x

Belize Cattle Tracker

Technical Documentation
Version 1

Group 2 Members:
Joanne Y.
Chimezirim A.
Rene S.
Miguel A.
Juan S.
Cameron T.

Submitted To:
CMPS4131
Mr. Manuel Medina
April 6, 2022

Table of Contents

Overview

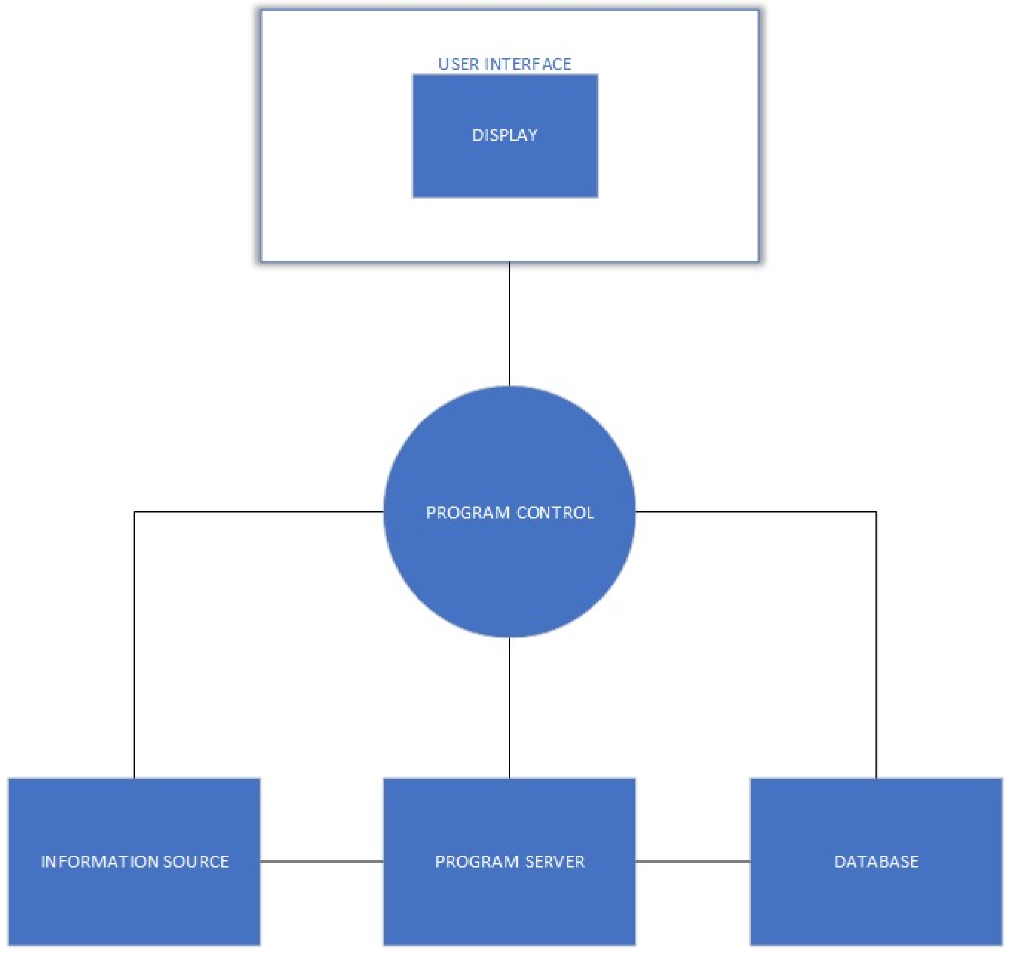
This document should provide an overview of the program implementation of the system, especially aspects of the code that were used to complete the unit testing of key functions. This document also covers the types of software and components of the system (database layout, interface connects, etc.)

Objective

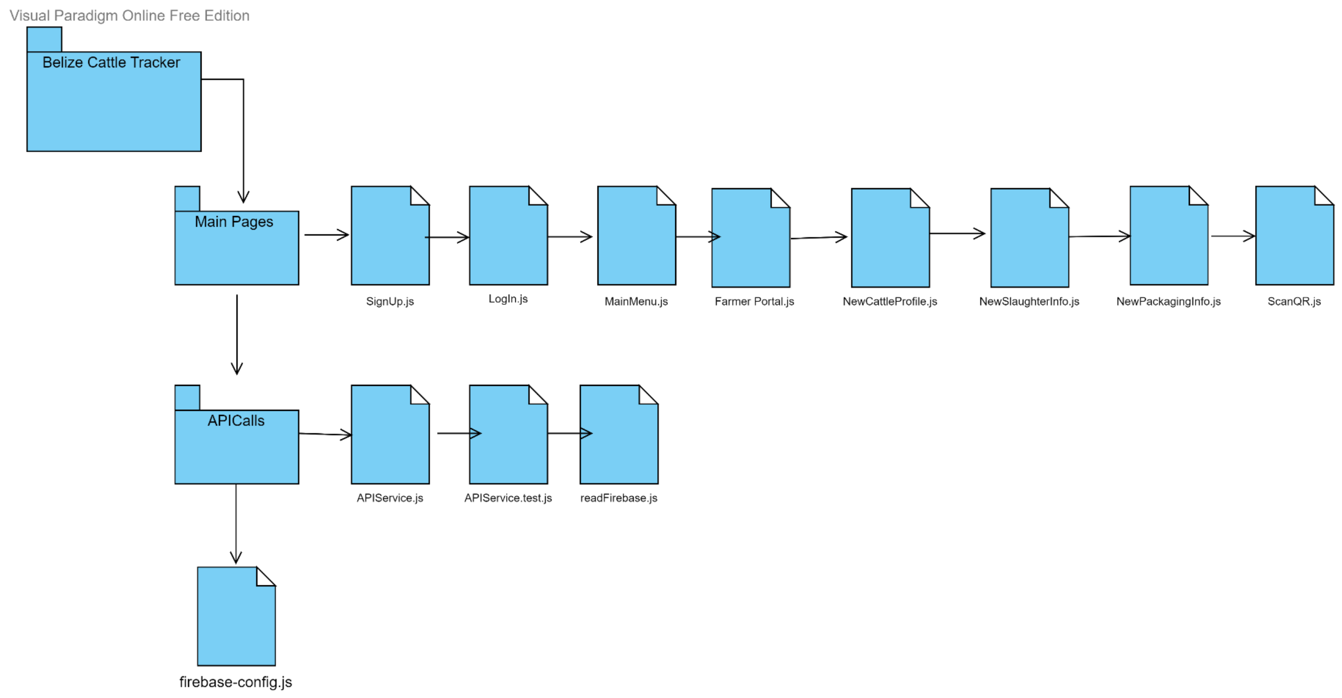
The main objective for this documentation is to deliver developers and testers with an in-depth understanding of the structure and reasons for coding a function using certain data structures. It assists a developer with enough knowledge about the system to further expand and maintain the system (for future uses).

Program Structure

General System Structure



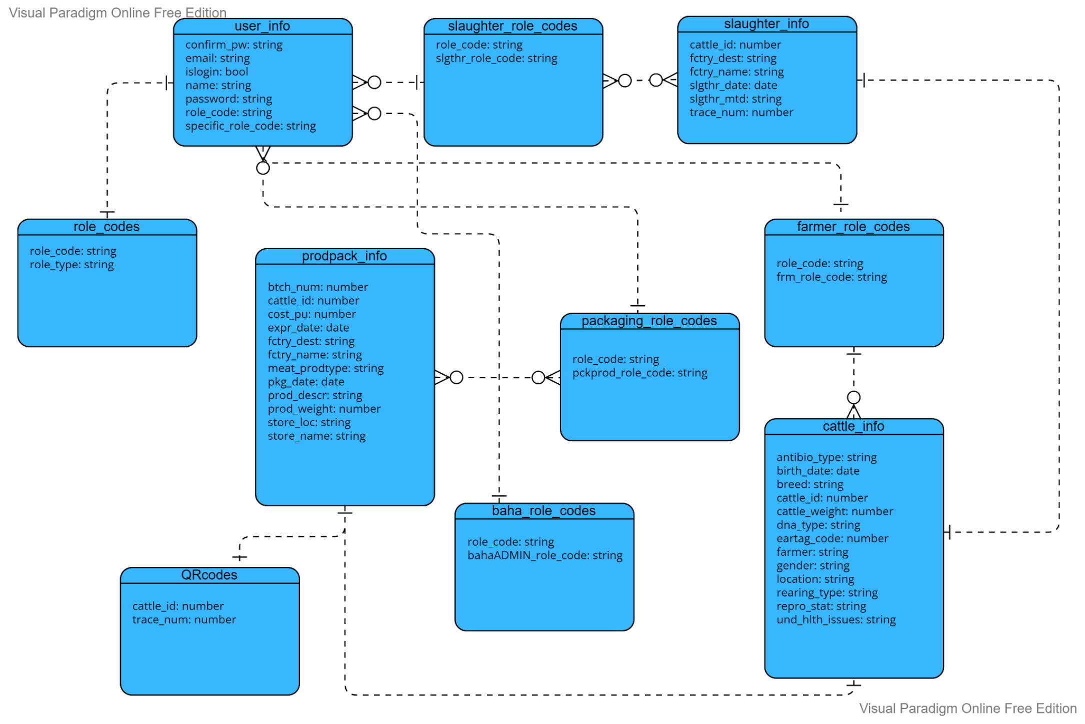
Program File Structure



Database Structure

Database Design Schema

Below is a clear representation of what the **database schema** works/looks like with the assumption that the Firestore Database is in terms of a relational SQL based database.



Technologies Used

Front-end Software

What Front-End?: ReactJS

Reason for ReactJs: React is a framework for creating user interfaces that does exactly what it says on the tin, offering developers complete freedom and flexibility. Developers utilize JavaScript and JSX to construct applications. React's strongest features are its components and declarative coding. The frontend of your project is the layer that is immediately available to users and presented on user screens. The React JavaScript toolkit for frontend development allows you to create high-quality user interfaces for web apps. This library works with Virtual DOM and allows you to embed HTML code in JavaScript (Brainhub, 2021).

Back-end Software

What Backend? JavaScript or Node.JS

Reason for Node.Js: Node JS is widely recognized as the best technology for hosting and operating web servers for React apps. Node employs a node package manager, or NPM, to install all new updates and packages, which is one of the main reasons for this. To make the compilation process easier, Node bundles single React apps. This is accomplished by utilizing proper Node modules and a web pack. Both React and Node JS are used in today's development ecosystems for various reasons. As a result, it's critical to examine their relevant use case as well as their development methodology. React JS is typically used to create front-end user interface components, whereas Node JS handles backend data (Programmers.io, 2021).

Database Software
What DB Platform: Firebase (Firestore Database)

Type of DB: Non Relational, Nosql

Reason for Firebase: The newest database for mobile app development is Firestore. Cloud Firestore is more powerful than a real-time database, with more capabilities, quicker queries, and more scalability. Cloud Firestore is a NoSQL cloud database that is both versatile and scalable. For client and server-side programming, it is utilized to store and sync data. Google Cloud Platform and Firebase are used for mobile, web, and server development (JavaTPoint, n.d.).

IDE
What IDE: Visual Studio Code

Why VS Code: Visual Studio Code is a lightweight code editor that includes features for debugging, task execution, and version management. Its goal is to give only the tools a developer requires for a speedy code-build-debug cycle, leaving more sophisticated processes to IDEs with greater features (Visual Studio Code, n.d.).

System Testing

Use Cases Tested

UC1- Create Cattle Profile- Allows a farmer/BAHA to create new cattle profiles of registered cattles.

UC5- Add Product Information- Allows Packaging Managers to add on packaging information to previously added cattle slaughter and birth information.

UC7- Generate Reports- Allows authorized personeels from BAHA to generate reports on different cattles based on different filters such as cattleID, DOB, etc.

UC-8 Add Slaughter Information- Allows Slaughterhouse Managers to add on slaughter information to previously added cattle birth information.

Testing Framework

Testing framework used was: Jest

Why Jest: Jest is a testing framework for client-side JavaScript and, more especially, React applications. The official Jest website has further information about the platform. The test results are shown in a treeview, and you can quickly browse to the test source from there. The test status is displayed next to the test in the editor, along with the option to run or debug it fast (PhPStorm, 2022).

Commands Used for Testing

Integration testing: yarn test *or* npm test

Unit testing: yarn test -- -t 'name of the test case'

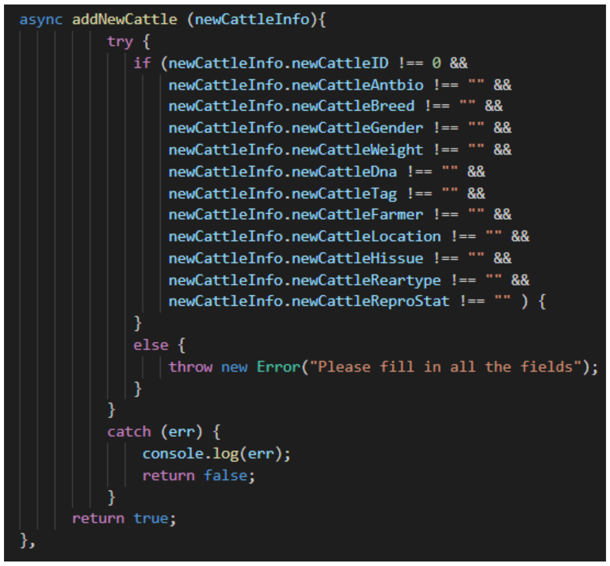
Example: If you would like to test UC1, you can use the command below

```
yarn test -- -t 'TC-1 Add cattle info successfully to the database with valid data'
```

Test Cases and Testing Components

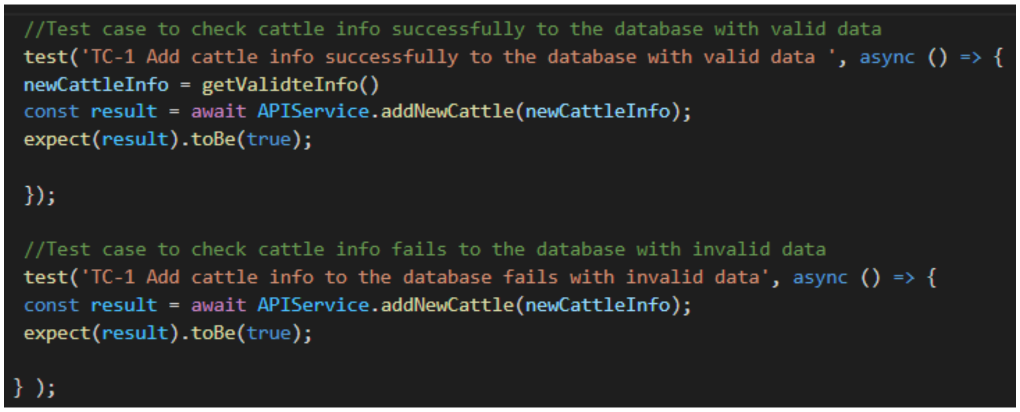
UC1- Create Cattle Profile

Component



```
async addNewCattle (newCattleInfo){
        try {
          if (newCattleInfo.newCattleID !== 0 &&
              newCattleInfo.newCattleAntbio !== "" &&
              newCattleInfo.newCattleBreed !== "" &&
              newCattleInfo.newCattleGender !== "" &&
              newCattleInfo.newCattleWeight !== "" &&
              newCattleInfo.newCattleDna !== "" &&
              newCattleInfo.newCattleTag !== "" &&
              newCattleInfo.newCattleFarmer !== "" &&
              newCattleInfo.newCattleLocation !== "" &&
              newCattleInfo.newCattleHissue !== "" &&
              newCattleInfo.newCattleReartype !== "" &&
              newCattleInfo.newCattleReproStat !== "" ) {
          }
          else {
              throw new Error("Please fill in all the fields");
          }
        }
        catch (err) {
            console.log(err);
            return false;
        }
      return true;
},
```

Test



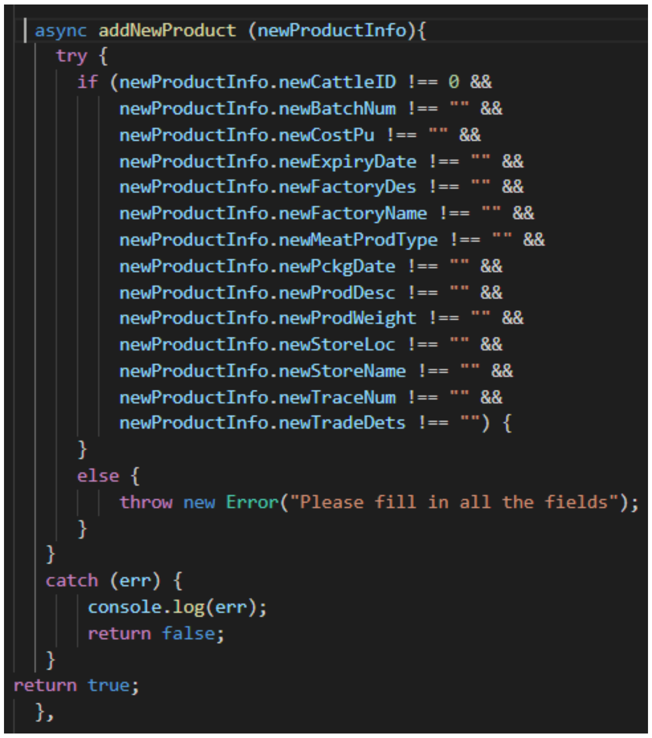
```
//Test case to check cattle info successfully to the database with valid data
test('TC-1 Add cattle info successfully to the database with valid data ', async () => {
newCattleInfo = getValidteInfo()
const result = await APIService.addNewCattle(newCattleInfo);
expect(result).toBe(true);

});

//Test case to check cattle info fails to the database with invalid data
test('TC-1 Add cattle info to the database fails with invalid data', async () => {
const result = await APIService.addNewCattle(newCattleInfo);
expect(result).toBe(true);

} );
```

AddNewCattle:

- The function takes one parameter, newCattleInfo object (type: any)
- A try statement is used inside the function to test if the code produces any errors.
- An if statement is used inside the try statement to validate if the variables of the parameter passed are not null
- Return false if the parameter does not pass the validation and log the error to the console or Return true if the parameter passes the validation

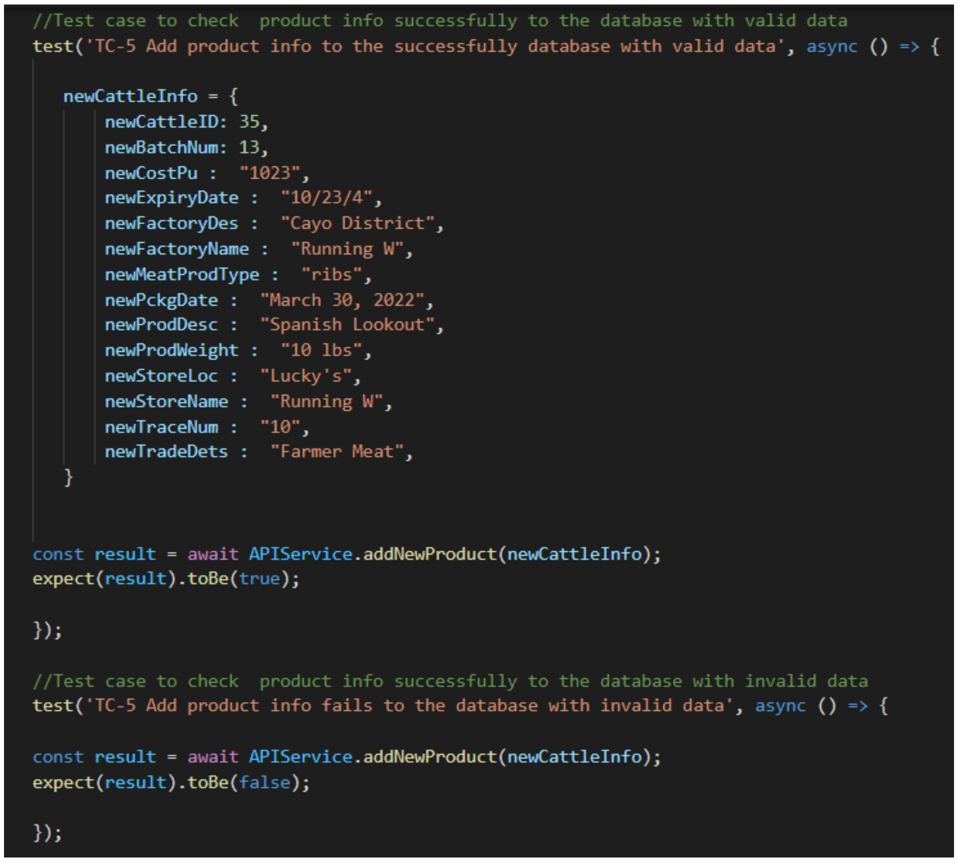
UC5- Add Product Information

Component



```
async addNewProduct (newProductInfo){
  try {
    if (newProductInfo.newCattleID !== 0 &&
        newProductInfo.newBatchNum !== "" &&
        newProductInfo.newCostPu !== "" &&
        newProductInfo.newExpiryDate !== "" &&
        newProductInfo.newFactoryDes !== "" &&
        newProductInfo.newFactoryName !== "" &&
        newProductInfo.newMeatProdType !== "" &&
        newProductInfo.newPckgDate !== "" &&
        newProductInfo.newProdDesc !== "" &&
        newProductInfo.newProdWeight !== "" &&
        newProductInfo.newStoreLoc !== "" &&
        newProductInfo.newStoreName !== "" &&
        newProductInfo.newTraceNum !== "" &&
        newProductInfo.newTradeDets !== "") {
    }
    else {
        throw new Error("Please fill in all the fields");
    }
  }
  catch (err) {
    console.log(err);
    return false;
  }
return true;
 },
```

Test



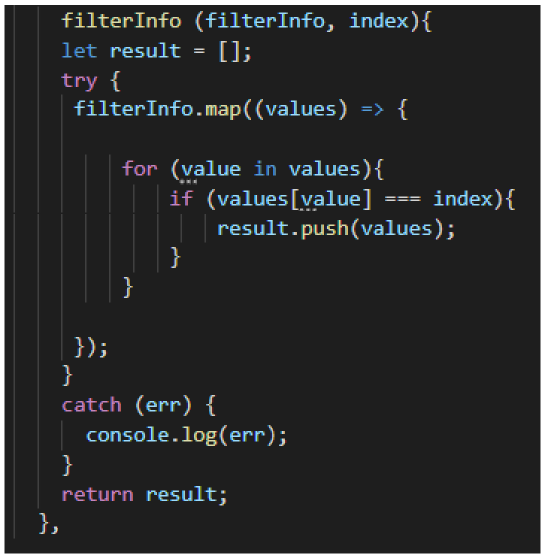
```
//Test case to check  product info successfully to the database with valid data
test('TC-5 Add product info to the successfully database with valid data', async () => {

   newCattleInfo = {
       newCattleID: 35,
       newBatchNum: 13,
       newCostPu :  "1023",
       newExpiryDate :  "10/23/4",
       newFactoryDes :  "Cayo District",
       newFactoryName :  "Running W",
       newMeatProdType :  "ribs",
       newPckgDate :  "March 30, 2022",
       newProdDesc :  "Spanish Lookout",
       newProdWeight :  "10 lbs",
       newStoreLoc :  "Lucky's",
       newStoreName :  "Running W",
       newTraceNum :  "10",
       newTradeDets :  "Farmer Meat",
   }


const result = await APIService.addNewProduct(newCattleInfo);
expect(result).toBe(true);

});

//Test case to check  product info successfully to the database with invalid data
test('TC-5 Add product info fails to the database with invalid data', async () => {

const result = await APIService.addNewProduct(newCattleInfo);
expect(result).toBe(false);

});
```

AddNewProduct:

- The function takes one parameter, newProductInfo object (type: any)
- A try statement is used inside the function to test if the code produces any errors.
- An if statement is used inside the try statement to validate if the variables of the parameter passed are not null
- Return false if the parameter does not pass the validation and log the error to the console or Return true if the parameter passes the validation

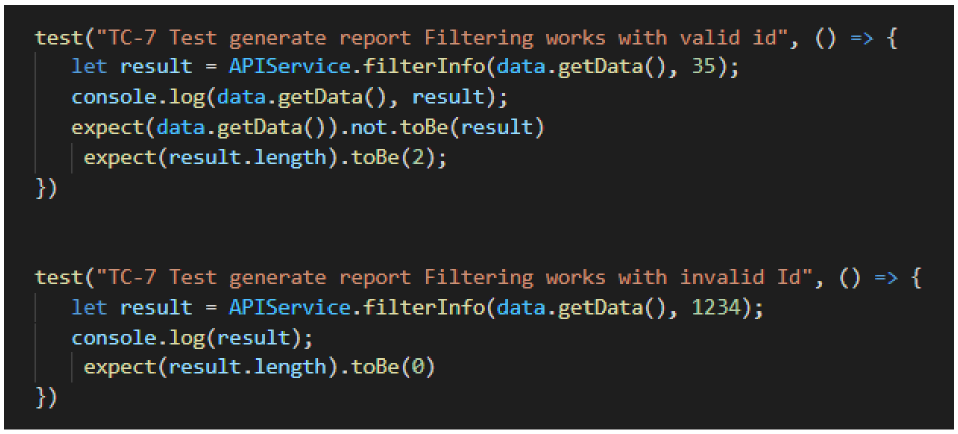
UC7- Generate Reports

Component



```
  filterInfo (filterInfo, index){
  let result = [];
  try {
   filterInfo.map((values) => {

     for (value in values){
       if (values[value] === index){
         result.push(values);
       }
     }

   });
  }
  catch (err) {
    console.log(err);
  }
  return result;
},
```

Test



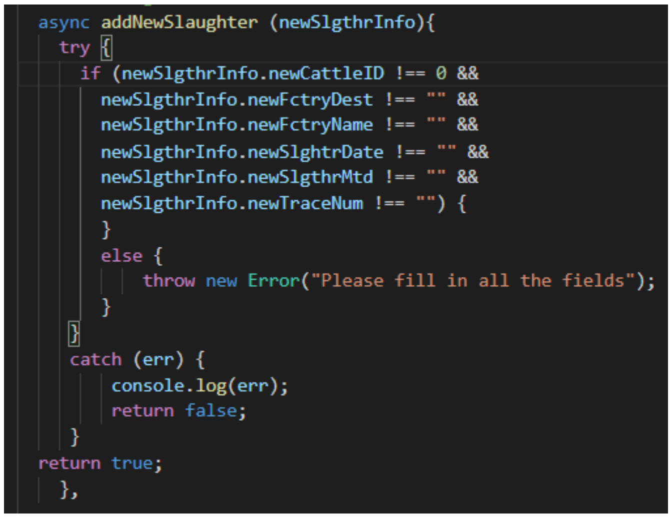
```
test("TC-7 Test generate report Filtering works with valid id", () => {
   let result = APIService.filterInfo(data.getData(), 35);
   console.log(data.getData(), result);
   expect(data.getData()).not.toBe(result)
    expect(result.length).toBe(2);
})


test("TC-7 Test generate report Filtering works with invalid Id", () => {
   let result = APIService.filterInfo(data.getData(), 1234);
   console.log(result);
    expect(result.length).toBe(0)
})
```

FilterInfo:

- The function takes two parameters, a newProductInfo object and index (type: any)
- A result array is initialized inside the function
- A try statement is used inside the function to test if the code produces any errors.
- The values from filterInfo are iterated over until a match with the index parameter is found and if the matching value is found, the values are pushed into the result array
- Log the error to the console if the value is not found or Return result if the matching value is found

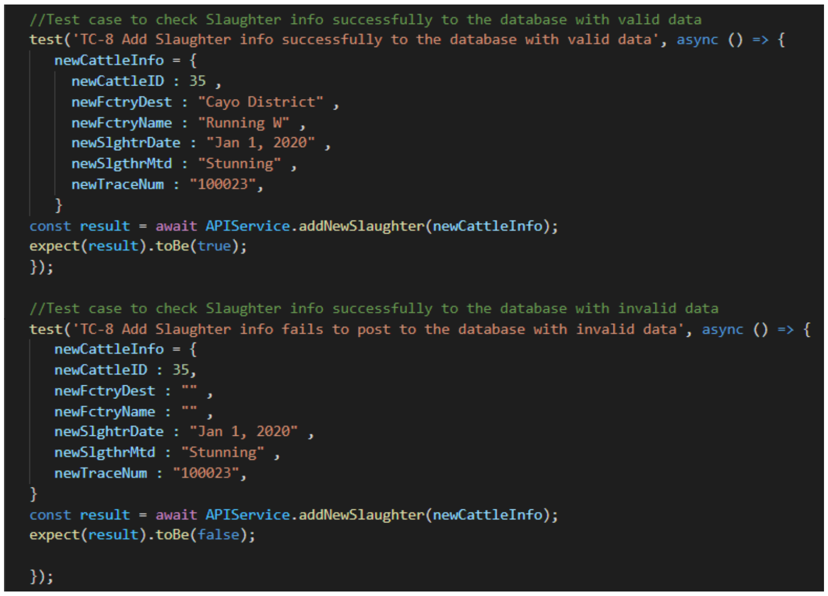
UC-8 Add Slaughter Information

Component



```
async addNewSlaughter (newSlgthrInfo){
  try {
    if (newSlgthrInfo.newCattleID !== 0 &&
      newSlgthrInfo.newFctryDest !== "" &&
      newSlgthrInfo.newFctryName !== "" &&
      newSlgthrInfo.newSlghtrDate !== "" &&
      newSlgthrInfo.newSlgthrMtd !== "" &&
      newSlgthrInfo.newTraceNum !== "") {
      }
      else {
          throw new Error("Please fill in all the fields");
      }
  }
   catch (err) {
      console.log(err);
      return false;
   }
return true;
  },
```

Test



```
//Test case to check Slaughter info successfully to the database with valid data
test('TC-8 Add Slaughter info successfully to the database with valid data', async () => {
   newCattleInfo = {
     newCattleID : 35 ,
     newFctryDest : "Cayo District" ,
     newFctryName : "Running W" ,
     newSlghtrDate : "Jan 1, 2020" ,
     newSlgthrMtd : "Stunning" ,
     newTraceNum : "100023",
   }
const result = await APIService.addNewSlaughter(newCattleInfo);
expect(result).toBe(true);
});

//Test case to check Slaughter info successfully to the database with invalid data
test('TC-8 Add Slaughter info fails to post to the database with invalid data', async () => {
   newCattleInfo = {
   newCattleID : 35,
   newFctryDest : "" ,
   newFctryName : "" ,
   newSlghtrDate : "Jan 1, 2020" ,
   newSlgthrMtd : "Stunning" ,
   newTraceNum : "100023",
}
const result = await APIService.addNewSlaughter(newCattleInfo);
expect(result).toBe(false);

});
```

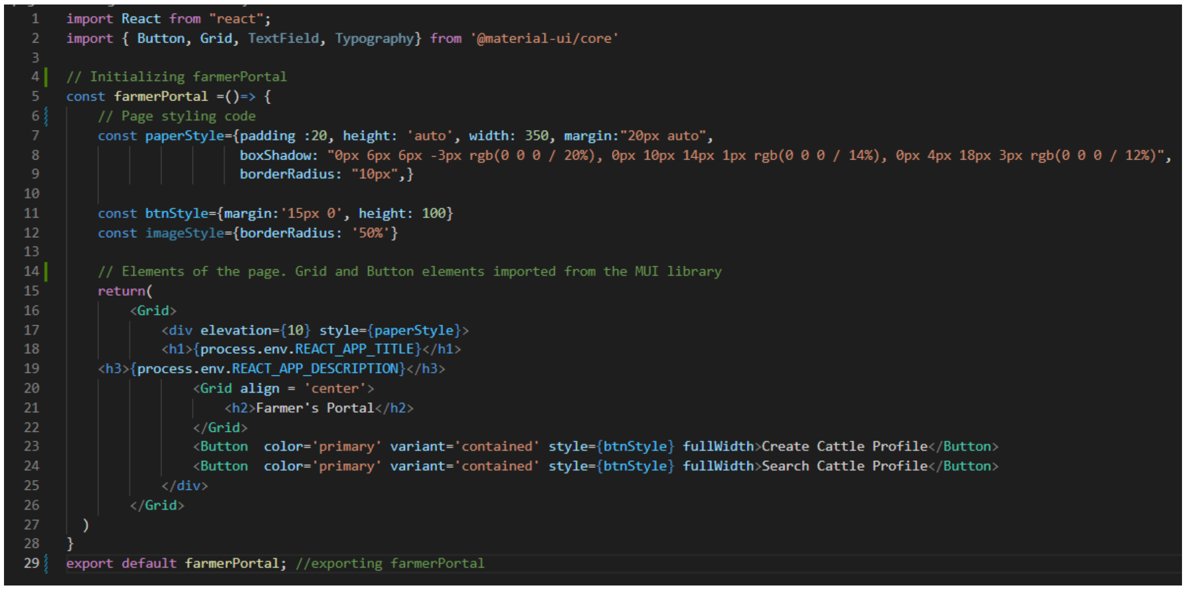
AddNewSlaughter:

- The function takes one parameter, newSlgthrInfo object (type: any)
- A try statement is used inside the function to test if the code produces any errors.
- An if statement is used inside the try statement to validate if the variables of the parameter passed are not null
- Return false if the parameter does not pass the validation and log the error to the console or Return true if the parameter passes the validation

Test Cases and Interface Components

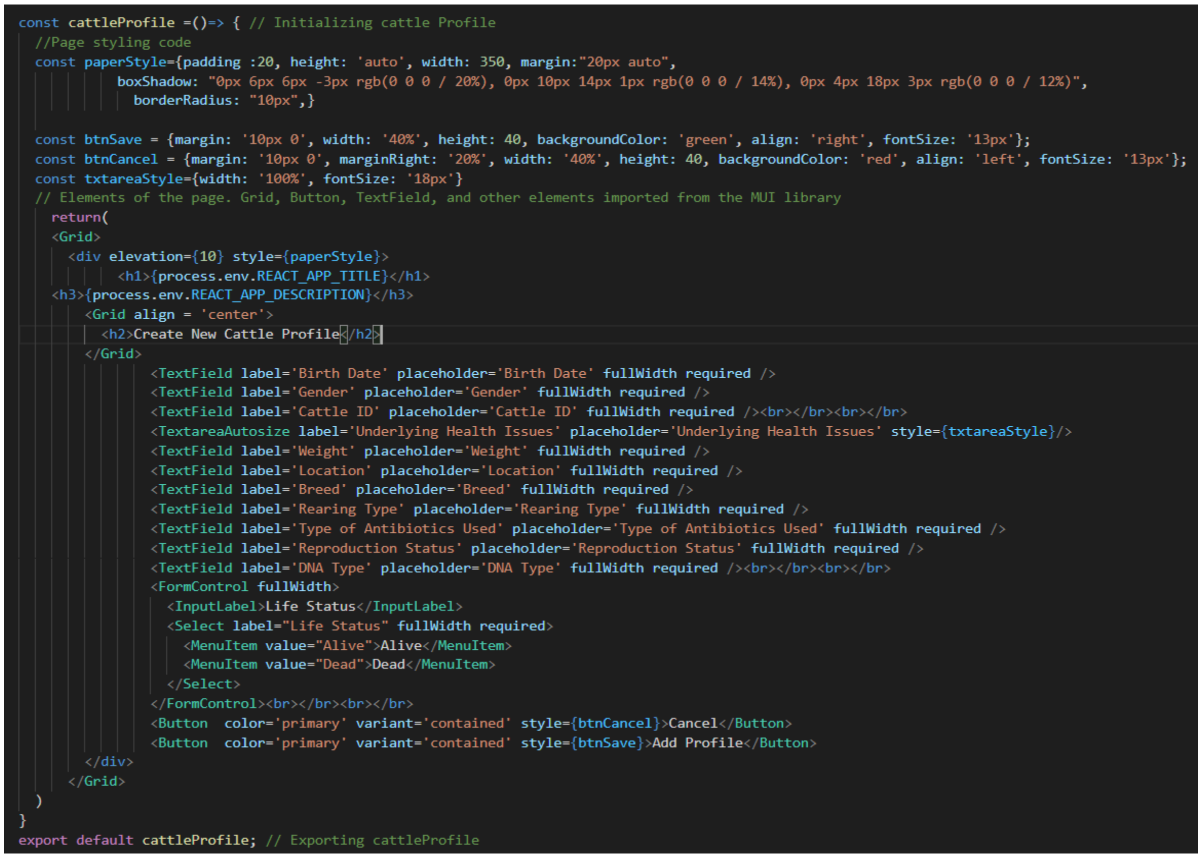
UC1- Create Cattle Profile

Farmer Portal Code



```
import React from "react";
import { Button, Grid, TextField, Typography} from '@material-ui/core'

// Initializing farmerPortal
const farmerPortal =()=> {
    // Page styling code
    const paperStyle={padding :20, height: 'auto', width: 350, margin:"20px auto",
                      boxShadow: "0px 6px 6px -3px rgb(0 0 0 / 20%), 0px 10px 14px 1px rgb(0 0 0 / 14%), 0px 4px 18px 3px rgb(0 0 0 / 12%)",
                      borderRadius: "10px",}

    const btnStyle={margin:'15px 0', height: 100}
    const imageStyle={borderRadius: '50%'}

    // Elements of the page. Grid and Button elements imported from the MUI library
    return(
        <Grid>
            <div elevation={10} style={paperStyle}>
            <h1>{process.env.REACT_APP_TITLE}</h1>
    <h3>{process.env.REACT_APP_DESCRIPTION}</h3>
                <Grid align = 'center'>
                    <h2>Farmer's Portal</h2>
                </Grid>
                <Button  color='primary' variant='contained' style={btnStyle} fullWidth>Create Cattle Profile</Button>
                <Button  color='primary' variant='contained' style={btnStyle} fullWidth>Search Cattle Profile</Button>
            </div>
        </Grid>
  )
}
export default farmerPortal; //exporting farmerPortal
```

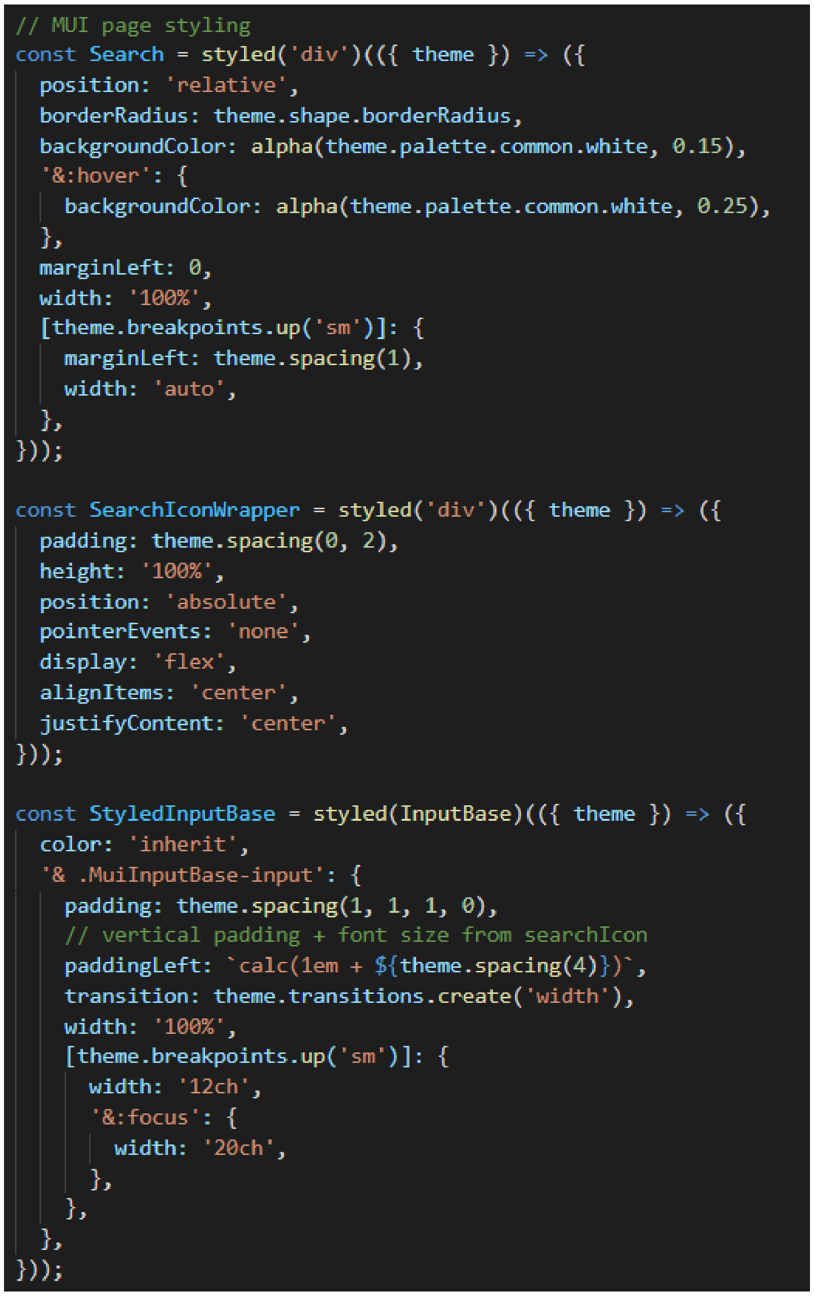
Create New Cattle Profile Code



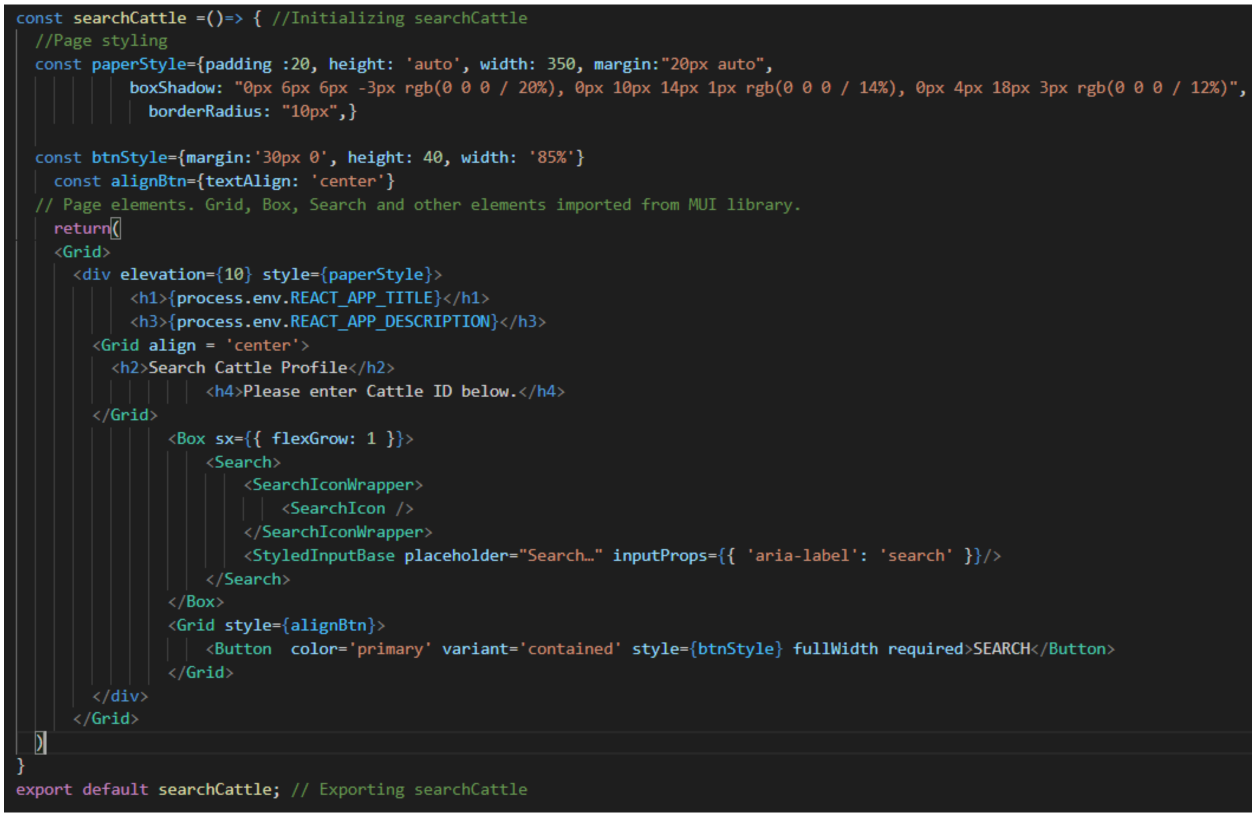
```
const cattleProfile =()=> { // Initializing cattle Profile
  //Page styling code
  const paperStyle={padding :20, height: 'auto', width: 350, margin:"20px auto",
            boxShadow: "0px 6px 6px -3px rgb(0 0 0 / 20%), 0px 10px 14px 1px rgb(0 0 0 / 14%), 0px 4px 18px 3px rgb(0 0 0 / 12%)",
              borderRadius: "10px",}

  const btnSave = {margin: '10px 0', width: '40%', height: 40, backgroundColor: 'green', align: 'right', fontSize: '13px'};
  const btnCancel = {margin: '10px 0', marginRight: '20%', width: '40%', height: 40, backgroundColor: 'red', align: 'left', fontSize: '13px'};
  const txtareaStyle={width: '100%', fontSize: '18px'}
  // Elements of the page. Grid, Button, TextField, and other elements imported from the MUI library
    return(
    <Grid>
      <div elevation={10} style={paperStyle}>
            <h1>{process.env.REACT_APP_TITLE}</h1>
    <h3>{process.env.REACT_APP_DESCRIPTION}</h3>
        <Grid align = 'center'>
          <h2>Create New Cattle Profile</h2>
        </Grid>
                <TextField label='Birth Date' placeholder='Birth Date' fullWidth required />
                <TextField label='Gender' placeholder='Gender' fullWidth required />
                <TextField label='Cattle ID' placeholder='Cattle ID' fullWidth required /><br></br><br></br>
                <TextareaAutosize label='Underlying Health Issues' placeholder='Underlying Health Issues' style={txtareaStyle}/>
                <TextField label='Weight' placeholder='Weight' fullWidth required />
                <TextField label='Location' placeholder='Location' fullWidth required />
                <TextField label='Breed' placeholder='Breed' fullWidth required />
                <TextField label='Rearing Type' placeholder='Rearing Type' fullWidth required />
                <TextField label='Type of Antibiotics Used' placeholder='Type of Antibiotics Used' fullWidth required />
                <TextField label='Reproduction Status' placeholder='Reproduction Status' fullWidth required />
                <TextField label='DNA Type' placeholder='DNA Type' fullWidth required /><br></br><br></br>
                <FormControl fullWidth>
                  <InputLabel>Life Status</InputLabel>
                  <Select label="Life Status" fullWidth required>
                    <MenuItem value="Alive">Alive</MenuItem>
                    <MenuItem value="Dead">Dead</MenuItem>
                  </Select>
                </FormControl><br></br><br></br>
                <Button  color='primary' variant='contained' style={btnCancel}>Cancel</Button>
                <Button  color='primary' variant='contained' style={btnSave}>Add Profile</Button>
        </div>
      </Grid>
  )
}
export default cattleProfile; // Exporting cattleProfile
```

UC5- Add Product Information

Search Cattle Code

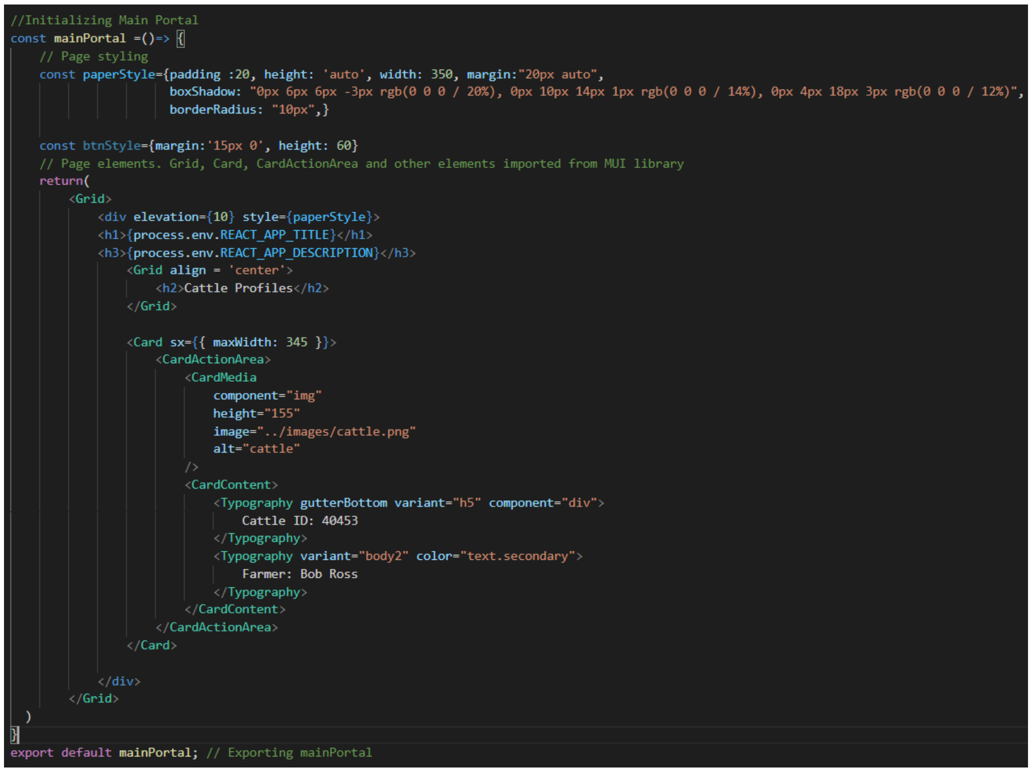


```
// MUI page styling
const Search = styled('div')(({ theme }) => ({
  position: 'relative',
  borderRadius: theme.shape.borderRadius,
  backgroundColor: alpha(theme.palette.common.white, 0.15),
  '&:hover': {
    backgroundColor: alpha(theme.palette.common.white, 0.25),
  },
  marginLeft: 0,
  width: '100%',
  [theme.breakpoints.up('sm')]: {
    marginLeft: theme.spacing(1),
    width: 'auto',
  },
}));

const SearchIconWrapper = styled('div')(({ theme }) => ({
  padding: theme.spacing(0, 2),
  height: '100%',
  position: 'absolute',
  pointerEvents: 'none',
  display: 'flex',
  alignItems: 'center',
  justifyContent: 'center',
}));

const StyledInputBase = styled(InputBase)(({ theme }) => ({
  color: 'inherit',
  '& .MuiInputBase-input': {
    padding: theme.spacing(1, 1, 1, 0),
    // vertical padding + font size from searchIcon
    paddingLeft: `calc(1em + ${theme.spacing(4)})`,
    transition: theme.transitions.create('width'),
    width: '100%',
    [theme.breakpoints.up('sm')]: {
      width: '12ch',
      '&:focus': {
        width: '20ch',
      },
    },
  },
}));
```



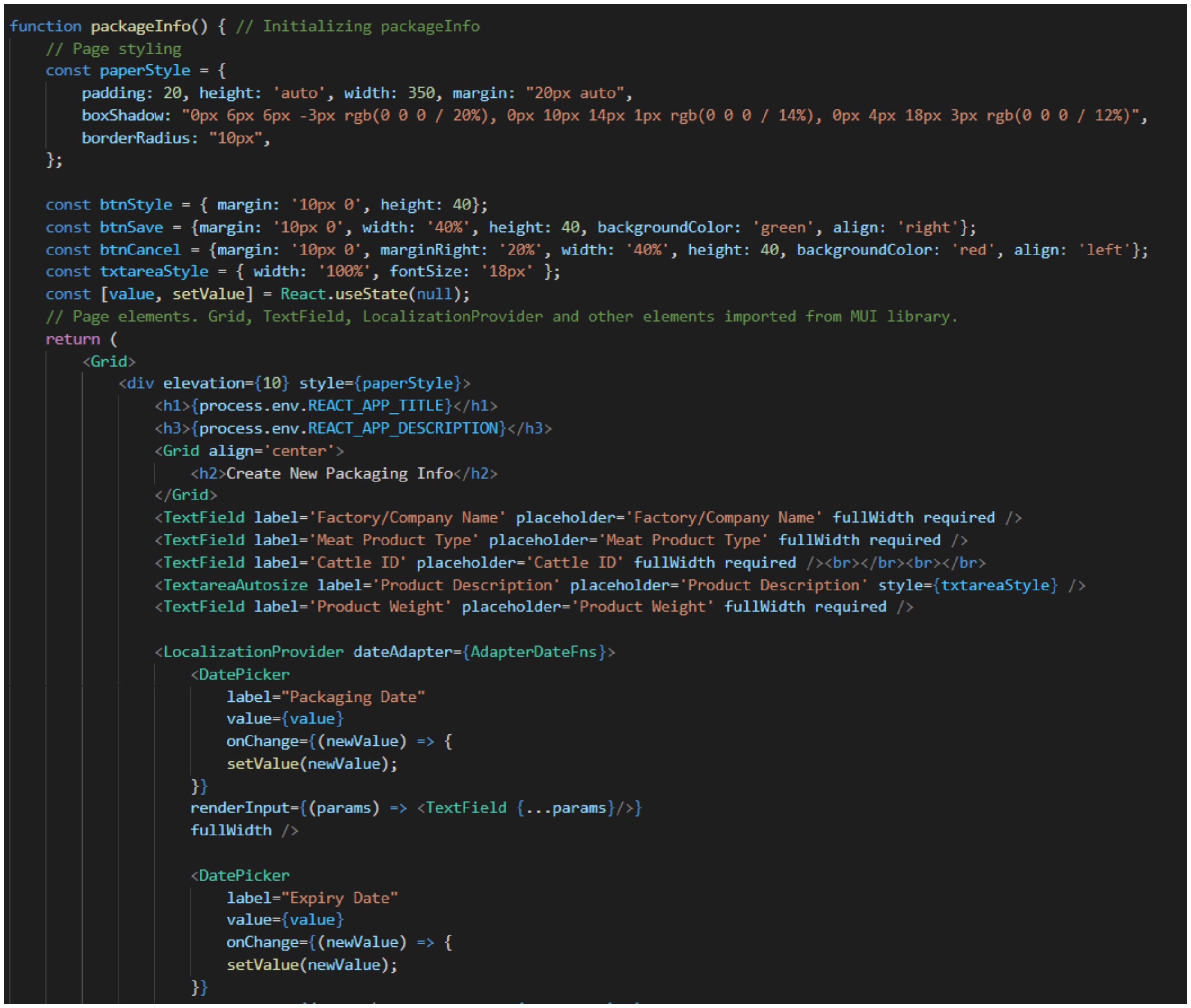
```
const searchCattle =()=> { //Initializing searchCattle
  //Page styling
  const paperStyle={padding :20, height: 'auto', width: 350, margin:"20px auto",
            boxShadow: "0px 6px 6px -3px rgb(0 0 0 / 20%), 0px 10px 14px 1px rgb(0 0 0 / 14%), 0px 4px 18px 3px rgb(0 0 0 / 12%)",
             borderRadius: "10px",}

  const btnStyle={margin:'30px 0', height: 40, width: '85%'}
    const alignBtn={textAlign: 'center'}
  // Page elements. Grid, Box, Search and other elements imported from MUI library.
    return(
    <Grid>
      <div elevation={10} style={paperStyle}>
            <h1>{process.env.REACT_APP_TITLE}</h1>
            <h3>{process.env.REACT_APP_DESCRIPTION}</h3>
        <Grid align = 'center'>
          <h2>Search Cattle Profile</h2>
                    <h4>Please enter Cattle ID below.</h4>
        </Grid>
                <Box sx={{ flexGrow: 1 }}>
                    <Search>
                        <SearchIconWrapper>
                            <SearchIcon />
                        </SearchIconWrapper>
                        <StyledInputBase placeholder="Search…" inputProps={{ 'aria-label': 'search' }}/>
                    </Search>
                </Box>
                <Grid style={alignBtn}>
                    <Button  color='primary' variant='contained' style={btnStyle} fullWidth required>SEARCH</Button>
                </Grid>
        </div>
      </Grid>
  )
}
export default searchCattle; // Exporting searchCattle
```

Cattle Profile Code

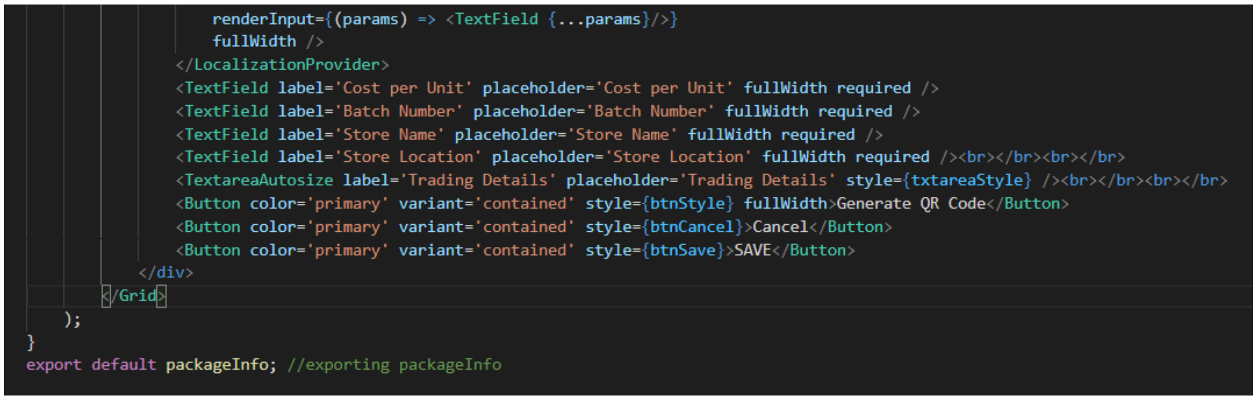


```
//Initializing Main Portal
const mainPortal =()=> {
    // Page styling
    const paperStyle={padding :20, height: 'auto', width: 350, margin:"20px auto",
                      boxShadow: "0px 6px 6px -3px rgb(0 0 0 / 20%), 0px 10px 14px 1px rgb(0 0 0 / 14%), 0px 4px 18px 3px rgb(0 0 0 / 12%)",
                      borderRadius: "10px",}

    const btnStyle={margin:'15px 0', height: 60}
    // Page elements. Grid, Card, CardActionArea and other elements imported from MUI library
    return(
        <Grid>
            <div elevation={10} style={paperStyle}>
            <h1>{process.env.REACT_APP_TITLE}</h1>
            <h3>{process.env.REACT_APP_DESCRIPTION}</h3>
                <Grid align = 'center'>
                    <h2>Cattle Profiles</h2>
                </Grid>

                <Card sx={{ maxWidth: 345 }}>
                    <CardActionArea>
                        <CardMedia
                            component="img"
                            height="155"
                            image="../images/cattle.png"
                            alt="cattle"
                        />
                        <CardContent>
                            <Typography gutterBottom variant="h5" component="div">
                                Cattle ID: 40453
                            </Typography>
                            <Typography variant="body2" color="text.secondary">
                                Farmer: Bob Ross
                            </Typography>
                        </CardContent>
                    </CardActionArea>
                </Card>

            </div>
        </Grid>
  )
}
export default mainPortal; // Exporting mainPortal
```

Package Information Code



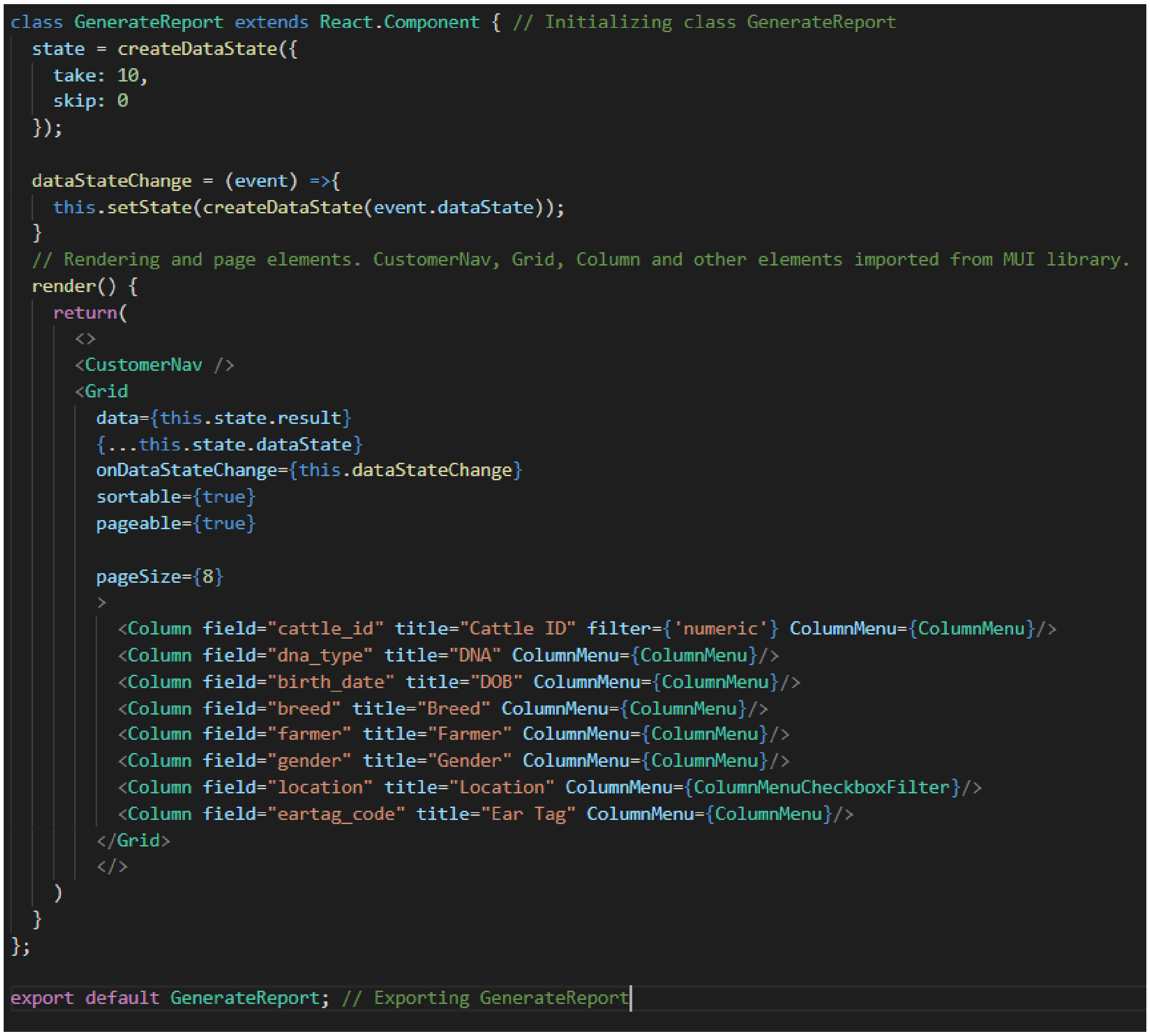
```
function packageInfo() { // Initializing packageInfo
    // Page styling
    const paperStyle = {
        padding: 20, height: 'auto', width: 350, margin: "20px auto",
        boxShadow: "0px 6px 6px -3px rgb(0 0 0 / 20%), 0px 10px 14px 1px rgb(0 0 0 / 14%), 0px 4px 18px 3px rgb(0 0 0 / 12%)",
        borderRadius: "10px",
    };

    const btnStyle = { margin: '10px 0', height: 40};
    const btnSave = {margin: '10px 0', width: '40%', height: 40, backgroundColor: 'green', align: 'right'};
    const btnCancel = {margin: '10px 0', marginRight: '20%', width: '40%', height: 40, backgroundColor: 'red', align: 'left'};
    const txtareaStyle = { width: '100%', fontSize: '18px' };
    const [value, setValue] = React.useState(null);
    // Page elements. Grid, TextField, LocalizationProvider and other elements imported from MUI library.
    return (
        <Grid>
            <div elevation={10} style={paperStyle}>
                <h1>{process.env.REACT_APP_TITLE}</h1>
                <h3>{process.env.REACT_APP_DESCRIPTION}</h3>
                <Grid align='center'>
                    <h2>Create New Packaging Info</h2>
                </Grid>
                <TextField label='Factory/Company Name' placeholder='Factory/Company Name' fullWidth required />
                <TextField label='Meat Product Type' placeholder='Meat Product Type' fullWidth required />
                <TextField label='Cattle ID' placeholder='Cattle ID' fullWidth required /><br></br><br></br>
                <TextareaAutosize label='Product Description' placeholder='Product Description' style={txtareaStyle} />
                <TextField label='Product Weight' placeholder='Product Weight' fullWidth required />

                <LocalizationProvider dateAdapter={AdapterDateFns}>
                    <DatePicker
                        label="Packaging Date"
                        value={value}
                        onChange={(newValue) => {
                        setValue(newValue);
                    }}
                    renderInput={(params) => <TextField {...params}/>}
                    fullWidth />

                    <DatePicker
                        label="Expiry Date"
                        value={value}
                        onChange={(newValue) => {
                        setValue(newValue);
                    }}
```



```
                    renderInput={(params) => <TextField {...params}/>}
                    fullWidth />
                </LocalizationProvider>
                <TextField label='Cost per Unit' placeholder='Cost per Unit' fullWidth required />
                <TextField label='Batch Number' placeholder='Batch Number' fullWidth required />
                <TextField label='Store Name' placeholder='Store Name' fullWidth required />
                <TextField label='Store Location' placeholder='Store Location' fullWidth required /><br></br><br></br>
                <TextareaAutosize label='Trading Details' placeholder='Trading Details' style={txtareaStyle} /><br></br><br></br>
                <Button color='primary' variant='contained' style={btnStyle} fullWidth>Generate QR Code</Button>
                <Button color='primary' variant='contained' style={btnCancel}>Cancel</Button>
                <Button color='primary' variant='contained' style={btnSave}>SAVE</Button>
            </div>
        </Grid>
    );
}
export default packageInfo; //exporting packageInfo
```

UC7- Generate Reports

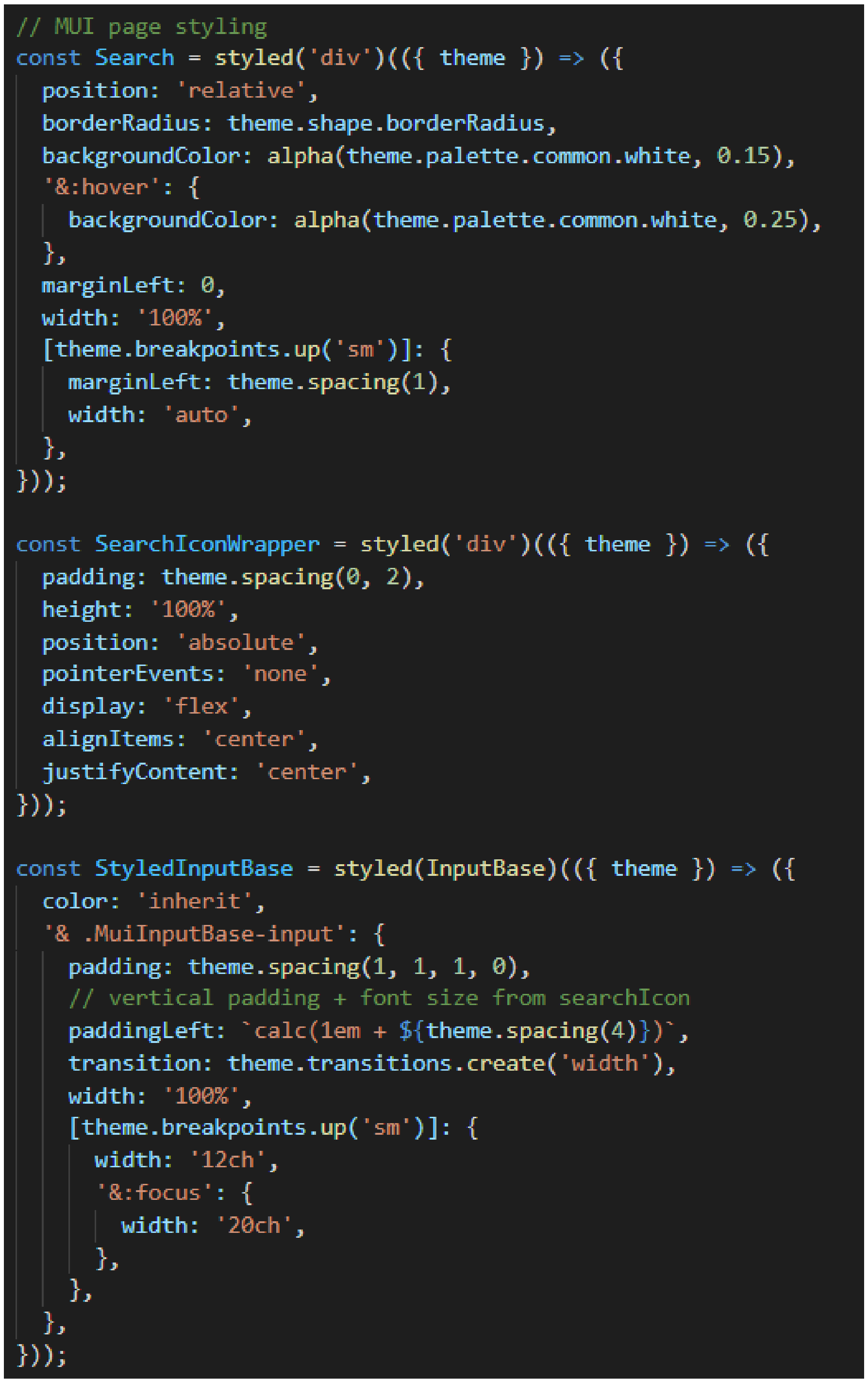
Generate Report Code



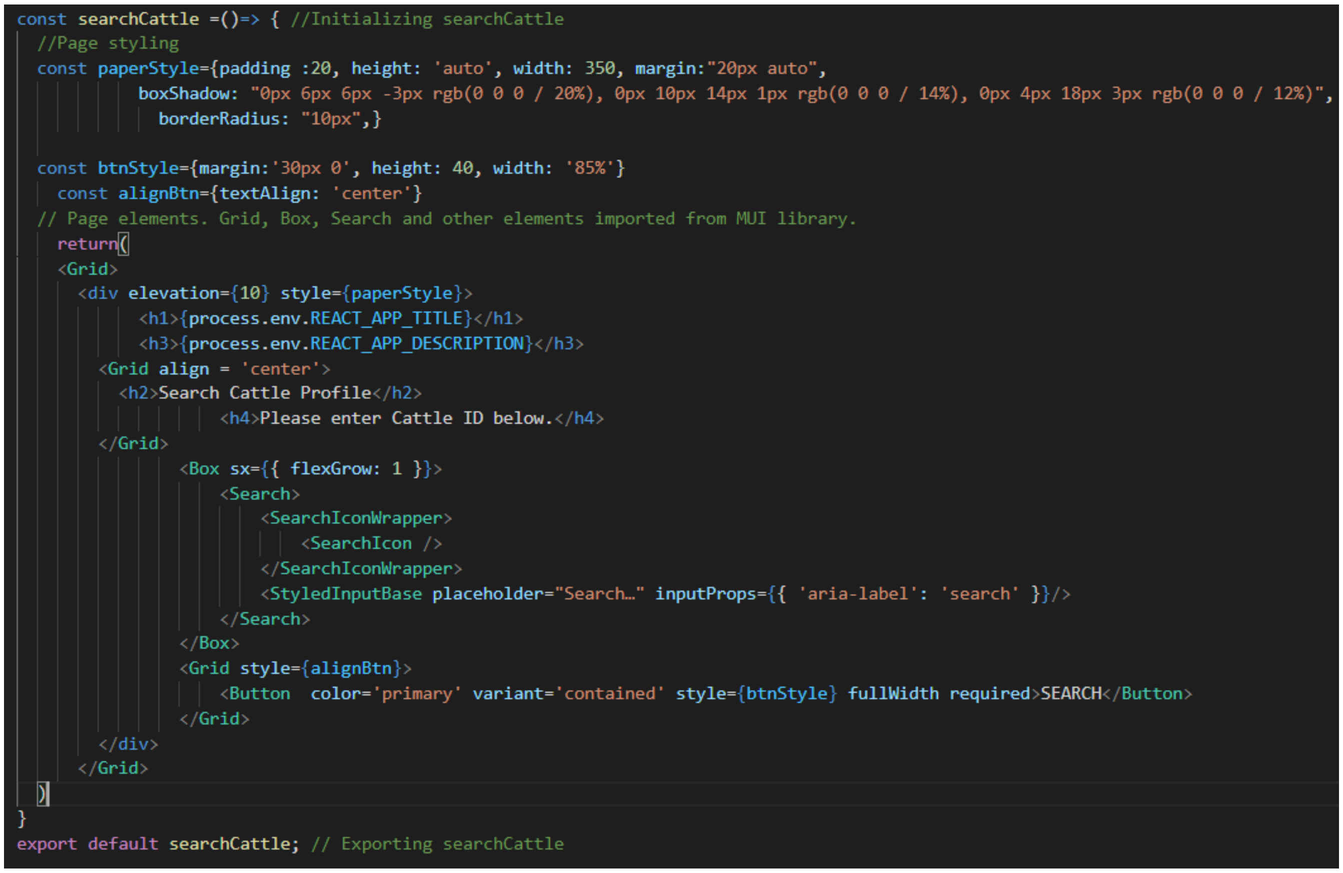
```
class GenerateReport extends React.Component { // Initializing class GenerateReport
  state = createDataState({
    take: 10,
    skip: 0
  });

  dataStateChange = (event) =>{
    this.setState(createDataState(event.dataState));
  }
  // Rendering and page elements. CustomerNav, Grid, Column and other elements imported from MUI library.
  render() {
    return(
      <>
      <CustomerNav />
      <Grid
        data={this.state.result}
        {...this.state.dataState}
        onDataStateChange={this.dataStateChange}
        sortable={true}
        pageable={true}

        pageSize={8}
        >
          <Column field="cattle_id" title="Cattle ID" filter={'numeric'} ColumnMenu={ColumnMenu}/>
          <Column field="dna_type" title="DNA" ColumnMenu={ColumnMenu}/>
          <Column field="birth_date" title="DOB" ColumnMenu={ColumnMenu}/>
          <Column field="breed" title="Breed" ColumnMenu={ColumnMenu}/>
          <Column field="farmer" title="Farmer" ColumnMenu={ColumnMenu}/>
          <Column field="gender" title="Gender" ColumnMenu={ColumnMenu}/>
          <Column field="location" title="Location" ColumnMenu={ColumnMenuCheckboxFilter}/>
          <Column field="eartag_code" title="Ear Tag" ColumnMenu={ColumnMenu}/>
        </Grid>
        </>
    )
  }
};

export default GenerateReport; // Exporting GenerateReport
```

UC-8 Add Slaughter Information

Search Cattle Code

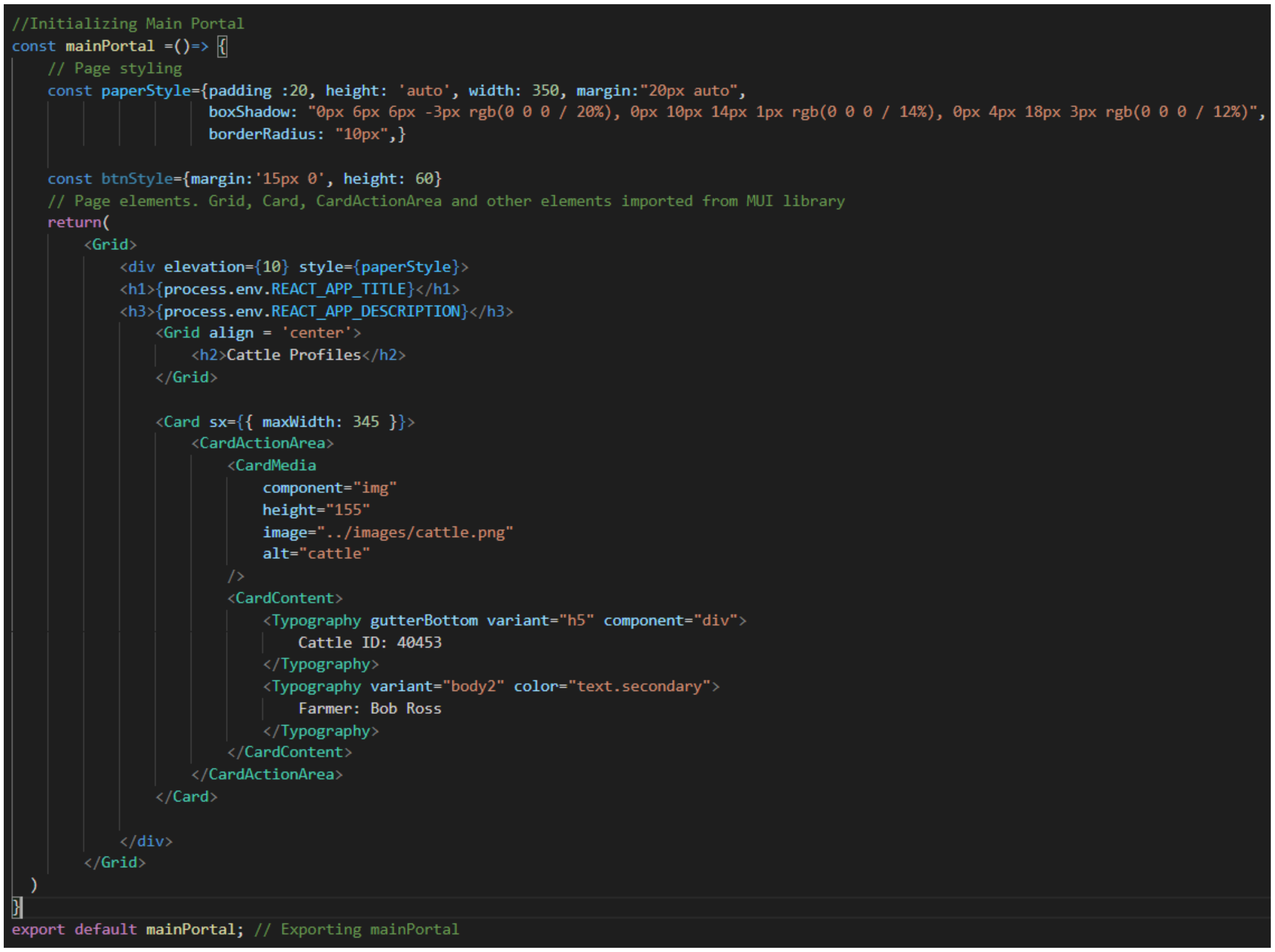


```
// MUI page styling
const Search = styled('div')(({ theme }) => ({
  position: 'relative',
  borderRadius: theme.shape.borderRadius,
  backgroundColor: alpha(theme.palette.common.white, 0.15),
  '&:hover': {
    backgroundColor: alpha(theme.palette.common.white, 0.25),
  },
  marginLeft: 0,
  width: '100%',
  [theme.breakpoints.up('sm')]: {
    marginLeft: theme.spacing(1),
    width: 'auto',
  },
}));

const SearchIconWrapper = styled('div')(({ theme }) => ({
  padding: theme.spacing(0, 2),
  height: '100%',
  position: 'absolute',
  pointerEvents: 'none',
  display: 'flex',
  alignItems: 'center',
  justifyContent: 'center',
}));

const StyledInputBase = styled(InputBase)(({ theme }) => ({
  color: 'inherit',
  '& .MuiInputBase-input': {
    padding: theme.spacing(1, 1, 1, 0),
    // vertical padding + font size from searchIcon
    paddingLeft: `calc(1em + ${theme.spacing(4)})`,
    transition: theme.transitions.create('width'),
    width: '100%',
    [theme.breakpoints.up('sm')]: {
      width: '12ch',
      '&:focus': {
        width: '20ch',
      },
    },
  },
}));
```



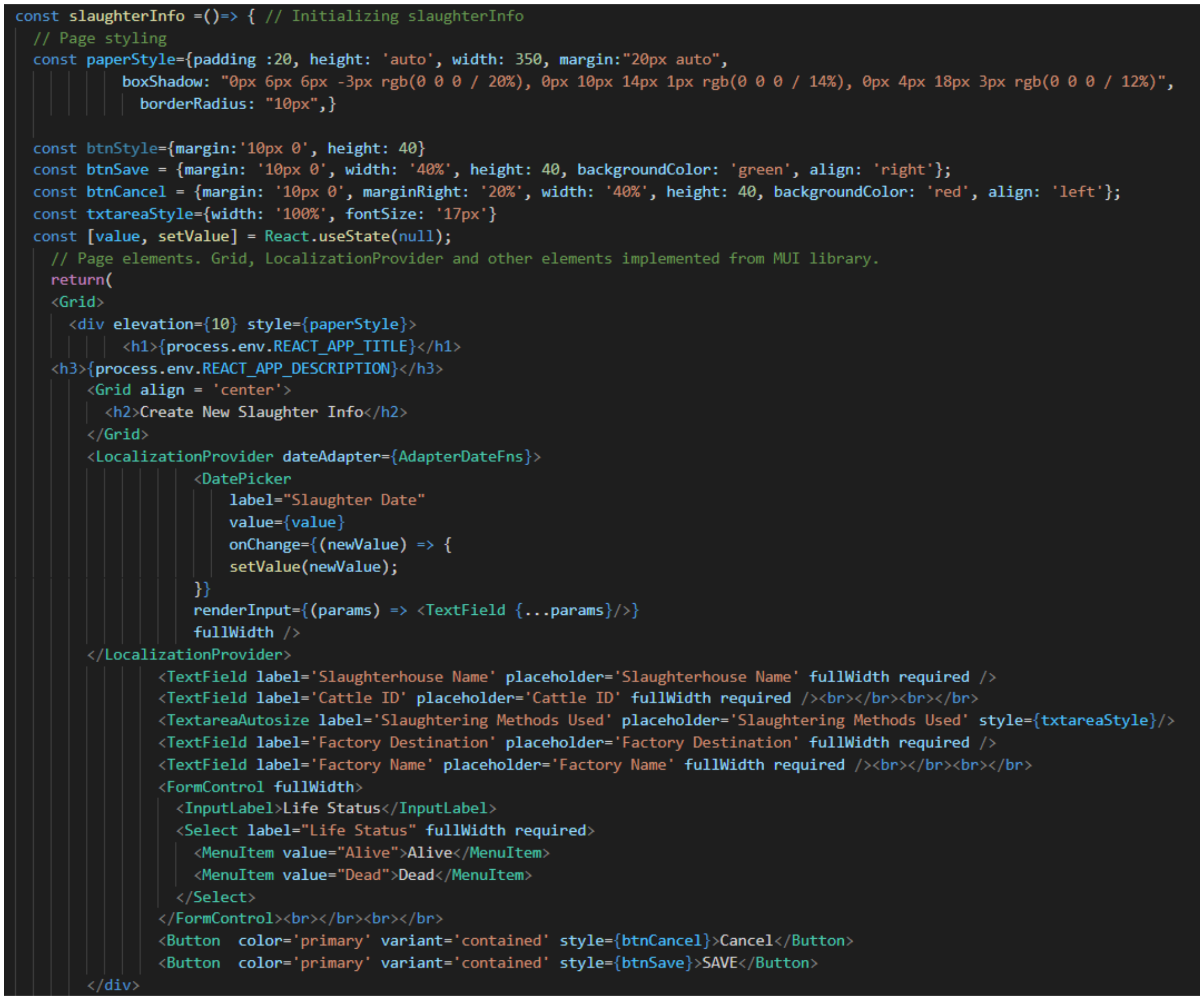
```
const searchCattle =()=> { //Initializing searchCattle
  //Page styling
  const paperStyle={padding :20, height: 'auto', width: 350, margin:"20px auto",
            boxShadow: "0px 6px 6px -3px rgb(0 0 0 / 20%), 0px 10px 14px 1px rgb(0 0 0 / 14%), 0px 4px 18px 3px rgb(0 0 0 / 12%)",
              borderRadius: "10px",}

  const btnStyle={margin:'30px 0', height: 40, width: '85%'}
    const alignBtn={textAlign: 'center'}
  // Page elements. Grid, Box, Search and other elements imported from MUI library.
    return(
    <Grid>
      <div elevation={10} style={paperStyle}>
            <h1>{process.env.REACT_APP_TITLE}</h1>
            <h3>{process.env.REACT_APP_DESCRIPTION}</h3>
        <Grid align = 'center'>
          <h2>Search Cattle Profile</h2>
                    <h4>Please enter Cattle ID below.</h4>
        </Grid>
                <Box sx={{ flexGrow: 1 }}>
                    <Search>
                        <SearchIconWrapper>
                            <SearchIcon />
                        </SearchIconWrapper>
                        <StyledInputBase placeholder="Search…" inputProps={{ 'aria-label': 'search' }}/>
                    </Search>
                </Box>
                <Grid style={alignBtn}>
                    <Button  color='primary' variant='contained' style={btnStyle} fullWidth required>SEARCH</Button>
                </Grid>
        </div>
      </Grid>
  )
}
export default searchCattle; // Exporting searchCattle
```

Cattle Profile Code

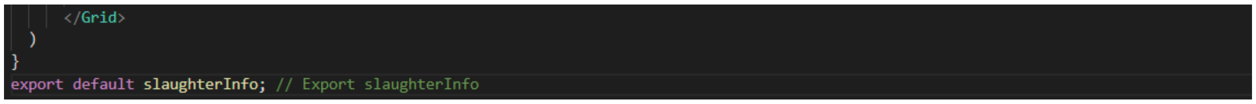


```
//Initializing Main Portal
const mainPortal =()=> {
    // Page styling
    const paperStyle={padding :20, height: 'auto', width: 350, margin:"20px auto",
                    boxShadow: "0px 6px 6px -3px rgb(0 0 0 / 20%), 0px 10px 14px 1px rgb(0 0 0 / 14%), 0px 4px 18px 3px rgb(0 0 0 / 12%)",
                    borderRadius: "10px",}

    const btnStyle={margin:'15px 0', height: 60}
    // Page elements. Grid, Card, CardActionArea and other elements imported from MUI library
    return(
        <Grid>
            <div elevation={10} style={paperStyle}>
            <h1>{process.env.REACT_APP_TITLE}</h1>
            <h3>{process.env.REACT_APP_DESCRIPTION}</h3>
                <Grid align = 'center'>
                    <h2>Cattle Profiles</h2>
                </Grid>

                <Card sx={{ maxWidth: 345 }}>
                    <CardActionArea>
                        <CardMedia
                            component="img"
                            height="155"
                            image="../images/cattle.png"
                            alt="cattle"
                        />
                        <CardContent>
                            <Typography gutterBottom variant="h5" component="div">
                                Cattle ID: 40453
                            </Typography>
                            <Typography variant="body2" color="text.secondary">
                                Farmer: Bob Ross
                            </Typography>
                        </CardContent>
                    </CardActionArea>
                </Card>

            </div>
        </Grid>
    )
}
export default mainPortal; // Exporting mainPortal
```

Slaughter Information Code



```
const slaughterInfo =()=> { // Initializing slaughterInfo
  // Page styling
  const paperStyle={padding :20, height: 'auto', width: 350, margin:"20px auto",
            boxShadow: "0px 6px 6px -3px rgb(0 0 0 / 20%), 0px 10px 14px 1px rgb(0 0 0 / 14%), 0px 4px 18px 3px rgb(0 0 0 / 12%)",
              borderRadius: "10px",}

  const btnStyle={margin:'10px 0', height: 40}
  const btnSave = {margin: '10px 0', width: '40%', height: 40, backgroundColor: 'green', align: 'right'};
  const btnCancel = {margin: '10px 0', marginRight: '20%', width: '40%', height: 40, backgroundColor: 'red', align: 'left'};
  const txtareaStyle={width: '100%', fontSize: '17px'}
  const [value, setValue] = React.useState(null);
    // Page elements. Grid, LocalizationProvider and other elements implemented from MUI library.
    return(
    <Grid>
      <div elevation={10} style={paperStyle}>
            <h1>{process.env.REACT_APP_TITLE}</h1>
    <h3>{process.env.REACT_APP_DESCRIPTION}</h3>
        <Grid align = 'center'>
          <h2>Create New Slaughter Info</h2>
        </Grid>
        <LocalizationProvider dateAdapter={AdapterDateFns}>
                    <DatePicker
                        label="Slaughter Date"
                        value={value}
                        onChange={(newValue) => {
                        setValue(newValue);
                    }}
                    renderInput={(params) => <TextField {...params}/>}
                    fullWidth />
        </LocalizationProvider>
                <TextField label='Slaughterhouse Name' placeholder='Slaughterhouse Name' fullWidth required />
                <TextField label='Cattle ID' placeholder='Cattle ID' fullWidth required /><br></br><br></br>
                <TextareaAutosize label='Slaughtering Methods Used' placeholder='Slaughtering Methods Used' style={txtareaStyle}/>
                <TextField label='Factory Destination' placeholder='Factory Destination' fullWidth required />
                <TextField label='Factory Name' placeholder='Factory Name' fullWidth required /><br></br><br></br>
                <FormControl fullWidth>
                  <InputLabel>Life Status</InputLabel>
                  <Select label="Life Status" fullWidth required>
                    <MenuItem value="Alive">Alive</MenuItem>
                    <MenuItem value="Dead">Dead</MenuItem>
                  </Select>
                </FormControl><br></br><br></br>
                <Button  color='primary' variant='contained' style={btnCancel}>Cancel</Button>
                <Button  color='primary' variant='contained' style={btnSave}>SAVE</Button>
        </div>
```



```
      </Grid>
  )
}
export default slaughterInfo; // Export slaughterInfo
```

References

Firebase: Firestore vs. Realtime Database - Javatpoint. www.javatpoint.com. (n.d.). Retrieved April 7, 2022, from https://www.javatpoint.com/firebase-firestore-vs-realtime-database

Jest: PhpStorm. PhpStorm Help. (n.d.). Retrieved April 7, 2022, from https://www.jetbrains.com/help/phpstorm/running-unit-tests-on-jest.html#ws_jest_installation

Microsoft. (2021, November 3). *Visual studio code frequently asked questions*. RSS. Retrieved April 7, 2022, from https://code.visualstudio.com/docs/supporting/FAQ

Programmers.io. (2021, December 15). *Reasons to use react with node JS for web development*. Programmers.io. Retrieved April 7, 2022, from https://programmers.io/web-development-using-reactjs-with-nodejs/

React front-end development - 6 things to consider before choosing. React Front-End Development - 6 Things To Consider. (n.d.). Retrieved April 7, 2022, from https://brainhub.eu/library/reasons-to-choose-react/#:~:text=To%20give%20you%20a%20gentle,it%20works%20with%20Virtual%20DOM