Where To Go From Here
Frontend
With modern front-end frameworks, there is more push towards CSS in JS methodologies with which you are not going to need these. However, you should still learn BEM at-least, which would prove helpful in the long run.

These are not specific to React though, you can use them in any framework or app.
Before you start this, you should have a good understanding of what single page applications are, how they work, and what are some of the positive and negative aspects of single page applications.

You can fulfill all your testing needs with these three:
- Jest
- Enzyme
- Cypress

You should understand different types of testing and learn how to write these with the type-checkers:
- TypeScript
- Flow

Progressive Web Apps:
- PRPL Pattern
- RAIL Model
- Performance Metrics
- Using Lighthouse
- Using DevTools

Calculate, measure, and improve performance.

APIs used in PWAs:
- Location
- Notifications
- Device Orientation
- Payments
- Credentials

- Storage
- Web Sockets
- Service Workers

Unit Testing
- Mocha
- Chai
- Ava
- Karma
- Jasmine
- Protractor

Integration Testing
- Jest
- Enzyme
- Cypress

Functional Testing
- Mocha
- Chai
- Ava
- Karma
- Jasmine
- Protractor

CSS in JS
- CSS Modules
- Emotion
- Radium
- Glamorous

Vuex

RxJS
- ngRx

Angular

Vue.js

Pick a Framework
- React.js
Web Assembly or WASM is the binary instructions generated from high level languages such as Go, C, C++, or Rust. It is faster than JavaScript and WASM 1.0 has already shipped in the major browsers. It is being touted by some to eventually replace JavaScript but I seriously doubt that it would happen or see it happening anytime soon.
Backend
Back-end

1. Pick a Language
   There are myriads of different options
   - Python
   - Ruby
   - PHP
   - Node.js
   - TypeScript (Optional)
   - Go
   - Rust

2. Practice what you learnt
   - Exercice and make some command line applications with your picked language

   Sample Ideas
   - Implement some command that you use e.g. 'ls'
   - Command that fetches and saves reddit posts on /r/programming
   - Command that gives you directory structure in JSON format
   - Command that reads JSON from above and creates directory structure
   - Think of some task that you do everyday and try to automate that

3. Learn Package Manager
   - Learn how to use package manager for the language that you picked, e.g. PHP has composer, Node.js has NPM and yarn, Python has pip, Ruby has gems etc.

   Package managers help you bring external dependencies in your application and to distribute your own packages.

For the beginners, if you are just getting into back end development, I would recommend you to pick one of the scripting languages. For the quick-and-easy, go with Node.js or PHP. If you have already been doing backend with some scripting language then don't pick another scripting language and have a look at Go, Rust or Clojure, it will definitely give you a new perspective.

Find the latest version of this roadmap along with resources and other roadmaps
http://roadmap.sh
### Standards and Best Practices
Each of the languages has its own standards and best practices for doing things. Study them for your picked language. For example, PHP has PHP-FIG and PSRs, with Node.js there are many different driven by community etc.

### Make and Distribute Some Package/Library
Now go ahead and create a package and distribute it for others to use, and make sure to follow the standards and best practices that you have learnt this far.

### Contribute to Some Open Source Project
Search for some projects on GitHub and open some pull requests in open source projects. Some ideas for that:
- Refactor and implement the best practices that you learnt
- Look into the open issues and try to resolve
- Add any additional functionality

### Learn Relational Databases
There are several options here. However, if you learn one, others should be fairly easy. Pick MySQL for now but learn how they are different and the use cases.

### Learn about Testing
There are several different testing types, but for now learn about how to write Unit and Integration tests in the language you picked. Understand different testing terminologies such as mocks, stubs etc.

### Write Tests for the Practical Steps Above
Go ahead and write the unit tests for the practical tasks that you implemented in the steps before.

### Practical Time
Create a simple application using everything that you have learnt this far. It should have registration, login and CRUD. Create a blog, for example where anyone can register and get a public profile page. Create a blog, for example, where anyone can register and get a public profile page. Write a public profile page. Create a blog, for example, where anyone can register and get a public profile page.

Make sure to write tests, follow the standards and best practices. Also for the database, add the indexes, use proper storage engines and make sure to analyze the queries before using them in the application.
Learn a Framework

Practical time
Make the same application you made in 1 to the framework of your choice

Learn a NoSQL Database
First understand what they are, how they are different from relational databases and why they are needed. There are several different options. Have a look at different options and see how they differ. If you have to pick one, pick MongoDB

Once you have learnt, implement caching strategy in application you built in step 11

Caching
Learn how to implement app level caching using Redis or Memcached

Creating RESTful APIs
Understand REST and learn how to make RESTful APIs and make sure to read the part about REST from the original paper of Roy Fielding

Authentication/Authorization Methodologies
Learn about the differences and how to implement them

Message Brokers
Learn about the message brokers, understand the “Why” and pick one. There are multiple options but I would go for RabbitMQ or Kafka. Learn how to use RabbitMQ for now, if you want to pick one.

Depending upon the project and the language you picked, you might or might not need a framework. There are several different options
For PHP – Laravel or Symfony and Slim or Lumen for micro-frameworks
For Node.js – Express.js, Hapi.js
For Go – I prefer to code without framework
For others, search and find the suitable ones for the language you picked

Learn MongoDB for now but make sure to look how it compares with others
MongoDB
RethinkDB
Cassandra
Couchbase

OAuth
Basic Authentication
Token Authentication
JWT
OpenID

RabbitMQ
Kafka
Learn a Search Engine
As the application grows, simple queries on your database aren't going to cut it out and you will have to resort to a search engine. There are multiple options, each having its own differences.

- ElasticSearch
- Solr
- Sphinx

- Apache
- Nginx
- Caddy
- MS IIS

Learn How to Use Docker

Knowledge of Web Servers
There are several different options here, look at the different options, understand their differences and limitations.

Learn how to use Web Sockets

Learn GraphQL
While it is not required, feel free to have a look at it and see what it is all about and why they are calling it the new REST.

Look into Graph Databases
Again not required but you should have a little understanding of what they have to offer.

All the things that weren't mentioned above
Profiling, Static Analysis, DDD, SOAP. Go Figure!

Keep Exploring
Good Luck