

Frontend Code Review #1:

This review is in edit mode

Here you can edit files already in your review

- Click the red 'x's to remove files and revisions
- Click the 'Add Revisions' menu item in a file's toolbar to add newer (or older) revisions

Details

Objectives

General Comments

Unresolved 0

Resolved 0

Number of files included: 143

13

documentation

assets

css

bootstrap.min.css

demo-documentation.css

material-dashboard.css

img

faces

marc.jpg

apple-icon.png

cover.jpeg

favicon.png

mask.png

new_logo.png

reactlogo.png

sidebar-1.jpg

sidebar-2.jpg

sidebar-3.jpg

/documentation/assets/img/sidebar-1.jpg Added

1628b3f x


zhangfan

we need to remove these redundant files after design

Reply · Edit · Delete · Mark as needs resolution · Add to favourites · 15:59

Add a file comment.

New version



- Defects: there are many redundant content.
- Suggestions: Make sure remove all the contents we don't need.

Add a file comment. Click on source lines to add an inline comment.

```
1 import Dashboard from "layouts/Dashboard/Dashboard.jsx";
2 import Landing from "layouts/Landing/Landing.jsx";
3
4 const indexRoutes = [
5   { path: "/admin", component: Dashboard },
6   { path: "/creator", component: Dashboard},
7   { path: "/", component: Landing},
8 ];
9
10 export default indexRoutes;
```

zhangfan

The architecture looks very clear. Maybe later should redirect the page to login page if the user hasn't logged before.

Reply · Edit · Delete · Mark as needs resolution · Add to favourites · 16:00

- Defects: In this way, people who haven't logged in could visit any page with url. It doesn't make sense.
- Suggestions: Add something to record if people have logged in, and if not, redirect to login; else show this pages. BTW, We could also use some cookies to record the state of login. But of course, these are all future work.

```

152         }}
153     />
154 </h4>
155 <Table
156     tableHeaderColor="success"
157     tableHead=[["Vertical", "# of Title Needed", "# of Title Completed", "Assignee"]]

```



zhangfan

According to the last client meeting, it looks we don't need assignee any more. :)

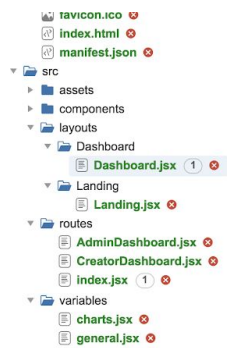
[Reply](#) · [Edit](#) · [Delete](#) · [Mark as needs resolution](#) · [Add to favourites](#) · 16:08

```

158     tableData=[["Education", "30", "25", "Dakota Rice"],
159               ["Science", "25", "31", "Minerva Hooper"],
160               ["Sports", "40", "33", "Sara Rodriguez"],
161

```

- Defects: According to the last meeting with clients, we don't need assignee anymore. If I am wrong, correct me.
- Suggestions: remove that.



```

47     handleDrawerToggle = () => {
48         this.setState({ mobileOpen: !this.state.mobileOpen });
49     };
50
51     resizeFunction() {
52         if (window.innerWidth >= 960) {
53             this.setState({ mobileOpen: false });
54         }
55     }

```



zhangfan

Do we need to fit the window inner width to the phone screen window?

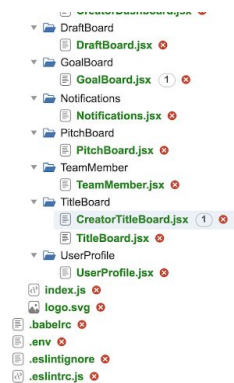
[Reply](#) · [Edit](#) · [Delete](#) · [Mark as needs resolution](#) · [Add to favourites](#) · 16:32

```

55     }
56     componentDidMount() {
57         if (navigator.platform.indexOf("Win") > -1) {
58             const ps = new PerfectScrollbar(this.refs.mainPanel);
59         }
60         window.addEventListener("resize", this.resizeFunction);
61     }

```

- Defects: not a big deal, but do we need to consider the phone screen window? If I am wrong, correct me.
- Suggestions: add code to adapt the mobile screen shot.



```

79     }
80
81     render() {
82         const { classes } = this.props;
83         return (
84             <div>
85                 <GridContainer className={classes.titleBoardContainer}>
86                     <GridItem xs={12} sm={12} md={12}>

```



zhangfan

Should we use const variables to replace the numbers?

[Reply](#) · [Edit](#) · [Delete](#) · [Mark as needs resolution](#) · [Add to favourites](#) · 16:36

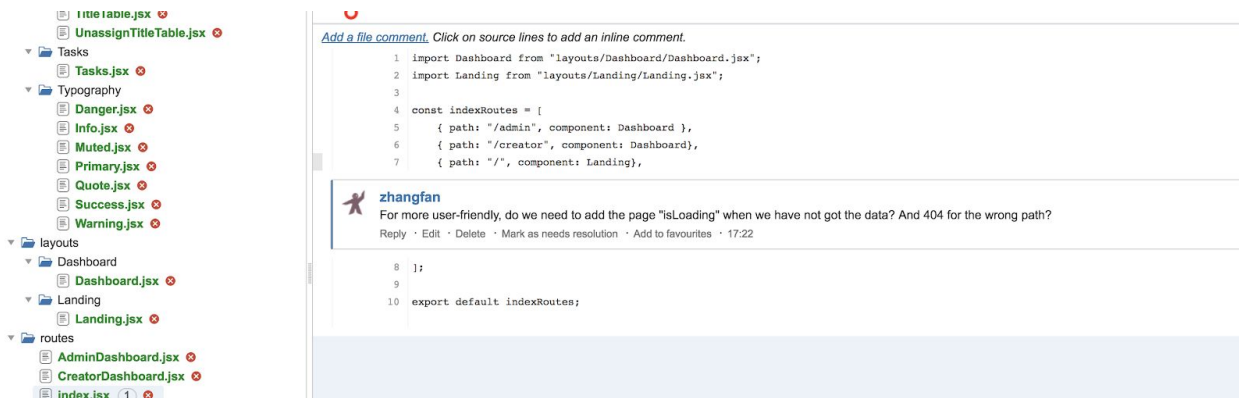
```

87         <Card>
88             <CardHeader color="warning" className={classes.cardHeader}>
89                 <div>
90                     <h4 className={classes.cardTitleWhite}>
91                         <Book className={classes.headerIcon} />
92                         Assigned Title Table
93                     </h4>
94                     <p className={classes.cardCategoryWhite}>
95                         Title assigned to content writer in this week
96                     </p>
97                 </div>

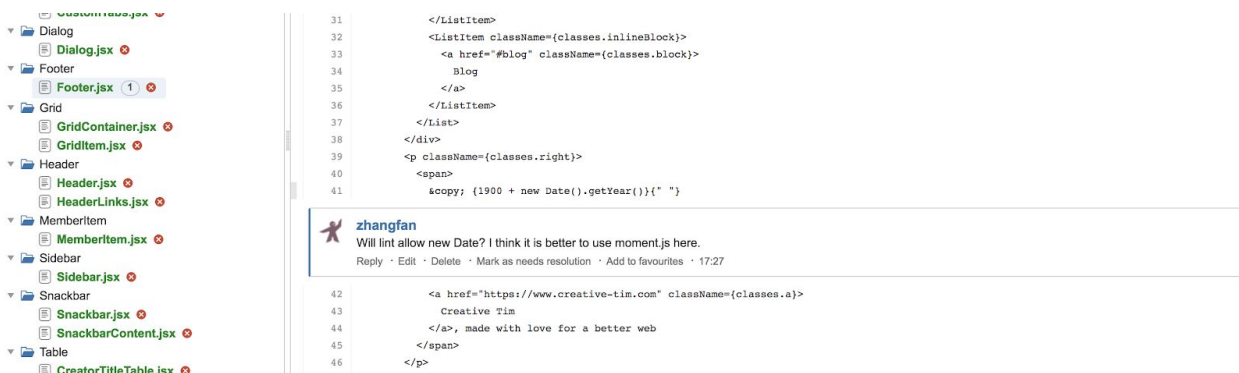
```

- Defects: It is not big deal in the bingging. But if the code grows, using numbers to represent styles, parameters will be trouble. What's more, it is good to have product style to make the style looks consistent.

- Suggestions: Use const for each numbers. For example: const XS = 12



- Defects: It will be confused if users go to the pages and see nothing just because the data haven't been loaded.
- Suggestions: add loading pages later after we add reducer, actions and connect to backend API.



- Defects: Not big deal. It will have ESLint(if we open that check) warning when using new Date.
- Suggestions: Use moment.js library rather than use new Date.

The screenshot shows a code editor with a file explorer on the left and a code editor on the right. The file explorer shows a project structure with folders like Grid, Header, MemberItem, Sidebar, Snackbar, Table, Tasks, and Typography. The code editor shows the content of 'Header.jsx', which includes imports for Material-UI components and a function definition for 'Header'.

```

3 import PropTypes from "prop-types";
4 // @material-ui/core components
5 import withStyles from "@material-ui/core/styles/withStyles";
6 import AppBar from "@material-ui/core/AppBar";
7 import Toolbar from "@material-ui/core/Toolbar";
8 import IconButton from "@material-ui/core/IconButton";
9 import Hidden from "@material-ui/core/Hidden";
10 // @material-ui/icons
11 import Menu from "@material-ui/icons/Menu";
12 // core components
13 import HeaderLinks from "../HeaderLinks.jsx";
14
15 import headerStyle from "assets/jss/material-dashboard-react/components/headerStyle.jsx";
16
17 function Header({ ...props }) {
18   function makeBrand() {
19     var name;

```

The screenshot shows a code editor with a file explorer on the left and a code editor on the right. The file explorer shows a project structure with folders like 'Custom tabs.jsx', 'Dialog', 'Footer', 'Grid', 'Header', 'MemberItem', 'Sidebar', 'Snackbar', and 'Table'. The code editor shows a JSX element for a table with pagination. The table has columns 'id', 'name', 'email', and 'password'. The pagination component is located at the bottom of the table footer.

Summary:

Frontend Code Review #2:

Due to technical difficulties with crucible, we conducted a second code review of the frontend on Github directly.

10

src/routes/index.jsx

Show comments

...

...

@@ -0,0 +1,10 @@

1

+ import Dashboard from "layouts/Dashboard/Dashboard.jsx";

2

+ import Landing from "layouts/Landing/Landing.jsx";

3

+

4

+ const indexRoutes = [

5

+ { path: "/admin", component: Dashboard },

6

+ { path: "/creator", component: Dashboard},

7

+ { path: "/", component: Landing},

8


+];

9

+


10

+ export default indexRoutes;



russomp 11 minutes ago

not sure if we want separate routes or if we want to render different components based on user type returned from API; we should discuss this further.



Reply...

6

+ { path: "/creator", component: Dashboard},

7

+ { path: "/", component: Landing},

8

+];

9

+

10

+ export default indexRoutes;

150

src/variables/charts.jsx

Show comments

...

...

@@ -0,0 +1,150 @@

1

+ // #####

2

+ // // // javascript library for creating charts

3

+ // #####

4

+ var Chartist = require("chartist");

5

+

6

+ // #####

7

+ // // // variables used to create animation on charts

We need to think about how we want to organize our routes as a team. We currently have the frontend organized into two distinct sets of routes -- admin and writer routes -- each with different idea management boards. This will work, but we may want to consider using general routes and rendering specific tables based on user types.

```
32 + class Dashboard extends React.Component {
33 +   state = {
34 +     value: 0
35 +   };
36 +   handleChange = (event, value) => {
```



russomp 18 minutes ago

event is unused; maybe consider using an _ to suggest that the variable is not used



Reply...

```
37 +   this.setState({ value });
38 + };
39 +
40 + handleChangeIndex = index => {
41 +   this.setState({ value: index });
42 + };
43 + render() {
44 +   const { classes } = this.props;
45 +   return (
46 +     <div>
47 +       <GridContainer>
48 +         <GridItem xs={12} sm={12} md={12}>
49 +           <TextField
50 +             id="date"
51 +             label="Date"
52 +             type="date"
53 +             defaultValue={new Date().toISOString().substring(0, 10)}
54 +             className={classes.textField}
55 +             InputLabelProps={{
56 +               shrink: true,
57 +             }}
58 +           />
59 +         </GridItem>
60 +       </GridContainer>
61 +     </div>
62 +   );
63 + }
```

This is just a stylistic suggestion. Since event is unused, it might be nice to use a variable name that signals to developers that come onto the project that the value is never consumed. This can be an _ or explicit name like notUsed.

```

97 + class DraftBoard extends React.Component {
98 +   constructor(props) {
99 +     super(props);
100 +     this.state = {
101 +       showDialog: false,
102 +       showLog: false,
103 +       dialogTitle: "Create new draft"
104 +     };
105 +   }
106 +   handleCreateGoal = () => {

```



russomp an hour ago

consider using bind in the constructor rather than arrow function



Reply...

```

107 +   this.setState({

```



russomp an hour ago

+ 😊 ...

consider using the functional version of setState since React batches state changes for performance reasons

```

this.setState(prevState => ({ dialogTitle: "Create new draft", showDialog: !prevState.showDialog
}));

```



Reply...

```

108 +     dialogTitle: "Create new draft",
109 +     showDialog: !this.state.showDialog
110 +   })
111 + };
112 +
113 +   cancelCreateGoal = () => {

```



russomp an hour ago

consider using bind in the constructor rather than arrow function



Reply...

```

114 +   this.setState({

```

Using bind in the constructor is currently recommended in the React docs, since it is an explicit bind of this to the given class function. This ensures that the value of this is maintained when the function is passed down to other components via props. Either way works, but arrow functions make an implicit binding and a full copy, rather than a shallow bind prox.

It is also recommended to use the functional version of set state to avoid race conditions during async batch updates.

```

202 +         <Table
203 +             tableHeaderColor="rose"
204 +             tableHead=[["Title", "Pitch creator", "Reporter", "Status"]]

```



russomp 22 minutes ago

who is the reporter?



Reply...

```

205 +         tableData=[
206 +             ["If we have something wrong to im...", "Kerry Deng", "Dakota Rice","Approved"],
207 +             ["God bless the world..", "Kerry Deng", "Dakota Rice","Approved"],
208 +             ["How could mortal do such...", "Kerry Deng", "Minerva Hooper", "Approved"],
209 +             ["The most important news happend...", "Kerry Deng", "Minerva Hooper", "Parked"],
210 +             ["Do you know how many chicken...", "Kerry Deng", "Minerva Hooper", "Parked"],
211 +         ]
212 +     >
213 +     </Table>
214 +     </CardBody>
215 + </Card>
216 + </GridItem>
217 + </GridContainer>
218 + </div>
219 + );
220 + }
221 + }
222 +
223 + Dashboard.propTypes = {
224 +     classes: PropTypes.object.isRequired
225 + };
226 +
227 + export default withStyles(dashboardStyle)(Dashboard);

```

The team needs to discuss terminologies after our most recent client meeting and nail down a consistent vocabulary.


```
65 + function TitleBoard(props) {
```



russomp an hour ago

title board is no longer needed after recent success-critical stakeholder feedback



Reply...

```
66 +   const { classes } = props;
67 +   return (
68 +     <GridContainer className={classes.titleBoardContainer}>
69 +       <GridItem xs={12} sm={12} md={6}>
70 +         <Card>
71 +           <CardHeader color="primary" className={classes.cardHeader}>
72 +             <div>
73 +               <h4 className={classes.cardTitleWhite}>
74 +                 <Book className={classes.headerIcon} />
75 +                 Unassigned Title Table
76 +               </h4>
77 +               <p className={classes.cardCategoryWhite}>
78 +                 Title need to be assigned
79 +               </p>
80 +             </div>
81 +           <FormControl className={classes.formControl}>
82 +             <InputLabel className={classes.tableHeaderLabel} htmlFor="vertical-simple">Vertical</InputLabel>
83 +             <Select
84 +               className={classes.tableHeaderLabel}
85 +               value="Lifestyle"
86 +               inputProps={{
87 +                 name: 'vertical',
88 +                 id: 'vertical-simple',
89 +               }}
90 +             >
```

The team also needs to update the different boards based on feedback from our most recent meeting with success-critical stakeholders.

150 src/variables/charts.jsx

Show comments

...

...

@@ -0,0 +1,150 @@

1

+ // #####

2

+ // // // javascript library for creating charts

3

+ // #####

4

+ var Chartist = require("chartist");

5

+

6

+ // #####

7

+ // // // variables used to create animation on charts

8

+ // #####

9

+ var delays = 80,

10

+ durations = 500;

11

+ var delays2 = 80,

12

+ durations2 = 500;

13

+ +

14

+ // #####

15

+ // // // Daily Sales

16

+ // #####

17

+

18

+ const pitchChart = {

19

+ data: {

20

+ labels: [

21

+ "Jan",

22

+ "Feb",

23

+ "Mar",



russomp

13 minutes ago

these custom charts are really cool 🍷



Reply...

19

+ data: {

20

+ labels: [

21

+ "Jan",

22

+ "Feb",

23

+ "Mar",