.exports = {  addUser : async (req,res) => {  const { name, email, password } = req.body;   const uid = blake2b(email).slice(0,10);  await db.collection('users').doc(uid).set({  uid,  name,  email,  password:blake2b(password)  }).then((data)=>{  console.log(data);  res.status(200).send('success');  }).catch((err)=>{  console.log(err);  res.status(503).send('error');  })  }, deleteUser : async (req,res) => {  const { uid } = req.body;   await db.collection('users').doc(uid).delete().then((data)=>{  res.status(200).send('success');  }).catch(err=>{  console.log(err);  res.status(503).send('error');  });   },   updateUser : async (req,res) => {  const { uid, updateFields } = req.body;  if(updateFields.password)  updateFields.password = blake2b(updateFields.password)    await db.collection('users').doc(uid).update(updateFields).then((data)=>{  res.status(200).send('success');  }).catch(err=>{  console.log(err);  res.status(503).send('err');  })  } } |



Src : <https://blake2.net/>

**8. Testing**

**8.1 Basic Website Testing**

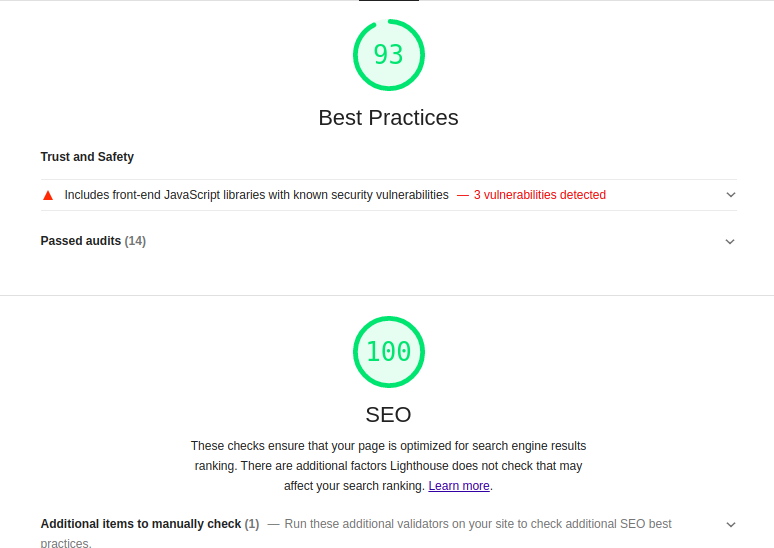
As the work we are doing involves mostly APIs, we used postman to test out the APIs. The results of the tested APIs are below.

<https://documenter.getpostman.com/view/10291679/TVK5e2bN> We have published the postman collection with respective test APIs and their responses.

|  |  |  |
| --- | --- | --- |
| Test | Expected Output | Observed Output |
| Testing secured API without auth header | “noAuthHeader” |  |
| Testing secured API with wrong / tempered auth header | “JWTVerifyFailed” |  |
| Test secured route with proper Auth header | Responds successfully with data |  |
| Successful Login | Responds with JWT token signed from the backend. |  |
| Login with wrong credentials | “invalidPass” |  |
| Try to make POST requests by tempering data (intercept attacks) | “RSADecryptError” |  |

**Google Lighthouse**

Here you can see we have achieved 100 in Search Engine Optimization without any paid plugin like the other websites and we have also achieved 93 in best practice which denotes that our code is well optimized for the compilation.

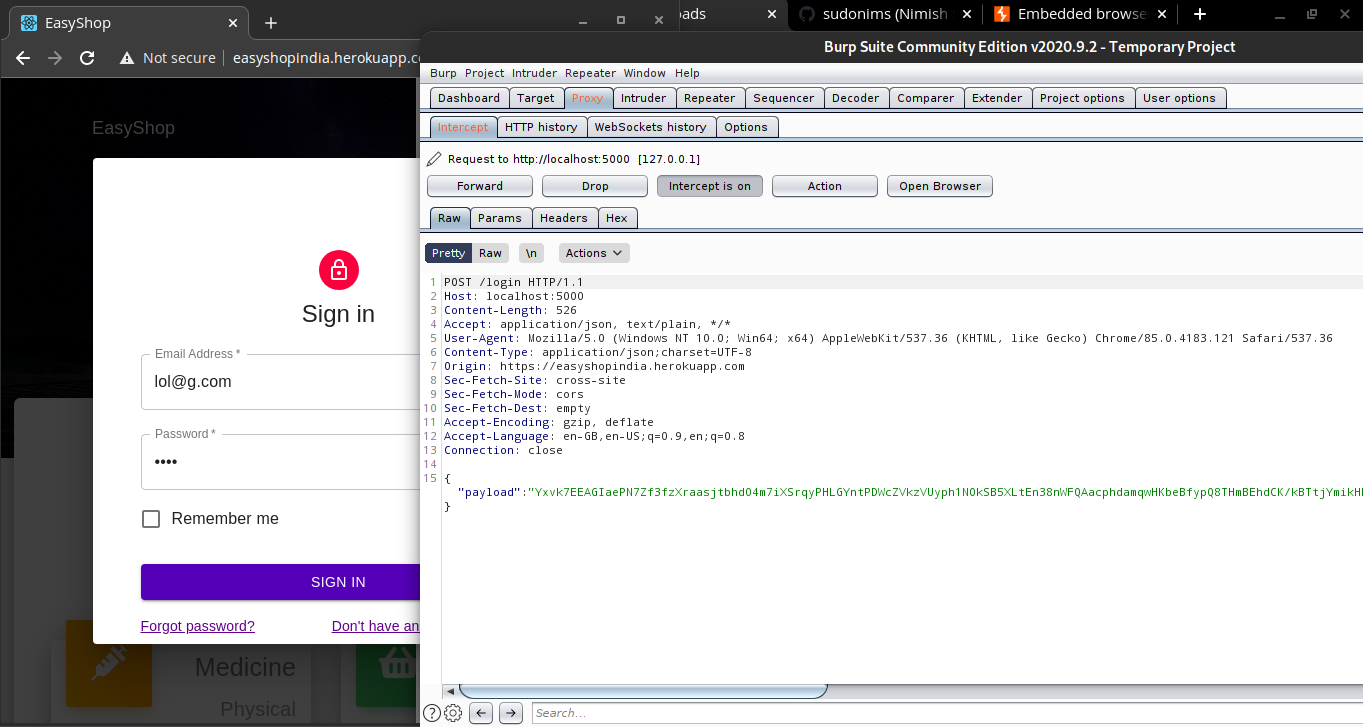


*Google Lighthouse testing*

**8.2 Security Testing**

* Burp Suite Intercept testing

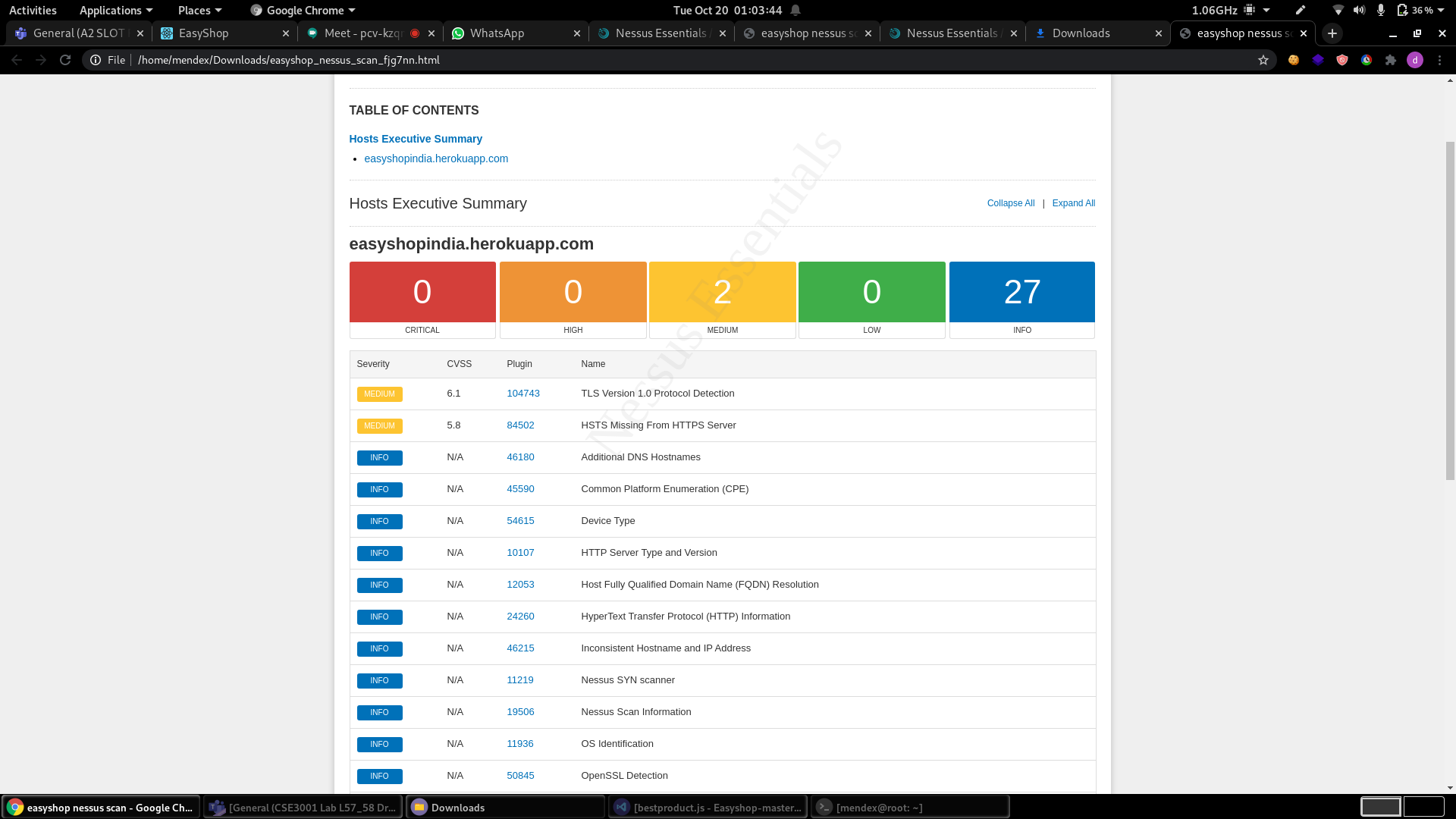
In burpsuite we use fuzzer to acknowledge all the vulnerability in our website. Via fuzzer we can see the all post request that will be made by website to the database or server. So we intercept all the traffic for our particular website and we can see all the requests have been made to the database server via our website. And we can clearly see our website made a login post request to the database server and it shows “payload” instead of login-id and password information. And you can’t see the exact data that the user has entered, instead the raw data penetration tester only sees a hash of the request. So you can’t see the exact data nor you can see what type of data request has been sent. It will send all types of requests in just one line of “payload”.

****

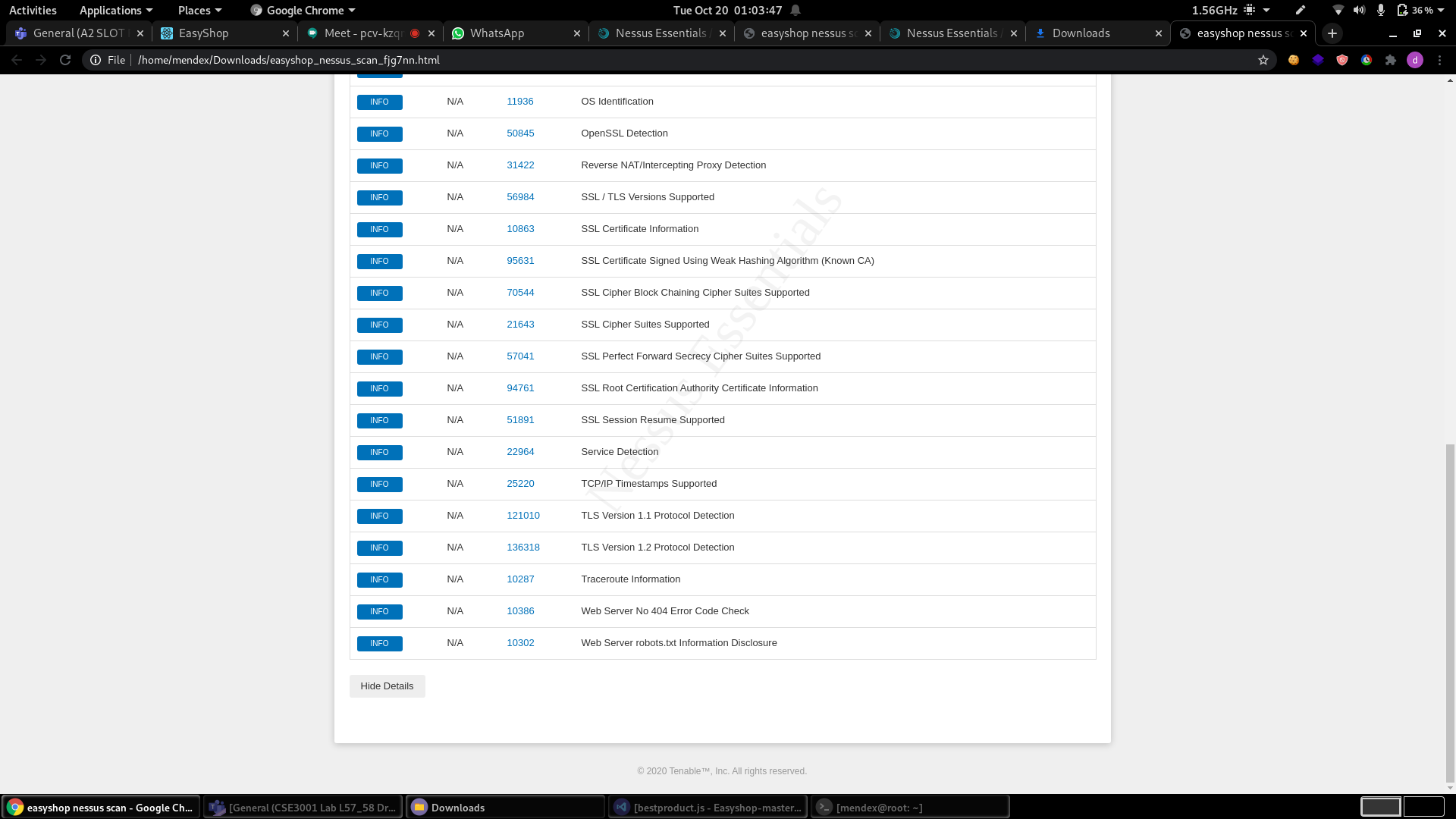
*Burp Suite intercept*

* Nessus testing

Nessus is a remote security scanning tool, which scans a computer and raises an alert if it discovers any vulnerabilities that malicious hackers could use to gain access to any computer connected to the network. We tested our website on Nessus Tool and got 2 medium level vulnerabilities and 27 negligible vulnerabilities. Both the Medium level vulnerabilities are not for our website, but are present on the platform we have used to host our website - Herokuapp

****

*Nessus test*

This image lists the vulnerabilities found in our website by Nessus 

*Vulnerabilities found by the Nessus test*

**9. FUTURE WORK AND CONCLUSION:**

**FUTURE WORK:**

Future work that can be implemented with the project is integration of more shops and an option for home delivery done by the shop. As in this project, we have developed a web app that facilitates consumers to buy goods from small local retail shops.

An android/IOS application can also be developed as it will help us reach a wider market as surveys show most Indians prefer to use an app over a website for shopping online.

**CONCLUSION:**

This project has successfully made a web app for the given problem statement. And has met most of the requirements stated at the beginning of the project.

The language used is HTML, CSS, and JS and the database used is Google Firebase. It is a web application useful for any person looking to buy groceries. The application is flexible, easy to use with interactive UI.

**10. References**

Functional Overview of the Website: <https://miro.com/app/board/o9J_kog83XA=/>

API test link : <https://www.getpostman.com/collections/c6da3b13ae81ffccd11a>

NodeJS for BackEnd Development : <https://nodejs.org/en/>

ExpressJS for initializing server: <https://expressjs.com/>

jsonwebtoken for token auth: <https://jwt.io/>

Nodersa for encrypting req.body: <https://www.npmjs.com/package/node-rsa>

Openssl to generate rsa keys: <https://www.openssl.org/>

Firebase Database: <http://firebase.google.com/>

ReactJS for Front-End Development : <https://reactjs.org/>

Material-UI for frontend framework: <https://material-ui.com/>

Axios for making requests to server: <https://www.npmjs.com/package/axios>