Design document

Name: Lucas Jacobs
Pcn: 490692
Class: S3 CB-05
Teachers: Jacco Snoerren, Maja Pesic
Date: 25-11-2022
Word count: 1057

Version History

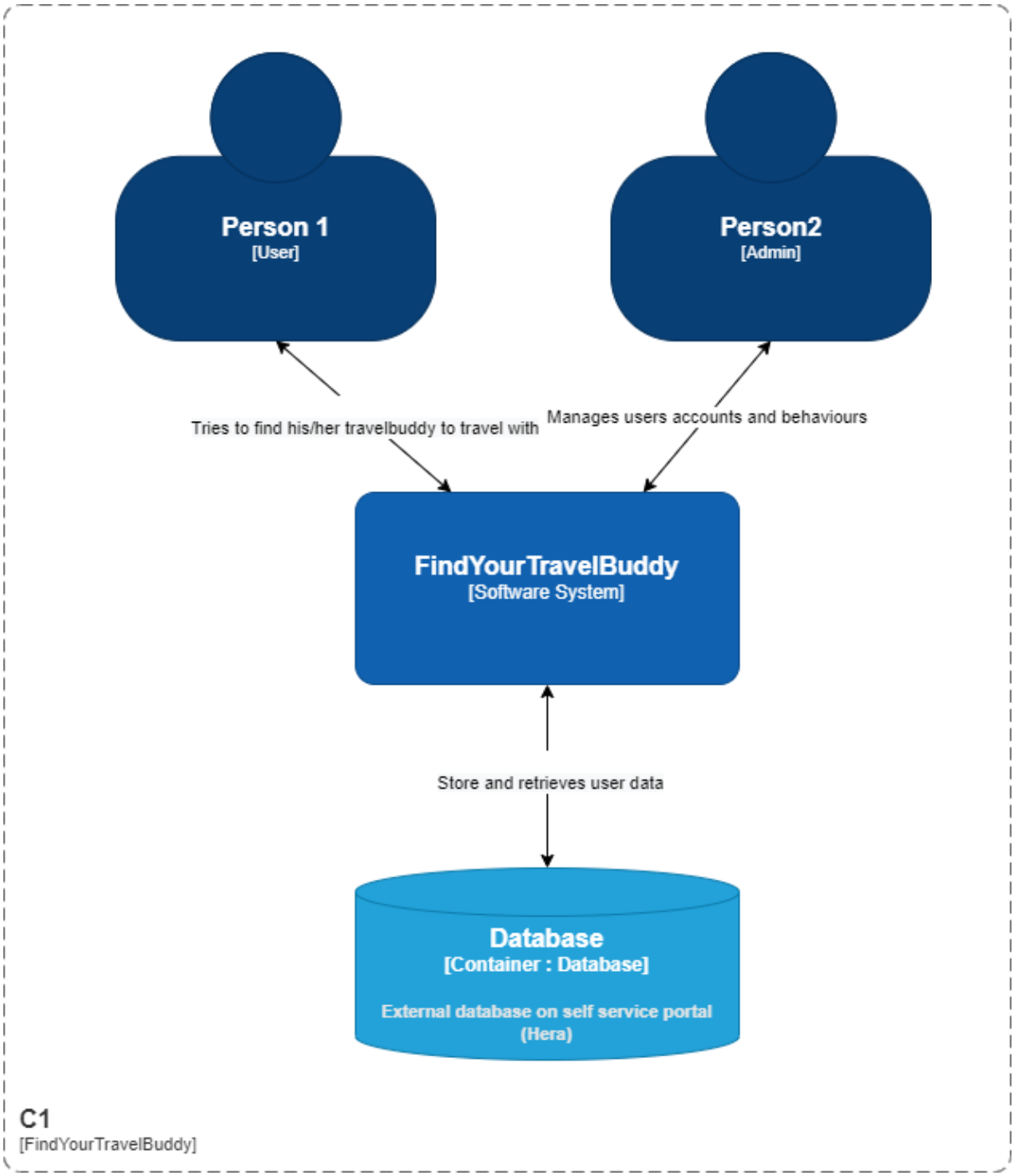
Version	Date	Author(s)	Changes	State
1	2022-10-07	Lucas Jacobs	Begin the design document. the C4 architecture, how is SOLID guaranteed, and Important design decisions.	In progress
2	2022-11-25	Lucas Jacobs	Added the diagram of the CI/CD pipeline	In progress

Contents

C4 architecture

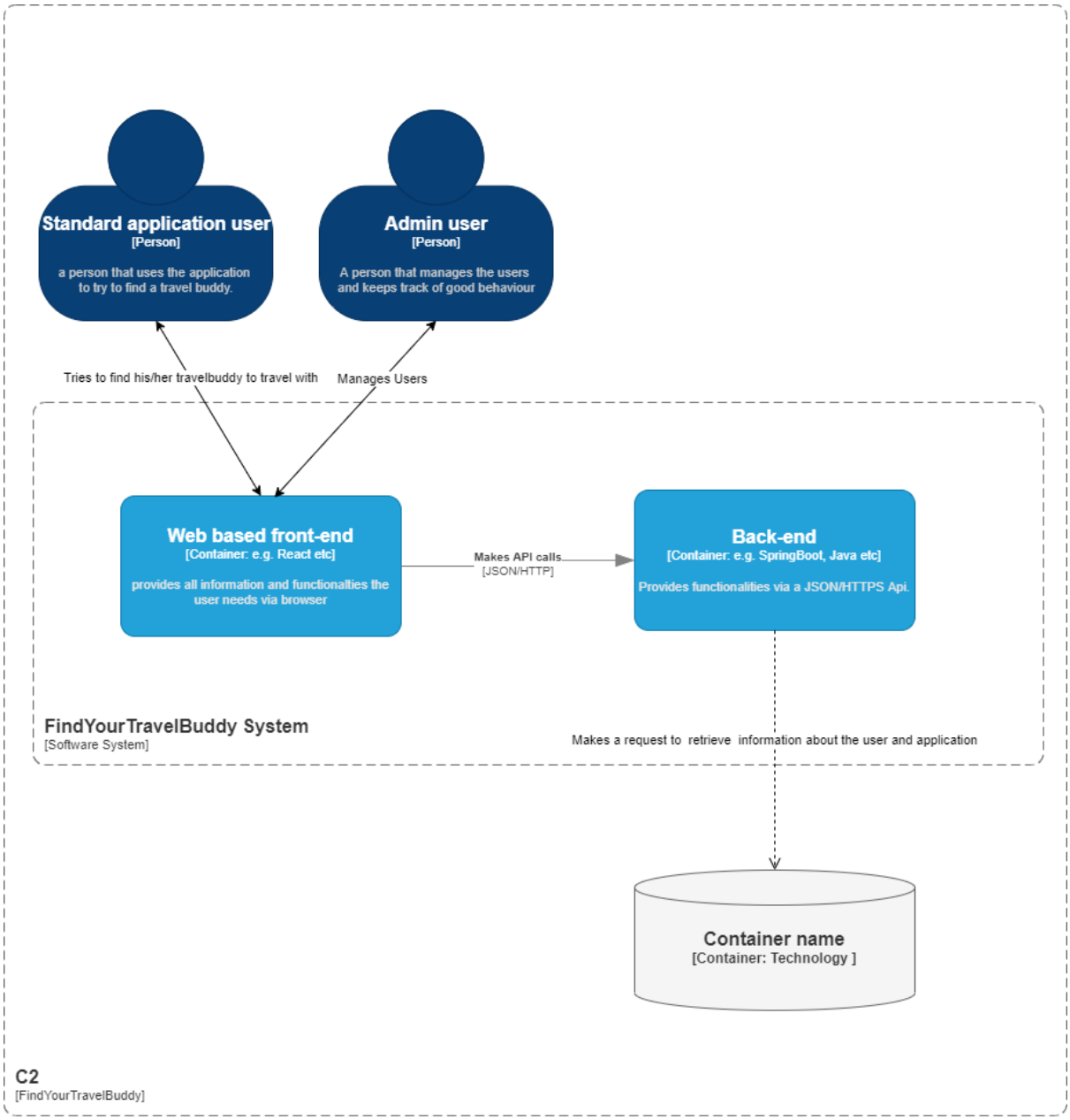
System context (C1): System, users, dependencies

Stretch: Single software system
Audience: everyone



Containers (C2): Overall shape technology choices

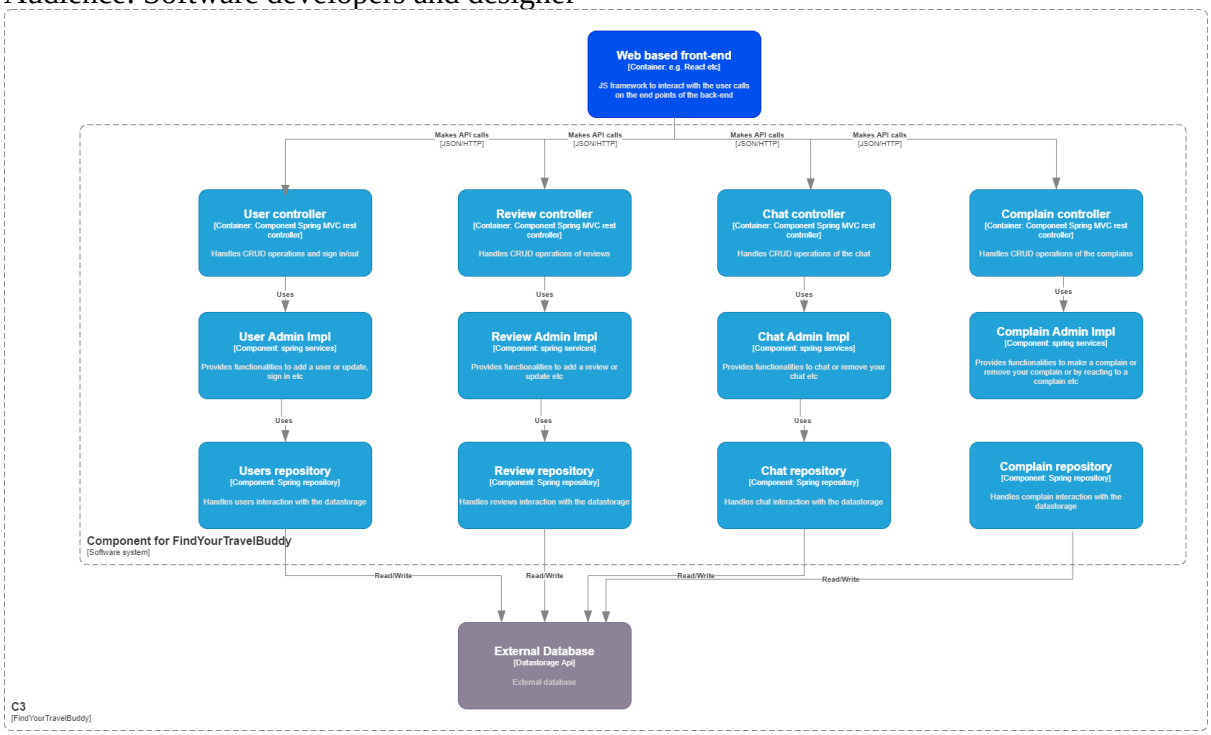
Stretch: Single software system
Audience: Technical people



Components (C3): Components and their interactions

Stretch: Single component

Audience: Software developers and designer

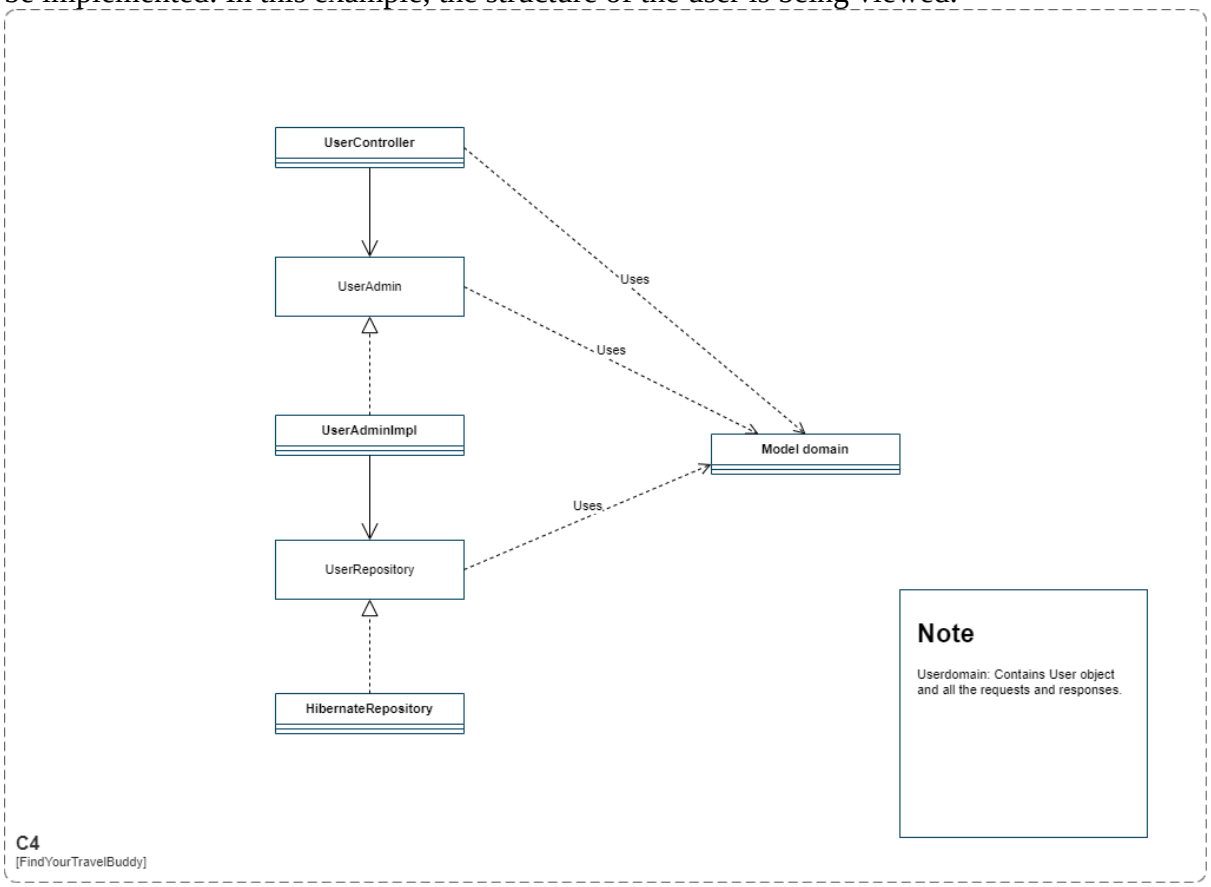


Separate concerns

To separate all of our concerns I have divided everything to be responsible for one thing. These functions are controller, logic, and repository. First of all the controllers are used to receive an API call by JSON/HTTP to get a request from the front-end. This controller will handle the CRUD operations and other functionalities depending on the component. Moreover, the controller will use the logic layer to check certain validations if the incoming data is correct. After that, the logic layer will make use of calling the repository. The repository is only responsible to read/writing specific data to a database.

Code (C4): Implementation details

Stretch: Single component
Elements in this structure: classes, interfaces, objects
Audience: Software developers and designer
This architecture is the main structure that will be used for every component that is going to be implemented. In this example, the structure of the user is being viewed.



How SOLID will be guaranteed

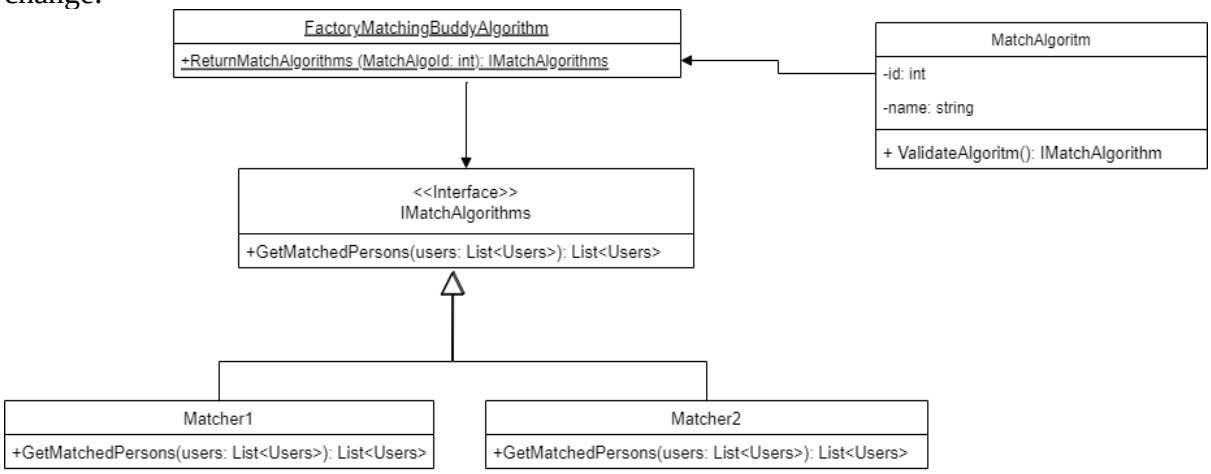
First of all, how is dependency inversion guaranteed in my coding structure? Wikipedia says the principle states:

1. “High-level modules should not import anything from low-level modules. Both should depend on abstractions.” (Wikipedia, 2022)
2. “Abstractions should not depend on details. Details (concrete implementations) should depend on abstractions.” (Wikipedia, 2022)

To make sure that every concrete implementation depends on abstraction, I will use interfaces to separate the layers. In my code structure, we have two interfaces that are separating the high- and low-level modules. The ‘AdminImpl’ class is separated from the lower level class “Repository”. This is done by the interface between them. The interface will be implemented in the ‘Repository’ class, whereas the ‘AdminImpl’ class will use the interface.
The same thing is going on with the classes Controller’ and ‘AdminImpl’. To conclude, now everything depends on abstraction, both the high- and low-level modules.

Next the S in SOLID, which stands for the single responsibility principle. This means that every class, method, function, etc, should only have one purpose and responsibility. This means that a class only should be used for example to add a user, and not that it also contains something about adding an animal. In my application, everything will only have one purpose to change. You can see the structure of my C4 architecture which will also be applied to every component to maintain the single responsibility.

To continue, The L in SOLID, means the Liskov Substitution principle. For now, this will be applied to my factory patterns. This is because in this design you will call a method based on one interface. See below for more clearance.
Furthermore, The O in SOLID, which means open/closed principle. With this principle, every entity in coding (classes, methods, etc) should be open for extension and closed for modification. In my application, When I use multiple algorithms to match potential travel buddies, I will make use of factory patterns to avoid big methods that you constantly need to change.



Example on how the factory pattern is going to look.

Lastly, the I in SOLID, interface segregation principle. This will be guaranteed by having a good look at separating the methods in interfaces. We do not want to force code to implement certain methods when it is not needed.

Usage of Spring Boot framework

For this application, I will use the Spring Boot framework for my back end. Spring Boot is a framework for java that provides multiple services. It helps for building your application and it offers to implement dependency injection.

Dependency Injection

Using Spring Boot for dependency injection is easy and fast to use. It is handy because in my application I am using the dependency inversion principle and this prevents you from initializing your objects because it is not possible to initialize an interface or an abstract class. Therefore, spring boot comes in handy by supporting dependency injection. The only thing you need to do is put some annotations in your application.

Lombok usage

When using the Lombok library you will prevent writing code that you repeat in the same places (boilerplate code), for instance, a constructor or property.

Models

When using Lombok in your models, will save a lot of time writing code. The reason is that when you put certain annotations above your model it will automatically make you standard constructors and properties based on the variables you have filled in.

Framework

The frameworks that I am using for my project are the following:

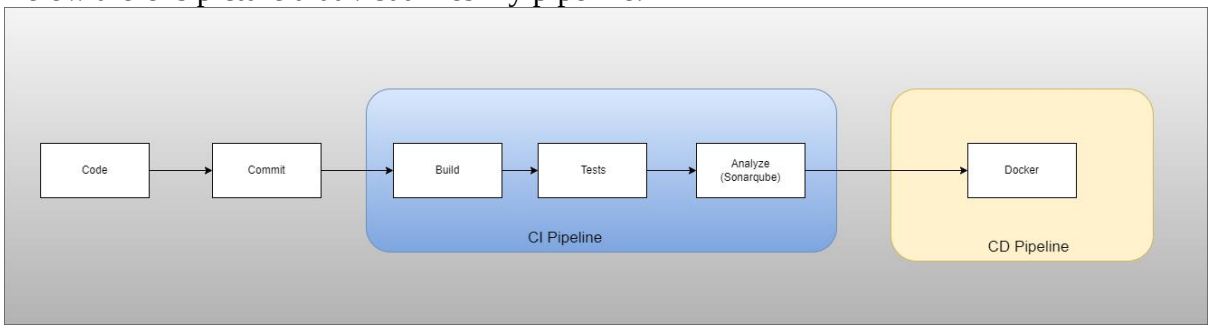
- Front-end: React JS
- Back-end: Spring boot

Front-end: React JS

First to begin, what is React and what can you do with it? React is originally made by Facebook and it aims so that you can easily create fast user interfaces, in my case for a website. (techtmagic, 2022)
Second of all, why am I using React JS for this project? React has itself that you naturally have to isolate UI components. By doing this everything is loosely coupled which means it is easy to write unit tests and maintain my application. (builtin, 2022)

Diagram CI/CD Pipeline

Below there is picture that visualizes my pipeline.



Reference

10 Key Reasons Why You Should Use React for Web Development. (2022, 03 15). Retrieved from techmagic: https://www.techmagic.co/blog/why-we-use-react-js-in-the-development/#why-use-react-js-for-web-development

Dependency inversion principle. (2022, 09 14). Retrieved from Wikipedia: https://en.wikipedia.org/wiki/Dependency_inversion_principle

Starting Out in Software Engineering? Don't Bother Learning React JS. (2022, 06 08). Retrieved from builtin: https://builtin.com/software-engineering-perspectives/dont-learn-reactjs