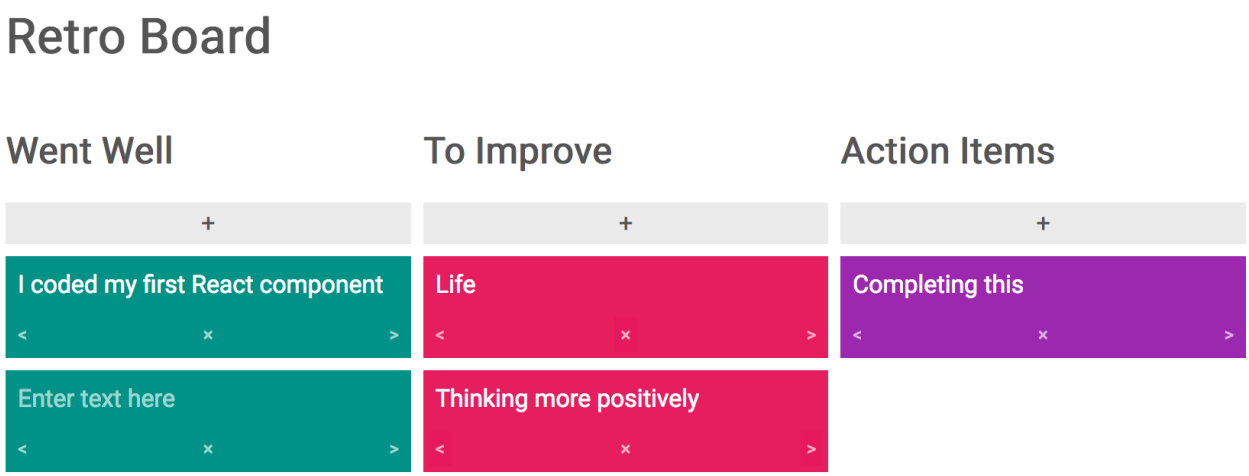
Project 1 Description

Retrospective Board

Due Date: **April 1, 2019**[1]

For your first project, you will be creating a retrospective board. A retrospective board is used within the *Scrum* workflow. Scrum is the most popular Agile style of software project management used to guide the software development lifecycle. Scrum style management is based on four basic ceremonies for each sprint or software release: the 1.) Sprint Planning meeting, 2.) Daily Stand-up, 3.) Sprint Review, and 4.) Sprint Retrospective. As a part of the Sprint Retrospective, team members will use a retrospective board to reflect on what went well and what they need to improve on for next time.

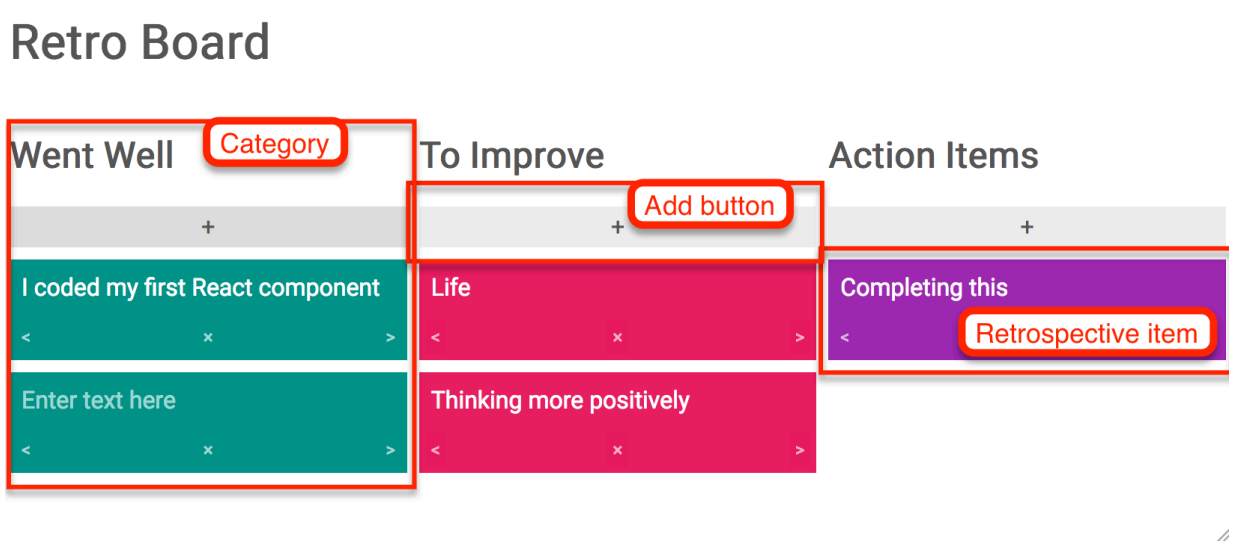
This project is inspired by *FunRetro*. If you are not familiar with a retrospective board, please visit the *FunRetro* website and create a new board.



[1] The project is due on this date, but as long as you make a valid attempt, you can resubmit your work by the last day of class (May 1, 2019) for a better grade.

Project Requirements

Your retrospective board must meet the following requirements:

UI Requirements

The Retro Board should have three categories: 1.) Went Well 2.) To Improve and 3.) Action Items. The categories must be displayed in this order. They must be arranged in such a way that the user can easily tell what UI elements on the page belong to what category.
Each category should contain an add button or another type of UI element. When clicked or activated, the app will add a new retrospective item to the given category.
The user should be able to type text inside of each retrospective item. This text will be stored in state. The only time the text can change is when the user is typing inside the retrospective item.
The user should be able to delete each retrospective item. When deleted, the retrospective item should be removed from state.
The user should be able to move the retrospective item into a different category by clicking on a left or right arrow. If there is not a category to the left, then clicking the arrow should move the retrospective item to the rightmost category. The same applies to if there is no category to the right but in reverse. When the item moves from one category to the next, the retrospective item's user input, visual structure, and appearance (with the exception of color) should stay the same.
For the final UI requirement, you must pick one of the following:

1. Add a “thumbs up” and “thumbs down” button or clickable elements to your retrospective items. Clicking on one of these buttons should increase the number of “thumbs up” or “thumbs down” votes. Display the number of “thumbs up” and “thumbs down” votes in each retrospective item.
2. Validate user input. Make text within the retrospective item required. When the retrospective item loses focus or is submitted (the type of event will depend on how you design the functionality of your app), the text should be validated in some way. Suggestions are to remove the retrospective item from the board or display an error message.
3. Add functionality to change the layout of the retrospective board. Include some type “layout switcher” UI element or elements. When clicked or activated, the layout of the categories will change between a horizontal and vertical layout. For an example, please visit the *FunRetro* website and create a new board.

An example of a “layout switcher” from FunRetro.

Code Requirements
This application should contain at least two React components. At least one of the React component should extend the React Component class.
This application should use state within the React component and update state correctly with *this.setState()*.
The application should handle events within React components.
Styling must be included, but you will not be graded on how visually appealing your app is or how well your CSS is written.

An example of HTML and CSS can be found within are GitHub repository within the *projects/project1/example-styles*. If you copy all or a majority of the HTML and CSS, you must credit Matina Patsos and Jamal Taylor on your project.

Tips

Work on one or both of these two things first before diving in: 1.) rendering static, unfunctional JSX with dummy data 2.) designing your data model (what your state will look like and how you will modify state).

If you need to, design your data model in isolation. Remember, *this.setState* uses *Object.assign().*

Make use of high order functions with arrays (e.g. map, filter, find, reduce) when setting state. E.g. *this.setState({ myObj: this.state.myObj.reduce(...) });*

If you find using more than one component challenging, begin with a single component. Then break your component down into at least two separate components later.

With your event handlers (methods), log the value first before modifying state e.g. *console.log(e.target.value)*. This way, you know whether or not your event handler is working.

When looping through an array, you may find it helpful to put in placeholder data first before using real data.

If you need to loop through an object inside JSX, convert it into an array with *Object.values()* or *Object.entries()*.

Project Submission

You must create a new git repository for your first project. Within your project, you will need to include a *package.json* and *README.md* file in the project root directory. All files must be included to install, build, and run your application. You are limited to Node.js packages. Instructions on how to install or start your application must be written in the *README.md*. You can find an example template of a *README.md* here: https://gist.github.com/PurpleBooth/109311bb0361f32d87a2

While not required to do so, you can style your *README.md* with Markdown. A cheatsheet for Markdown can be found here: https://github.com/adam-p/markdown-here/wiki/Markdown-Cheatsheet)

Before or on the day the project is due, you must share a link to your GitHub repository to both on the instructors in Slack.

Grading Rubric

You will need to complete all the coding requirements *and* receive 24 out of 27 points in order to pass. Note that if we cannot start or install your application, it is an automatic failure. For full descriptions, please see the UI and coding requirements above.

Short Description	Points
Application Setup	
Is the application easy to install and start?	REQUIRED TO PASS
Does the application include a README.md file within the root of the git repository and with clear instructions on how to install and start?	1
Application Functionality	
Does it work? Are there any major bugs?	1
Code Quality	
Does the code run without errors or warnings in the console?	1
Meeting Requirements (THIS IS NOT THE FULL DESCRIPTION. REFER TO UI REQUIREMENT ABOVE.)	
The Retro Board should have three categories ...	4
Each category should contain an add button or ...	4
The user should be able to type text inside ...	4
The user should be able to delete each retrospective item ...	4
The user should be able to move the retrospective item ...	4
For the final UI requirement, you must pick one of the following ...	4
Does it meet all the coding requirements?	REQUIRED TO PASS