**Freddy candy shop**

**Overview**

A frontend web application to manage Freddy candy orders

**Folder Structure**

public

* + - assests
      * css
      * js
      * Images
    - other
    - index.html
    - login.html
    - order.html

**Pages/ Features**

**User Login**

|  |  |
| --- | --- |
| Summary | User is required to provide user name and password to access the application |
| Basic flow: | 1. User enters a valid user name and password  2. An invalid user credential will display and error message  3. A valid request returns an access token and a refresh token which is set in the session storage in the browser  4. user is then redirected to the dashboard (index.html) |

**Dashboard**

|  |  |
| --- | --- |
| Summary | Provide a statistical analysis of orders |
| Basic flow: | 1. Displays sales records for today, last and last month  2. Displays a toggle view of weekly revenues and yearly revenues  3. The bar graph arranges data stating from current and previous day or month for both weekly revenue and yearly revenue respectively  4. Display records of bestsellers in a table |

**Orders**

|  |  |
| --- | --- |
| Summary | Searching for orders |
| Basic flow: | 1. Enter the name of order in the input field and click of the search icon  2. Shows all orders related to the search input and display orders in a table |

**Custom Functions**

|  |  |  |  |
| --- | --- | --- | --- |
| Name | params | Purpose | Code |
| getRefreshToken | none | This called when  request status return 401.  It gets a new access token and the request is retried without breaking the application | /\*\*     \* get regresh token     \* @returns string     \*/    const getRefreshToken = async () => {      const request = await fetch("https://freddy.codesubmit.io/refresh", {        method: "POST",        headers: {          Authorization: `Bearer ${refresh\_token}`,          "Content-Type": "application/json",        },      });      if (request.status === 200) {        const response = await request.json();        sessionStorage.setItem("access\_token", response?.access\_token);        return response?.access\_token;      } else {        logout();      }    }; |
| reOrderDaysList | none | This fuction returns an ordered days of the week starting from current day (today) and previous day (yesterday) | /\*\*   \* Reoder week days to start with current day and previous day   \* @returns arry   \*/  const reOrderDaysList = () => {    let list, currentObj, previousObj, current;    current = today;    list = ["Sun", "Mon", "Tue", "Wed", "Thur", "Fri", "Sat"];    let copyList = list.slice();    if (current === 0) {      previousObj = list.pop();      currentObj = list.shift();    } else {      currentObj = list[current - 1];      previousObj = list[current - 2];      const removed = list.splice(current - 1, 2);    }    return ["today", "yesterday", ...list];  }; |
|  |  |  |  |

|  |  |  |  |
| --- | --- | --- | --- |
| Name | params | Purpose | Code |
| renderChart | Lables: string [],  Data:  Number[] | Draws the revenue bar chart | /\*\*     \* Draw bar chart     \* @param {string[]} labels     \* @param {number[]} data     \*/    const renderChart = (labels, data) => {      chart = new Chart(canvas, {        type: "bar",        data: {          labels: labels,          datasets: [            {              // label: '# of Votes',              data: data,              backgroundColor: [                "rgba(255, 99, 132, 0.2)",                …              borderColor: [                "rgba(255, 99, 132, 1)",…              ],              borderWidth: 1,            },          ],        },        options: {          scales: {            y: {              beginAtZero: true,            },          },          plugins: {              legend: {                  display: false,                position: "top",              },              title: {                display: false,                text: "Whom'st let the dogs out",              },          }        },      });    }; |
| converToArray | Object {} | Convert sales data into a array | /\*\*   \* Converting object of array to array of objects   \* @param {object} data   \*/  const convertToArray =(data)=>{      return Object.keys(data).map(          (key) => data[key]        );  } |
| reOrderMonthList | none | This function returns an ordered months of the year starting from current month (November) and previous month (October) | /\*\*   \* Reoder months of the year to start with current month and previous month   \* @returns arry   \*/  const reOrderMonthList = () => {    let list, currentObj, previousObj, current;    current = currentMonth;    list = ["Jan", "Feb","Mar","Apr","May","Jun", "Jul","Aug","Sep","Oct","Nov","Dec",    ];    let copyList = list.slice();    if (current === 0) {      previousObj = copyList.pop();      currentObj = copyList.shift();    } else {      currentObj = copyList[current];      previousObj = copyList[current - 1];      const removed = copyList.splice(current - 1, 2);    }    return ["this month", "last month", ...copyList];  }; |
| reOrderValues | Object{} | The function return an array starting from current day/month and previous day/month | /\*\*   \* return an array of values   \* @param {object}   \* @returns array   \*/  const reOrderValues = (data) => {    let coppy = convertToArray(data);    const removed = coppy.splice(today - 2, 2).reverse();    return [...removed, ...coppy].map((x) => x?.total);  }; |