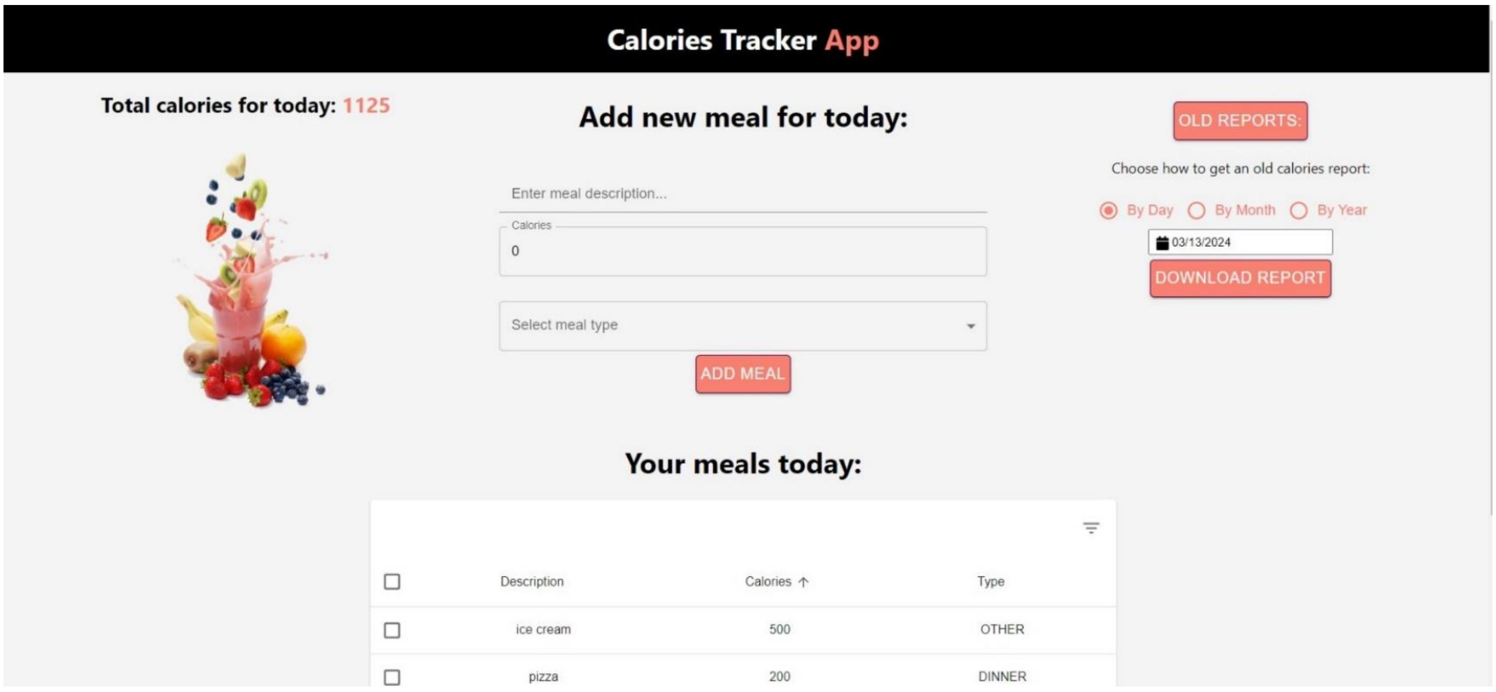
Calorie Management Client Application

Final Project in Front-End Development

Submitted by:

Name	ID	Email	Mobile Number
Daniella Boaz	209371913	Daniellaboaz22@gmail.com	0549000716
Gal Kalev	318657632	galk2508@gmail.com	0526262143



	Description	Calories ↑	Type
☐	ice cream	500	OTHER
☐	pizza	200	DINNER

Web Page:

https://daniellaboaz.github.io/Calorie_Management-_App/

Video Link:

https://youtu.be/k3c6b1xiDbU

GitHub Link:

https://github.com/daniellaboaz/Calorie_Management-_App.git

table of contents:

project structure:

| - CaloriesApp

| - - Component folder

| - - - - - CalendarShow.js

| - - - - - ControlsCounter.js

| - - - - - ControlsInput.js

| - - - - - DataService.js

| - - - - - ErrorClass.js

| - - - - - MUI_EnhancedTable.js

| - - - - - Navbar.js

| - - - - - OldMealsReport.js

| - - - - - ShowMealsOfToday.js

| - - App.js

| - - idb.js

| - - App.css

Back to table of contents:

Src folder:

App.js:

```
// Daniella Boaz (209371913), Gal Kalev (318657632)
//src/App.js
// External libraries
import React, { useState, useEffect } from 'react';
import idb from './idb';
import 'react-calendar/dist/Calendar.css';
import Button from '@mui/material/Button';
//import components:
import CalendarShow from './components/CalendarShow';
import OldMealsReport from './components/OldMealsReport';
import Navbar from './components/Navbar';
import ShowMealsOfToday from './components/ShowMealsOfToday';
import fetchAttributesForDate from './components/DataService';
import ErrorHandling from './components/ErrorClass';
//display total calories of today:
import ControlsCounter from './components/ControlsCounter';
// Controls for adding new meals:
import ControlsInput from './components/ControlsInput';
import './App.css';

// in this the main page where we fetch the data from idb
const App = () => {
  const [meals, setMeals] = useState([]);
  const [calories, setMealCalories] = useState(0);
  const [mealType, setMealType] = useState(null);
  const [mealDescripton, setMealDescripton] = useState('');
  const [totalCalories, setTotalCalories] = useState(0);
  const [showMealsReport, setShowMealsReport] = useState(false);
  const [selectedAttributes, setSelectedAttributes] = useState(null);


  // Fetch meals of today from the database and display them
  const fetchMeals = async () => {
    try {
      console.log('Fetching meals for today...');
      await idb.openCaloriesDB('caloriesdb', 1);
      const today = new Date();
      today.setHours(0, 0, 0, 0);
      const mealsToday = await fetchAttributesForDate(today, setSelectedAttributes, 'today');
      console.log('Meals fetched for today:', mealsToday);

      if (mealsToday && mealsToday.length > 0) {
        const sumCalories = mealsToday.reduce((total, meal) =>
          total + (parseInt(meal.calorie, 10) || 0), 0);
        setMeals(mealsToday);
        setTotalCalories(sumCalories);
      } else {
        console.log('No meals recorded for today.');
```

Back to table of contents:

```
        setMeals([]);
        setTotalCalories(0);
      }
    } catch (error) {
      console.error('Error fetching meals for today:', error);
      throw new ErrorHandling('fetch');     }

  };

  //upload the data after refreshing the page
  useEffect(() => {
    fetchMeals(); // Fetch meals when the component mounts
  }, []);

  //add a new meal today and display
  const onAddMealsClick = async () => {
    if (calories <= 0 || mealType === null || mealDescripton === '') {
      alert('Input must not be empty or is illegal');
      return;
    } try {
      await idb.openCaloriesDB('caloriesdb', 1);
      const today = new Date();
      today.setHours(0, 0, 0, 0);
      const formattedDate = today.toISOString().split('T')[0];

      await idb.addCalories({
        calorie: calories,
        category: mealType,
        description: mealDescripton,
        date: formattedDate,
      });
      // Call the fetchMeals directly after adding a new meal to display it
      await fetchMeals();
    } catch (error) {
      console.error(error);
       throw ErrorHandling('addMeal');
    }
    // setting the controls to default
    setMealCalories(0);
    setMealDescripton('');
    setMealType('');
  };

  //delete a meal from the list and the database
  const deleteMeal = async (mealId) => {
    try {
      await idb.openCaloriesDB('caloriesdb', 1);
      await idb.deleteCalorie(mealId);
      setShowMealsReport(false);
      // Fetch meals again after deletion
      await fetchMeals();
    } catch (error) {
```

Back to table of contents:

```
    console.error(error);
    throw ErrorHandling('delete');
  }
};

//Show/hide calendar on button click
const clickToShowCalendar = () => {
  const calendarContainer = document.getElementsByClassName('calendar-container')[0];
  if (calendarContainer) {
    if (calendarContainer.style.display === 'none') {
      calendarContainer.style.display = 'block';
      setShowMealsReport(false); // Hide meals report when showing calendar
    } else {
      calendarContainer.style.display = 'none';
    };
  };
};

return (
  <div className='App_home'>
    <Navbar />
    <table className='app_table'>{/*divide the page to 3 equal sections*/}
      <tbody>
        <tr>
          <td className='app_table_cell'>{/*display total calories of today*/}
            <ControlsCounter totalCalories={totalCalories} />
          </td>
          <td className='app_table_cell'>{/* Controls for adding new meals*/}
            <ControlsInput
              calories={calories}
              mealDescripton={mealDescripton}
              mealType={mealType}
              setMealCalories={setMealCalories}
              setMealDescripton={setMealDescripton}
              setMealType={setMealType}
              onAddMealsClick={onAddMealsClick}
            />
          </td>
          <td className='app_table_cell'>{/* calander for old meald report*/}
            <div className='app_constrols_report'>
              <Button className='btn' onClick={clickToShowCalendar} variant='contained'
                color='primary' sx={{ marginTop: '10px', marginBottom: '10px' }}>
                Old reports:
              </Button>
              <CalendarShow
                setShowMealsReport={setShowMealsReport}
                setSelectedAttributes={setSelectedAttributes}
              />
            </div>
          </td>
        </tr>
      </tbody>
      </table>
      {/* a list of today meals*/}
      <ShowMealsOfToday meals={meals} deleteMeal={deleteMeal} />
      {/* a list of previos meals from selected date*/}
      {showMealsReport ? (
        <OldMealsReport
          selectedAttributes={selectedAttributes}
          deleteMeal={deleteMeal}
        />
      ) : null}
    </div>
  );
};
export default App;
```

Back to table of contents:

idb.js:

```
// Daniella Boaz (209371913), Gal Kalev (318657632)
// src/idb.js
const idb = {
  db: null,

  // Open or create the IndexedDB database for the calories database.
  openCaloriesDB: async (dbName, version) => {
    return new Promise((resolve, reject) => {
      const request = indexedDB.open(dbName, version);
      request.onerror = (event) => {
        reject(`Error opening database: ${event.target.error}`);
      };
      request.onupgradeneeded = (event) => {
        const db = event.target.result;
        // Create an object store if it doesn't already exist
        if (!db.objectStoreNames.contains('caloriesdb')) {
          db.createObjectStore('caloriesdb', { keyPath: 'id', autoIncrement: true });
        }
      };
      request.onsuccess = (event) => {
        idb.db = event.target.result;
        resolve(idb);
      };
    });
  },

  // Add calories data to the IndexedDB.
  addCalories: async (caloriesData) => {
    return new Promise((resolve, reject) => {
      if (!idb.db) {
        reject('Database is not initialized');
        return;
      }
      const transaction = idb.db.transaction(['caloriesdb'], 'readwrite');
      const store = transaction.objectStore('caloriesdb');
      // Add the current date to the caloriesData object
      const currentDate = new Date().toISOString();
      caloriesData.date = currentDate;
      const request = store.add(caloriesData);
      request.onsuccess = (event) => {
        resolve(event.target.result);
      };
      request.onerror = (event) => {
        reject(`Error adding calories: ${event.target.error}`);
      };
    });
  },
```

Back to table of contents:

```
  // Retrieve all calories data from data base.
  async getAllCalories() {
    return new Promise((resolve, reject) => {
      if (!idb.db) {
        reject('Database is not initialized');
        return;
      }
      const transaction = idb.db.transaction(['caloriesdb'], 'readonly');
      const store = transaction.objectStore('caloriesdb');
      const request = store.getAll();
      request.onsuccess = (event) => {
        resolve(event.target.result);
      };
      request.onerror = (event) => {
        reject(`Error getting calories: ${event.target.error}`);
      };
    });
  },

  // Delete a specific calorie entry from the data base.
  async deleteCalorie(calorieId) {
    return new Promise((resolve, reject) => {
      if (!idb.db) {
        reject('Database is not initialized');
        return;
      }
      const transaction = idb.db.transaction(['caloriesdb'], 'readwrite');
      const store = transaction.objectStore('caloriesdb');
      const request = store.delete(calorieId);
      request.onsuccess = (event) => {
        resolve(event.target.result);
      };
      request.onerror = (event) => {
        reject(`Error deleting calorie: ${event.target.error}`);
      };
    });
  }
};

export default idb;
```

Back to table of contents:

Src/Components folder:

CalendarShow.js:

```
// Daniella Boaz (209371913), Gal Kalev (318657632)
// src/components/CalendarShow.js
// calendar to get a report of specific date
import React, { useState } from 'react';
import '../App.css';
import fetchAttributesForDate from './DataService';
import ErrorHandling from './ErrorClass';
//Import from MUI
import Button from '@mui/material/Button';
import Radio from '@mui/material/Radio';
import RadioGroup from '@mui/material/RadioGroup';
import FormControlLabel from '@mui/material/FormControlLabel';
import FormControl from '@mui/material/FormControl';
import DatePicker from 'react-datepicker';
import 'react-datepicker/dist/react-datepicker.css';

// React Date Picker calendar formatted as 'DD/MM/YYYY'
const ByDayCalendar = ({ selectedDate, setSelectedDate }) => {
    return (
        <DatePicker
            className='cal-style'
            showIcon
            maxDate={new Date()}
            selected={selectedDate}
            onChange={(date) => setSelectedDate(date)}
        />
    );
};

// React Date Picker calendar formatted as 'MM/YYYY'
const ByMonthCalendar = ({ selectedDate, setSelectedDate }) => {
    return (
        <DatePicker
            className='cal-style'
            showIcon
            maxDate={new Date()}
            selected={selectedDate}
            onChange={(date) => setSelectedDate(date)}
            dateFormat='MM/yyyy'
            excludeDates={[
                1661990400000, 1664582400000, 1667260800000, 1672531200000,
            ]}
            showMonthYearPicker
        />
    );
};
```

Back to table of contents:

```
// React Date Picker calendar formatted as 'YYYY'
const ByYearCalendar = ({ selectedDate, setSelectedDate }) => {
    return (
        <DatePicker
            className='cal-style'
            showIcon
            maxDate={new Date()}
            selected={selectedDate}
            onChange={(date) => setSelectedDate(date)}
            showYearPicker
            dateFormat='yyyy'
        />
    );
};
const CalendarShow = ({ setShowMealsReport, setSelectedAttributes }) => {
    const [selectedDate, setSelectedDate] = useState(new Date());
    // Set selectedaView as the chosen calendar
    const [selectedView, setSelectedView] = useState('');

    const selectDate = async () => {
        try {
            // Fetch meals for the selected date based on the selectedView
            await fetchAttributesForDate(selectedDate, setSelectedAttributes, selectedView);
            setShowMealsReport(true);
            // Hide the calendar after selecting a date
            document.getElementsByClassName('calendar-container')[0].style.display = 'none';
        } catch (error) {
            console.error('Error fetching attributes:', error);
           throw new ErrorHandling('fetch');

        }
    };
    return (
        <div className='calendar-container' style={{ display: 'none' }}>
            <p>Choose how to get an old calories report:</p>
            <FormControl component='fieldset'>
                {/* Radio buttons to choose the preferred calendar*/}
                <RadioGroup
                    row
                    aria-labelledby='demo-row-radio-buttons-group-label'
                    name='row-radio-buttons-group'
                    value={selectedView}
                    onChange={(event) => setSelectedView(event.target.value)}
                >
                    <FormControlLabel
                        value='byDay'
                        control={<Radio name='cal' />}
                        label='By Day'
                    />
```

Back to table of contents:

```
                    <FormControlLabel
                        value='byMonth'
                        control={<Radio name='cal' />}
                        label='By Month'
                    />
                    <FormControlLabel
                        value='byYear'
                        control={<Radio name='cal' />}
                        label='By Year'
                    />
                </RadioGroup>
            </FormControl>
            {/* Show the calendar by the user preference */}
            <div className='cal-container'
            style={{ display: selectedView === 'byDay' ? 'block' : 'none' }}>
                <ByDayCalendar selectedDate={selectedDate} setSelectedDate={setSelectedDate} />
            </div>

            <div className='cal-container'
            style={{ display: selectedView === 'byMonth' ? 'block' : 'none' }}>
              <ByMonthCalendar selectedDate={selectedDate} setSelectedDate={setSelectedDate} />
            </div>

            <div className='cal-container'
            style={{ display: selectedView === 'byYear' ? 'block' : 'none' }}>
               <ByYearCalendar selectedDate={selectedDate} setSelectedDate={setSelectedDate} />
            </div>
            <Button className='btn'
            onClick={selectDate}
            variant='contained'
            sx={{ marginTop: '20px' }}>
                Download report
            </Button>
        </div>
    );
};

export default CalendarShow;
```

Back to table of contents:

ControlsCounter.js:

```
// Daniella Boaz (209371913), Gal Kalev (318657632)
// src/components/ControlsCounter.js
//display total calories of today
import React from 'react';
import '../App.css';

const ControlsCounter = ({ totalCalories }) => {
  return (
    <div className='app_controls_counter'>
      {/*show total calories of today*/}
      <h2>Total calories for today: <span>{totalCalories}</span></h2>
      {/*show image to design the page*/}
      <img src={`${process.env.PUBLIC_URL}/images/pic.png`}
        alt='pic'
        style={{ width: '40%', height: 'auto' }}
      />
    </div>
  );
};

export default ControlsCounter;
```

Back to table of contents:

ControlsInput.js:

```
// Daniella Boaz (209371913), Gal Kalev (318657632)
// src/components/ControlsInput.js
// Controls for adding new meals
import React from 'react';
//Material-UI imports and styling:
import TextField from '@mui/material/TextField';
import Button from '@mui/material/Button';
import { FormControl, InputLabel, Select, MenuItem, Input } from '@mui/material';
import '../App.css';

const ControlsInput = ({
  calories,
  mealDescripton,
  mealType,
  setMealCalories,
  setMealDescripton,
  setMealType,
  onAddMealsClick,
}) => {
  return (
    <div className='app_constrols_Input_muster'>
      <div className='app_constrols_Input'>
        <h1>Add new meal for today:</h1>
        {/*adding meal description */}
        <FormControl fullWidth sx={{ marginTop: '20px' }} >
          <InputLabel htmlFor='meal-des'>Enter meal description...</InputLabel>
          <Input sx={{ textAlign: 'center' }}
            type='text'
            placeholder='Enter meal description...'
            value={mealDescripton}
            onChange={(e) => setMealDescripton(e.target.value)}></Input>
        </FormControl>
        {/*adding meal calories */}
        <TextField type='number' label='Calories' inputProps={{ min: 0 }}
          value={calories} onChange={(e) => setMealCalories(e.target.value)}
          fullWidth margin='normal' />
        {/*adding meal type */}
        <FormControl fullWidth sx={{ marginTop: '20px', marginLeft: 0 }}>
          <InputLabel id='meal-type-placeholder' htmlFor='meal-name'>
            Select meal type
          </InputLabel>
          <Select
            labelId='meal-type-placeholder'
            value={mealType || ''}
            onChange={(e) => setMealType(e.target.value)}
            label='Select meal type'
            fullWidth
          >
            <MenuItem value='BREAKFAST'>BREAKFAST</MenuItem>
            <MenuItem value='LUNCH'>LUNCH</MenuItem>
```

Back to table of contents:

```
            <MenuItem value='DINNER'>DINNER</MenuItem>
            <MenuItem value='OTHER'>OTHER</MenuItem>
          </Select>
        </FormControl>
      </div>
      {/*adding meal to the database and display it in the list*/}
      <Button
        className='btn'
        onClick={onAddMealsClick}
        variant='contained'
        color='primary'
        sx={{ marginTop: '20px' }}>
        Add Meal
      </Button>
    </div>
  );
};

export default ControlsInput;
```

Back to table of contents:

DataService.js:

```
// Daniella Boaz (209371913), Gal Kalev (318657632)
// src/components/DataService.js
import idb from '../idb';
import ErrorHandling from './ErrorClass';

// Fetching the meals based on the date
const fetchAttributesForDate = async (selectedDate, setSelectedAttributes, selectedView) => {
    try {
        const selectedYear = selectedDate.getFullYear();
        const selectedMonth = selectedDate.getMonth() + 1;
        const selectedDay = selectedDate.getDate();
        const attributes = await idb.getAllCalories();

        let filteredAttributes;

        if (selectedView) { // Filter the meals based on the calendar view
            switch (selectedView) {
                case 'today':
                case 'byDay':
                    filteredAttributes = attributes.filter((item) => {
                        const itemDate = new Date(item.date);
                        const itemYear = itemDate.getFullYear();
                        const itemMonth = itemDate.getMonth() + 1;
                        const itemDay = itemDate.getDate();
                        return selectedYear === itemYear &&
                            selectedMonth === itemMonth &&
                            selectedDay === itemDay;

                    });
                    break;
                case 'byMonth':
                    filteredAttributes = attributes.filter((item) => {
                        const itemDate = new Date(item.date);
                        const itemYear = itemDate.getFullYear();
                        const itemMonth = itemDate.getMonth() + 1;
                        return selectedYear === itemYear &&
                            selectedMonth === itemMonth;
                    });
                    break;
                case 'byYear':
                    filteredAttributes = attributes.filter((item) => {
                        const itemDate = new Date(item.date);
                        const itemYear = itemDate.getFullYear();
                        return selectedYear === itemYear;
                    });
                    break;
                default:
                    break;
            }
```

Back to table of contents:

```
            // Checking if there are meals from that date
            if ((!filteredAttributes || filteredAttributes.length === 0) &&
                selectedView !== 'today') {
                alert('No calories from that time');
                setSelectedAttributes(null);
                return [];
            } else {
                setSelectedAttributes(filteredAttributes);
                return filteredAttributes;
            }
        }

    } catch (error) {
        console.error('Error fetching attributes for date:', error);
        throw new ErrorHandling('fetch');

    }
};

export default fetchAttributesForDate;
```

Back to table of contents:

Navbar.js:

```
// Daniella Boaz (209371913), Gal Kalev (318657632)
// src/components/Navbar.js
// display settings for the page
import React from 'react';
import '../App.css';

// The header bar at the top of the page
const Navbar = () => {
    return (
        <div className='app_navbar'>
          <h1>Calories Tracker <span>App</span></h1>
        </div>
    );
  };

export default Navbar;
```

Back to table of contents:

OldMealsReport.js:

```
// Daniella Boaz (209371913), Gal Kalev (318657632)
// src/components/OldMealsReport.js
// show the report from the day selected
import React, { useState } from 'react';
import Button from '@mui/material/Button';
import '../App.css';
import EnhancedTable from './MUI_EnhancedTable';

const OldMealsReport = ({ selectedAttributes, deleteMeal }) => {
    // Show only if there is a report
    const [isReportVisible, setReportVisibility] = useState(true);

    if (selectedAttributes && selectedAttributes.length > 0 && isReportVisible) {
        // Get the first date of the selected report
        const firstSelectedDate = new Date(selectedAttributes[0].date);
        // Get the last date of the selected report
        const lastSelectedDate =
            new Date(selectedAttributes[selectedAttributes.length - 1].date);

        const firstDay = firstSelectedDate.getDate().toString().padStart(2, '0');
        const lastDay = lastSelectedDate.getDate().toString().padStart(2, '0');

        const firstMonth = (firstSelectedDate.getMonth() + 1).toString().padStart(2, '0');
        const lastMonth = (lastSelectedDate.getMonth() + 1).toString().padStart(2, '0');

        const firstYear = firstSelectedDate.getFullYear();
        const lastYear = lastSelectedDate.getFullYear();

        let formattedDate;

        // Setting 'formattedDate' based of the range of the selected date
        if (firstDay === lastDay && firstMonth === lastMonth && firstYear === lastYear) {
            formattedDate = `${firstDay}/${firstMonth}/${firstYear}`;// By day
        }
        else if (firstDay !== lastDay && firstMonth === lastMonth && firstYear === lastYear) {
            formattedDate = `${firstDay}-${lastDay}/${firstMonth}/${firstYear}`;// By month
        }
        else if (firstDay !== lastDay && firstMonth !== lastMonth && firstYear === lastYear) {
            formattedDate = `${firstDay}/${firstMonth}-${lastDay}/${lastMonth}/${firstYear}`;
            // By year
        }

        return (
            <div>
                <div className='app_meals_container_wrapper'>
                    <h1>Meals Report from: {formattedDate}</h1>
                </div>
                <div className='app_meals_container_wrapper' >
                    {/* Show a list of old reports in table */}
                    <EnhancedTable rows={selectedAttributes} deleteMeal={deleteMeal} />
```

Back to table of contents:

```
                    <Button
                        className='btn'
                        variant='contained'
                        style={{ width: '20%' }}
                        onClick={() => setReportVisibility(false)}>
                        Hide report
                    </Button>{/* Button to hide the report from the page*/}
                </div>
            </div>
        );
    }
    return null;
};

export default OldMealsReport;
```

Back to table of contents:

ShowMealsOfToday.js:

```
// Daniella Boaz (209371913), Gal Kalev (318657632)
// components/ShowMealsOfToday.js
//show a list of meals added today

import React from 'react';
import '../App.css';
import EnhancedTable from './MUI_EnhancedTable';

const ShowMealsOfToday = ({ meals, deleteMeal }) => {
  return (
    <div>
      <div className='app_meals_container_wrapper'>
        <h1>Your meals today:</h1>
      </div>
      {/* display the meals */}
      <EnhancedTable rows = {meals} deleteMeal={deleteMeal}/>
    </div>
  );
};

export default ShowMealsOfToday;
```

Back to table of contents:

ErrorClass js:

```
// Daniella Boaz (209371913), Gal Kalev (318657632)
//src/components/ErrorClass.js

//Class for errors when an error is thrown
class ErrorHandling extends Error {
    constructor(action) {
        let message;
        switch (action) {
            case 'fetch':
                message = 'Error fetching data';
                break;
            case 'delete':
                message = 'Error deleting data';
                break;
            case 'addMeal':
                message = 'Error adding meal to data';
                break;
            default:
                message = 'An error occurred';
        }
        super(message);
        this.name = this.constructor.name;
    }
}

export default ErrorHandling;
```

Back to table of contents:

MUI_EnhancedTable.js:

```
// Daniella Boaz (209371913), Gal Kalev (318657632)
// src/components/MUI_EnhancedTable.js
// The table that displaying the meals description, type and calories for today and for old
report
import React from 'react';
// Material-UI imports and styling:
import PropTypes from 'prop-types';
import { alpha } from '@mui/material/styles';
import Box from '@mui/material/Box';
import Table from '@mui/material/Table';
import TableBody from '@mui/material/TableBody';
import TableCell from '@mui/material/TableCell';
import TableContainer from '@mui/material/TableContainer';
import TableHead from '@mui/material/TableHead';
import TablePagination from '@mui/material/TablePagination';
import TableRow from '@mui/material/TableRow';
import Toolbar from '@mui/material/Toolbar';
import Typography from '@mui/material/Typography';
import Paper from '@mui/material/Paper';
import Checkbox from '@mui/material/Checkbox';
import IconButton from '@mui/material/IconButton';
import Tooltip from '@mui/material/Tooltip';
import DeleteIcon from '@mui/icons-material/Delete';

// Function to get set the order when sorting the columns
function descendingComparator(a, b, orderBy) {
  if (b[orderBy] < a[orderBy]) {
    return -1;
  }
  if (b[orderBy] > a[orderBy]) {
    return 1;
  }
  return 0;
}

// Function that gets the comparator based on the order and orderBy
function getComparator(order, orderBy) {
  return order === 'desc'
    ? (a, b) => descendingComparator(a, b, orderBy)
    : (a, b) => -descendingComparator(a, b, orderBy);
}

// unction to stabilize sorting order
function stableSort(array, comparator) {
  const stabilizedThis = array.map((el, index) => [el, index]);
  stabilizedThis.sort((a, b) => {
    const order = comparator(a[0], b[0]);
    if (order !== 0) {
      return order;
    }
    return a[1] - b[1];
  });
  return stabilizedThis.map((el) => el[0]);
}

// Define the columns and by the meals attributes (description, type and calories)
const headCells = [
  {
    id: 'des',
    numeric: true,
    disablePadding: false,
    label: 'Description',
  },
  {
    id: 'calories',
    numeric: true,
    disablePadding: false,
    label: 'Calories',
  },
  {
    id: 'type',
    numeric: true,
    disablePadding: false,
    label: 'Type',
  },
]

// Component for the table header
function EnhancedTableHead(props) {
  const { onSelectAllClick, order, orderBy, numSelected, rowCount } =
    props;

return (
  <TableHead>
    <TableRow>
      <TableCell padding='checkbox'>
        <Checkbox
          color='primary'
          indeterminate={numSelected > 0 && numSelected < rowCount}
          checked={rowCount > 0 && numSelected === rowCount}
          onChange={onSelectAllClick}
          inputProps={{
            'aria-label': 'select all desserts',
          }}
        />
      </TableCell>
      {headCells.map((headCell) => (
        <TableCell
          key={headCell.id}
          align='center'
          sortDirection={orderBy === headCell.id ? order : false}
        >
```

Back to table of contents:

```
          {headCell.label}
        </TableCell>
      ))}
    </TableRow>
  </TableHead>
);
}

// Prop types for EnhancedTableHead component
EnhancedTableHead.propTypes = {
  numSelected: PropTypes.number.isRequired,
  onRequestSort: PropTypes.func.isRequired,
  onSelectAllClick: PropTypes.func.isRequired,
  order: PropTypes.oneOf(['asc', 'desc']).isRequired,
  orderBy: PropTypes.string.isRequired,
  rowCount: PropTypes.number.isRequired,
};

// Component for the toolbar above the table
function EnhancedTableToolbar(props) {
  const { numSelected, deleteMeal, selected, setSelected } = props;

    // Function to delete selected meals
  const deleteMealFromTable = () => {
    selected.forEach((item) => {
      deleteMeal(item);
    });
    setSelected([]);
  }


return (
  <Toolbar
    sx={{
      pl: { sm: 2 },
      pr: { xs: 1, sm: 1 },
      ...(numSelected > 0 && {
        bgcolor: (theme) =>
          alpha(theme.palette.primary.main, theme.palette.action.activatedOpacity),
      }),
    }}
  >
    {numSelected > 0 ? (
      <Typography
        sx={{ flex: '1 1 100%' }}
        color='inherit'
        variant='subtitle1'
        component='div'
      >
        {numSelected} selected
      </Typography>
    ) : (
      <Typography
        sx={{ flex: '1 1 100%' }}
        variant='h6'
        id='tableTitle'
        component='div'
      >
      </Typography>
    )}

    {numSelected > 0 && (
      <Tooltip title='Delete'>
        <IconButton onClick={deleteMealFromTable}>
          <DeleteIcon />
        </IconButton>
      </Tooltip>
    )}
  </Toolbar>
);
}


// Prop types for EnhancedTableToolbar component
EnhancedTableToolbar.propTypes = {
  numSelected: PropTypes.number.isRequired,
};
//   ! ----------------------------------------!
// Main EnhancedTable component
function EnhancedTable(props) {
  const [order, setOrder] = React.useState('asc');
  const [orderBy, setOrderBy] = React.useState('calories');
  const [selected, setSelected] = React.useState([]);
  const [page, setPage] = React.useState(0);
  const [rowsPerPage, setRowsPerPage] = React.useState(5);

  let { rows } = props;

  // Reverse the order of the rows so it will show last meal at the top of the table
  const rowsR = React.useMemo(() => {
    return [...rows].reverse();
  }, [rows]);

  // Event handler for sorting
  const handleRequestSort = (event, property) => {
    const isAsc = orderBy === property && order === 'asc';
    setOrder(isAsc ? 'desc' : 'asc');
    setOrderBy(property);
  };

  // Event handler for selecting all rows
  const handleSelectAllClick = (event) => {
    if (event.target.checked) {
      const newSelected = rowsR.map((n) => n.id);
```

Back to table of contents:

```
      setSelected(newSelected);
      return;
    }
    setSelected([]);
  };

  // Event handler for individual row selection
  const handleClick = (event, id) => {
    const selectedIndex = selected.indexOf(id);
    let newSelected = [];

    if (selectedIndex === -1) {
      newSelected = newSelected.concat(selected, id);
    } else if (selectedIndex === 0) {
      newSelected = newSelected.concat(selected.slice(1));
    } else if (selectedIndex === selected.length - 1) {
      newSelected = newSelected.concat(selected.slice(0, -1));
    } else if (selectedIndex > 0) {
      newSelected = newSelected.concat(
        selected.slice(0, selectedIndex),
        selected.slice(selectedIndex + 1),
      );
    }
    setSelected(newSelected);
  };

  // Event handler for changing page
  const handleChangePage = (event, newPage) => {
    setPage(newPage);
  };

   // Event handler for changing rows per page
  const handleChangeRowsPerPage = (event) => {
    setRowsPerPage(parseInt(event.target.value, 10));
    setPage(0);
  };

  // Function to determine if a row is selected
  const isSelected = (id) => {
    const index = selected.indexOf(id);

    return index !== -1;
  };

// Calculate the number of empty rows
  const emptyRows =
    page > 0 ? Math.max(0, (1 + page) * rowsPerPage - rows.length) : 0;

    // Visible rows based on sorting, page, and rows per page
  const visibleRows = React.useMemo(
    () =>
      stableSort(rowsR, getComparator(order, orderBy)).slice(
```

Back to table of contents:

```
      page * rowsPerPage,
      page * rowsPerPage + rowsPerPage
    ),
  [order, orderBy, page, rowsPerPage, rowsR] // Include rows in the dependency array
);

// Return the JSX for the EnhancedTable component
return (
  <Box sx={{ width: '100%', justifyContent: 'center', display: 'flex', }}>
    <Paper sx={{ width: '50%', mb: 2, }}>
      <EnhancedTableToolbar
      numSelected={selected.length}
      selected={selected}
      setSelected={setSelected}
      deleteMeal={props.deleteMeal}
      />
      <TableContainer>
        <Table
          sx={{ minWidth: 750 }}
          aria-labelledby='tableTitle'

        >
          <EnhancedTableHead
            numSelected={selected.length}
            order={order}
            orderBy={orderBy}
            onSelectAllClick={handleSelectAllClick}
            onRequestSort={handleRequestSort}
            rowCount={rows.length}
          />
          <TableBody>
            {visibleRows.map((row, index) => {
              const isItemSelected = isSelected(row.id);
              const labelId = `enhanced-table-checkbox-${index}`;

              return (
                <TableRow
                  hover
                  onClick={(event) => handleClick(event, row.id)}
                  role='checkbox'
                  aria-checked={isItemSelected}
                  tabIndex={-1}
                  key={row.id}
                  selected={isItemSelected}
                  sx={{ cursor: 'pointer' }}
                >
                  <TableCell padding='checkbox'>
                    <Checkbox
                      color='primary'
                      checked={isItemSelected}
                      inputProps={{
                        'aria-labelledby': labelId,
```

Back to table of contents:

```
                        }}
                      />
                    </TableCell>
                      <TableCell
                      padding='normal'
                      component='th'
                      id={labelId}
                      scope='row'
                      align='center'

                    >{row.description} {/* Set each meal in a row */}
                    </TableCell>
                    <TableCell align='center'>{row.calorie}</TableCell>
                    <TableCell align='center'>{row.category}</TableCell>
                  </TableRow>
                );
              }))}
              {emptyRows > 0 && (
                <TableRow>
                  <TableCell colSpan={6} />
                </TableRow>
              )}
            </TableBody>
          </Table>
        </TableContainer>
        <TablePagination
          rowsPerPageOptions={[5, 10, 25]}
          component='div'
          count={rows.length}
          rowsPerPage={rowsPerPage}
          page={page}
          onPageChange={handleChangePage}
          onRowsPerPageChange={handleChangeRowsPerPage}
        />
      </Paper>
    </Box>
  );
}

export default EnhancedTable;
```

Back to table of contents:

App.css:

```css
/* Daniella Boaz (209371913), Gal Kalev (318657632)
src/App.css
styling guide*/
.App_home {
  position: relative;
  height: 100vh;
  width: 100%;
  background: #f4f4f4;
  overflow-x: hidden;
}

/* styling at the top of the page */
.app_navbar {
  display: flex;
  align-items: center;
  justify-content: center;
  text-align: center;
  color: #fff;
  background: black;
  width: 100%;
  height: 10vh;

  position: relative;
  z-index: 1;
}

/* make importent content in red */
span {
  color: salmon;
}

/*where to display the calendar*/
.app_constrols_report {
  display: flex;
  flex-direction: column;
  align-items: center;
  justify-content: center;
  text-align: center;
  margin-top: 5%;
}

/* Controls for adding new meals*/
.app_constrols_Input {
  top: 30px;
  display: flex;
  justify-content: space-around;
  align-items: center;
  flex-direction: column;
  text-align: left;
}
```

Back to table of contents:

```css
.app_constrols_Input_btn {
  justify-content: center;
  flex-direction: row;
}

.app_constrols_Input_muster {
  justify-content: space-around;

  padding: 5px;
}

/* show list of meals buttons design*/
.scrollableContainer button {
  text-align: center;
  color: white;
  background: red;
  font-size: 18px;
  border-radius: 5px;
  border: 1px solid black;
}

/* radio buttons forthe meal type */
.radio_label {
  margin-right: 40px;
  display: inline;
  clear: none;
  padding: 8px;
  flex-direction: row;
}

/* the styling to display meals */
.app_meals_container {
  display: flex;
  flex-direction: column;
  align-items: center;
}

.app_meals_container_wrapper {
  text-align: center;
  margin-bottom: 20px;

}

.app_meals_container_warapper_inner {
  width: 50%;
  display: flex;
  align-items: center;
  align-items: stretch;
  justify-content: space-between;
  border: 1px solid black;
```

Back to table of contents:

```
  margin: 10px auto;
  padding: 5px;
  font-size: 18px;
  font-weight: 800;
  border-radius: 5px;
  overflow: hidden;
  flex-wrap: wrap;
  flex-direction: column;
}


/* styling for Material-UI Card and Button components */
.app_meals_container_warapper_inner>.MuiCard-root {
  width: 100%;
  margin: 10px auto;
  margin-bottom: 10px;
}

.app_meals_container_warapper_inner>.MuiCard-root>.MuiCardContent-root {
  display: flex;
  flex-direction: column;
  align-items: flex-start;
}

.app_meals_container_warapper_inner>.MuiCard-root>.MuiCardContent-root>div {
  margin-bottom: 10px;
}

.app_meals_container_warapper_inner>.MuiCard-root>.MuiCardContent-root>div:last-child {
  margin-bottom: 0;
}

.app_meals_container_warapper_inner>.MuiCard-root>.MuiButtonBase-root {
  margin-top: 10px;
}

/* the same style to all the buttons*/
.app_table_cell .btn {
  padding: 5px;
  outline: none;
  text-align: center;
  font-size: 18px;
  color: white;
  background: salmon;
  border-radius: 5px;
  border: 1px solid rgb(79, 2, 74);
  margin-bottom: 5px;
  margin-top: 5px;
}

.calendar-container {
  display: flex;
```

Back to table of contents:

```
  flex-direction: column;
  align-items: center;
}

.calendar-container .btn {
  padding: 5px;
  outline: none;
  text-align: center;
  font-size: 18px;
  color: white;
  background: salmon;
  border-radius: 5px;
  border: 1px solid rgb(79, 2, 74);
  margin-bottom: 5px;
  margin-top: 5px;
}

.app_meals_container_wrapper .btn {
  padding: 5px;
  outline: none;
  text-align: center;
  font-size: 18px;
  color: white;
  background: salmon;
  border-radius: 5px;
  border: 1px solid rgb(79, 2, 74);
  margin-bottom: 5px;
  margin-top: 5px;
}

/* show list of meals */
.scrollableContainer {
  max-height: 180px;
  overflow-y: auto;
  margin: 0 auto;
  height: auto;
  width: 50%;
  padding: 16px;
  align-items: center;
  align-content: center;
  justify-content: center;
  flex-direction: column;
}

/* design the page equaly to 3 */
.app_table {
  border-collapse: collapse;
  width: 100%;
  table-layout: fixed;

}
```

Back to table of contents:

```css
.app_table_cell {
  padding: 0;
  vertical-align: top;
  text-align: center;
}

/* the counter of total calories for today */
.app_controls_counter {
  display: inline-block;
  min-height: max-content;
}
```