CIS-285

GROUP 5

TO DO LIST PROJECT

# Description

We decided to create a web application that acts as a to-do-list. The function of a to-do-list is very foundational to other types of essential business models, such as a shopping cart or schedule calendar. The project was coded in javascript using the React framework. Most of our team had not used javascript or react before so it was a pleasant learning experience.

# Features

In order to make the to-do-list function as intended, different features had to be implemented. These features are as follows:

* Add task
* Remove task
* Mark task as complete

These features are necessary to the functionality of our project.

* Add Task

The user inputs a string into the textbox on the home screen and it adds the string as a task to the list.

* Remove Task

Next to the task inside of the list there is a red ‘x’. If the user clicks the red ‘x’ the task will be removed from the list and removed from the display on the screen.

* Mark task as complete

Also next to the task inside of the list there is a checkbox. When the user clicks the check box the task is struck through in text to imply that the task has been complete. For example, if the task says “do laundry”, when the box is checked it will say “~~do laundry~~”.

# Source Code

## Todo.js

import { Checkbox, IconButton, ListItem, Typography } from "@material-ui/core";

import CloseIcon from "@material-ui/icons/Close";

import React from "react";

function Todo({ todo, toggleComplete, removeTodo }) {

function handleCheckboxClick() {

toggleComplete(todo.id);

}

function handleRemoveClick() {

removeTodo(todo.id);

}

return (

<ListItem style={{ display: "flex", color: "#FFCB05" }}>

<Checkbox checked={todo.completed} onClick={handleCheckboxClick} />

<Typography

variant="body1"

style={{

textDecoration: todo.completed ? "line-through" : null,

}}

>

{todo.task}

</Typography>

<IconButton onClick={handleRemoveClick} color="secondary">

<CloseIcon />

</IconButton>

</ListItem>

);

}

export default Todo;

## TodoForm.js

import { Button, TextField } from "@material-ui/core";

import React, { useState } from "react";

import uuid from "uuid";

import { makeStyles } from "@material-ui/core/styles";

const useStyles = makeStyles((theme) => ({

customHoverFocus: {

"&:hover, &.Mui-focusVisible": { backgroundColor: "#FFCB05" },

},

}));

function TodoForm({ addTodo }) {

const [todo, setTodo] = useState({

id: "",

task: "",

completed: false,

});

function handleTaskInputChange(e) {

setTodo({ ...todo, task: e.target.value });

}

function handleSubmit(e) {

e.preventDefault();

if (todo.task.trim()) {

addTodo({ ...todo, id: uuid.v4() });

setTodo({ ...todo, task: "" });

}

}

const classes = useStyles();

return (

<form className="todo-form" onSubmit={handleSubmit}>

<TextField

label="Task"

type="text"

name="task"

value={todo.task}

onChange={handleTaskInputChange}

variant="filled"

InputLabelProps={{

style: { color: "#FFCB05" },

}}

inputProps={{ style: { color: "#FFCB05" } }}

/>

<Button

className={classes.customHoverFocus}

variant="contained"

type="submit"

>

Submit

</Button>

</form>

);

}

export default TodoForm;

## TodoList.js

import { List } from "@material-ui/core";

import React from "react";

import Todo from "./Todo";

function TodoList({ todos, removeTodo, toggleComplete }) {

return (

<List>

{todos.map((todo) => (

<Todo

key={todo.id}

todo={todo}

removeTodo={removeTodo}

toggleComplete={toggleComplete}

/>

))}

</List>

);

}

export default TodoList;

# Analysis

We used a tool called Sonarcloud to analyze and review our code. Sonarcloud is useful for providing a clean layout and finding bugs and duplications to improve our code in every aspect. Sonarcloud helped us find hone in on whatever bugs we had during development so that they could be quickly dealt with.

# Test Cases

Test case #1 deals with ensuring the to-do list can add items. Here we have added three items, one for going to work, one grocery shopping, and one for making dinner.

Graphical user interface, website

Description automatically generated

Test case #2 deals with ensuring the to-do list can delete items properly. Here we delete one of the three items, which was previously for going to work.

Graphical user interface, website

Description automatically generated

Test case #3 deals with ensuring the to-do list can check off items that were completed. Here we check off two of the three items, and a strikethrough goes through them once they are checked.

Graphical user interface, text, website

Description automatically generated

# Conclusion

We achieved what we set out to do in this project by creating a functional to-do-list. Though it is a simple looking application it holds uses for many other types of programs. For example, if we were given more time, we could have developed this same application into a working schedule/calendar. We would have to add functionality for each day of the week and possibly have a reminder system and all sorts of new user/UI functionality. Or we possibly could have created an online shopping cart with this code as well. A shopping cart and a to-do-list are not too different fundamentally. The add/remove functions could simply remain the same. We would need to add functionality for the items being added and the price totaling, possibly even add a database for all the items. These are just a few ideas that we think are realistic for us to create using this to-do-list as a base.