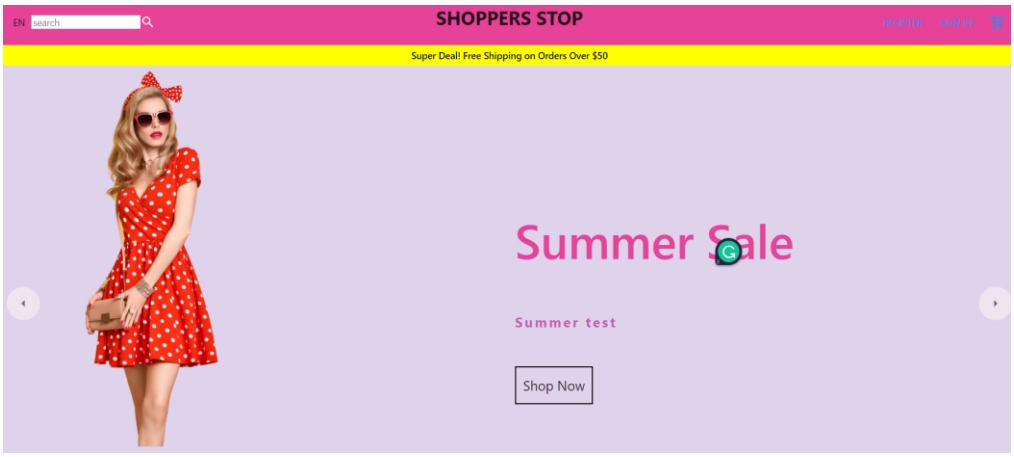
SHOPPERS STOP



30/07/2022

Shopping Website (MERN)

The project is a web-based full stack application. It is a Fashion shopping website based on Mongo DB, Express, ReactJS, and NodeJS (MERN).

We have also implemented Stripe, SMS, Email, and WebSocket services.

Developed By: Shivangi Patel & Maryam Faraji

Shoppers Stop

SHOPPING WEBSITE (MERN)

PURPOSE

As a part of this course, we were expected to develop and deploy a Web application with intermediate web application development practices. We decided to develop a Shopping App that is a full stack application based on MERN (Mongo DB, Express, ReactJS, and NodeJS) where we can apply the knowledge skills we have learned in this course as well as previous such as Full Stack Web Development I(CPSC-2600) and Software practices (2350).

While working on this moderately-complex web application project we learned web development practices including security and authentication, third-party APIs, web accessibility, advanced UI design techniques, contemporary front-end architectures and frameworks, and hosting and deployment.

REQUIREMENT GATHERING AND PLANNING

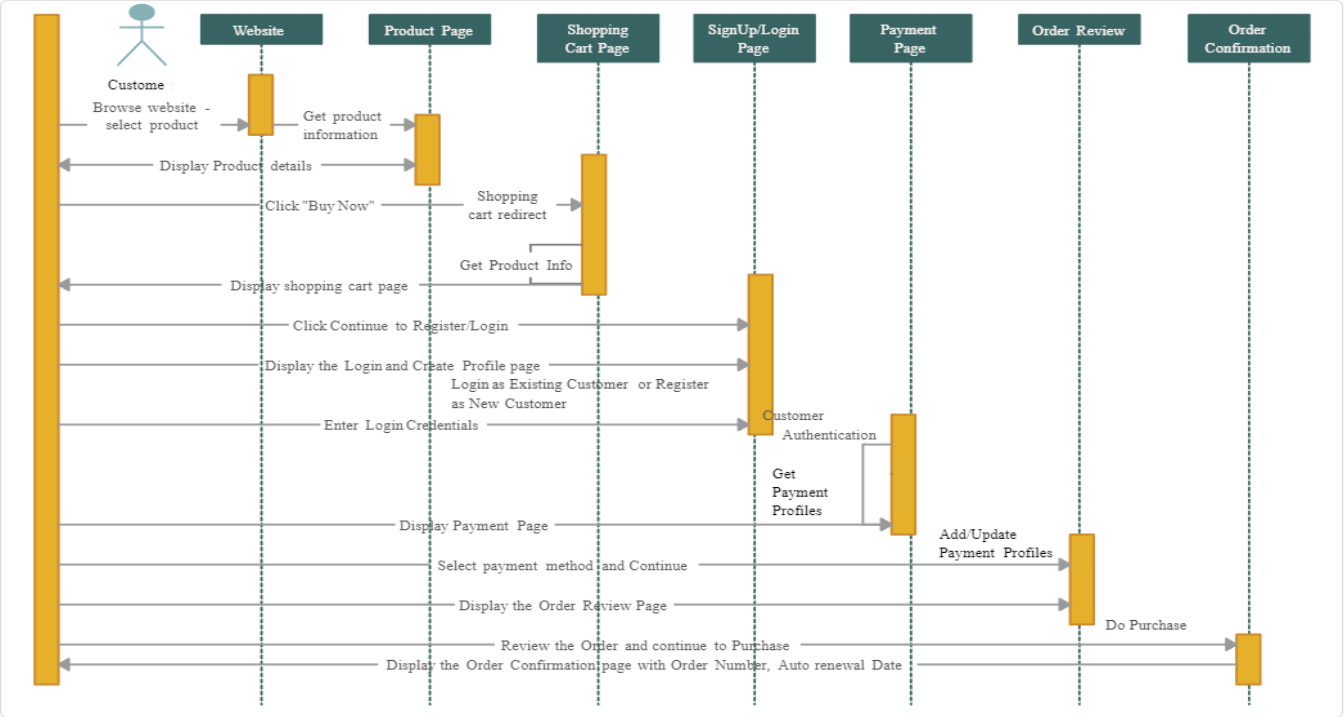
We started researching our project by end of the June 2021 and visited several websites and YouTube channels, here is a list of some.

- https://www.youtube.com/c/Codevolution
- https://www.youtube.com/c/Freecodecamp
- https://www.youtube.com/watch?v=2yddJwL1Tpg

Later we decided on the features of our project, we referred following sequence diagram for shopping applications in general that we found on the internet on the following website

- https://creately.com/diagram/example/i3069s4u1/sequence-diagram-online-shopping

DIAGRAM



We further plan our project based on the diagram with very little changes. We decided on what technologies and libraries we will use in our project detail description of which is in Implementation section below.

MEETINGS AND TASK DISTRIBUTION

We met every Saturday during lab time and spent time afterward discussing the direction of the project and making decisions. The task of research and coding was divided in a way that each of us had an equal contribution to the project development. The scrum and sprints were made on basis of our discussions, and we use to talk on slack and zoom calls about the development and completion of the sprints and helped each other to complete the task as planned. The GitHub commit messages also serve as documentation of changes made to the app code and contributions of both to the project.

IMPLEMENTATION

- Frontend framework: ReactJS with React hooks and Redux(state management).
- Backend framework: NodeJS and ExpressJS.
- Database: MongoDB.
- UI Design Techniques: We majorly used styled Components along with material UI.
-

SECURITY, AUTHENTICATION, AND AUTHORIZATION

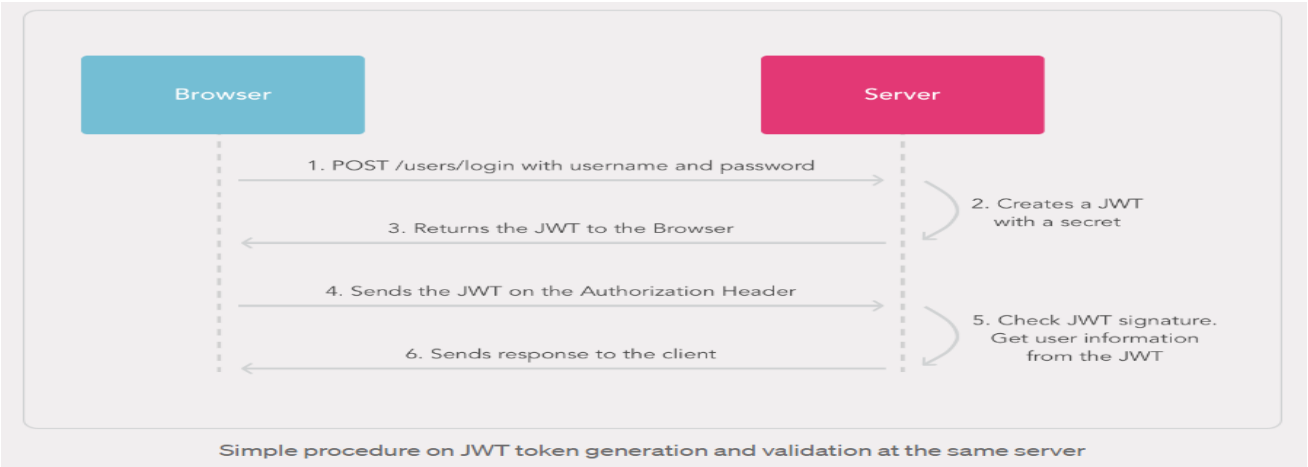
In order to secure the user's password, we are using the crypto-js library and encrypting the password right away when a user registers.

```
const CryptoJS = require("crypto-js");
```

```
router.post("/register", async (req, res) => {
  var encryptPassword = CryptoJS.AES.encrypt(
    req.body.password,
    process.env.PASS_SEC
  ).toString();
```

Authentication enables us to keep our networks secure by permitting only authenticated users or processes to log in and access the website. it's a key step because that does not allow unauthorized users to gain access to sensitive information.

In our application, we have used JWT to authenticate the user. It is a JSON-based format of the security token and uses an API authentication mechanism.



Simple procedure on JWT token generation and validation at the same server

As shown in diagram above, when registered user logs in to our website our server(backend) generates JWT token via "jsonwebtoken" library and returns access token to browser in response.
Here is the screenshot from the code. We have set the expiry for 3 days. We are adding id and admin to the token for verifying them for authorization.

```
    const accessToken = jwt.sign(
      {
        id: user._id,
        isAdmin: user.isAdmin,
      },
      process.env.JWT_SEC,
      {
        expiresIn: "3d",
      }
    );
```

Only with help of tokens, user can check the products, buy them, access the details of the product, access the payment page, etc.

We are authorizing Admin to access a certain part of the website. For Example, getting a list of all users, editing the product details, adding new products, etc. is done only by the Admin.

We have used the JWT token to verify the admin and use the following code as middleware to routes that are meant to be accessed only by Admin.

```
const verifyTokenAndAdmin = (req, res, next) => {
  verifyToken(req, res, () => {
    if (req.user.isAdmin) {
      next();
    } else {
      res.status(403).json("You are not alowed to do that!");
    }
  });
};
```

```
//Get all Users
router.get("/", verifyTokenAndAdmin, async (req, res) => {
```

```
  const query = req.query.new;
  // new is the name of the query
  // by this we can write any query we want
  try {
    // if there is a query it is gonna return 5 latest users
    const users = query
      ? await User.find().sort({ _id: -1 }).limit(5)
      : await User.find();
    res.status(200).json(users);
  } catch (err) {
    res.status(500).json(err);
  }
});
```

PAYMENT GATEAWAY INTEGRATION, SMS, AND EMAIL SERVICES

For user who buys product from our website they have to enter their credit/debit card information and for Anyone involved with the processing, transmission, or storage of card data must comply with the PCI DSS(Payment Card Industry Data Security Standard). It is a set of regulations created by major credit card companies on how people should be using credit card data to keep cardholders safe. These are some of the steps we should take care of

- Having a website firewall to ensure that we are in good traffic, and we are blocking malicious attackers.
- Changing default passwords
- Protecting cardholder data
- How to transmit the data
- Using SSL Certificate (to make sure no one in the middle sniffing the traffic
- Some measures to prevent Phishing (a method of trying to gather personal information using deceptive emails and websites)

If our website does not comply with PCI DSS,

- Fines are imposed by the payment card industry,
- Pay all remediation, assessment, forensic analysis, and legal fees,
- Merchant accounts will be suspended.

To implement this checklist to make sure we are following PCI DSS Compliance, we used Stripe as our payment third-party. Stripe has been audited by an independent PCI Qualified Security Assessor (QSA) and is certified as a PCI Level 1 Service Provider.

We have implemented transactional SMS (Short message service) and Email services to notify the user of successful payment. The user will receive the SMS whether the payment succeeded or failed.

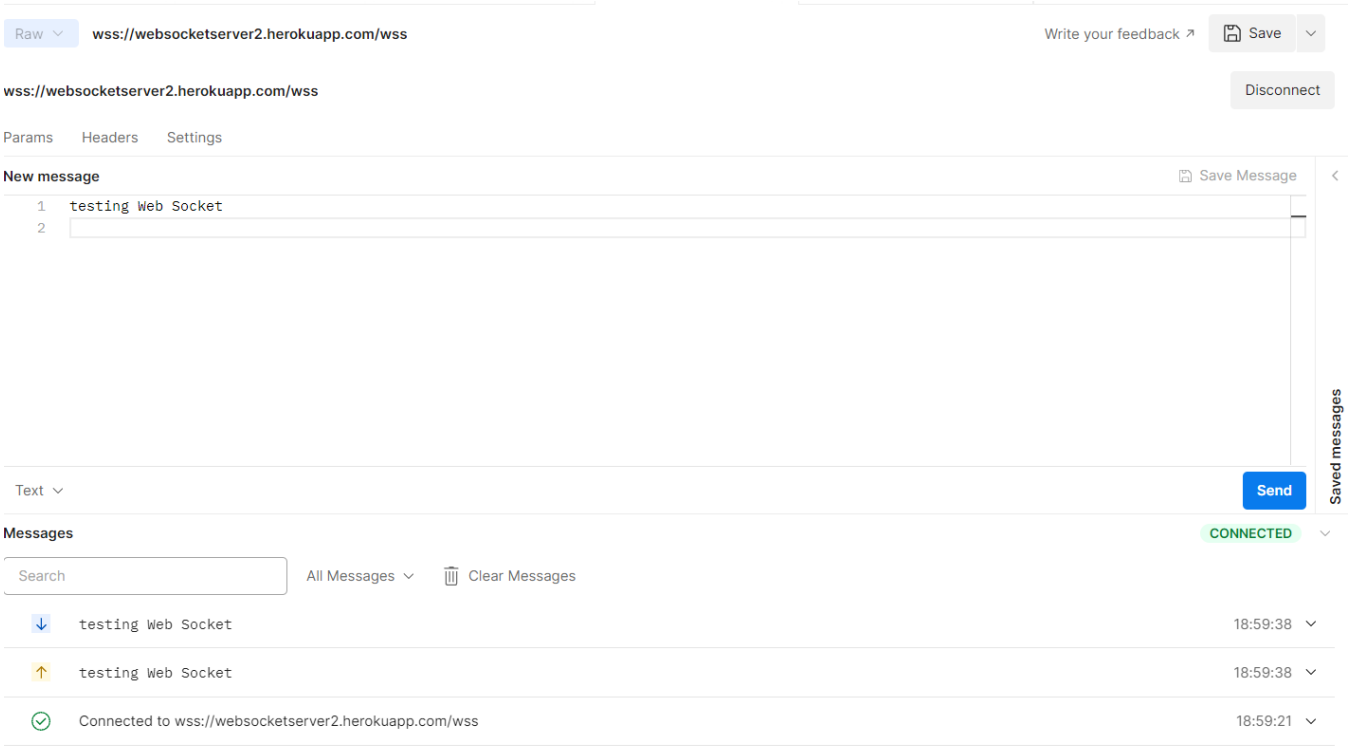
We are using Twilio for SMS services and SendGrid for Email services.

For now, only one number and email Id are used for the project.

WEB-SOCKETS

The **WebSocket API** is an advanced technology that makes it possible to open a two-way interactive communication session between the user's browser and a server. With this API, you can send messages to a server and receive event-driven responses without having to poll the server for a reply.

We have created a Web-socket Server for Customer Support. A user can communicate his concerns to Admin and Admin can respond him/her.



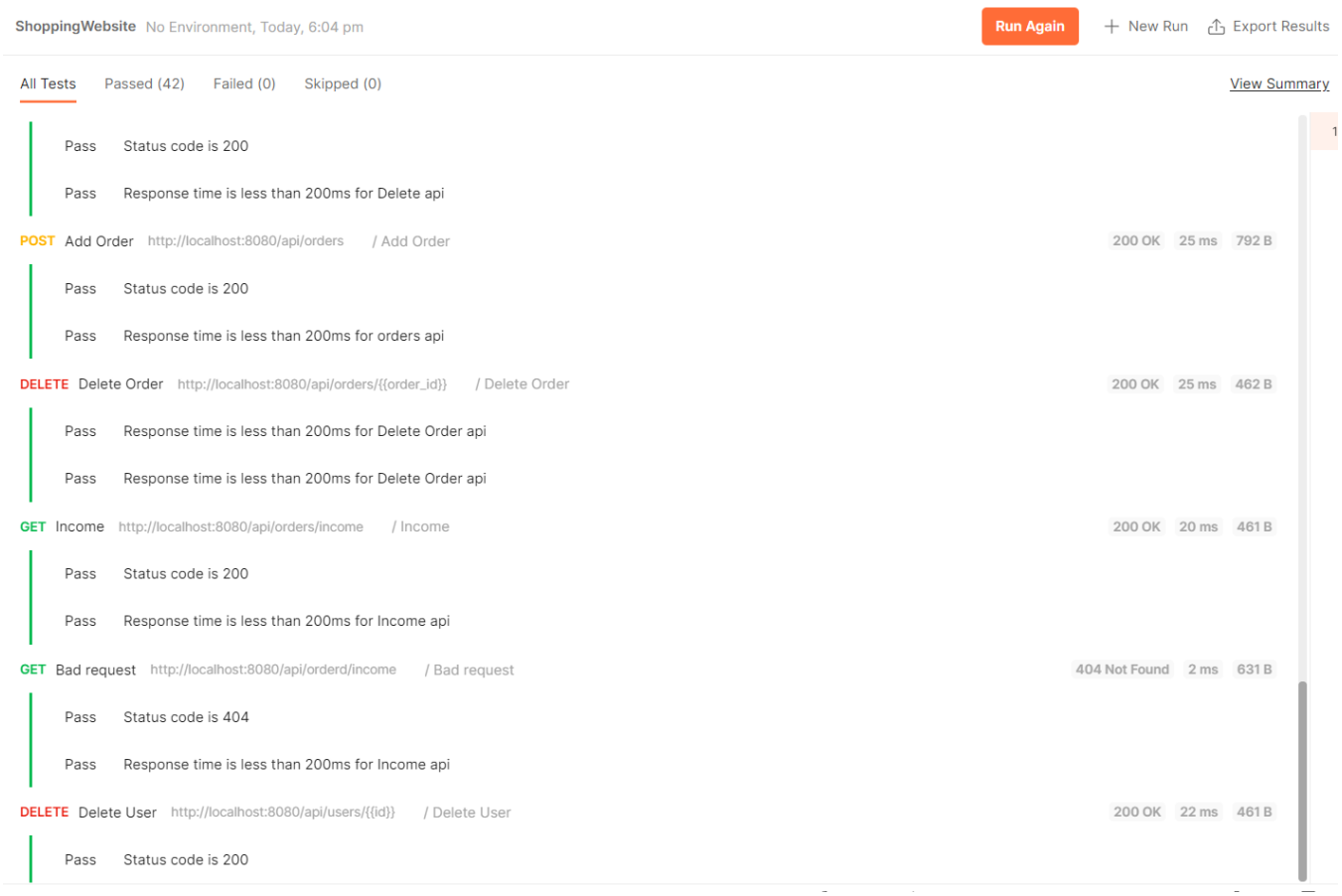
- Our WebSocket server is deployed at following URL https://websocketserver2.herokuapp.com/
- Screenshot shows that is tested successfully.

TESTING

We have tested all our API with both automated and manual methods. We have used Postman for testing our API endpoints. We chose postman as we can write and run tests using JavaScript language.

Considerations for testing

- Service should respond quickly enough for the intended users.
- Server should respond with the correct values (Positive Testing)
- Service should handle exceptions and illegal values i.e., Negative Testing
- service should be stable under expected and unexpected user loads



ACCESSIBILITY

There are a set of recommendations for making Web content more accessible, primarily for people with disabilities they called the Web Content Accessibility Guidelines (WCAG).

We have followed a few recommendations such as

- "alt tags for images.
- "captions" for the images
- "for" tags for forms
- Our website allows keyboard navigations
- We have good color contrast
- Use Heading Hierarchies to Structure and Organize Content

CONTINUOUS DEPLOYMENT

We have implemented Continuous Integration and Continuous deployment for our website.

- We have connected our GitHub repository with Heroku while deployment. Every push to the main branch of our Git repo will deploy a new version of this app. Deploys happen automatically.

- We used the open-source platform Docker for containerization. We have written GitHub actions to containerize the image so that every time we push a change and image is updated automatically.

- We have tested this successfully.

FUTURE PLANS

- Implementing Dashboard for Admin.
- Setting up TLS for PCI DSS.
- Integrating promotional emails and SMS for discounts and offers.
- Auto-chat for our support feature.

SUBMISSION

GITHUB URL : HTTPS://GITHUB.COM/SHIVANGIPATEL1/SHOPPING-WEBSITE
HEROKU URL: HTTPS://CPSC2650PROJECT.HEROKUAPP.COM/
DOCKER URL:
HTTPS://HUB.DOCKER.COM/REPOSITORY/DOCKER/SHIVANGIPATEL9170/SHOPPING-MERN

REFRENCES

- https://developer.mozilla.org/en-US/docs/Web/API/WebSockets_API
- https://learning.postman.com/docs/writing-scripts/test-scripts/
- https://stripe.com/docs
- https://www.twilio.com/docs/all
- https://docs.sendgrid.com/
- https://www.thapatechnical.com/2021/04/how-to-deploy-mern-projects-on-heroku.html
- https://www.youtube.com/c/LamaDev
- https://www.youtube.com/watch?v=_tiecDrW6yY
- https://www.youtube.com/watch?v=mbsmsi7l3r4
- https://www.youtube.com/results?search_query=styled+components+react
-