6 HOW TO **PLAN** AND **BUILD** A REACT APPLICATION

FROM THE EARLIER "THINKING IN REACT" LECTURE:

- 1 Break the desired UI into components
- 2 Build a **static** version (no state yet)
- 3 Think about state management + data flow



- This works well for small apps with **one page and a few features**
- In real-world apps, we need to adapt this process

6 HOW TO **PLAN** AND **BUILD** A REACT APPLICATION

1 Gather application requirements and features

This is just a rough overview. In the real-world, things are never this linear

- 2 Divide the application into pages
 - Think about the overall and page-level UI
- 3 Divide the application into **feature categories**
- 4 Decide on what **libraries** to use (technology decisions)

PROJECT **REQUIREMENTS** FROM THE BUSINESS

STEP 1

- Very simple application, where users can order one or more pizzas from a menu
- Requires no user accounts and no login: users just input their names before using the app
- The pizza menu can change, so it should be loaded from an API

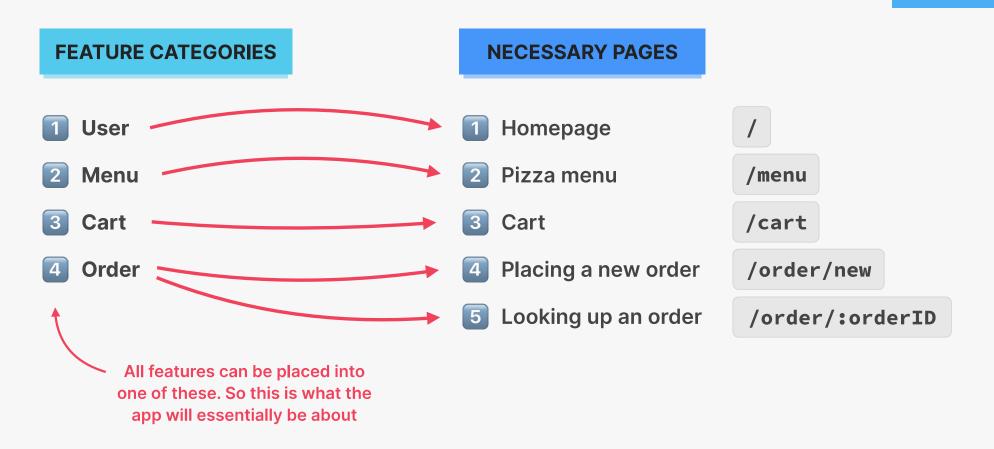


- Users can add multiple pizzas to a cart before ordering
- Ordering requires just the user's name, phone number, and address
- f possible, GPS location should also be provided, to make delivery easier
- User's can mark their order as "priority" for an additional 20% of the cart price
- Orders are made by sending a POST request with the order data (user data + selected pizzas) to the API
- Payments are made on delivery, so no payment processing is necessary in the app
- Each order will get a unique ID that should be displayed, so the user can later look up their order based on the ID
- Users should be able to mark their order as "priority" order even after it has been placed

From these requirements, we can understand the features we need to implement

FEATURES + PAGES

STEP 2 + 3



STATE MANAGEMENT + TECHNOLOGY DECISIONS

STEP 3 + 4

STATE "DOMAINS" / "SLICES"

These usually map quite nicely to the app features

User — Global UI state (no accounts, so stays in app)

Menu — Global remote state (menu is fetched from API)

Global UI state (no need for API, just stored in app)

4 Order — Global remote state (fetched and submitted to API)

TYPES OF STATE

This is just one of many tech stacks we could have chosen

Routing

Styling

Remote state management

Ul State management

.... React Router

tailwindcss

React Router

Redux

The standard for React SPAs

Trendy way of styling applications that we want to learn

New way of fetching data right inside React Router (v6.4+) that is worth exploring ("render-as-you-fetch" instead of "fetch-on-render"). Not really state management, as it doesn't persist state.

State is fairly complex. Redux has many advantages for UI state. Also, we want to practice Redux a bit more