**Steps (with examples) for creating a form app**

**Key**

App.js

Form.js

formSchema.js

Order.js

1. Create the React App
   1. ‘npx create-react-app ( choose app name )’
   2. Install dependencies ( ie yup, axios, etc )
2. Create component files you need (Form.js, User.js, formSchema.js)
3. Set up Form.js (import React and export default)
4. Create pizza function’s props const to be used throughout the form (Add to this as you go)
5. Create your form fields ( <form> <label> <input> )
6. Add at least the main form values ( name, type, value, onChange )
   1. onChange will need to equal an event; for now name that event and create the actual event on step 8
   2. When setting values to checkboxes and radios use ‘checked’ instead of ‘value’
7. Use a slice of state to keep track of form values
   1. State needs to start with empty form fields
8. Create the onChange event
   1. Set the event to include the details you need from event.target so that when something changes on any one of the form values, the onChange event will fire, the callback executes and stores the info needed
   2. Set the event to include a variable the sets the type to either a checkbox or the value (use ternary)
   3. Set the value of the entire form, using a spread operator, and then add to it
9. Create the submit button (set it to disabled) in the JSX
   1. Set state for the disabled state; start as true
   2. Need to set the {disabled} to equal disabled in the return
10. Import \* as yup from ‘yup’ and export default
11. Build out schema form shape
12. Include every item from form and include any requirements you want for validation or leave blank after type
13. When the state of the form changes we will fire an effect that will check the schema
    1. Pass the entire state (formValues) into the schema
    2. This will make sure that all is valid before the submit button will activate
    3. Need to set the {formValues} to equal values in the return
14. Create a slice of state to keep track of errors to show the user what needs to be fixed
15. Along with creating the function (inputChange) you also need to declare that function with name and value
16. This declaration will also need to be set for validation errors; yup.reach
17. Ues .validate(value)
18. Use .then in the event that the validation passes
19. Use .catch in the event that the validation fails; err.errors[0]
20. Need to set the {inputChange} event to equal change in the return
21. Need to create div to show the errors on the screen by using JSX
22. Declare the onSubmit and define the variable you are setting it to
    1. Set up the event handler for that variable to include event.preventDefault()
23. Set up Order.js (import React and export default)
24. Create the order function’s props const to be used throughout the form
25. Create an if statement to tell what to do if the order is not yet appended to the page
26. Create JSX for the details you want to be appended to the page from the order
27. Declare the formSubmit function and create the payload (newOrder) by pulling the details from the form
28. Use .trim() for the newOrder and pull each detail from the form
29. Declare the postNewOrder function and use axios to .post the order to the api
30. Return promise; .then, .catch
31. You can now use the res.data for whatever you need it for

A picture containing timeline

Description automatically generated

Form.js

Text

Description automatically generated

App.js

Text

Description automatically generated

formSchema.js

Text

Description automatically generated

Order.js