TEAM.NAME V2

TEAM_03

~~MOZILLA: CODE-COVERAGE~~
REACT-BOOTSTRAP

Deliverable 2

Members:
Alexei COREIBA
Gaganpreet KABERWAL
Harsh PATEL
Kohilan MOHANARAJAN
Kanstantsin ILIOUKEVITCH

Supervisor:
Prantar BHOWMIK
Anya TAFLIOVICH

March 11, 2020

Contents

1 Disclaimer

During Deliverable 1 we made the choice of selecting a Mozilla project, code-coverage, as our open source project for this term. Sadly, we were unable to continue with the project due to many issues relating to building it. With the developers living in France, their reluctance's to give us dev environment secrets, and there being poor documentation on the project in general, we were unable to build the project in a dev environment even until Thursday, March 5th. Because of this, we've asked Anya to switch projects to React-Bootstrap, and thankfully this was approved.

React-Bootstrap is a frontend framework meant to accelerate the development of frontend web development. It incorporates components that we know and love from Bootstrap into the React framework. The original repository can be found here: github. Our forked repository can be found here: github. The implementation of different Bootstrap elements can be found in the 'src' file within the main repository. Each JavaScript file is a single Bootstrap element that can be used to the public. Each component must include proper documentation and testing in order to be considered an 'active' component and be included into the master branch. Their testing is done through Enzyme. Enzyme is a JavaScript framework that helps automate React component testing. Since most components are independent of each other, only unit testing is done prior to being commited to their master branch. Their tests can be found in the 'test' folder on their master branch. Bugs definitely are missed, which can be seen by their number of issues meaning there is a need for implementing bug fixes. There is also a large collection of Bootstrap elements that have yet to be ported to this project. This gives us the possibility of selecting one of these components to develop later on in Deliverable 4.

The rest of our development process will remain as was described in Deliverable 1, but the only difference being the project which we will work on.

2 Promising and Interesting Bugs

All the bugs that are listed below were selected due to the possibility of completion keeping in mind our tight deadline. As we've technically restarted our project as of March 4th, we have 5 days to get ourselves accustomed to the code base, learn how to build, and learn how their testing is done. This means that we must select bugs that are possible to be fixed within a sitting and test cases or scenarios can be made within the same sitting. All the bugs are not extremely technically challenging, but are not being actively fixed by the developers and are thus great options for our team.

2.1 Potential Bug 1

Our first bug that we've decided to look into was the following: github-issue #3654. This bug is tied to the Carousel Bootstrap component which can be found here: Carousel. The bug is described as follows: when at the first index of the Carousel, and scrolling to left to the last index, the animation is shown as going right instead of going left. The location of the bug can be found here: github-file. Specifically, the logic behind the scrolling animations seems to have been improperly implemented. An estimate into the hours needed to develop a fix for this issue is 3.

2.2 Potential Bug 2

Our second bug that we've decided to look into was the following: github-issue #3589. This bug is tied to the Dropdown Bootstrap component which can be found here: Dropdown. The bug is described as follows: when setting the 'flip' attribute to false, the attribute is not set to false, meaning that the dropdown still flips. The 'flip' attribute is mentioned in the following file: github-file. The bug seems to be that the flip attribute is never properly passed into the dropdown component and is thus never truly set. This would be the first point of investigation into the root of the issue. An estimate into the hours needed to develop a fix for this issue is 4.

2.3 Potential Bug 3

Our third bug that we've decided to look into was the following: github-issue #3497. This bug is tied to the Navs Bootstrap component which can be found

here: Navs. The bug is described as follows: TabPane, with unmountOnExit, doesn't keep it's transition that was originally set. The proper behaviour is that the transition should follow through. This bug seems to be traced to the following line in the following file: github-file-line. There is a potential logic change to fix the bug suggested in the comments which is a simple check to ensure that the pane is not unmounted prior to the animation executing. An estimate into the hours needed to develop a fix for this issue is 3.

2.4 Potential Bug 4

Our fourth bug that we've decided to look into was the following: github-issue #4667. This issue may not necessarily be a bug, but is an interesting observation of the setup behind React-Bootstrap. The original HTML DOM components are not necessarily easily accessible through React-Bootstrap. This issue is marked with 'needs more info' which is both accurate and not. Typically, the HTML file is dynamically rendered by React-Boostrap depending on what is needed, and you would start off with a clean base. So as this would be an interesting bug to look into, this may not be feasible given our tight deadline. This issue would have to be further analyzed into how React can access the original HTML DOM. There can be a patch made by using JQuery to fetch a required property but this is not recommended by the developers. An estimate into the hours needed to develop a fix for this issue is 15.

2.5 Potential Bug 5

Our fifth bug that we've decided to look into was the following: github-issue #4933. This bug is tied to the Carousel Bootstrap component that is linked in 2.1. The bug is described as follows: when on IE, the Carousel 'onSlideEnd' is never triggered. The comments further mention that this is not directly an issue with React-Bootstrap but a JavaScript issue on IE. A patch can be implemented to ensure that this event works properly on IE. The devs do mention that they don't necessarily support IE and therefore this issue should be considered as low priority for the team, but a possible choice. Since this issue is tied to an IE issue, a workaround must be thoroughly considered, and one hasn't been explored at this time. An estimate into the hours needed to develop a fix for this issue is 10.

3 Selected Bugs

3.1 Selected Bug 1: 2.1 (#3654)

3.1.1 Implementation Details

The bug fix can be found on lines 159-160. This was a simple logical error with figuring out which direction the slide animation should be set to. Previously, they would slide in the 'next' direction if they were at the last index and going to the first index or if the previous index - or current index - is less than the next index. This last clause is where the issue lies, as if you were at the first index and going to the last index - in the 'prev' direction - it would always be true, even though you are meant to go in the 'prev' direction. The fix that was implemented was to ensure that you are not going from the first index to the last index. By implementing this, the direction would not be set to 'next' and by default be set to 'prev' on line 164. This fix doesn't affect any other component within the project and thus doesn't have any negative impact on the rest of the codebase.

3.1.2 Testing Details

We created Enzyme unit tests to ensure that the correct direction was being set during the animation of the transition. This is standard for any issue that is being worked on throughout this codebase.

For our acceptance testing, we created a dummy app which can be found on our first github repository within the 'test-app' folder. This is a blackbox test to ensure that the issue is no longer occurring and the component is acting as it should be. We created a carousel object and tested whether going in the 'prev' direction on the first index to see whether the slide animation would be correct.

3.2 Selected Bug 2: 2.2 (#3589) - BLOCKED

This issue was officially selected as we thought we would be able to address it within the limited amount of time that we had for this deliverable. Upon further investigation, we have discovered that this issue is not directly linked to the React-Bootstrap project. The underlying technology that this relies on is popper.js which is not used directly by React-Bootstrap but another

project that is then used within React-Bootstrap. The project that is responsible for this is React-Overlays, and as this is not within the same project we are not able to address this issue. We have left a comment under the thread of this issue on github to ensure that our findings are communicated with anyone else whom comes across this issue. Our comment states that this issue should be linked to any issue currently linked in that repository that mentions the same behaviour. Since this issue was no longer within our scope, we have moved it to the blocked lane.

3.3 Selected Bug 3: 2.3 (#3497)

3.3.1 Implementation Details

The bug fix can be found on lines 139. This was another logical error that dealt with the TabPane component. The TabPane component was meant to retain its transition regardless of whether the `unmountOnExit` field is set to `true` or not. Previously, the component only checked if the component was active and if `unmountOnExit` was set to true in order to run the transition, without checking the `Transition` field. To fix this, we updated the check (`!active && unmountOnExit`) to include a check for

```
!(active || Transition) && unmountOnExit.
```

This fix is isolated, as it is related to the component's animation, thus not affecting the rest of the codebase.

3.3.2 Testing Details

Similarly to the first bug, we created Enzyme test units to ensure that the component would send back null if it was not active or transitioning, and `unmountOnExit` was set to true. Unfortunately, we were not able to finish the tests in time.

For our acceptance testing, we used the same dummy app from before to do black box testing on the component, to ensure that it was acting as it should. We created a tabPane object with `unmountOnExit` set to true, and tested whether or not the transition occurred.

3.4 How the changes affect the codebase

The changes are isolated to each component, so aside from the one line in those individual components that they fix, they don't change the codebase drastically. The files that were changed were `src/TabPane.js` and `src/Carousel.js` for the fixes, and `test/CarouselSpec.js` and `test/TabSpec.js` for the tests. As our dummy app was outside of the `react-bootstrap` repository, it does not affect the codebase. The code changes can be found in the `bug/RBT3-3497` and `origin/bug-Carousel-not-sliding-correctly-3654` branches.

4 Software Development Process

Our choice of software development process was the Kanban Method. We have successfully employed certain practices to utilize this methodology to its full potential. The visualization tool that was used was Github Project Board. Although our original plan was to use swiftkanban.com for visualization, we ran into multiple issues with setting up the board and the whole process was not efficient or user friendly. Thus we decided to switch to github. We added the required columns needed to tailor to our specific project, namely:

1. Todo
2. Planning
3. In Development
4. Blocked
5. Code Review
6. Acceptance tests
7. Done

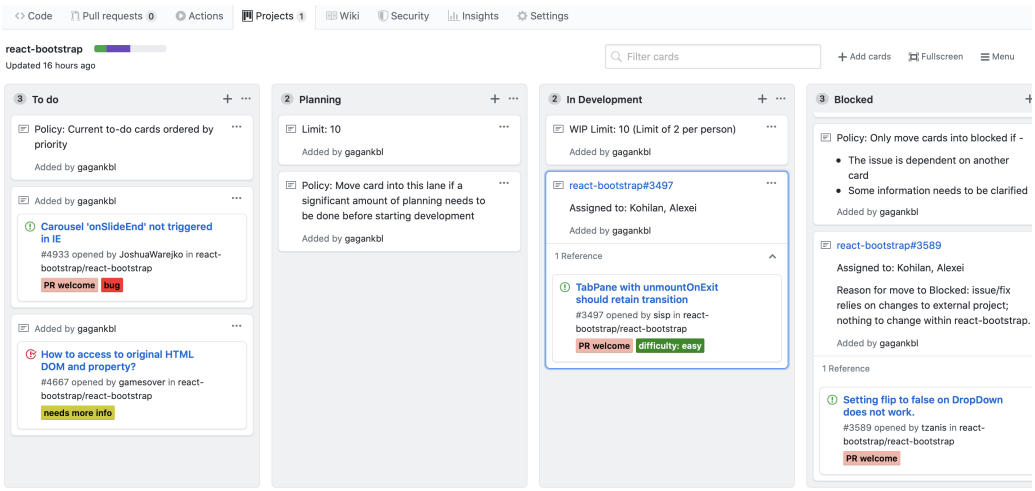

Figure 1: Github Project Board

Each column was given a specific policy that each team member much adhere to before moving a specific issue into that column. Work in progress limits were set for the needed columns. The global limit for in progress items was set to 10 (with a limit of 2 issues per person), but certain columns needed a different limit. For example: Code Review and Acceptance Testing limits were set to 5. This was to constrict each person to only peer review/ test one issue at a time which helps in improving the quality of the delivered work. All these policies and limits are visible on the board which can be accessed with the link above.

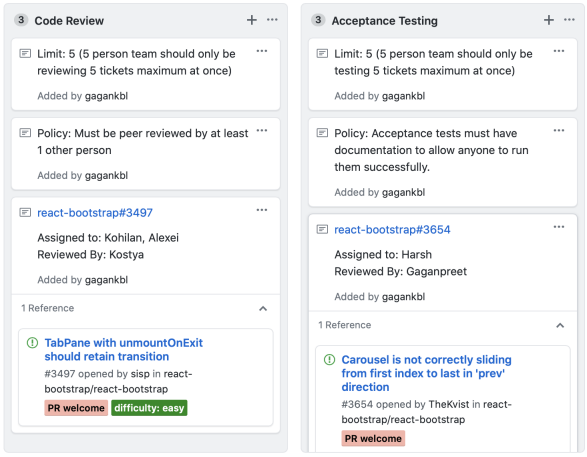

Figure 2: WIP Limits and Policies

The Code Review column was a necessity as it helped us implement feedback loops. With the help of pair programming, each member was able to receive quick and useful feedback from at least one other person. Regular meetings also added to feedback and helped deliver a correct and efficient solution.

There were also scenarios where a certain issue could no longer be worked on due to reasons not under control of the team. Such scenarios included: lack of communication from the project admin side, not being supplied with the correct security tokens, the issue getting closed before we implemented the fix etc. The Kanban method allowed us to adjust to these scenarios with ease and resolve any bottlenecks as efficiently as possible. An issue would be

moved to the blocked column, opening up more space for issues that can be picked up. Since the ToDo column's policy was to organize issues by priority, choosing the next ticket to work on was of minimal difficulty for any team member.

When choosing to work on a new issue, risks were evaluated and a scientific method for generating a hypothesis was used to allow team-level understanding of the problem and come up with steps for improvement (for the issue or for the process itself). This enabled deeper collaboration and allowed the team to reach a shared consensus. These techniques helped us manage the workflow and make the process smoother to adjust to any changes that took place.

Due to the last minute pivot, we were in a tough situation with very little time and a lot to do. We utilized the ability to have daily meetings during which we discussed our progress and the issues that we were experiencing. These meetings were done through Discord as it was easily accessible to all team members and had the useful feature of sharing screens.

Utilizing the share screen feature, we were able to do many things together such as:

- Explanation and walkthroughs of code
- Group code review
- Pair programming

The pair programming is an example where we implemented something from the XP process.

Thanks to our choice of Kanban, with a little of XP, we were able to move quickly enough to do a full pivot of our project and meet the deadline successfully.