Sprint # 01 - Team 34

09 March 2021

Ishwor Giri, Muhammad Shahzaib Tahir Awan

FRONTEND INTRODUCTION

All the source code related to frontend can be found inside frontend app folder

Entry point is **index.js** and all components are put together using imports from various js files or modules.

All web pages feature a navigation bar with items "Home," "About," "Login," and "Sign Up." It also has a placeholder for a logo, which is temporarily occupied by an alternate text reading "Beer Game."

Index

The *Index.js* imports *App.js* from "./components/App" directory.

App.js centralises activity.

It imports all other components through their respective directories as well as BrowserRouter as Router, Switch, Route, and Link from React.

It makes use of <Switch> protocol, which looks through its children <Route> and renders the first result that matches the current URL it holds. All the routing to the app is handled using this and other components can refer to these endpoints using <Link to=""> for quick render without reloading pages instead of using anchor tags from html.

Login and Sign Up

Login.js , Signup.js

Login and Sign Up pages make use of simple Bootstrap forms and labels to extract user information: email and password.

The user can input their information and the information can be stored in a database only when such a database is connected. Functions related to the backend are not implemented yet. One can do so by simply creating functions inside respective class components and handling using onclick and setstate properties of javascript and reactjs.

Homepage

homepage.js

The Homepage simply gives the user an option to choose between acting as a **Player** or an **Instructor**. uses Link to route to respective pages /player/ and /instructor/.

INSTRUCTOR AND PLAYER VIEWS

PlayerView.js, InstructorView.js

Join as: Instructor

Once the user has chosen to join as an instructor, the user will be redirected to *InstructorView.js* directory (./instructor/).

InstructorView.js gives the user the following options:

1. **Create Game**

    CreateGame.js

In order to create a new game, the instructor has to fill in the following information: Enable Wholesaler (type: checkbox,) enable Retailer (type: checkbox,) whether to Share Activity with Other Players (type: checkbox,) Week Length, Backorder Cost, Password for the game, and Demand Pattern through a dropdown with hardcoded values (can be changed later.)

    NOTE: NOT THE FINAL FORM STRUCTURE, needs more properties to be added

2. **Create Demand Pattern**

For now, the button doesn't redirect the user to any useful directory as it has been left empty intentionally and it may be used for constructing further functions.

3. **View Created Games List**

Additionally, the Instructor View displays a table with the following information: Game ID, Number of Active Players, Status, and Options. The instructor can modify the status of the game by simply clicking on the "Modify" button and can also "View Details."
Table is created using another functional component that takes the id of the game as props and later can be used to fetch details for the specific game to display actual values and correct buttons.

```
<GameInfo id="1" />
```

For now, modify and view details are dynamic and redirects to another page for modifying and viewing details of the specific game using routes as shown below.

/game/modify/{gameid}
/game/details/{gameid}

If the instructor clicks on the MODIFY button then he/she shall be redirected to another Component from GameEdit.js where one can implement functions to display details of the game inside the form and changes can be done . Currently it only displays game id which is being fetched from the url using this.props.match.params.gid

One thing to consider here is that the component is being exported wrapped with a withRouter function that allows use to access url parameters.

```
export default withRouter(GameView)
withRouter imported from react-router-dom
```

Also the props are being sent using this routing setup inside App.js function.

```
Route exact path="/game/modify/:gid">
             <GameEdit />
            </Route>
```

Same applies for VIEW details which contain features to freeze game, plot, print password and other game related options but no implementation for visual layout has been done yet.

```
  <Route exact  path="/game/details/:gid" >
             <GameView />
            </Route>
```

Some of the components also use componentDidMount function to check if the component has rendered on the browser and the title of the page should be changed.

```
  componentDidMount() {
    document.title = " Game View Details";

  }
```

Join as: Player

Once the user has chosen to play as a player, the user will be redirected to *PlayerView.js* directory (./player/).

The user shall have the option to join a new game, for which he has to enter an Unique Code provided by the Instructor and their password. On submitting this information, the user shall be redirected to another page *PlayerChooseRole.js* where he will have the option to choose a role: Manufacturer, Distributor, Wholesaler, or Retailer.

For now the Submit button will not work because we don't have any backend handling for this form submission.
On clicking “Play Game,” the player shall be redirected to the actual game view - *PlayGame.js* - and can start playing the game.

One Another Column, user can see the ongoing game(only one) and can continue the game without entering any game id, password , role.
As the Project Requirement we have added a Create Game button that will redirect to the same create game that was presented to the instructor but one can change and define according to the requirements for this project.

THE GAME SCREEN

PlayGame.js

The game screen makes use of `Tab` and `Tabs` components from the 'react-bootstrap' library to separate information into four views: Orders, History, Plot, and More Info.

Orders: view1()

Players can see all the information for the current week such as demand from DOWNSTREAM, back Order, Total Requirements, Beginning Inventory, Incoming Shipment and Total Availability, Units Shipped to Downstream, Ending inventor. There is also a form and button to submit the request to purchase from upstream. Right now, the button only calls a pop alert window and prints the number. But, one should implement the function to send the request to backend to handle the further process

downstream,upstream players must be replaced with respective names once the backend works.

History: View2()

This view shows all the information for the last 10 week data. It uses another functional component DisplayTable taking data as week

<DisplayTable week="1" /> should render week 1 details

Plot: View3()

Not implemented yet but it should contain all the graphing related options.

More Info: View4()

not implemented yet but it should contain the status of the game if the week is over and the status of other players if enabled by the instructor .

That is all we have implemented for now. Please add more as you take on this project and make this the best one of all.