Created API for the server to receive the database for the tables I used, the app.get would receive the table that I am trying to get information from the tables, employee, manager, hourly-emp, and salary-emp the 'app' is the express app which im using the express dependentcy in React also with the mysql dependentcy.

```
app.get("/api/getEmp", (req, res) => {
  db.query("SELECT * FROM employees", (err, result) => {
    res.send(result);
  });
});
```

While app.get received the tables app.post allows the API insert new tuples into the tables. It would read the information that the user was input using the body parser dependency in react. Also with all the insert, updates, and deletion I do time them to see how long it takes to run the task for to display for the user.

```
app.post("/api/insert", (req, res) => {
  const startTime = performance.now();
  const ssn = req.body.ssn;
  const dob = req.body.dob;
  const Fname = req.body.Fname;
  const Minit = req.body.Minit;
  const Lname = req.body.Lname;

  db.query(
    "INSERT INTO employees (ssn, dob, Fname, Minit, Lname) VALUES (?,?,?,?,?)",
    [ssn, dob, Fname, Minit, Lname],
    (err, result) => {
      console.log(result);
    }
  );
  const endTime = performance.now();
  const runtime = `${(endTime - startTime).toFixed(6)}`;
  res.status(200).json({ message: "Table updated successfully", runtime });
});
```

```
app.delete("/api/delete/:ssn", (req, res) => {
  const startTime = performance.now();
  const ssn = req.params.ssn;

  db.query("DELETE FROM employees WHERE ssn = ?", ssn, (err, result) => {
    if (err) console.log(err);
  });
  const endTime = performance.now();
  const runtime = ` ${(endTime - startTime).toFixed(6)} `;
  res.status(200).json({ message: "Table updated successfully", runtime });
});
```

This app.put is used to update the table but this one only updates the employee table by using the body parser to parse out the user input to be able to determine what changes are being made the specific tuple and would write a sql query that would get the user input to update but also go through an if statement to see if the attributes are being changed in the table if so they are push to be added to a array which the array is added to a sql query that would put the columns that are being updated if so. Which I also have it sending me responses through the console if any errors occur to during this task.

```
app.put("/api/update/", (req, res) => {
  const startTime = performance.now();
  const ssn = req.body.ssn;
  const updates = [];

  // Add non null and string values to the updates array
  if (req.body.dob && req.body.dob !== "") {
    updates.push("`dob` = ?");
  }
  if (req.body.Fname && req.body.Fname !== "") {
    updates.push("`Fname` = ?");
  }
  if (req.body.Minit && req.body.Minit !== "") {
    updates.push("`Minit` = ?");
  }
  if (req.body.Lname && req.body.Lname !== "") {
    updates.push("`Lname` = ?");
  }

  if (updates.length === 0) {
    // No non null and string values to update
    return res.status(400).json({ message: "No values to update" });
  }

  const values = [];
```

```
  values.push(ssn);

  db.query(
    `UPDATE employees SET ${updates.join(", ")} WHERE ssn = ?`,
    values,
    (err, result) => {
      if (err) {
        console.log(err);
        return res.status(500).json({ message: "Error updating table" });
      }

      if (result.affectedRows === 0) {
        // No rows were affected, which means the ssn value doesn't match any record in the database
        return res.status(404).json({ message: "Record not found" });
      }

      console.log(result);
      const endTime = performance.now();
      const runtime = ` ${(endTime - startTime).toFixed(6)} `;
      res.status(200).json({ message: "Table updated successfully", runtime });
    }
  );
});
```

The main app function for the react app which has the navigation bar component that I created with react bootstrap and I used the dependency routes to help me create hyperlinks to route the links for each component that I created to show the tables.

```
function App() {
  return (
    <>
    <div className='App1'>
    <Router>

      <h1 className='App'>Project 5</h1>
      <NavBar></NavBar>

      <Routes>
      <Route exact path='/' Component={Employee}>
      </Route>
      <Route path='/Employee' Component={Employee}>
      </Route>
      <Route path='/Managers' Component={Managers}>
      </Route>
      <Route path='/Salaries' Component={Salaries}>
      </Route>
      <Route path='/Hourly' Component={Hourly}>
      </Route>
      </Routes>


      </Router>
      </div>
      </>
  );
}
```

The setField function creates a array to input the user input from the insert form which also will set the error into a array if any are found in the form.
The formattedEmpList just collect the employees table but I have it where it format the date for the table to be displayed in that certain way also with the hourly, and salary table I have it formatting the currency of those tables. The useEffect syncs the api to get the table so that it can be displayed.

```
const setField = (field, value) => {
  setForm({ ...form, [field]: value });

  if(!!errors[field]){
    setErrors({...errors, [field]: null});
  }
};

const formattedEmpList = empList.map((emp) => {
  const dateObj = new Date(emp.dob); // Create a Date object from the dob string
  const formattedDate = `${dateObj.getMonth() + 1} / ${dateObj.getDate()} / ${dateObj.getFullYear()}`;
  return { ...emp, dob: formattedDate }; // Return a new object with the formatted date
});

//display the employee table
useEffect(() => {
  Axios.get("http://localhost:3001/api/getEmp").then((response) => {
    setList(response.data);
  });
```

The validateForm function is to just validate the insert form for a new manager, employee, hourly, or salary tuple but each of those function are different due based on the tables columns and its, requirement in the database.

```
const validateFrom = () => {
  const {ssn, dob, Fname, Lname} = form;
  const newErrors = {};

  if(!dob || dob === ''){
    newErrors.dob = 'Please enter date of birth.';
  }
  if(!ssn || ssn === ''){
    newErrors.ssn = 'Please enter ssn.';
  }else if (ssn.length < 9 || ssn.length > 9){
    newErrors.ssn = 'Not a ssn.';
  }
  else if (empList.find((val) => val.SSN === ssn)){
    newErrors.ssn = 'ssn already exists';
  }
  if(!Fname || Fname === ''){
    newErrors.Fname = 'Please enter first name.';
  }
  if(!Lname || Lname === ''){
    newErrors.Lname = 'Please enter last name.';
  }

  return newErrors;
};
```

This handleSubmit function handles the submit of all forms for its default but only for the insert form it would collect any validation that has not been met and would set the Error useState which allows to display the errors. And reset the insert form to null values.

```
const handleSubmit = (e) => {
  if (e) {
   e.preventDefault();
  }

  const errorsForm = validateFrom();

  if (Object.keys(errorsForm).length > 0){
    setErrors(errorsForm)
  } else {
    setForm({});
    console.log('Added')
  }

  //save employee
};
```

For the submit function it would look for any errors in the insert form and would call the handlesubmit fuction if any errors occur in the form, but if no errors occur it would post the user input to the posy API which would insert it to the table, since the I also trying to get the runtime I also request the runtime for how long the task it took to complete, to be able to display it to the user. Also make sure the form is reset, and handles for alerts and modal close.

```
const submitEmp = async (ssn) => {
  if (empList.find((val) => val.SSN === ssn) || ssn === '' || ssn.length > 9 || ssn.length < 9
  || dob === '' || Fname === '' || Lname === ''){
    handleSubmit();
  }else{
  Axios.post("http://localhost:3001/api/insert", {
    ssn: ssn,
    dob: dob,
    Fname: Fname,
    Minit: Minit,
    Lname: Lname,
  }).then((response) => {
    setRuntime2(response.data.runtime)
    return response.data
  });

  setList([
    ...empList,
    { SSN: ssn, dob: dob, Fname: Fname, Minit: Minit, Lname: Lname },
  ]);
  setForm({});
  addHandleClose();
  setAlert2(true);
}
};
```

This delete function works on collecting the specific id in this case it is SSN and when I retrieves the SSN it would delete is from the table in the database and also return the runtime for the time it took to complete and resets the table to show the recent table after the deletion without refreshing the whole page. Shows alert to complete deletion and closes the confirm deletion modal.

```
const deleteEmp = () => {
  Axios.delete(`http://localhost:3001/api/delete/${deleteBySSN}`).then((response) => {
    setRuntime3(response.data.runtime)
    return response.data
  }).then(() => {
    Axios.get("http://localhost:3001/api/getEmp").then((response) => {
    setList(response.data);
  });
  });
  deleteHandleClose();
  setAlert3(true);
};
```

This update function just collects the user input data that they inserted to update for the specific SSN and updates it based on what was inserted. And there is no validation for this update form. Also, this function refreshes the List useState which allows to display the table without refreshing the whole page every time the table is updated. And resets the form and, given the alert of how long it took to complete the task, which also closes the form modal.

```
const updateEmp = () => {
  Axios.put(`http://localhost:3001/api/update/`, {
    ssn: updateSSN,
    dob: updateDob,
    Fname: updatedFname,
    Minit: updatedMinit,
    Lname: updatedLname,
  }).then((response) => {
    setRuntime(response.data.runtime)
  }).then(() => {
    Axios.get("http://localhost:3001/api/getEmp").then((response) => {
    setList(response.data);
  })});
  setUpdatedSSN("");
  updatedDOB("");
  setUpdatedFname("");
  setUpdatedMinit("");
  setUpdatedLname("");
  setAlert(true);
  handleClose();
};
```

This useEffect is added to the hourly and salary components which get the manager and employees table for any changes that is made to be sure that the data is up to date and to help for the validation for their insertion forms. The Manager components has the save style of the function without calling the api/getMan because that is getting the manager table which is redundant for the manager component.

```
  useEffect(() => {
    Axios.get("http://localhost:3001/api/getEmp").then((response) => {
      setEmpList(response.data);
    });

    Axios.get("http://localhost:3001/api/getMan").then((response) => {
      setManList(response.data);
    });
  }, []);

  useEffect(() => {
    console.log("empList:", empList);
    console.log("manList:", manList);
  }, [empList, manList]);
```

This section of the jsx display the alerts when alert = true other wise it doesn't show alerts. Also allows the user to search the table based on first name, which is different for each table. Also has a button for the user to insert an employee.

```
            <div className="App">
            <Alert variant="success" show={alert} onClose={() =>{setAlert(false)}} dismissible>
          <Alert.Heading>Successfully Updated Employee!</Alert.Heading>
          <p>Runtime for update was {runtime} ms</p>
        </Alert>

        <Alert variant="success" show={alert2} onClose={() =>{setAlert2(false)}} dismissible>
          <Alert.Heading>Successfully Added Employee!</Alert.Heading>
          <p>Runtime for insertion was {runtime2} ms</p>
        </Alert>

        <Alert variant="danger" show={alert3} onClose={() =>{setAlert3(false)}} dismissible>
          <Alert.Heading>Employee Delete!</Alert.Heading>
          <p>Runtime for deletion {runtime3} ms</p>
        </Alert>
         </div>
            <Row>
              <Col xs="auto">
                <Form.Group as={Col} controlId="formGridEmail">
                  <Form.Control
                    type="text"
                    placeholder="Search By First Name"
                    className="mb-2"
                    onChange={(e) => {
                      setSearch(e.target.value);
                    }}
                  />
                </Form.Group>
              </Col>
              <Col xs="auto">
                <Button
                  variant="primary"
                  className="mb-2"
                  onClick={addHandleShow}
                >
                  Add Employee
                </Button>
              </Col>
            </Row>
```

This section of the jsx creates t modal for the add employee form which calls handleSubmit when form is submitted but sets the field for the form validation and the SSN for the database insertion. Also show form feedback if any errors occur and goes down for each value the form contain for each component I created. Also resets form and errors for when user closes modal or hits the cancel button. The end of the insert form which when the add is click calls the submitEmp function.

```
<Modal show={show} onHide={() => {
    setForm({});
    setErrors({});
    addHandleClose();}}>
  <Modal.Header closeButton>
    <Modal.Title>Add Employee</Modal.Title>
  </Modal.Header>
  <Modal.Body>
    <Form id="myForm" onSubmit={handleSubmit}>
      <Form.Group className="mb-3" controlId="validationCustom01">
        <FloatingLabel
          controlId="floatingInput"
          label="SSN"
          className="mb-3"
        >
          <Form.Control
            required
            maxLength={9}
            type="number"
            name="ssn"
            value={form.ssn}
            isInvalid={!!errors.ssn}
            onChange={(e) => {

              setField('ssn', e.target.value);
              setSSN(e.target.value);
            }}
            placeholder="SSN"
          ></Form.Control>
          <Form.Control.Feedback type="invalid">
           <div className="red">{errors.ssn}</div>
          </Form.Control.Feedback>
        </FloatingLabel>
      </Form.Group>
```

```
            </Form.Group>
          </Form>
        </Modal.Body>
        <ModalFooter>
          <Button
            variant="primary"
            type="submit"
            onClick={() => {
              submitEmp(ssn);
            }
          }
          >
            Add
          </Button>
          <Button variant="warning" onClick={() => {
          setForm({});
          setErrors({});
          addHandleClose();}}>
            Cancel
          </Button>
        </ModalFooter>
      </Modal>
```

This section of the jsx code I create the table with it's header and list down each tuple that is found in the table database but since I formatted the date is prints the new list of tuples same goes with the salary and hourly table.

```
<Table striped bordered hover variant="dark">
  <thead>
    <tr>
      <th>SSN</th>
      <th>DOB</th>
      <th>First Name</th>
      <th>Middle Initial</th>
      <th>First Name</th>
      <th>Modify</th>
    </tr>
  </thead>

  {
    // eslint-disable-next-line
    formattedEmpList
      .filter((val) => {
        if (search === "") {
          return val;
        } else if (
          val.Fname.toLowerCase().includes(search.toLowerCase())
        ) {
          return val;
        }
      })
      .map((val) => {
        return (
          <tbody>
            <tr>
              <td>{val.SSN}</td>
              <td>{val.dob}</td>
              <td>{val.Fname}</td>
              <td>{val.Minit}</td>
              <td>{val.Lname}</td>
              <td>
```

This section updates the user by SSN, which has the similar structure with the insertion form but doesn't contain the validation. Calls the upadteEmp function when the update button is click or cancel if the user decides to cancel the update.

```
<Button
  variant="secondary"
  onClick={() => {
    handleShow();
    setUpdatedSSN(val.SSN);
  }}
>
  Update
</Button>
<Modal show={showUpdate} onHide={handleClose}>
  <Modal.Header closeButton>
    <Modal.Title>Update Employee</Modal.Title>
  </Modal.Header>
  <Modal.Body>
    <Form onSubmit={handleSubmit}>
      <Form.Group
        className="mb-3"
        controlId="exampleForm.ControlInput1"
      >
        <Form.Label>Date of Birth</Form.Label>
        <Form.Control
          type="date"
          placeholder="1999-01-26"
          onChange={(e) => {
            updatedDOB(e.target.value);
          }}
        ></Form.Control>
    </Form>
  </Modal.Body>
  <Modal.Footer>
    <Button
      variant="primary"
      type="submit"
      onClick={() => {
        updateEmp(val.SSN);
      }}
    >
      update
    </Button>
    <Button variant="warning" onClick={handleClose}>
      Cancel
    </Button>
  </Modal.Footer>
</Modal>
```

The deletion part of the jsx code has a button for deletion but when it's pressed a modal pops up and make the user to confirm the deletion and if user confirms that when the tuple of the table if finally delete off the table unless they hit cancel.

```jsx
<Button
  variant="danger"
  onClick={() => {
    deleteHandleShow();
    setDeleteBySSN(val.SSN)
  }}
>
  Delete
</Button>

<Modal show={showConfirm} onHide={deleteHandleClose}>
  <Modal.Header closeButton>
    <Modal.Title>Confirm Delete</Modal.Title>
  </Modal.Header>
  <Modal.Body>
    <p>Are you Sure you want to delete Employee?</p>
  </Modal.Body>
  <Modal.Footer>
    <Button
      variant="secondary"
      type="submit"
      onClick={() => {
        deleteEmp(val.SSN);
      }}
    >
      Confirm
    </Button>
    <Button variant="warning" onClick={deleteHandleClose}>
      Cancel
    </Button>
  </Modal.Footer>
</Modal>
```