What are we building?

You will be building a basic e-commerce web application. It has the following features:

- You can browse & purchase items with or without an account
- Items are categorized, and there is a way to see items by category
- If you have an account you can see your order history
- If you have an account your current shopping cart can be seen on multiple devices
- An admin is a special type of user with the ability to create new items for the store

What does your store sell? That's up to you, as long as you keep it family friendly.

NEXT LESSON
Why?: What are we hoping to do?

What are we hoping to do?

SYNTHESIS

This project will touch on all of the things you've learned so far. To that end, it's a real test of your education up to this point as well as a great way to reinforce all of the content in the course.

COLLABORATION

This is your first real time working on a development team. To that end, you'll have to practice your social skills while maintaining your programming skills.

PLANNING

You'll have to act as an agile team. This means daily stand-ups, keeping a planning board, and using pull requests to modify the code base. On top of that you'll have to have code reviews and maintain a release schedule.

It will be important to use git and GitHub with a good workflow to prevent code problems. You'll have to review pull requests to make sure that the incoming code is safe and sanitary for the health of the code base.

It's hard work, but getting good at it makes sure that you're a good team member and a valuable asset to the project.

BUILDING SOMETHING BIGGER

Up until now projects have felt relatively small in scope (even though they might have required enormous amounts of work). This project can be as big as you want, since there are no end to features you can add to an e-commerce site.

How does a team function?

Daily stand-ups should be your primary form of communication before team-mates start working on the project. Each team member should report the following:

- What did I do yesterday?
- What am I working on today?
- What blockers to I have?

A **blocker** is anything outside of your control that is preventing progress on what you would (ideally) be working on today.

These reports should be relatively terse (the whole team stand-up should finish in 15 minutes or less), and should be **blameless**. The goal here is to improve the team dynamic over time, rather than turn it into a toxic cesspool.

EXAMPLE STANDUP

- **Alice**: Yesterday, I refactored most of the product and order routes, but there's still a bit left to do. Today I'm going to finish those routes. I have no blockers.
- **Bob**: Yesterday I finished scaffolding the orders page with dummy data. Today I want to make the UI interact with the live data. I am currently blocked by the unfinished routes that Alice is working on.
- **Alice**: That's good to know. The order changes are ready, it's just the product routes that need fixing. I could make a pull request right after stand-up so you could use the finished routes if you can review the pull.

PROJECT MANAGERS

You will be assigned a project manager. That might be an IA or an instructor. They will help to lead the daily stand-up, keep you honest with the work you're doing, and help move the team forward toward the goals.

They will also be your first line of communication if you are having any problems with your code, with your team dynamic, or in general. They are there to support your team as you

work through this project.

TIPS & TRICKS

- Try to create time and space for uninterrupted focus. You can use the pomodoro technique, or whatever you've found works well... but keep team chatter low while the team needs to code.
- Make sure to switch between front-end, server, and database tasks so that you've touched code across the app by the time it is finished.
- Pair with each member of your team for at least one day.
- Make small and frequent pull requests.
- To that end, make sure that your team chunks out tasks into very discrete, achievable goals.
- Protect your master branch. Require review of pull-requests before merging.
- Communicate frequently with the team early on so that the team is on the same page with what needs to happen to hit each goal. This will prevent "I thought"'s from creeping in.
- Challenge yourself. This is a chance to grow... if you're too comfortable you should either be mentoring or finding a way to stretch yourself.

PREVIOUS LESSON
Why?: What are we hoping to do?

NEXT LESSON
Requirements: Project Description

Project Description

STARTER REPO: If you would like, you may start with our boilerplate repo (not ecommerce related, just a node-express-react-postgresql app all in one)

TIER 1: MVP SHOPPING EXPERIENCE

AS A CUSTOMER/VISITOR, I WANT TO BE ABLE TO:

- access a deployed version of the website so I can browse and purchase products.
- view all available products so I can pick from a variety.
- view the details for an individual product (including product descriptions, photos and optionally, reviews), so that I can determine whether that particular item fits my needs
- add a product to my cart so I can collect my desired products in one place.
- edit my cart if I change my mind:
    - change the quantity of a product in my cart.
    - remove a product in my cart.
    - *No one else should be able to edit my cart except me.*
- "checkout" the items in my cart so I can purchase my desired goods.
    - *Think of a typical user experience on popular websites from a guest user and logged-in user perspective.*
    - *You can just start with by simulating the experience of checking out with a simple confirmation page.*
- create an account so I can have a logged-in experience.

AS A LOGGED-IN CUSTOMER, I WANT TO BE ABLE TO:

- have a persistent cart so I can revisit and pick up where I left off.
    - *Logged-in-user across multiple devices: I'm logged in on my mobile device and add some items to my cart. When I open the browser on my laptop and log in, I want to see those items in my cart.*
    - *No one else should be able to edit my cart except me.*

AS AN ADMINISTRATOR, I WANT TO BE ABLE TO:

- have validated data to ensure reliability.
    -

i.e. each customer that creates an account should only be able to do so once with a single email address.

- have full rights to make backend requests to add, edit, and remove products.
    - *No one else should have access.*
- view user information.
    - *No one else should have access.*

AS AN ENGINEER, I WANT TO:

- have a well-seeded database so that I am able to simulate a number of different scenarios for the user stories below.
    - *By doing this, you really set yourselves up to tackle many of the points throughout the tiers. In the long run, this will save you, potentially, tons of time.*
    - *For example, seed hundreds of products with dummy data so that when you get to the "pagination" user story, you won't have to worry about adding more products.*
    - *Likewise, add a bunch of users with products in their carts so editing the cart can be worked on without already having the "add to cart" functionality built out.*
- user data to be secure so that no one can unrightfully manipulate information.

TIER 2: E-COMMERCE ESSENTIALS

AS A CUSTOMER, I WANT TO BE ABLE TO:

- see all products that belong to a certain category.
- explore an aesthetically pleasing website so I can easily navigate around and enjoy the experience (UI/UX).
    - *This includes front-end data validations. For example, if certain fields of a form are required and must be in a specific format, this is obvious to the user.*
- have a persistent cart so I can revisit and pick up where I left off.
    - *There are two more experiences to consider here. Explore your favorite websites to see what the intended behavior is for the following cases:*
        - **Guest-only:** I don't want to create an account, but I want my cart to persist between browser refreshes.
        - **Guest-to-logged-in-user:** Initially, I'm not logged in, and I add items to my cart. When I eventually log in, I want to see those same items I added when I was logged in still in my cart, in addition to the items I may have had in my cart from a previous logged in session.

AS A LOGGED-IN CUSTOMER, I WANT TO BE ABLE TO:

- see my order history so I can remember my previously purchased items and their prices at the time of purchase.
- view and edit my user profile so I can update my information when necessary.

AS AN ADMINISTRATOR, I WANT TO BE ABLE TO:

- allow customers to have a variety of payment method options in order to increase checkout conversion.
    - *Begin by integrating Stripe*
    - For client side, use Stripe's prebuilt Checkout form, ideally with the "Custom" strategy. We recommend react-stripe-checkout in this case. Build a custom form and communicate with Stripe & your server via Stripe.js.
    - For server side, use the `stripe` npm library (API docs here, tutorial here) to accept tokens from your front-end app and send charges via the Stripe API.
- have access to a dashboard with the following functionality:
    - create and edit products with name, description, price and one or more photos, so that visitors can see the latest info on what we have to offer
    - create categories for items, so that users can continue to have useful filters as our inventory grows in variety
    - manage the availability of a product, so that users will know whether or not they can purchase that product
    - view a list of all orders and be able to filter orders by status (Created, Processing, Cancelled, Completed), so that I can more easily find the orders I'm interested in
    - change the status of the order (Created -> Processing, Processing -> Cancelled || Completed), so that others will know what stage of the process the order is in
    - promote other user accounts to have admin status, so that new administrators can have the same privileges I have
    - delete a user, so users who should not be able to log in anymore cannot

AS AN ENGINEER, I WANT:

- continuous integration and delivery (deployment) of the codebase so that there are lower rates of release failure.

TIER 3: EXTRA FEATURES & FLAIR

AS AN ADMINISTRATOR, I WANT TO BE ABLE TO:

- trigger password reset for a user (that is, the next time they successfully log in with their old password, they are prompted for a new one), so that I can be proactive in getting

users to change their passwords after a period of time
- ensure accurate product inventory so that we can be sure only available products are sold.
    - *For example, when a customer purchases an item, the quantity available is appropriately deducted.*
    - *Likewise, if a customer attempts to purchase a higher quantity of an item that is available, they will be alerted/notified that there isn't enough inventory.*
- offer customers discounts through promo codes so that we can incentivize purchases.

AS A CUSTOMER, I WANT TO BE ABLE TO:

- log in through third-party authentication so I can avoid creating an account specific to the website. *For example, Google OAuth.*

RECEIVE NOTIFICATIONS

- receive an email confirmation when placing an order so that I can easily reference it when needed without visiting my account.
- be notified when certain events occur so that I am informed of my actions.
    - *For example, when I add a product to my cart, there is a toast notification that pops up in the corner of the page with an appropriate message for that action.*

HAVE A USER-FRIENDLY EXPERIENCE

- filter through all products.
    - *This is an opportunity to dive into a "search" input field. You can filter all products using vanilla JavaScript, or look into Algolia (search-as-a-service).*
- browse through all products in a digestible way so that I am not overwhelmed with an endless list of products.
    - *Dive into pagination here!*
    - *This goes back to the initial seed in Tier 1. If you have a database seeded with thousands of products, there shouldn't be any blockers in order to tackle this user story. It also begs the question of whether we should fetch all of the products from the database or limit the response in intervals (e.g. 25 at a time) and show more only through a user action (e.g. clicking a “Next”/”Show More” button).*
    - *Keep in mind, if you already have the product filter feature built out, can you get pagination to work on the results as well?*
- view featured products so that I can get inspiration.
    - *For example, display the five most purchased products within a given period of time (i.e. yesterday or last week), or the most recently added products.*

- add products to a wishlist so that I can differentiate products I would like to purchase now (cart) versus products I might be interested in purchasing in the future (wishlist).

HAVE A SEAMLESS EXPERIENCE

- navigate the website successfully regardless of whether or not I am handicapped so that my experience isn't hampered.
    - *This is a great opportunity to dive into ADA Compliance (screen-reader friendliness, keyboard navigation, colorblind-friendly, etc.).*
    - *A11y Checklist*
- view a display to know when content is loading or there is an error so that I can manage my expectations.
    - *For example, loading spinners while the frontend is waiting for a backend response.*
    - *As a customer, if I visit a product page that doesn't exist, notify me that it doesn't and bring me to all products. Likewise, if I visit a page that outright doesn't exist, navigate me to the landing page.*

TIER 4: EVEN MORE

AS A CUSTOMER, I WANT TO BE ABLE TO:

- post products to my social media accounts so that I can share with my friends/followers.
    - *For example, integrating Facebook to create a post of a product's name, description, photo and link.*
- receive recommended products so that I can have a customized user experience and get inspiration.
    - *For example, based on products viewed (similar products; matching "tags").*

AS AN ADMINISTRATOR, I WANT TO BE ABLE TO:

- visualize relevant KPIs (key performance indicators) in the admin dashboard so that I can make educated business decisions.
    - *For example, a line graph of total sales over time.*

Code Reviews

These are the major features we're looking for at each code review, in addition to your other progress.

REVIEW 1: START OF DAY 2

- Database:
    - Table Definitions
- API
    - Route definitions
- Project board with user stories

REVIEW 2: START OF WEEK 2

- Database:
    - Database Adapter Code (methods for API)
    - Initial Seed files for users, products, and orders
- API
    - Working Routes

REVIEW 3: BEGINNING OF WEEK 3

- Front-End:
    - View and browse products
    - Cart experience (i.e. adding, removing, and checking out) for authenticated (logged in) users

REVIEW 4: BEGINNING OF WEEK 4

- Front-End:
    - Cart persistence for both unauthenticated (guest) and authenticated (logged in) users
    - Ability to create/update/delete products for admins
    - Ability to create/update/delete reviews for authenticated users

PREVIOUS LESSON
Requirements: Project Description

NEXT LESSON
Data Validation: Keep the DB Clean

Keep the DB Clean

As you work on your data models, please consider the types of data that you will receive, what you want to make required and how you will propagate those errors to the user.

PRODUCTS

- Must have title, description, price, and inventory quantity
- Must belong to at least one category
- If there is no photo, there must be a placeholder photo used

USERS

- Users must have a valid email address
- Users email must be unique

ORDER

- Orders must belong to a user OR guest session (authenticated vs unauthenticated)
- Orders must contain line items that capture the price, current product ID and quantity
- If a user completes an order, that order should keep the price of the item at the time when they checked out even if the price of the product later changes

REVIEWS

- All reviews must belong to a product
- All reviews must belong to a user
- All reviews must be at least X characters

How do I work with others?

NORMS

Within 15 minutes of day 01, you should create a team document which describes the norms (or a set a rules) that the team agrees to. These rules act as a source of truth for how the group behaves, manages work, and deals with conflict.

When you're in doubt of how to behave, you can just refer back to the norms. The norms are not absolute, in the sense that the team can go back and modify them, but they're a great place to start from.

PAIRING

Pair-programming requires each programmer to leave behind their ego, to bring their best self to the table, and to be fearless. It requires a form of generosity as well as humility, and to that end it's a great practice.

Everyone has some baggage from their day, their home life, and their experiences... it can be very hard to let that go and "have a clear mind" when you're working with someone else, but it's critical to you and your partner to do so.

The mechanics of pairing is relatively simple. There are two roles:

- The **Driver** is the person currently writing the code. Their role is the hands of the pair, and should minimally provide input on where the code is going.
- The **Navigator** is the person actively directing where the code should be going. They make decisions on the types of functions we need to write, how the API should be hooked up, etc.

Even though the navigator makes decisions about the code and where it's going, it's up to the driver to actually write the code. The navigator shouldn't tell you when to write `let` vs. `const`, shouldn't have to name a variable, etc. The driver has a surprising amount of autonomy in this process.

Both roles should be looking for bugs, making sure the code is relatively clean. Both roles are responsible for ultimately finishing a story, creating the pull request, and should equally understand the code going into the code base.

Each role should be prepared to explain (think "mentor") to the other when stuck. The driver and navigator are both prone to not knowing where to head next, and should rely on each other to patch misunderstandings and to build up their partner.

PREVIOUS LESSON
Data Validation: Keep the DB Clean

NEXT LESSON
Presentation: Show it off!

Show it off!

Each member of your team should be responsible for presenting roughly an equal amount of time.

BE PREPARED

- Have all your presentation materials open and ready on ONE laptop
- Be logged in (use an ingonito window if neccessary)
- Possibly have screenshots
- *ZOOM IN (cmd plus) your code editor*
- Have all relevant code files OPEN
    - Possible have relevant code samples open in gists

FOCUS ON CODE

- Filling out forms, clicking buttons, etc. is not the most valuable use of class time. Show enough of the UI to give us context.
- Don't ignore the UI, if you've chosen a cool component library or have some really unique interface you're proud of, make sure to show that.
- We've all been out on the same mission, and now is an opportuntity to learn from each other.
- Pick 3-4 concepts, features, or implementations you're particularly proud or excited by.

FOCUS ON PROCESS

- Talk us through where your git, project board, or other processes helped you solve your problems.
- Did you use your seed file in a way that helped coordinate your programming? Show us that.
- Did you use any wireframing/diagramming tools? Show us the supporting documents that helped your development.

HOW TO CHOOSE WHAT TO SHARE?

- Pick slices or aspects of your project that, had you not been on the team that built it that way, you would feel like you missed out if you never learned about it.

SET THE SCENE

- Tell us what the requirements/constraints your code is influenced by/solving for.
- Show us enough of the UI to set necessary context.
- Walk us through the code.

SHOW US THE FULL SLICE

Each portion of your demonstration should cover most, if not all, of the stack:

- React Components
- State Management
- HTTP Requests
- Database Adapter
- Table Definitions
- etc.

Walk us through the decisions that flow through the full stack of your slice.

PREVIOUS LESSON
Working Together: How do I work with others?

NEXT LESSON
Feedback: Let us know what you think!