

# **A NEW MODEL METHOD TO SECURE AN E-COMMERCE WEBSITE**

**TECHNICAL ANSWERS FOR REAL WORLD PROBLEMS  
(CSE3999)  
REPORT**

**Winter 2020-21**

*by*

**18BCI0179 - Shouya Maheshwari**

**18BCI190 - Laksh Gupta**

**18BCI0197 - Nimish Shah**

**18BCI0214 – Mihir Srivastava**

**in partial fulfillment for the award of the degree of**

**B. Tech**

**in**

**Computer Science and Engineering**



**VIT<sup>®</sup>**  

---

**Vellore Institute of Technology**  
(Deemed to be University under section 3 of UGC Act, 1956)

**School of Computer Science and Engineering**

**May, 2021**

# Title

## A NEW MODEL METHOD TO SECURE AN E-COMMERCE WEBSITE

### Discussion on Implementation

#### User Experience:

*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.*

*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.*

By using the website, users/customers should be able to get the product they searched for in a structured manner sorted by price or name or whatever the user has chosen. Meanwhile, admins can update their inventory by logging in to the website. Admin can sign up by contacting the support team.

#### Security:

The main aim of our project is to secure the web app as much as possible. The algorithms we are using are pretty standard, but the way we are sending the data on the server is different. So, basically we will be encrypting the data on the client side itself (it will be done by the browser locally) and then send the data directly. So,

For request body,

```
{  
  user:"Tanish"  
}
```

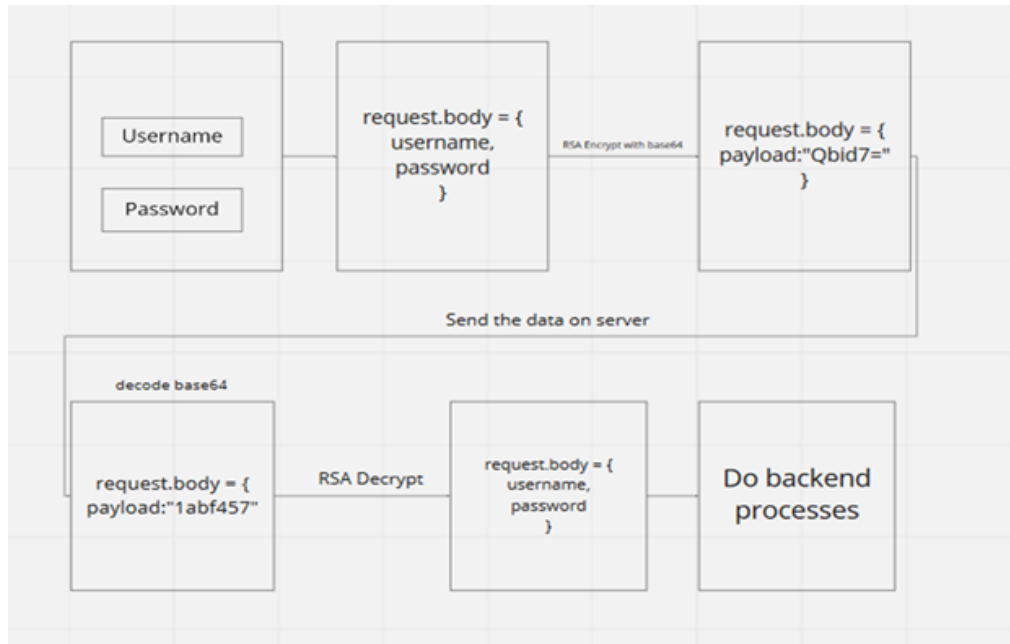
Normally, data would be sent like this, but we encrypt it and it will be sent like this,

```
{  
  payload:"Qbd3r=f" // Some base64 encoded string.  
}
```

On decoding the object would be, { { user:"Tanish" } }.

Along the client side encryption, we are also deploying server side encryption with the help of a one of the newest hashing algorithms BLAKE2b which is 512 bit in length.

Steps:



Client,

Form-data -> RSA encryption with 3072 bit key -> base64 encoding -> send the form data

Server,

Received base 64 form data -> decoding base64 to uint8 buffer -> RSA decryption -> received the form data in the backend controller (can do operations on it)

## Requirement of materials for the implementation of proposed method

### Hardware Requirements

*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.*

### Software Requirements

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", "mdbbootstrap": "^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",

## 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.*

## 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.*

## Security Requirements

security requirements are authentication, authorization, data protection, and nonrepudiation.

### Authentication

Authentication ensures that each entity involved in using a Web service—the requestor, the provider, and the broker (if there is one)—is what it actually claims to be. Authentication involves accepting credentials from the entity and validating them against an authority.

### Authorization

Authorization determines whether the service provider has granted access to the Web service to the requestor. Basically, authorization confirms the service requestor's credentials. It determines if the service requestor is entitled to perform the operation, which can range from invoking the Web service to executing a certain part of its functionality.

### Data Protection

Data protection ensures that the Web service request and response have not been tampered with en route. It requires securing both data integrity and privacy. It's worth mentioning that data protection does not guarantee the message sender's identity.

### Nonrepudiation

Nonrepudiation guarantees that the message sender is the same as the creator of the message. Now that we have an idea of what constitutes Web service security, we'll examine the top ten security factors affecting Web service implementation.

*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.*





## Detailed description of the work carried out

The main objective of our project is securing a website. To do that we have created an experimental website based on ecommerce as frauds and vulnerabilities in such websites are ample.

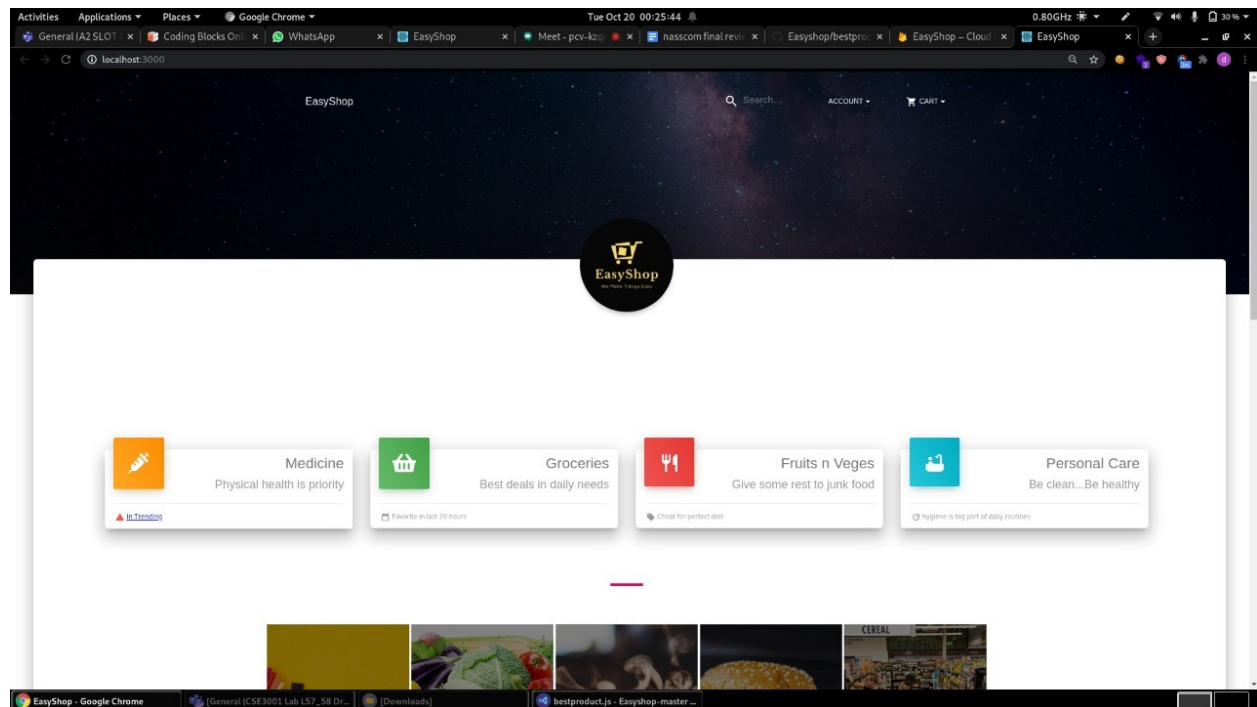
We have designed and implemented the ecommerce website in which the frontend includes Home page, Login and Admin as of now. In the backend in terms of security we have implemented JWT token and validation.

### JWT Token:

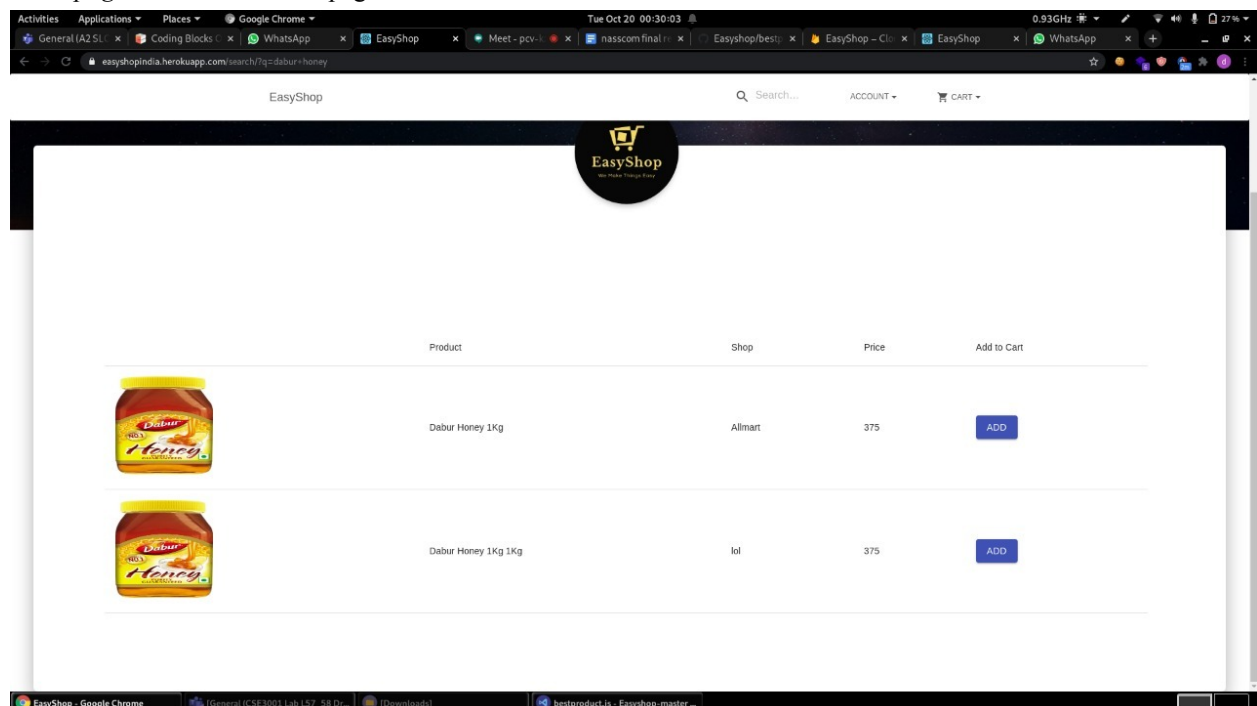
We have implemented token authentication for secured routes to authenticate the user. The token is being generated by the backend once a login request is made. The token then is stored in the browser local storage for uninterrupted sessions. The token then can be used to make requests to the backend by having an "authorization" header to verify requests.

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.	

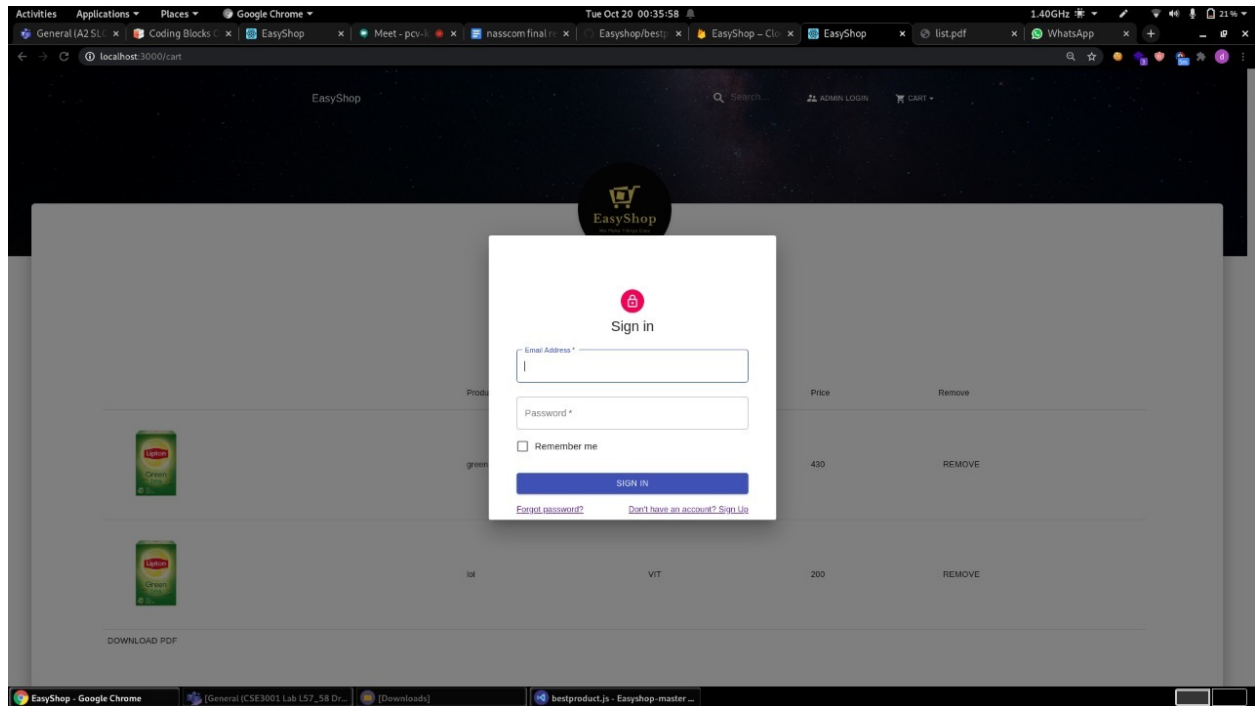
We have developed the basic structure of the website, and some basic security aspects.



*Home page* - This is the homepage of the website that we made



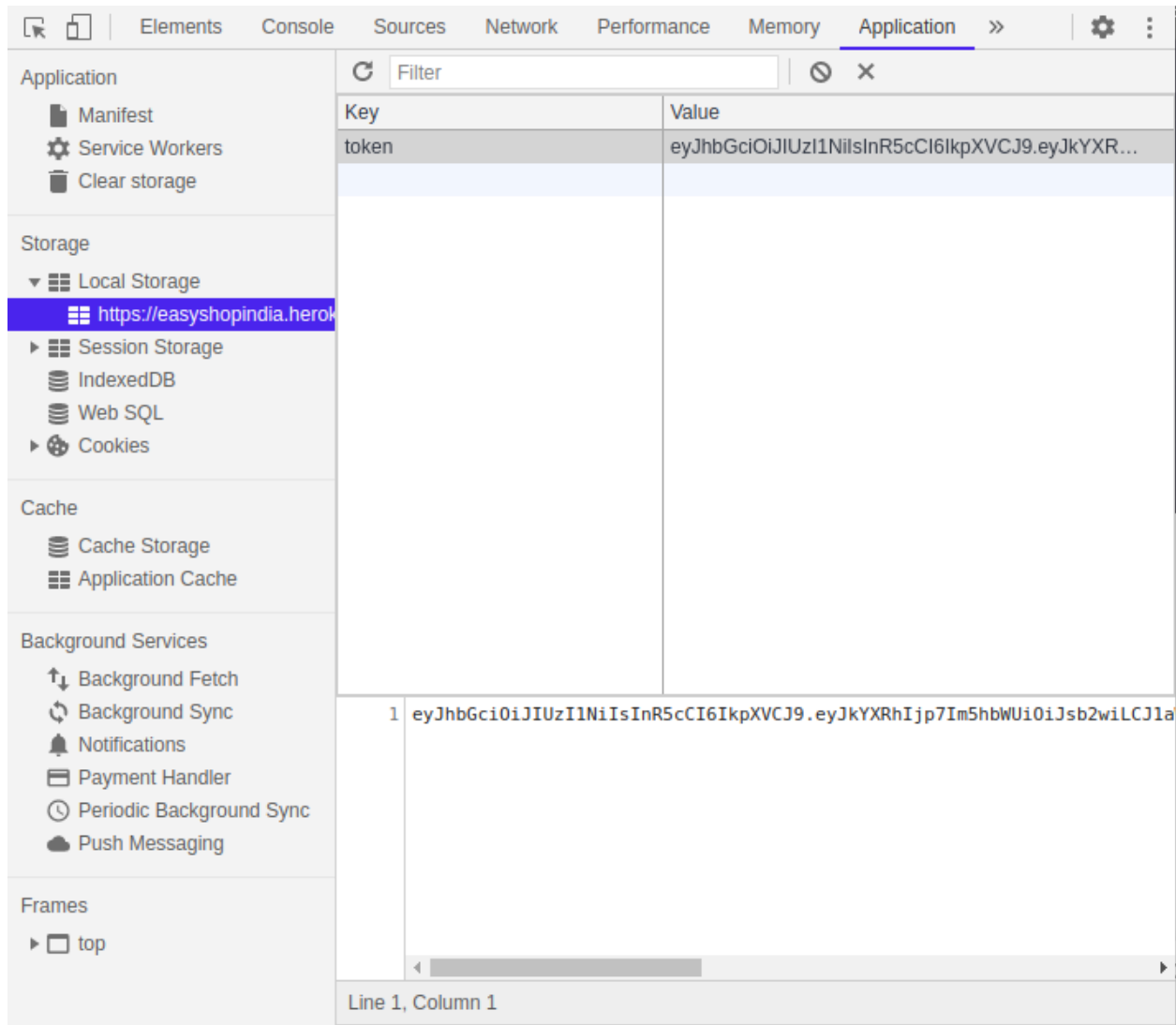
*Search page*- This is the search page. Upon searching for the product, this page will show up with the matched products.



*Admin Login-* The login page where shopkeepers can login to the website to manage their inventory.

easyshop-ace15	users	5ae462abbb
+ Start collection sample users >	+ Add document 09de02fe79 3ac5139270 5ae462abbb > d67d825418 e065f8a8a1 e341ef2961	+ Start collection + Add field 0 image_link: "https://www.amul.com/files/hits/amul-hits-1251.gif" 1 {image_link: "https://www...."} 2 {image_link: "https://encr..."} 3 {image_link: "https://lpi..."} email: "enzo_benzo@gmail.com" maps_url: "" name: "Enzo-Benzo" password: "d2d548838ef2c2eed01f5a1e6ece7dd4058500a0b98652bfae97e" products: [{quantity: 76, image_link:...}] uid: "5ae462abbb"

*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.