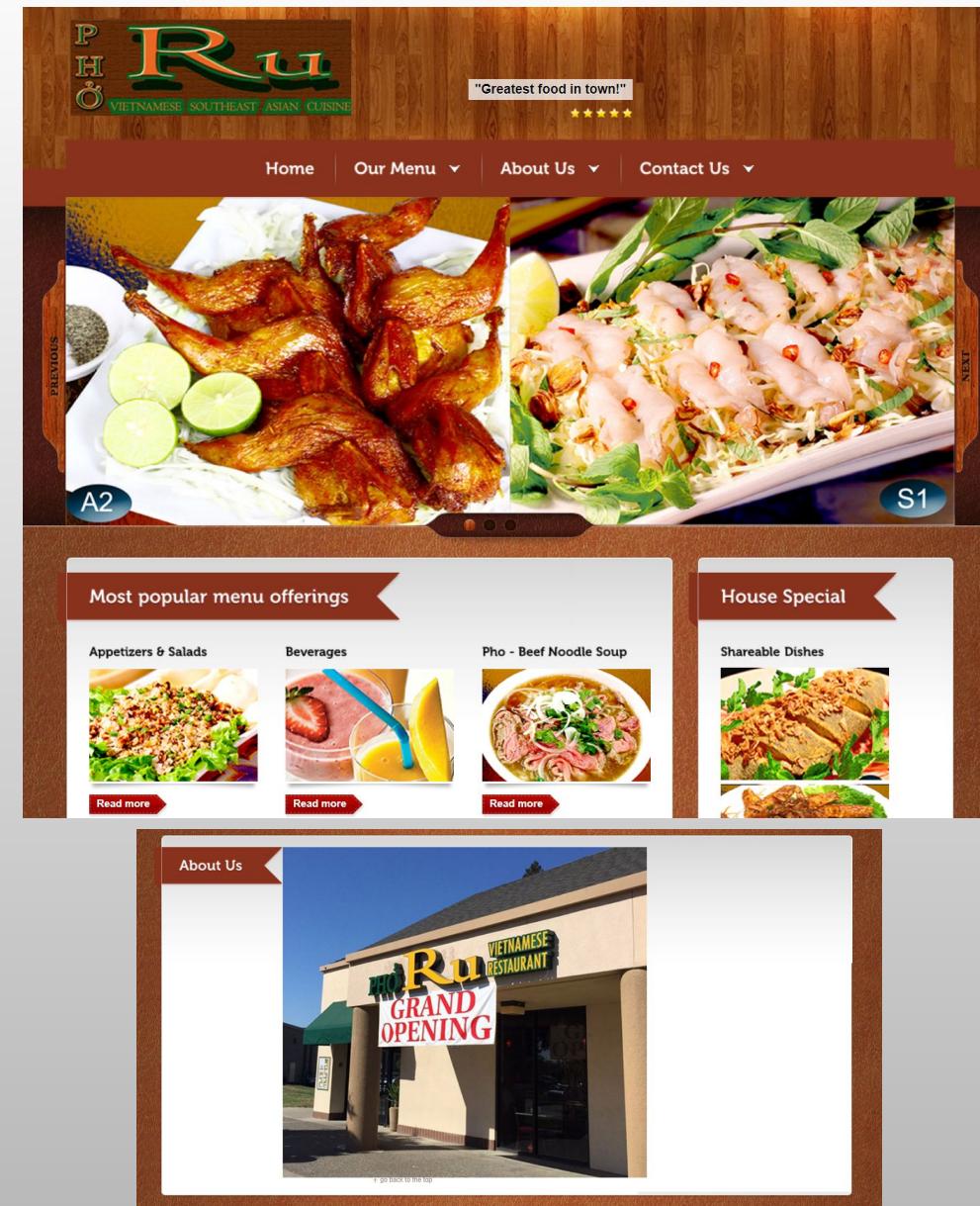


Pho Ru Website

Team Energy: Misba Tabassum, Massoud Bakhtyari,
Naba Riaz, Zoha Alvi, Nam Tran, Duc Nguyen

1. Project Overview

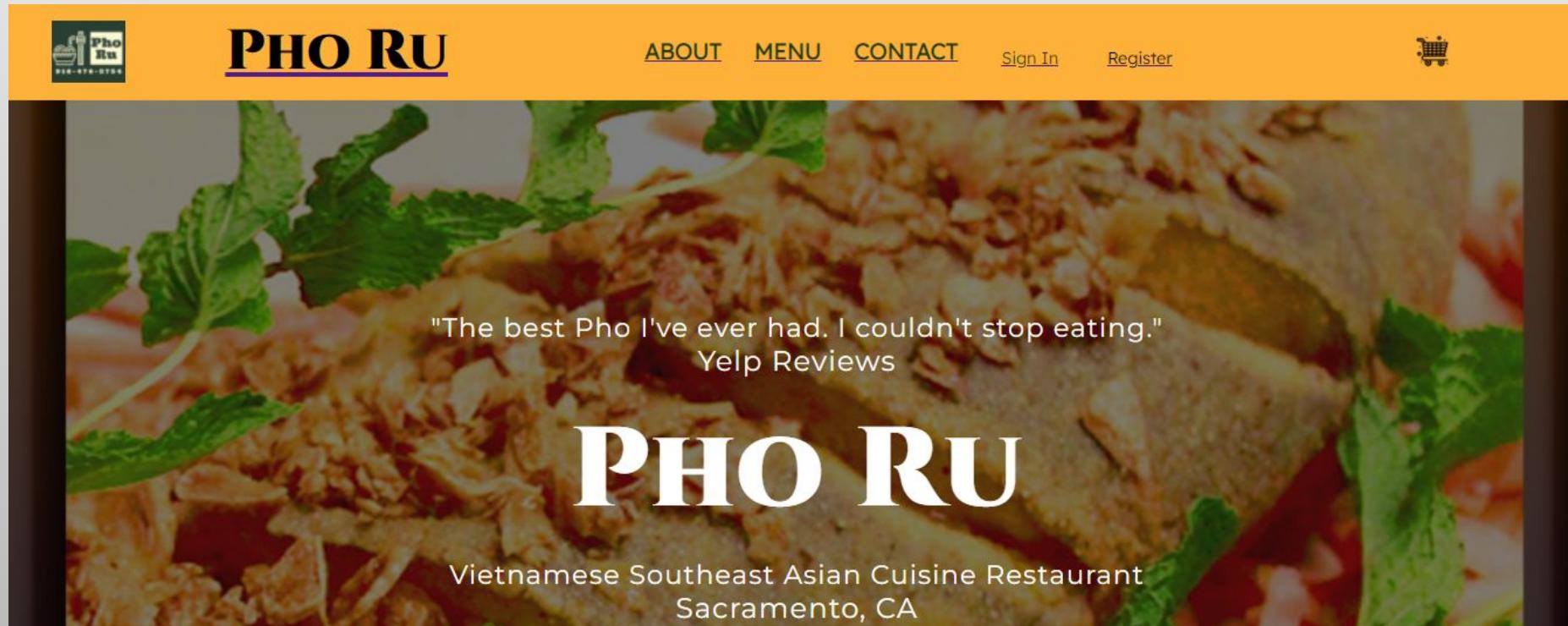
- Client
 - Lynn Nguyen, owner of Pho Ru
- Client's Business:
 - A Vietnamese Restaurant in Sacramento
- Client's “problem”:
 - Outdated website
 - Incorrect information (menu items, restaurant hours, pricing, etc.)
 - User Interface (UI) doesn't reflect the owner's current vision
 - Customers are unable to order online
- Proposed Solution
 - Customers can create an account
 - Customers can order online
 - Update/fix information
 - Better UI



2. Product Web Demo

- Link to the deployed website:

<https://626883441640c715162ae8f9--thriving-creponne-382db8.netlify.app/>



2. Product Mobile View Demo



3. Design - Register

- Bcrypt: password hashing
- Sequelize

The image shows a code editor with two files side-by-side.

Router File (./routes/auth.js):

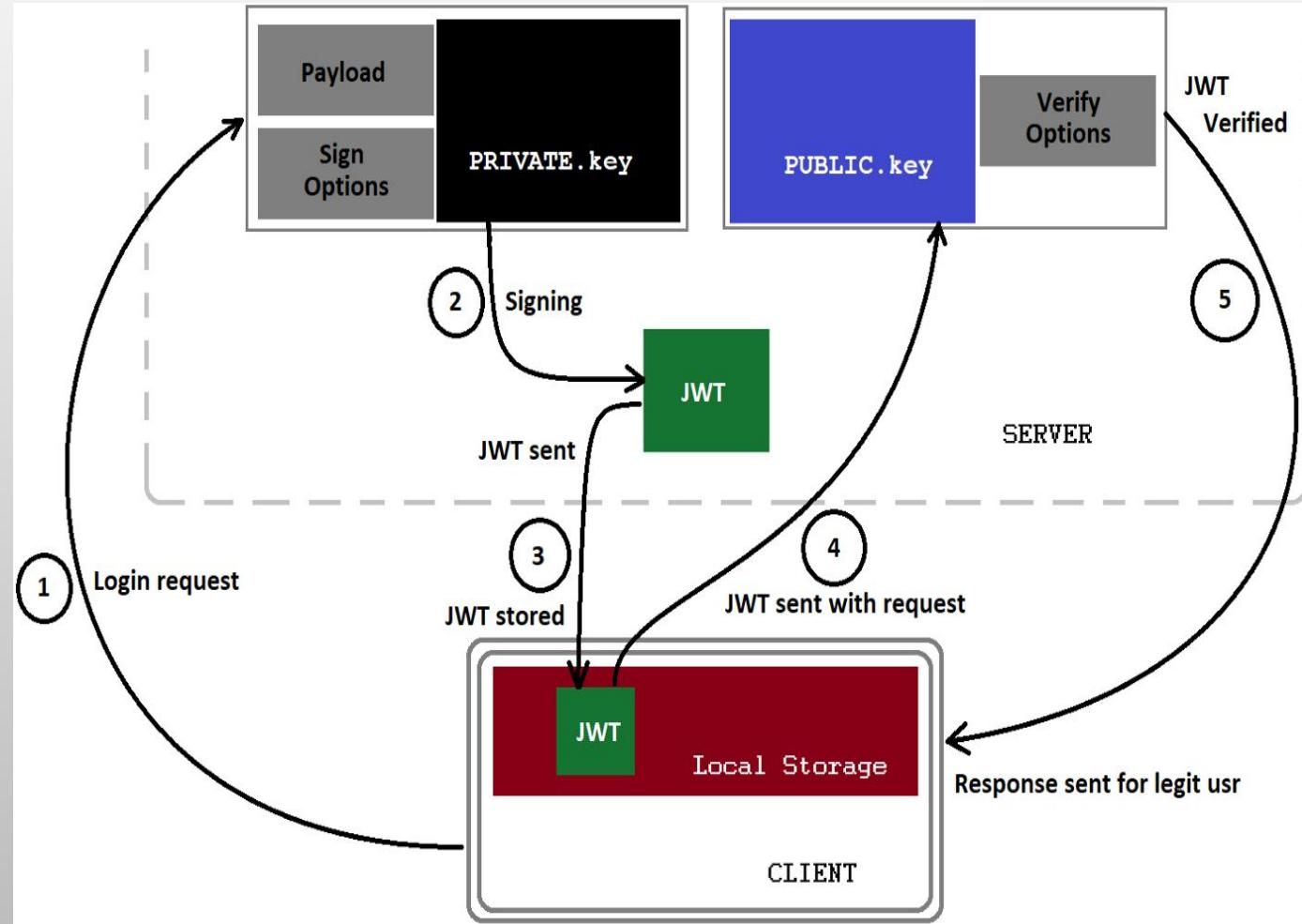
```
//- For register
router.post("/", async (req, res) => {
  const { name, last_name, username, password, question_one, question_two } = req.body;
  const user = await Users.findOne({ where: { username: username } });
  if(user) {
    res.json({ error: "User already Exist." });
  } else {
    bcrypt.hash(password, 10).then((hash) => {
      Users.create({
        name: name,
        last_name: last_name,
        username: username,
        password: hash,
        question_one: question_one,
        question_two: question_two,
      });
      res.json("Success");
    });
  }
});
```

Sequelize Model Definition (./models/Users.js):

```
module.exports = (sequelize, DataTypes) => {
  const Users = sequelize.define("Users", {
    name: {
      type: DataTypes.STRING,
      allowNull: false,
    },
    last_name: {
      type: DataTypes.STRING,
      allowNull: false,
    },
    username: {
      type: DataTypes.STRING,
      allowNull: false,
    },
    password: {
      type: DataTypes.STRING,
      allowNull: false,
    },
    question_one: {
      type: DataTypes.STRING,
      allowNull: false,
    },
    question_two: {
      type: DataTypes.STRING,
      allowNull: false,
    },
  },
```

	id	name	last_name	username	password	question_one	question_two	createdAt	updatedAt
▶	2	Massoud	Bakhtyari	massoud.bakhtyari@yahoo.com	\$2b\$10\$1zQbOkVOOSWexhyBKCNKZ.F9qFJy...dsads	CARMICHAEL	2022-04-21 05:41:17	2022-04-21 05:41:17	
		NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

3. Design - Sign In



```

const { verify } = require('jsonwebtoken');

// Create a middleware (a function with req, res, and next)
// . req: get stuff from the request
// . res: send stuff back
// . next: allow the request to move forward
const validateToken = (req, res, next) => {
  // get the access token being passed in the header
  const accessToken = req.header('accessToken');

  // if user are not logged in (no access token found),
  // send back an error object
  if (!accessToken) {
    console.log(accessToken);
    return res.json({ error: 'User not logged in!' });
  }

router.post("/login", async (req, res) => {
  const { username, password } = req.body;

  // Check if the user existed in our database
  // if: user not exist, send error feedback
  // else: user exists, check password
  // if: password not match, send error feedback
  // else: password correct, create and return token
  const user = await Users.findOne({ where: { username: username } });

  if (!user) {
    res.json({ error: "User Doesn't Exist." });
  } else {
    bcrypt.compare(password, user.password).then((match) => {
      if (!match) {
        res.json({ error: "Invalid Password." });
      } else {
        // Create token: pass in the data that needs to be secured.
        const accessToken = sign(
          { username: user.username, id: user.id },
          "SECRET"
        );
        res.json(accessToken);
      }
    });
  }
});

```

3. Design - Profile Page, Sign out



```
const logout = () => {
  localStorage.removeItem("accessToken");
  navigate("/");
  setAuthState(false);
};
```

3. Design - Shopping Cart

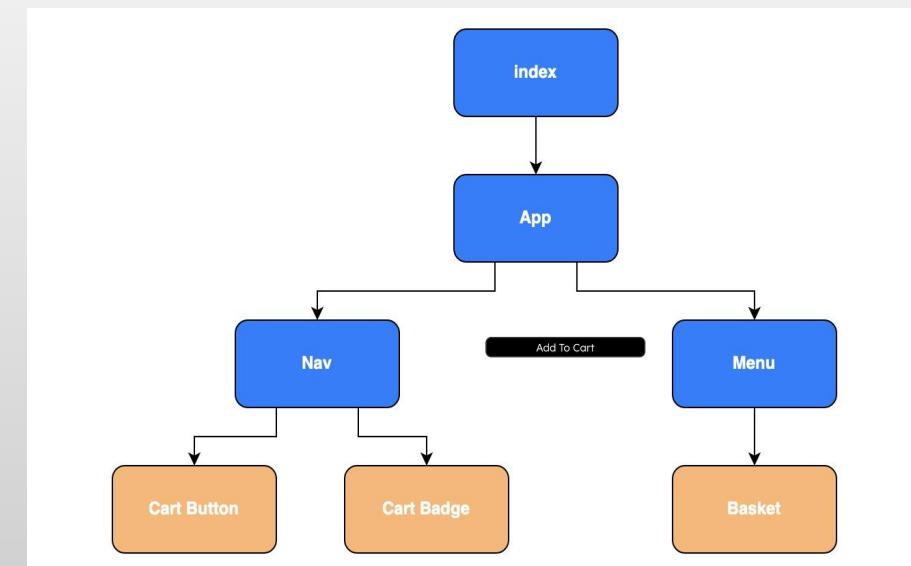
Multiple components are used on one page.

The screenshot shows a website for "PHO RU". The top navigation bar includes links for "ABOUT", "MENU", "CONTACT", "Sign In", and "Register". A shopping cart icon in the top right corner shows a count of 5. The main content area displays a grid of menu items:

- All**
- Appetizers and Salads**
- Pho - Beef Noodle Soup**
- Braised Duck Soup with Egg Noodle**
Mì Việt Tiêm
\$11.75
- Wonton Spicy Sate Soup w/ Egg or Rice Noodle**
Mì Hoành Thánh Súp Saté
\$10.75
- Thick Crab Soup q/ Tapioca Noodle or Udon**
Bánh Canh Cua
\$10.95

Each menu item has an "Add To Cart" button below it. On the left side, there is a sidebar with links for "All", "Appetizers and Salads", "Pho - Beef Noodle Soup", "Braised Duck Soup with Egg Noodle", "Wonton Spicy Sate Soup w/ Egg or Rice Noodle", "Thick Crab Soup q/ Tapioca Noodle or Udon", "Cart Items", "CHECKOUT", and "index".

Cart Button, Add-to-cart Button and Cart Items are on different components



The screenshot shows the "Cart Items" section of the website. It displays the following information:

Ru's Special Trio	5 x \$9.95
Items Price	\$49.75
Tax Price	\$3.61
Total Price	\$53.36

A "CHECKOUT" button is located at the bottom. Above the table, there is a shopping cart icon with a pink border and a red circle containing the number 5, representing the count of items in the cart.

How to keep track of user interaction with those elements and update the cart items correctly?

3. Design - Shopping Cart cont.

Current Presenter:
Duc

Cart Context: store global states and function

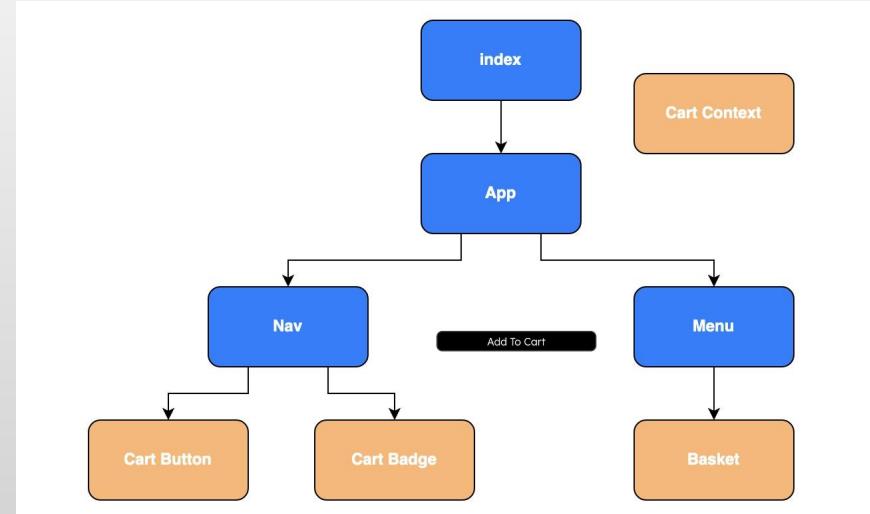
```
index.js      JS CartContext.js      JS CartContextSample.js 5 ● JS App.js      JS PaymentSc
Temp > JS CartContextSample.js > [e] CartContextProvider
1 import { createContext, useEffect, useState } from 'react';
2
3 //--- Create a Context for Cart
4 export const CartContext = createContext();
5
6 export const CartContextProvider = (props) => {
7   //--- Tracking states For Basket: tracking Cart's items
8   const [cartItems, setCartItems] = useState(...);
9   const [totalCartItems, setTotalCartItems] = useState(...);
10  const [isCartClicked, setIsCartClicked] = useState(false);
11
12 //--- Functions for Cart
13  const onAddToCart = (item) => {...};
14
15  const onRemoveFromCart = (item) => {...};
16
17  const toggleCartClicked = () => {...};
18
19 // const [state, dispatch] = useReducer(reducer, initialState);
20  const value = {
21    cartItems,
22    totalCartItems,
23    isCartClicked,
24    onAddToCart,
25    onRemoveFromCart,
26    toggleCartClicked,
27    setCartItems,
28    setTotalCartItems
29  }
30
31  return <CartContext.Provider value={value}>{props.children}</CartContext.Provider>
32}
33
```

Share the global states and functions with the App component

```
index.js      X
frontend > restaurant > src > JS index.js
1 import React from 'react';
2 import ReactDOM from 'react-dom';
3 import { CartContextProvider } from './helpers/CartContext';
4 import { AuthContextProvider } from './helpers/AuthContext';
5 import './index.css';
6 import App from './App';
7 import reportWebVitals from './reportWebVitals';
8
9 ReactDOM.render(
10   <React.StrictMode>
11     <AuthContextProvider>
12       <CartContextProvider>
13         <App />
14       </CartContextProvider>
15     </AuthContextProvider>
16   </React.StrictMode>,
17   document.getElementById('root')
18 );
```

```
CartContextSample.js 5      JS MenuSample.js X      JS Menu.js      JS App.js      JS PaymentScreen.js
Temp > JS MenuSample.js > [e] default
1 import { useState, useContext } from 'react';
2 import Basket from '../components/Basket';
3 import { CartContext } from '../helpers/CartContext';
4
5 function Menu() {
6
7   //--- Get Context value from CartContext in index.js
8   const { cartItems, onAddToCart, onRemoveFromCart, isCartClicked } = useContext(CartContext);
9
10  return (
11    <div>
12      {
13        isCartClicked ? (
14          <Basket
15            cartItems={cartItems}
16            onAddToCart={onAddToCart}
17            onRemoveFromCart={onRemoveFromCart}
18          />
19        ) : (
20          ...
21        )
22      }
23    </div>
24  );
25}
26
27 export default Menu;
```

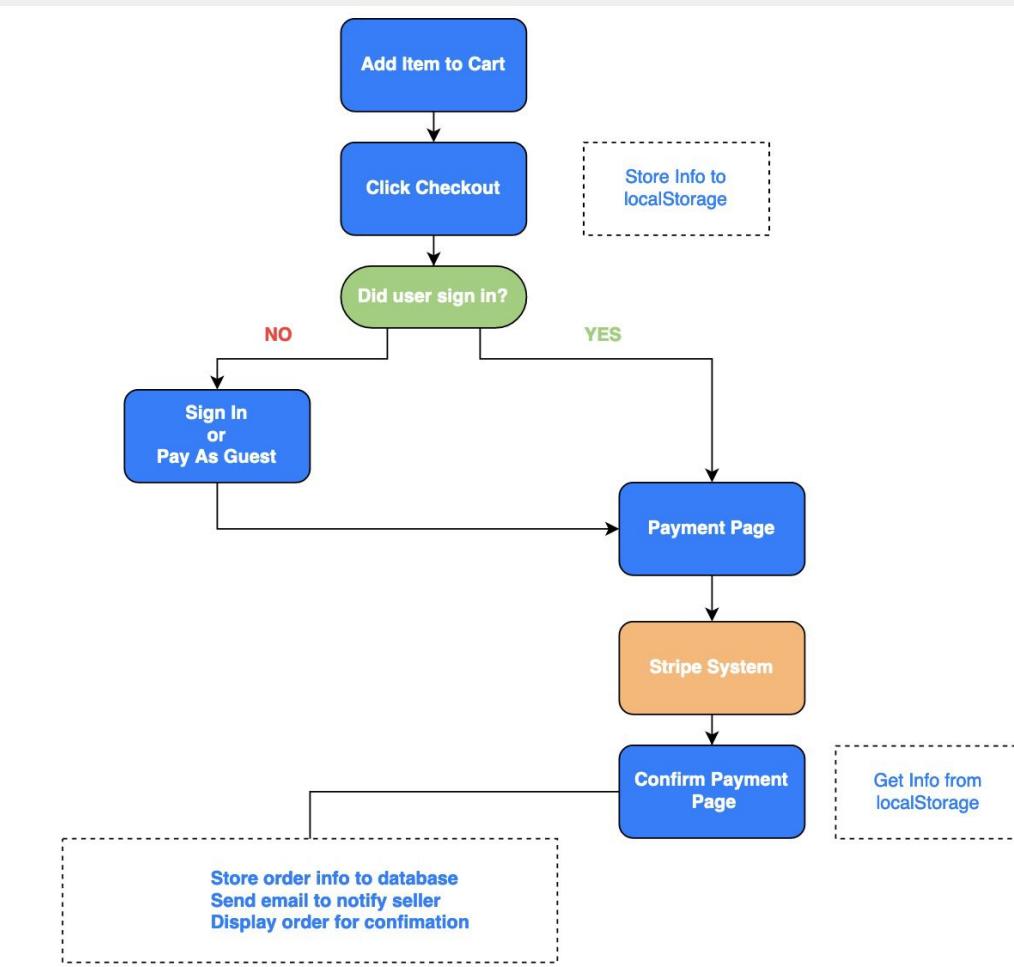
Use isCartClicked to toggle the Basket



Cart Items	
Ru's Special Trio	5 x \$9.95
Items Price	\$49.75
Tax Price	\$3.61
Total Price	\$53.36
CHECKOUT	

3. Design - Checkout Process

Current Presenter:
Duc

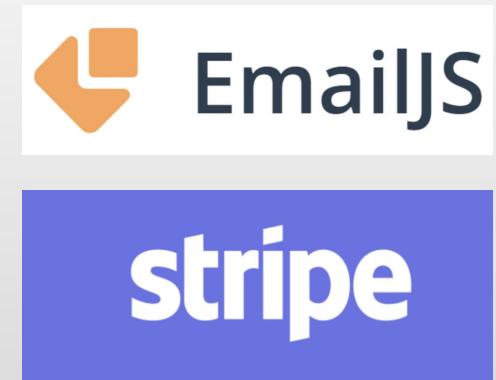


Send payment info to backend

```
if (!error) {
  try {
    const { id } = paymentMethod
    const response = await axios.post("http://localhost:3001/payment", {
      amount: d,
      description: cartItems,
      id
    })

    if (response.data.success) {
      console.log("Successful payment")
      setPaymentInProgress(false)
      setSuccess(true)
    }
  } catch (error) {
    console.log("Error", error)
  } else {
    console.log(error.message)
  }
}

useEffect(() => {
  if (success) {
    navigate('/confirmpayment')
  }
}, [success])
```

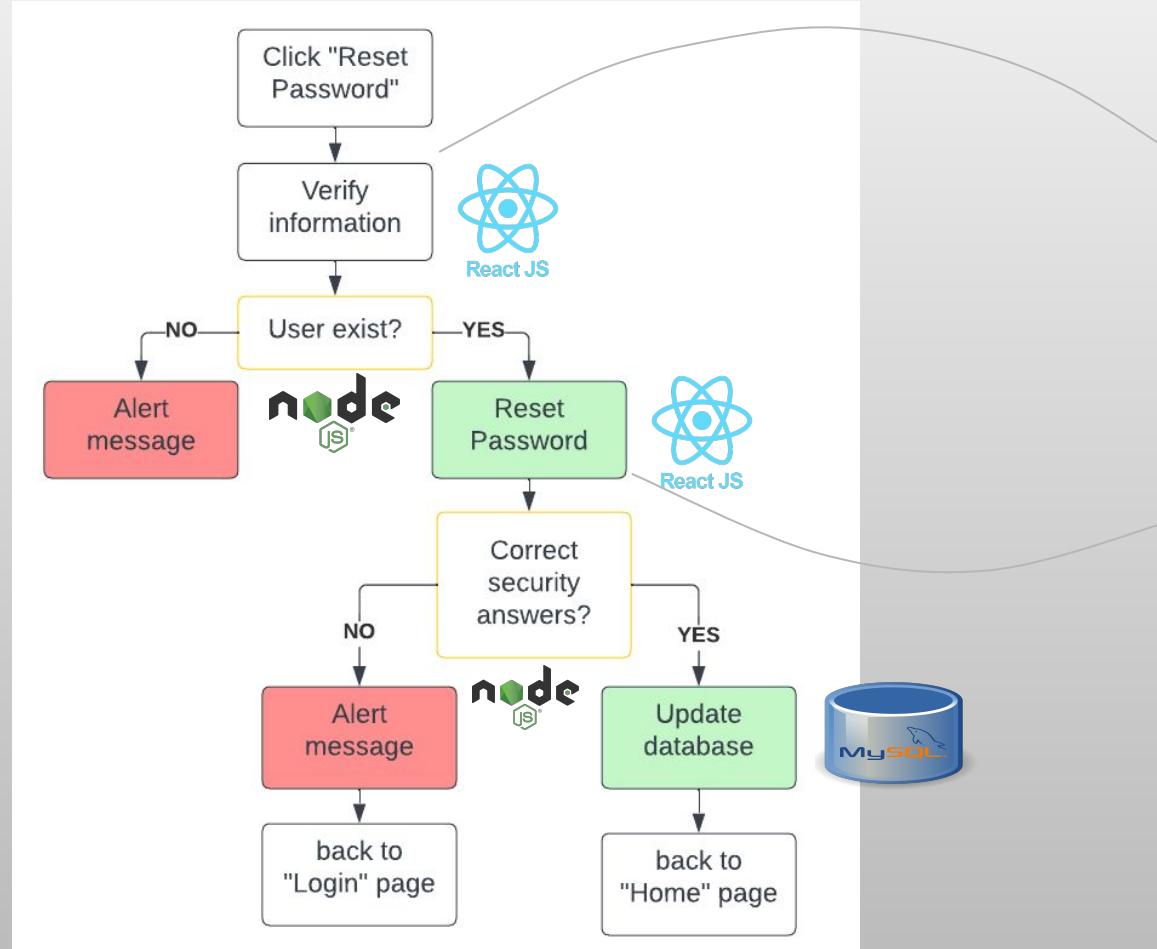


Store string to local storage

A screenshot of the Chrome DevTools Application tab. The Storage section shows Local Storage with a key-value pair: cartCheckoutInfo: {"cartItems": [{"id": "1", "type": "Appetizers and Salads", "name": "Ru's Special Trio", "vietnamese_name": "Tôm Âu V..."}]} and orderList: <p> Ru's Special Trio x 5 \$9.95

 <hr /> Tax Price ...</p>. The Session Storage section shows a key-value pair: http://localhost:3000: null.

3. Design - Reset Password



Verify Your Account

Please enter your email and name to verify your account.

Verify

Password Reset

Please answer the following security questions.

Question 1

Question 2

Please enter and confirm your new password.

Reset Password

4. Technologies and SCRUM

- SCRUM Roles
 - Product Owner, Scrum Master, and Scrum Team worked in collaboration with stakeholder
- SCRUM Practices Used
 - Sprint planning and refining requirements
 - Managing backlog and prioritizing work
 - Weekly standup meetings and daily communication via Discord or Zoom
 - Retrospectives

#

Welcome to #action-items!

This is the start of the #action-items channel.

nabariaz 10/15/2021

Sprint 1 to do list:

- Research on stacks: All
- Business event table (2 versions): Nam and Naba
- Context diagram (2 versions): Duc and Misba
- Execution report: Zoha
- Schedule a review meeting (sprint planning assignment - screenshot of the email mentioning we will be sharing the context diagram and screenshot of assignment portal): Duc (edited)

I-xxX_-Nam_-Xxx-| 11/15/2021

Softwares:

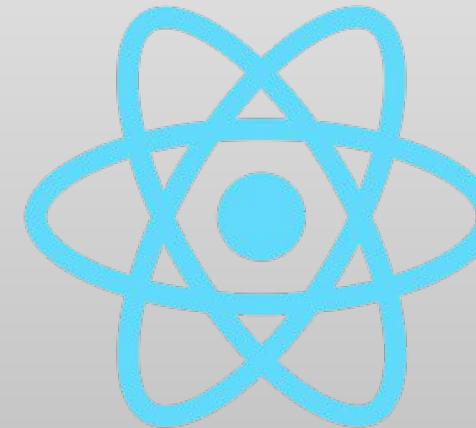
- Visual Studio Code (configured to git clone the repo) (edited)
- NodeJS (Backend) + Express (Also backend) + React (Frontend) (edited)
- Postman (API testing) (edited)
- MySQL Workbench (database) (edited)
- Heroku (free server-centric hosting, worry about it later I guess) (edited)
- Serverless hosting (maybe find a free one first, then find an actual one for deployment?) (edited)

Misba 02/02/2022

- Duc: Email Lynn for client meeting (menu pictures, about page content, etc)
- Massoud, Misba, Zoha: finish menu page
- Nam, Duc, Naba: finish ordering portion of menu page
- Email advisor before Feb16 (edited)

4. Technologies - Frontend

- ❖ React
- ❖ HTML/CSS

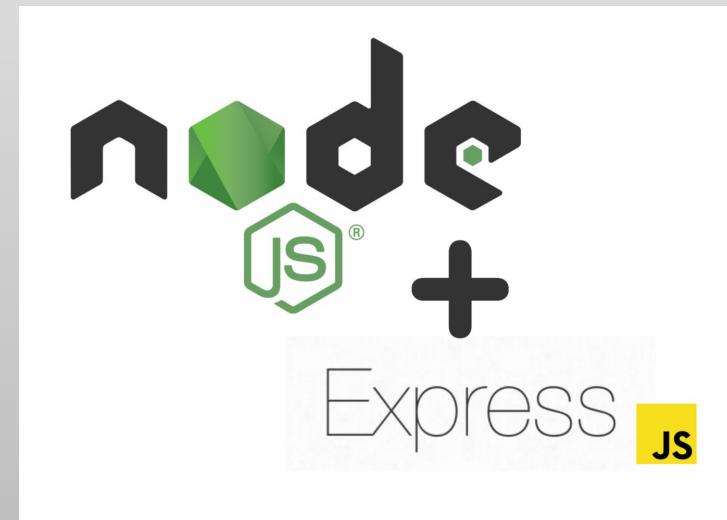


React JS



4. Technologies - Middleware

- ❖ Express
- ❖ Axios



4. Technologies - Backend

- ❖ Sequelize
- ❖ MySQL

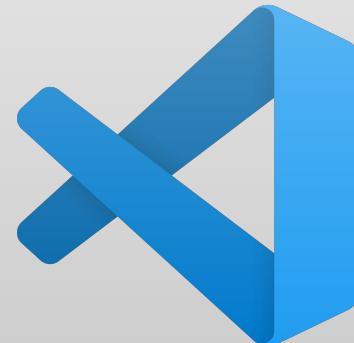


JavaScript Developers



4. Technologies - Other Tools

VS Code, MySQL Workbench, GitHub, PostMan, Insomnia, Selenium



5. Testing

- ❖ Functional testing: Manually
- ❖ API requests: Insomnia, Postman
- ❖ Integration Testing after each sprint
 - Peer Review
 - Make use of GitHub

The screenshot shows a Postman collection named "SeniorProject2022 / localhost". A POST request named "LoginUSER" is selected. The request details show a POST method to "http://localhost:3001/auth/login". The response is a 200 OK status with a response time of 85.5 ms and a size of 162 bytes, timestamped 2 Months Ago. The request body is set to "JSON" and contains the following JSON payload:

```
1 {  
2   "username": "johndoe@fakemail.com",  
3   "password": "pwd123"  
4 }
```

The response body shows a single JSON object containing a token:

```
1 {  
2   "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJ1c  
3   2VybmbFtZSI6ImpvaG5kb2VAZmFrZW1haWwuY29tIiwi  
4   aWQiojIsImlhdCI6MTY0NjMwMTEzMH0. rReBzF4aTP-  
5   PcvFFqp-Ie5F7Ub4Mb1gytLqzLRjFaXM"
```

5. Testing - Selenium Examples

❖ Selenium for automated test cases

```
1 # Test File 01: Navigate between pages
2
3 #-- Import webdriver from selenium
4 from selenium import webdriver
5 import time
6
7 #-- Provide path to ChromeDriver
8 PATH = '/Users/ducnguyen/Programs/chromedriver'
9
10 #-- Select the driver we want to use, and specify the webdriver
11 #-- for this web browser
12 driver = webdriver.Chrome(PATH)
13 driver.maximize_window()
14
15 #-- Open PhoRu website
16 driver.get('http://localhost:3000')
17
18 #-- To get the title of the website
19 print(driver.title)
20
21 #-- Check About Page button
22 time.sleep(2)
23 about = driver.find_element_by_class_name('about-btn')
24 about.click()
25 time.sleep(2)
26
```

```
#-- Check Cart button
time.sleep(2)
cart = driver.find_element_by_class_name('cart-btn-img')
cart.click()
time.sleep(2)
cart.click()

#-- Check Sign In button
time.sleep(2)
signin = driver.find_element_by_class_name('login-btn')
signin.click()

#-- Check Sign Up button
time.sleep(2)
signup = driver.find_element_by_class_name('login-btn-signup')
signup.click()

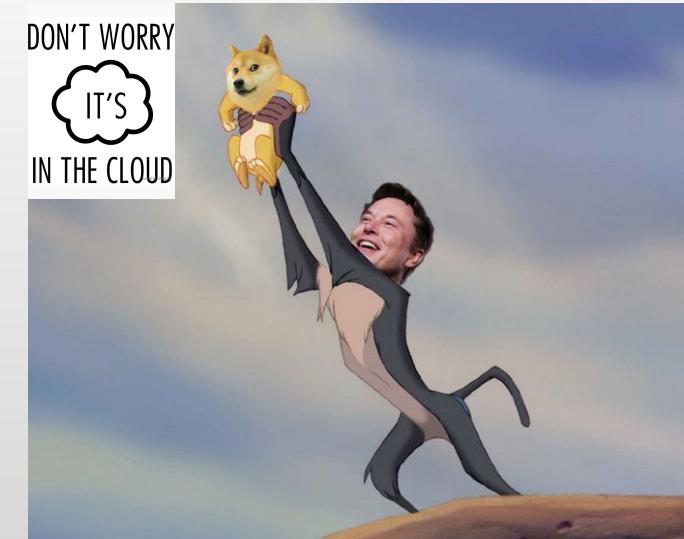
#-- To close the entire browser
time.sleep(3)
driver.quit()
```

6. Deployment

- Frontend: Netlify
- Backend: Heroku

COST:

- Netlify + Heroku: \$100 + \$7 = **\$107**/month
- EmailJs: **\$0** for 200 emails/month | **\$4** for 1000 emails/month



7. Lesson learned

- Things we learned

- Full Stack Web-Development: React, Express, MySQL
- New tools: Github, Postman, Jira, Selenium
- Team Work, Communication, Self-learning

- Insights/wisdom

- Perform regression testing after pushing code
- Let team members know which part you are working on to avoid conflicting changes
- Jira is a great organizational tool allowing us to organize stories and use the Kanban board
- Set achievable goals and delegate tasks efficiently
- Having the whole project planned out ahead of time is very helpful
- Learn to adapt to sudden changes



8. Conclusion

- Client is very satisfied with the product so far
- Nothing to be improved thus far

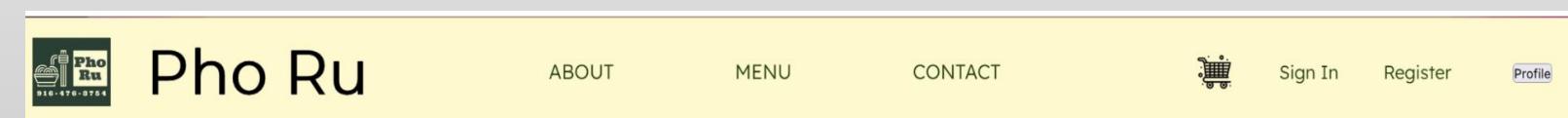


Pho Ru



Pho Ru

Logo Changes



Navigation Bar Changes

Thank you for listening!

