INDEX

CONTENTS	Page No.
1. ABSTRACT	4
2. INTRODUCTION	5
3. PROBLEM STATEMENT	6
4. OBJECTIVE	6
5. HARDWARE AND SOFTWARE REQUIREMENTS, TECHNOLOGIES USED	7
6. SHORT INTRODUCTION TO TECHNOLOGIES USED	8
7. SYSTEM DESIGN	12
8. SOURCE CODE	26
9. CONCLUSION	46
10. FUTURE ENHANCEMENT & BIBLIOGRAPY	47

EMPLOYEE MANAGEMENT SYSTEM

ABSTRACT

Employees are the backbone of any company therefore their management plays a major role in deciding the success of an organization. Employees Management Software makes it easy for the employer to keep track of all records. This software allows the administrator to edit employees, add new employees, transfer/promote/terminate employees. Each employee in the database is associated with a position can be added and edited when need arises. Employees can be transferred between positions easily without having to retype back their information in the database. You can check to see if there are duplicate positions/employees in the database. Most of all, the employer can assign tasks to employees and assess their progress in order to keep track of employee performance.

A flexible and easy to use Employee Management software solution for small and medium sized companies provides modules for personnel information management thereby organization and companies are able to manage the crucial organization asset— people. The combination of these modules into one application assures the perfect platform for re-engineering and aligning Human Resource processes along with the organizational goals. This system brings about an easy way of maintaining the details of employees working in any organization.

It is simple to understand and can be used by anyone who is not even familiar with simple employee system. It is user friendly and just asks the user to follow step by step operations by giving easy to follow options. It is fast and can perform many operations for a company.

The goal of this project is to design and develop an employee management system to fill existing gaps in the electronic management of employees.

INTRODUCTION

Employee Management System is a project made with an aim to replace manual Employee Management System (EMS) with more efficient computerized (EMS). This is a simple yet useful system for managing employee records with highest degree of accuracy and is meant to be used by company manager or appropriate authority person. Employee database management system is consistent of work-related and important personal information about an employee. In a nutshell, it is an online inventory of all employees of an organization.

❖ ADVANTAGES: -

- It can be used as general for any industry/company.
- Provide computerized system for maintaining records
- Improve workforce management efficiencies
- It increased accuracy of data input (error free)
- Saves time of management
- Huge data storage with less computer memory.
- Perfect entry and retrieval of data
- Securing employee information
- Very few manual errors
- Fewer compliance risks
- Low labour cost

PROBLEM STATEMENT

- **DIFFICULT TO SEARCH RECORD IN MANUAL (EMS)-** When there is no computerized system there is always a difficulty in searching of records, if the records are large in number.
- **SPACE CONSUMING-** After the no. of records become large the space for physical storage of file and records also increases if no computerized system is implemented.
- **COST CONSUMING-** As there is no computerized system to add each record paper will be needed which will increase the cost of the management of company.
- **FILE LOST-** When computerized system is not implemented file is always lost because of the human environment. Sometimes due to some human error, there may be a loss of records.
- **FILE DAMAGE-** When the computerized system is not there the file is always lost due to some accident like the spilling of water by some member to file accidentally. Besides some natural disaster like floods or fire may also damage the files.
- **TIME CONSUMING-** Preparing the list of employees and their information will take more time.

OBJECTIVES

In this world of growing technologies everything has been computerized. With large number of work opportunities the Human workforce has increased. Thus, there is a need of a system which can handle the data of such a large number of Employees. This project simplifies the task of maintaining records because of its user-friendly nature.

The objective of this project is to provide a comprehensive approach towards the management of employee information. This will be done by designing and implementing an HR management system that will bring up a major paradigm shift in the way that employee information is handled.

❖ The objectives of this system include:

- Design of a web-based HR management system to fulfill requirements such as project management, leave management, report generation to assist in performance appraisal, ESS and employee trainings.
- Well-designed database to store employee information.
- A user friendly front-end for the user to interact with the system.

Hardware and Software Requirements

❖ Software Requirements

- **Recommended: -**

  Windows 10 / Mac OS Mojave / Linux 7.5 or above

  Python 3.9.2

  Node 14.17.0

  Django 3.2.3

❖ Hardware Requirements

- **Recommended: -**

Any PC with CPUs released after 2006 with 1.5 GB or more RAM (Random Access Memory)

Disk Space -9 MB or more

Technologies Used

- **Front-end:** ReactJs
- **Back-end:** Django (Python Framework)
- **Database:** SQLite3 (Default in Django)

SHORT INTRODUCTION TO TECHNOLOGIES USED

Python 

Python is an interpreted, high-level, general-purpose programming language. Created by Guido Van Rossum and first released in 1991, python's design philosophy emphasizes code readability with its notable use of significant whitespace . Python is dynamically typed and garbage-collected. It supports multiple programming paradigms, including structured (Particularly, procedural), object-oriented, and functional programming. Python is often described as a "batteries included" language due to its comprehensive standard library.

Python was conceived in the late 1980s as a successor to the C language. Python 2.0, released in 2000, introduced features like list comprehensions and a garbage collection system capable of collecting reference cycles. Python 3.0, released in 2008, was a major revision of the language that is not completely backward-compatible, and much python 2 code does not run unmodified on python 3. Python is a multi-paradigm programming language. Object-oriented programming and structured programming are fully supported, and many of its features support functional programming and aspect-oriented programming (Including by metaprogramming and metaobjects (Magic methods)). Python uses dynamic typing and a combination of reference counting and a cycle-detecting garbage collector for memory management. It also features dynamic name resolution (late binding), which binds method and variable names during program execution.

Django 

Django is an open-source web application frame work written in Python. The primary goal of Django is to make the development of complex, data-based websites easier. Thus, Django emphasizes the reusability and pluggability of components to ensure rapid developments. Django consists of three major parts: model, view and template.

To build such a complicated web system, we need three major parts for each component: database, user interface and the functions to interact in between. Django framework provides sufficient functionalities to implement these three parts. Corresponding to database, user interface and functions in between, Django has model, template and view components to deal with each part respectively. Django's model component helps programmer to define and maintain tables in the database, while its template component helps to write html files using a combination of both html syntax and Django syntax. For those functions in between, Django provides a view component which reads the input from user interface and makes corresponding changes in the database.

ReactJS

ReactJS is a JavaScript framework that makes designing advanced and responsive interfaces hassle free. It is a highly scalable, super-performing framework, built and maintained by the folks at Facebook & Instagram. Further, it provides simple views and associations with each of the state behaviour in your application with out-of-the-box developing tools to write code and bring top quality user interfaces to life.

- **Why Choose ReactJS Framework?**

ReactJS has a rich library that allows you to express how your application should be built depending on your requirements. ReactJS even manages the dependent UI when any update or changes are made in the underlying data, reflecting across the application. It uses an auto refresh feature to update the changes based on state management.

- **ReactJS Framework's Key Features:**

ReactJS is highly flexible and its interfaces can be easily migrated to other libraries and frameworks. It has a unique functionality called 'markdown the /ibs' which allows it to comfortably debug while inspecting the root cause for failed cases.

Apart from this, there are a few more features listed below which make ReactJS framework one of the most popular frameworks in the current market.

- The ReactJS framework can be used for building cross-platform native mobile and web applications.
- It has rich component support, which can also be used as reusable components.
- It is much faster since underlying DOM manipulation utilized is called Virtual DOM.

- State management could be achieved by using Redux to share data across an application.
- JSX is a combination of JavaScript with XML which makes it faster and allows easy rendering.
- Performance and testing are easy for maintaining the current state and triggering actions along with the functions.

SQLite 

SQLite is an in-process library that implements a self-contained, serverless, zero-configuration, transactional SQL database engine. The code for SQLite is in the public domain and is thus free for use for any purpose, commercial or private. SQLite is the most widely deployed database in the world with more applications than we can count, including several high-profile projects.

SQLite is an embedded SQL database engine. Unlike most other SQL databases, SQLite does not have a separate server process. SQLite reads and writes directly to ordinary disk files. A complete SQL database with multiple tables, indices, triggers, and views, is contained in a single disk file. The database file format is cross-platform - you can freely copy a database between 32-bit and 64-bit systems or between big-endian and little-endian architectures. These features make SQLite a popular choice as an Application File Format. SQLite database files are a recommended storage format by the US Library of Congress. Think of SQLite not as a replacement for Oracle but as a replacement for fopen()

SQLite is a compact library. With all features enabled, the library size can be less than 600KB, depending on the target platform and compiler optimization settings. (64-bit code is larger. And some compiler optimizations such as aggressive function inlining and loop unrolling can cause the object code to be much larger.) There is a trade-off between memory usage and speed. SQLite generally runs faster the more memory you give it. Nevertheless, performance is usually quite good even in low-memory environments. Depending on how it is used.

Bootstrap

Bootstrap is a free and open-source CSS framework directed at responsive, mobile-first front-end web development. It contains CSS- and (optionally) JavaScript-based design templates for typography, forms, buttons, navigation, and other interface components.

Bootstrap is a HTML, CSS & JS Library that focuses on simplifying the development of informative web pages (as opposed to web apps). The primary purpose of adding it to a web project is to apply Bootstrap's choices of color, size, font and layout to that project. As such, the primary factor is whether the developers in charge find those choices to their liking. Once added to a project, Bootstrap provides basic style definitions for all HTML elements. The result is a uniform appearance for prose, tables and form elements across web browsers. In addition, developers can take advantage of CSS classes defined in Bootstrap to further customize the appearance of their contents. For example, Bootstrap has provisioned for light- and dark-colored tables, page headings, more prominent pull quotes, and text with a highlight.

Bootstrap also comes with several JavaScript components in the form of jQuery plugins. They provide additional user interface elements such as dialog boxes, tooltips, and carousels. Each Bootstrap component consists of an HTML structure, CSS declarations, and in some cases accompanying JavaScript code. They also extend the functionality of some existing interface elements, including for example an auto-complete function for input fields.

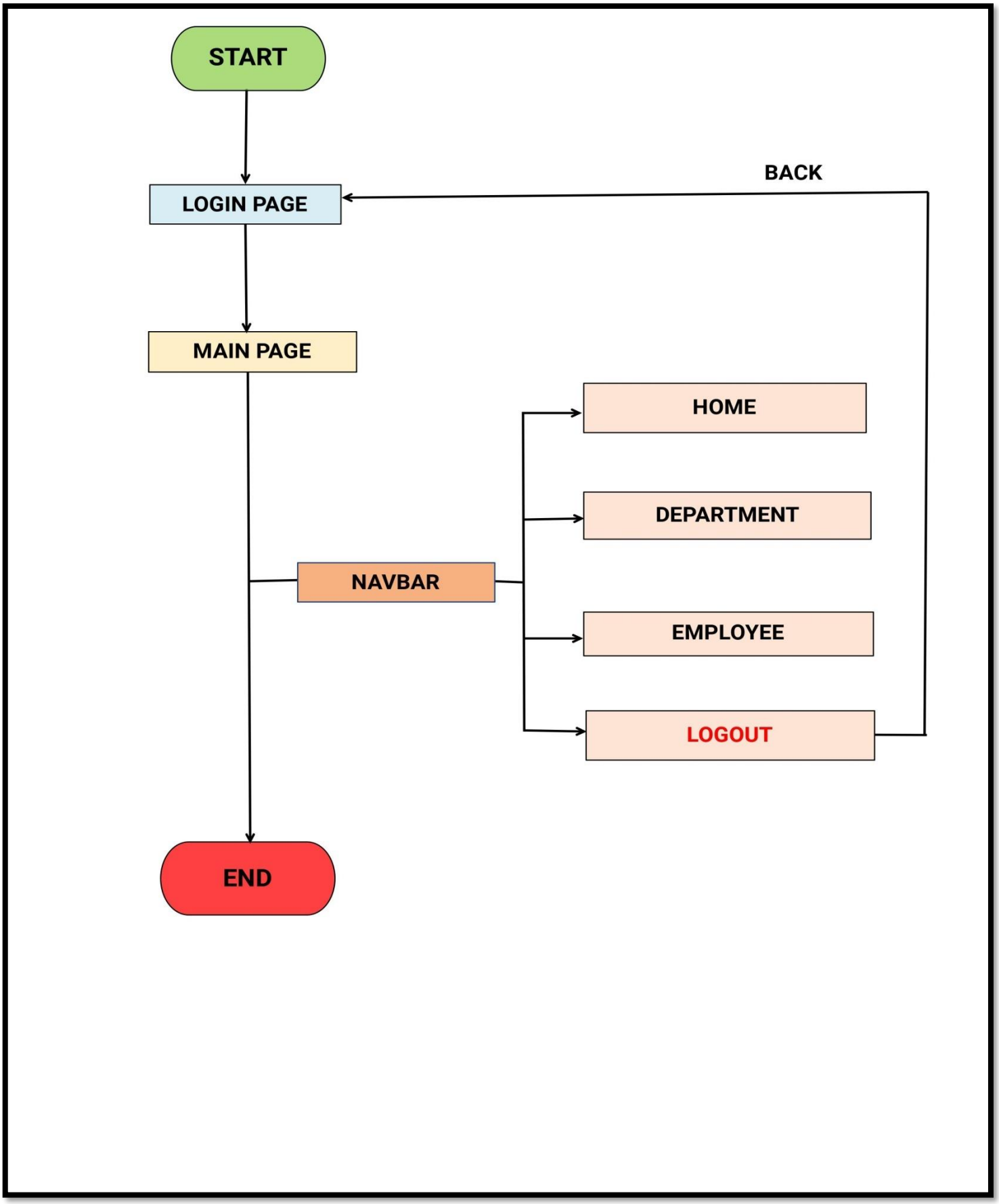
Nodejs

Node.js is an open-source, cross-platform, back-end JavaScript runtime environment that runs on the V8 engine and executes JavaScript code outside a web browser. Node.js lets developers use JavaScript to write command line tools and for server-side scripting—running scripts server-side to produce dynamic web page content before the page is sent to the user's web browser.

- Node.js is an open source server environment
- Node.js is free
- Node.js runs on various platforms (Windows, Linux, Unix, Mac OS X, etc.)
- Node.js uses JavaScript on the server

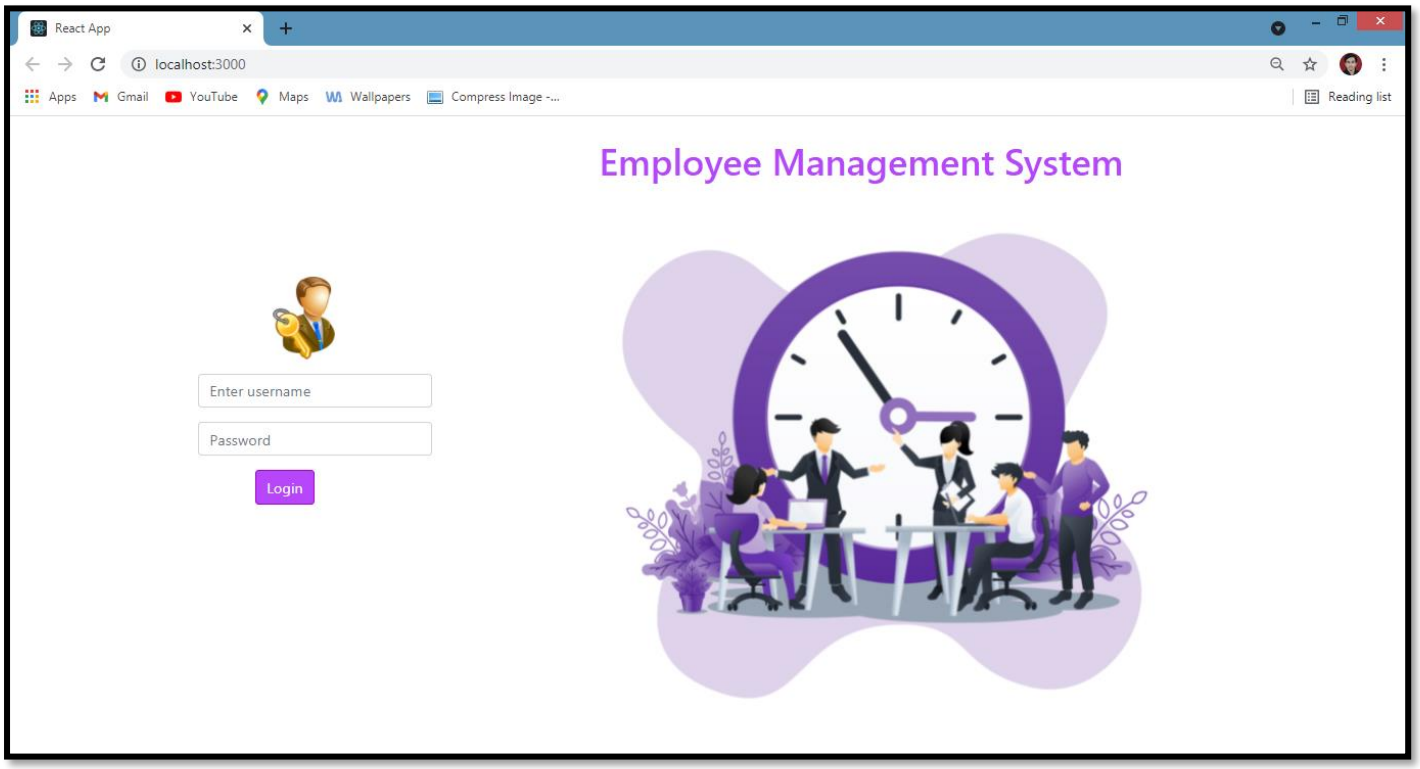
SYSTEM DESIGN

❖ Overall Employee Management System Flowchart



❖ LOGIN PAGE

➢ SCREENSHOT OF LOGIN PAGE

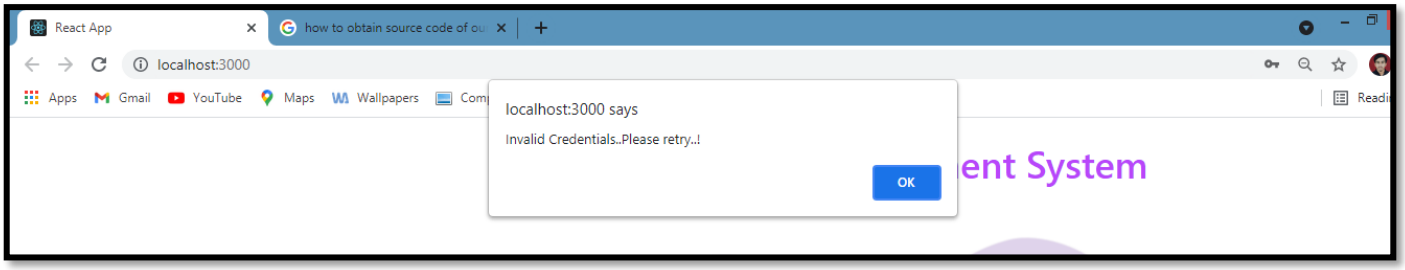


This is landing screen of our Employee Management System.

➢ System takes two inputs from user-

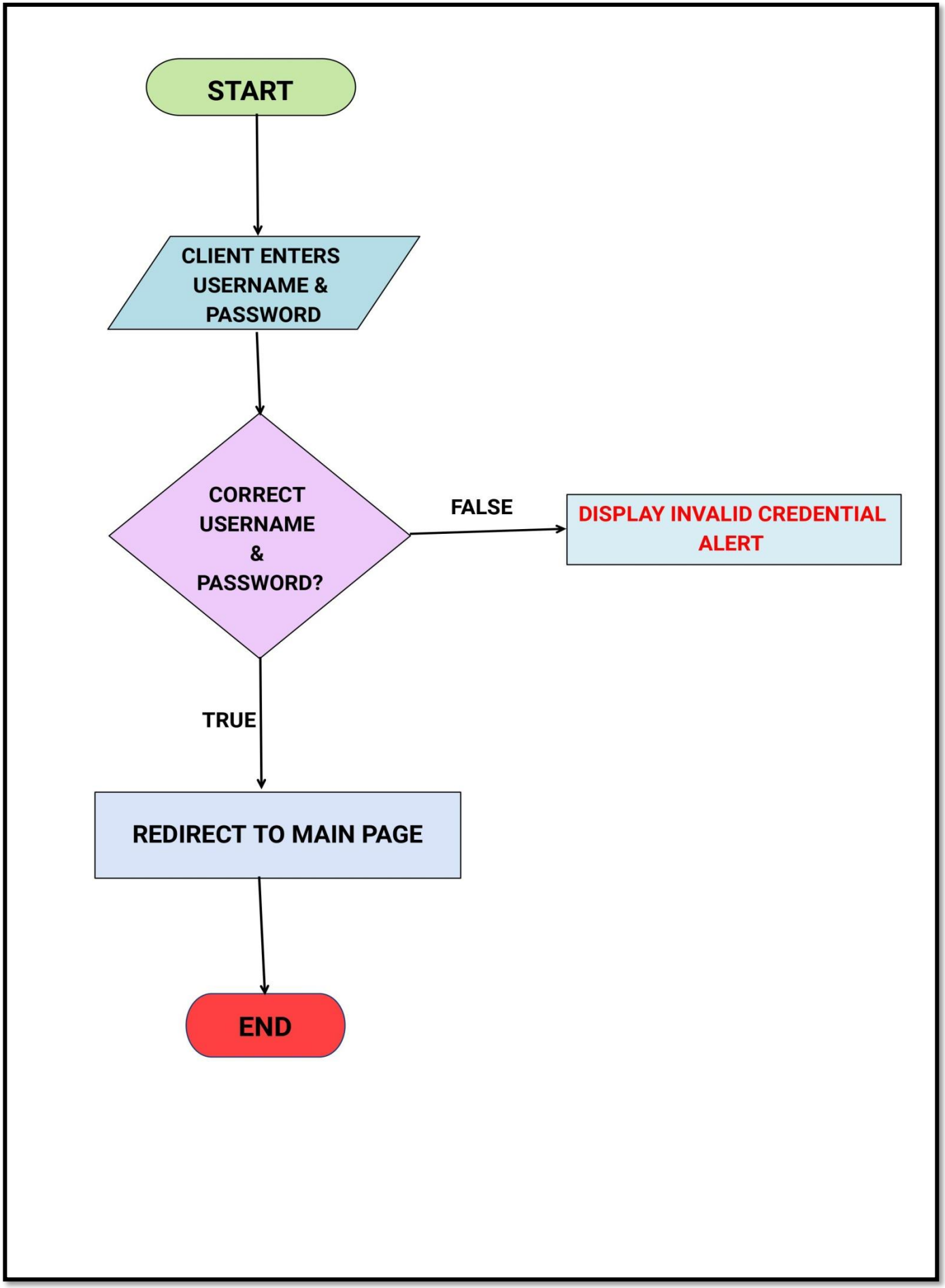
1. USERNAME 2. PASSWORD

They must be kept unique for privacy. If user **not write** correct username & password user will get **invalid credential alert.**



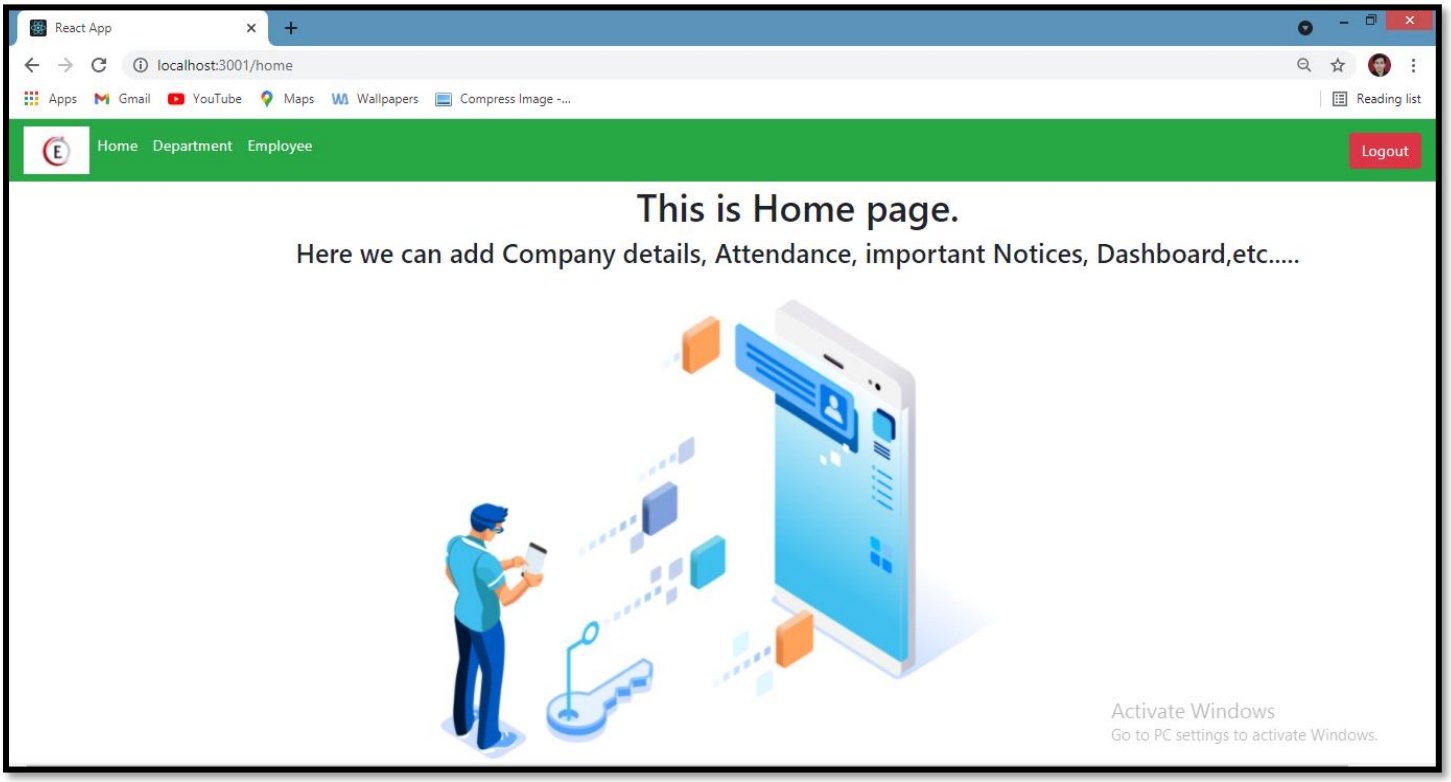
➢ Login button redirect user to Home Page (Main Page).

➢ LOGIN PAGE FLOWCHART



❖ HOME PAGE (Main Page)

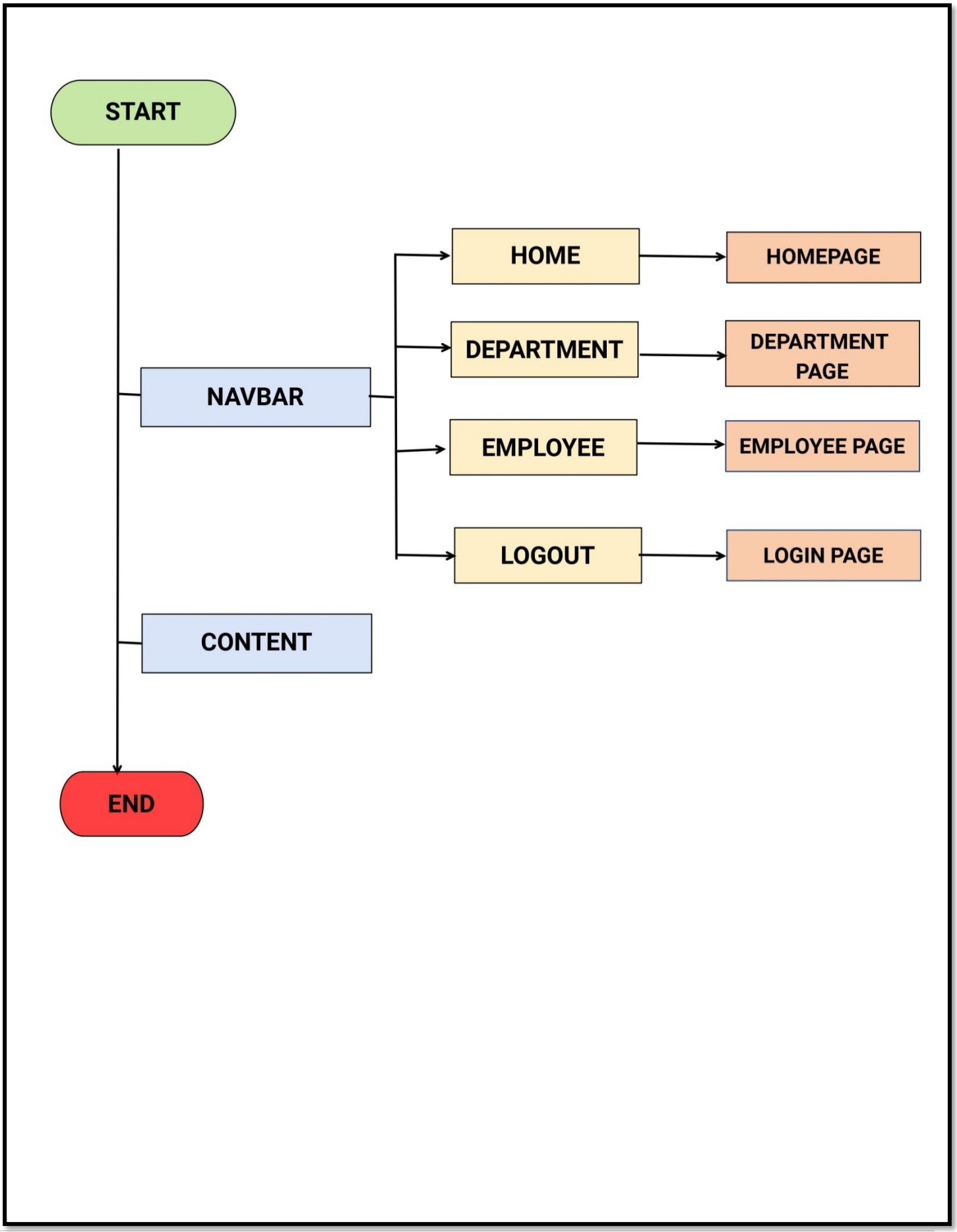
➢ SCREENSHOT OF HOME PAGE



- **This page consists of Navigation bar and Company / Industry related information.**

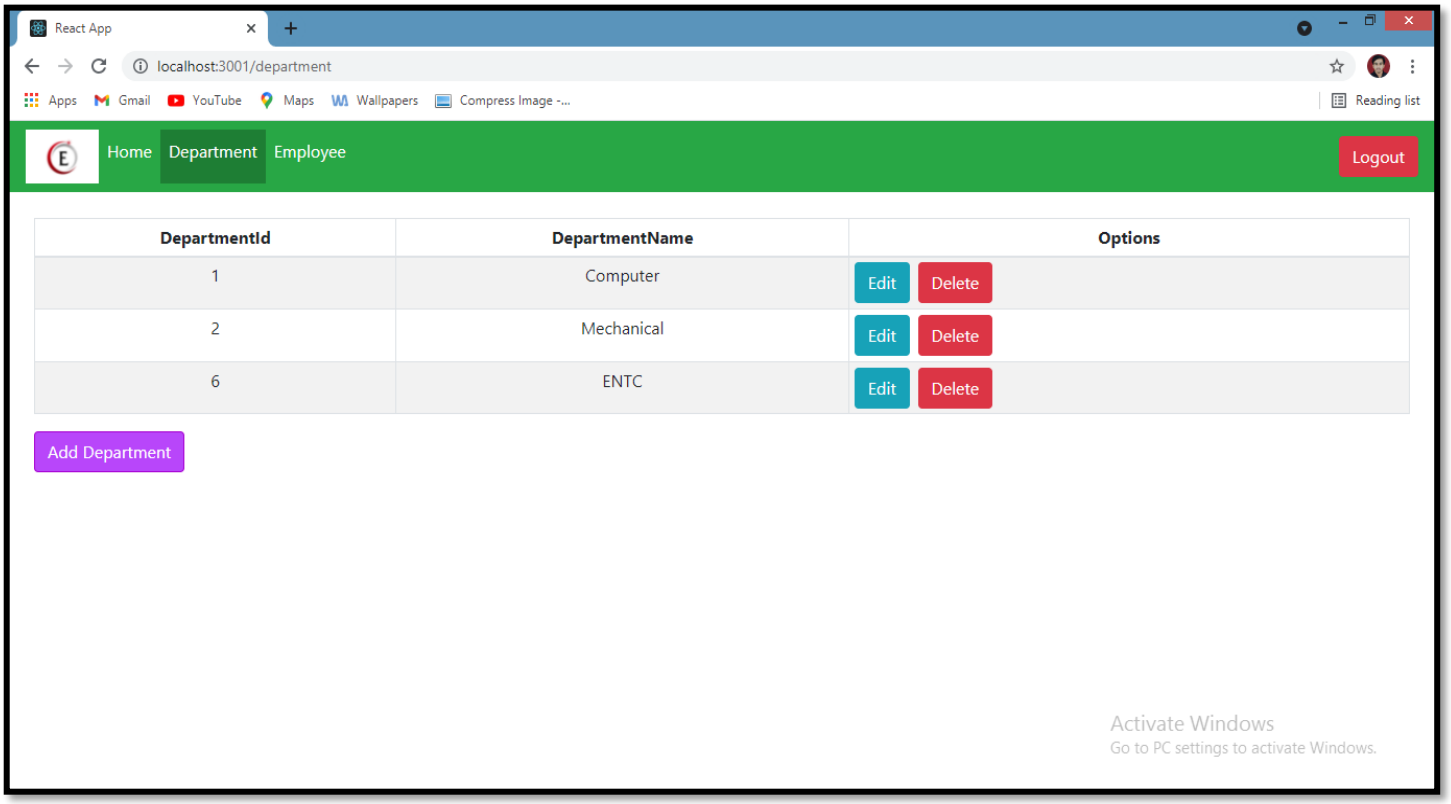
- **Navigation bar contains 3 tabs & 1 button-**
  I. Home
  II. Department (consists departments info.)
  III. Employee (consists departments info.)
  IV. Logout button (redirect again to login screen)

➢ FLOWCHART OF HOME PAGE (MAIN PAGE)



❖ DEPARTMENT PAGE

➢ SCREENSHOT OF DEPARTMENT PAGE



➢ **This page consists of 3 buttons with their respective tasks.**

I. Add Department (Click to add new department)

II. Edit (Click to update existing department)

III. Delete (Click to delete selected department)

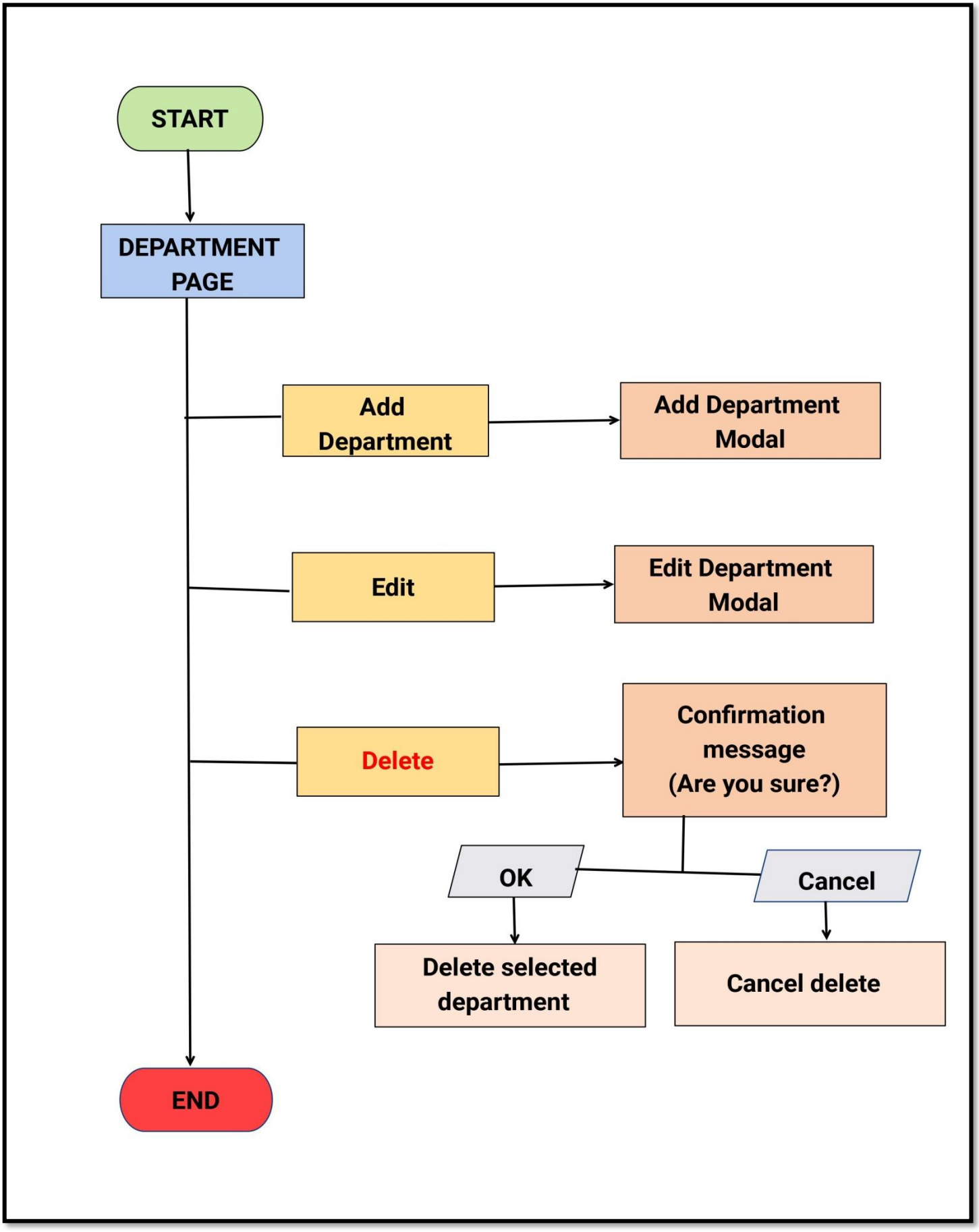
➢ **It also consists of Table with 3 Columns**

I. DepartmentId (new generated Id assign to each)

II. DepartmentName (Added Department Name)

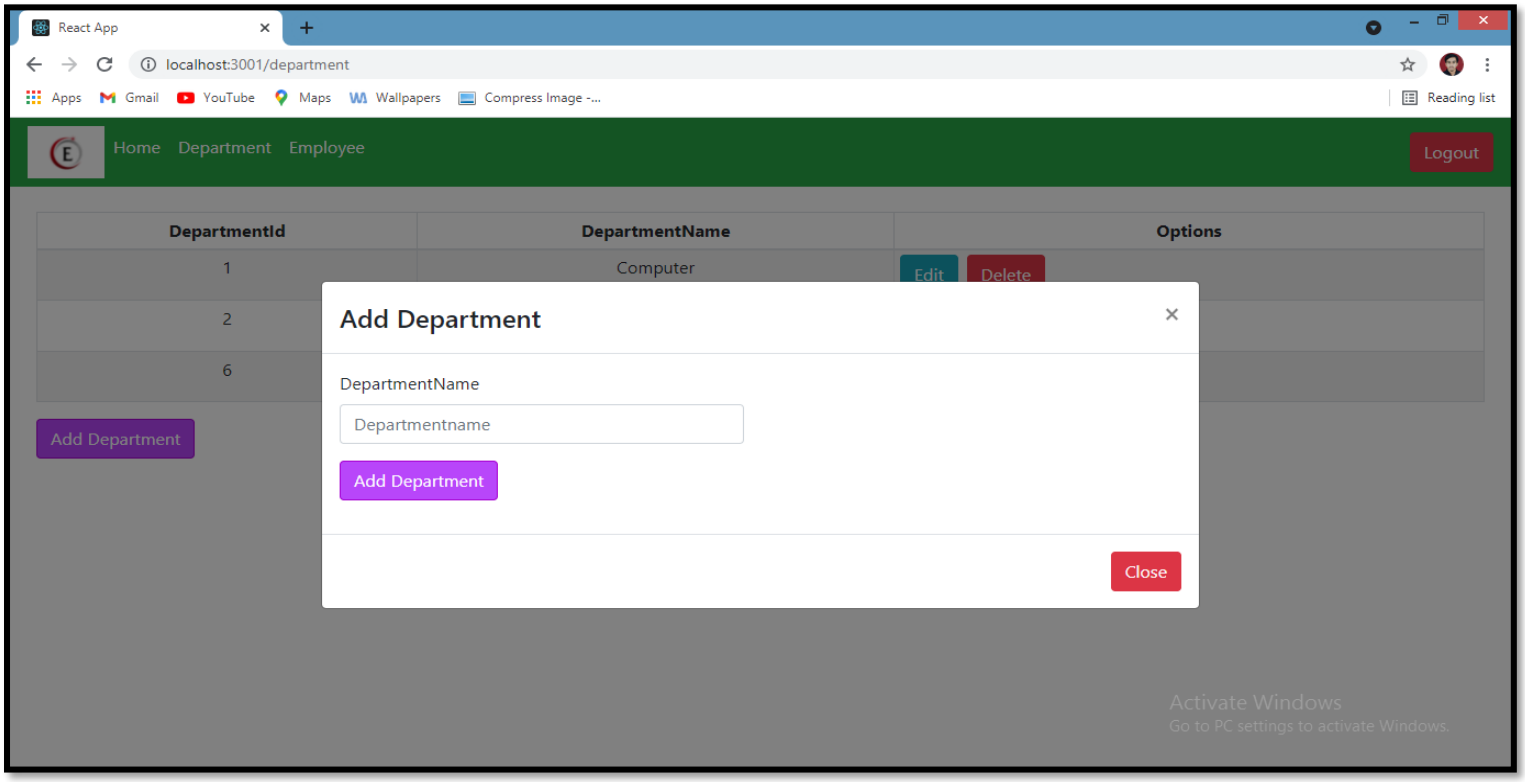
III. Options (Buttons - Edit & Delete)

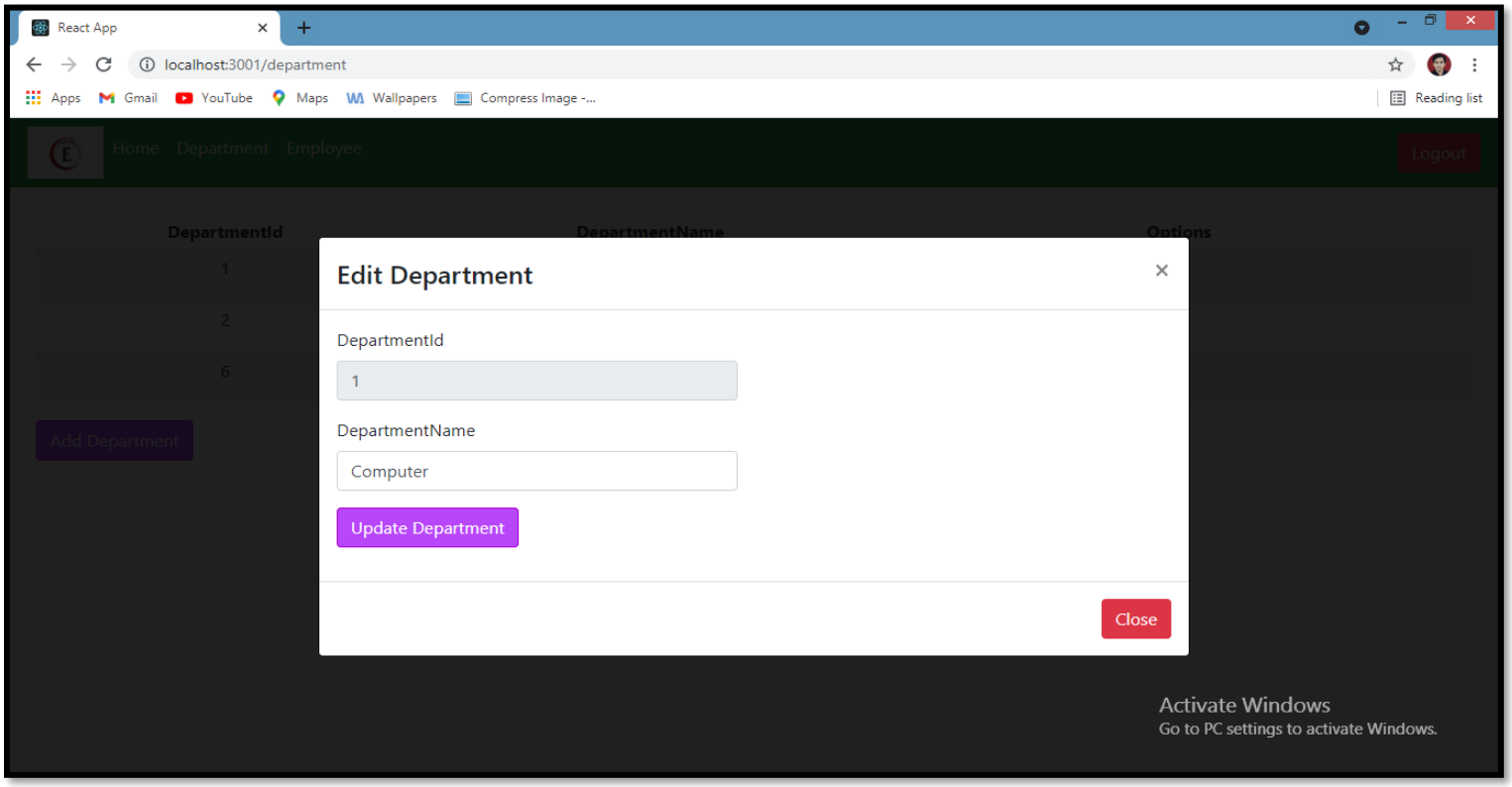
➢ FLOWCHART OF DEPARTMENT PAGE



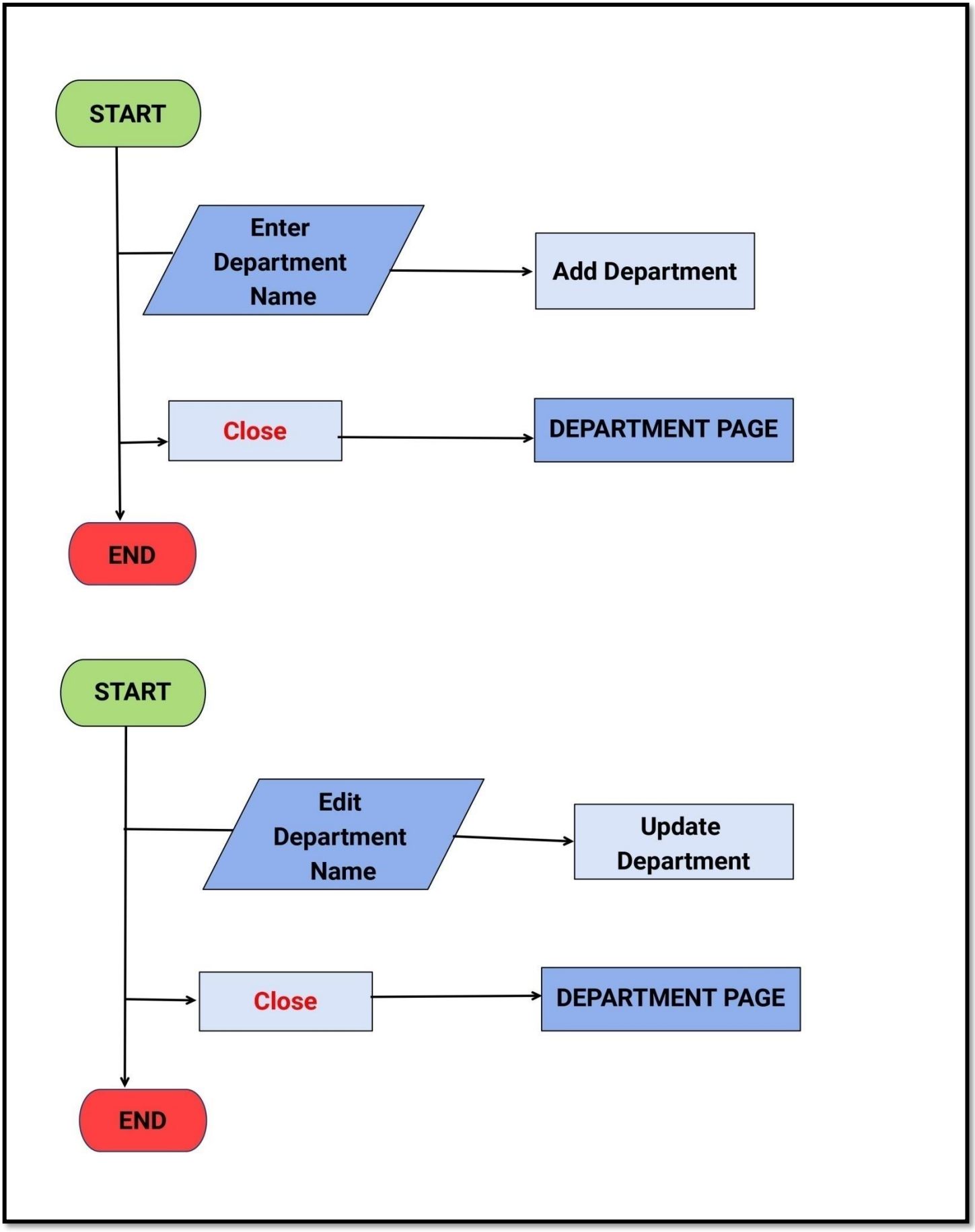
❖ ADD & EDIT DEP MODAL

➢ SCREENSHOT OF ADD & EDIT DEP MODAL



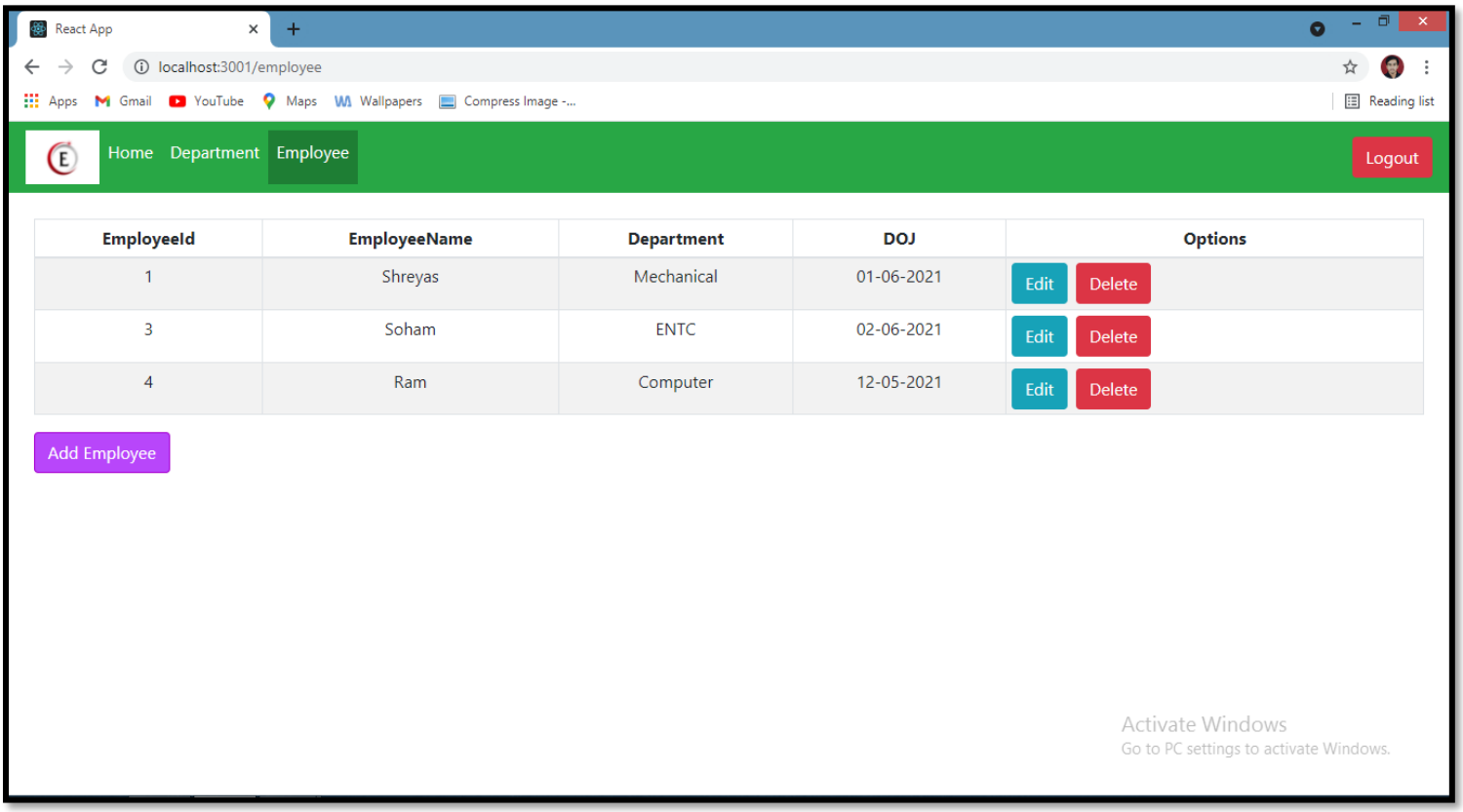


➢ FLOWCHART OF ADD & EDIT DEP MODAL



❖ EMPLOYEE PAGE

➢ SCREENSHOT OF EMPLOYEE PAGE



➢ **This page consists of 3 buttons with their respective tasks.**

I. Add Employee (Click to add new employee)

II. Edit (Click to update existing employee)

III. Delete (Click to delete selected employee)

➢ **It also consists of table with 5 Column.**

I. EmployeeId (new generated Id assign to each)

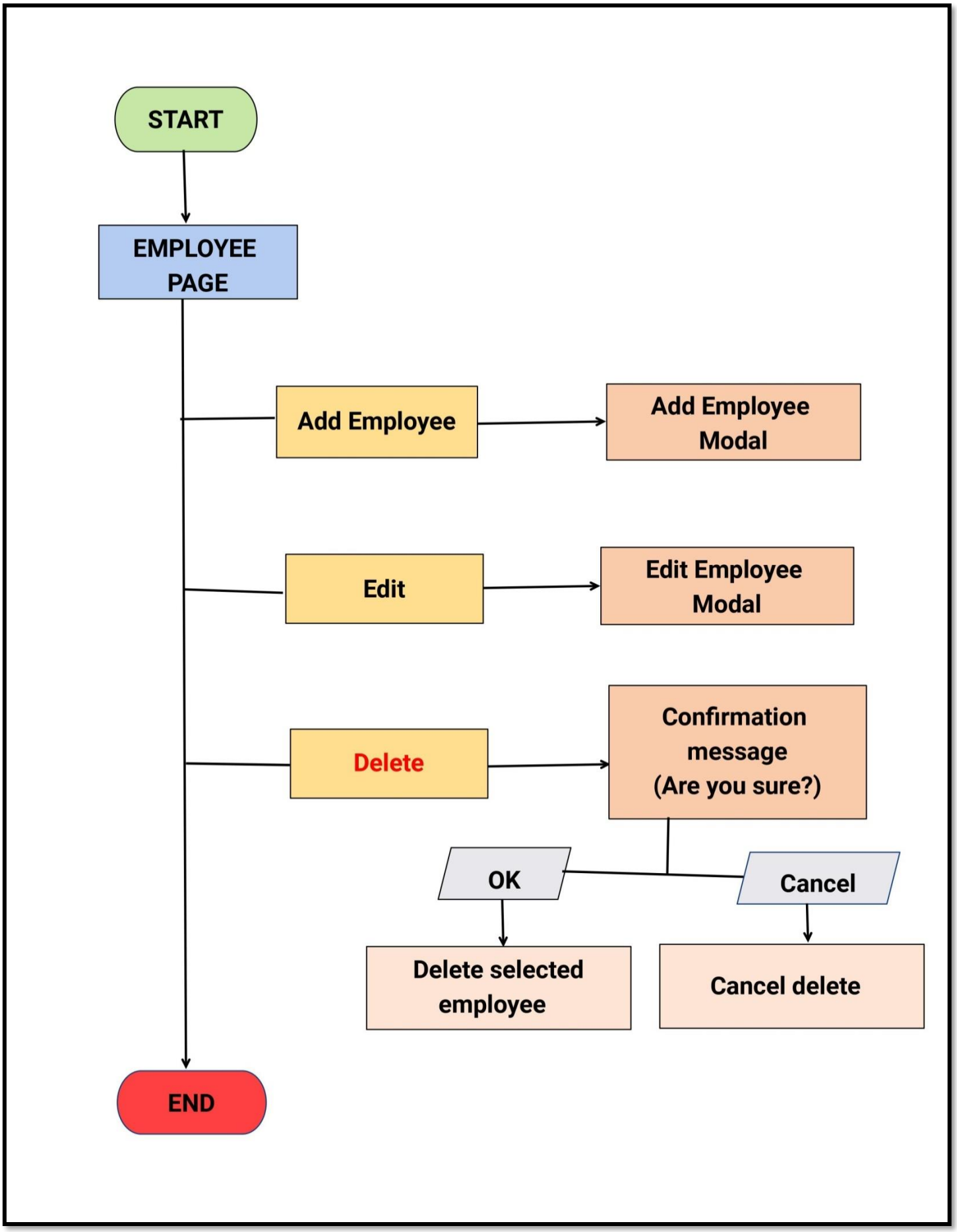
II. EmployeeName (Added Employee name)

III. Department (Select Department name)

IV. DOJ (Select date of joining)

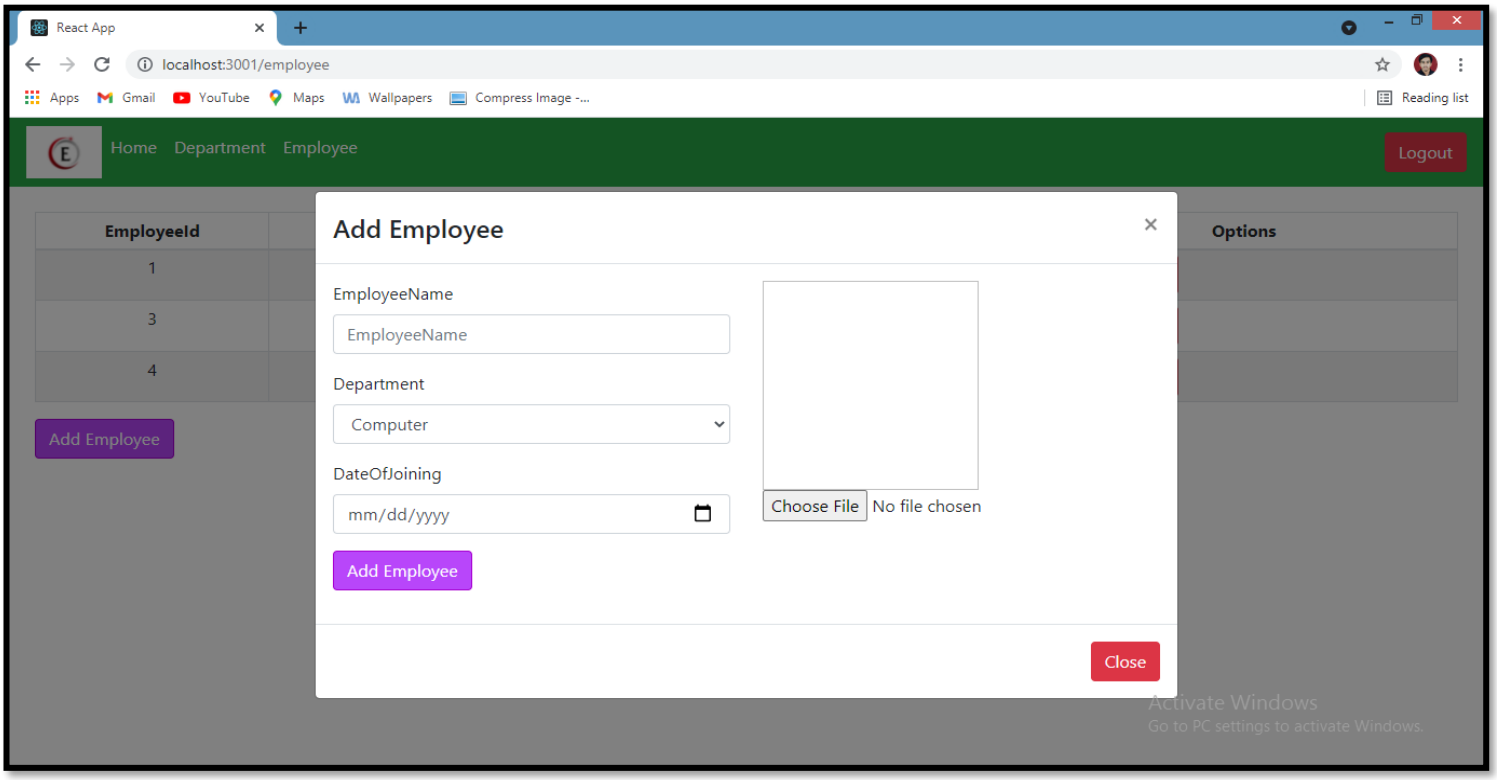
V. Options (Buttons - Edit & Delete)

➢ FLOWCHART OF EMPLOYEE PAGE



❖ ADD EMPLOYEE MODAL

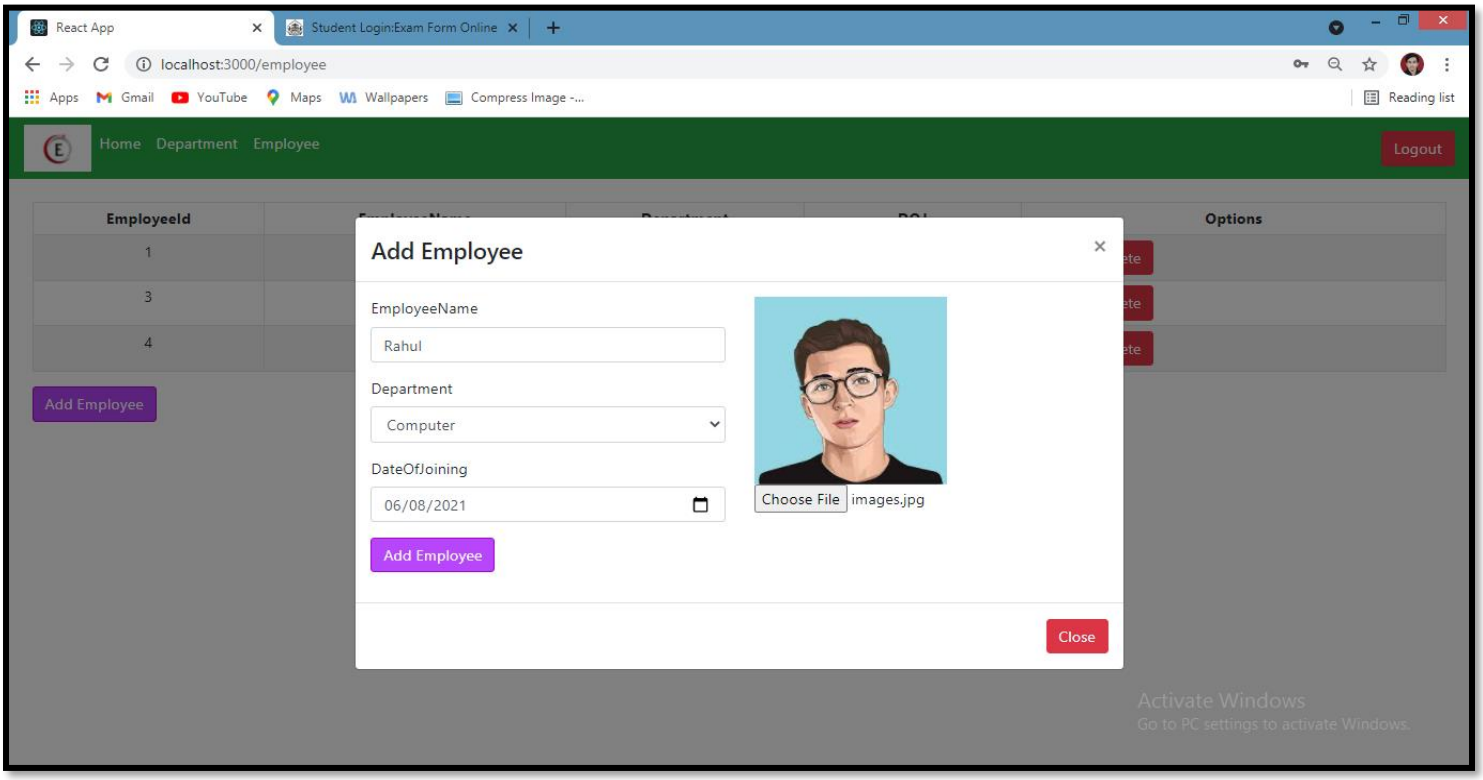
➢ SCREENSHOT OF ADD EMPLOYEE MODAL



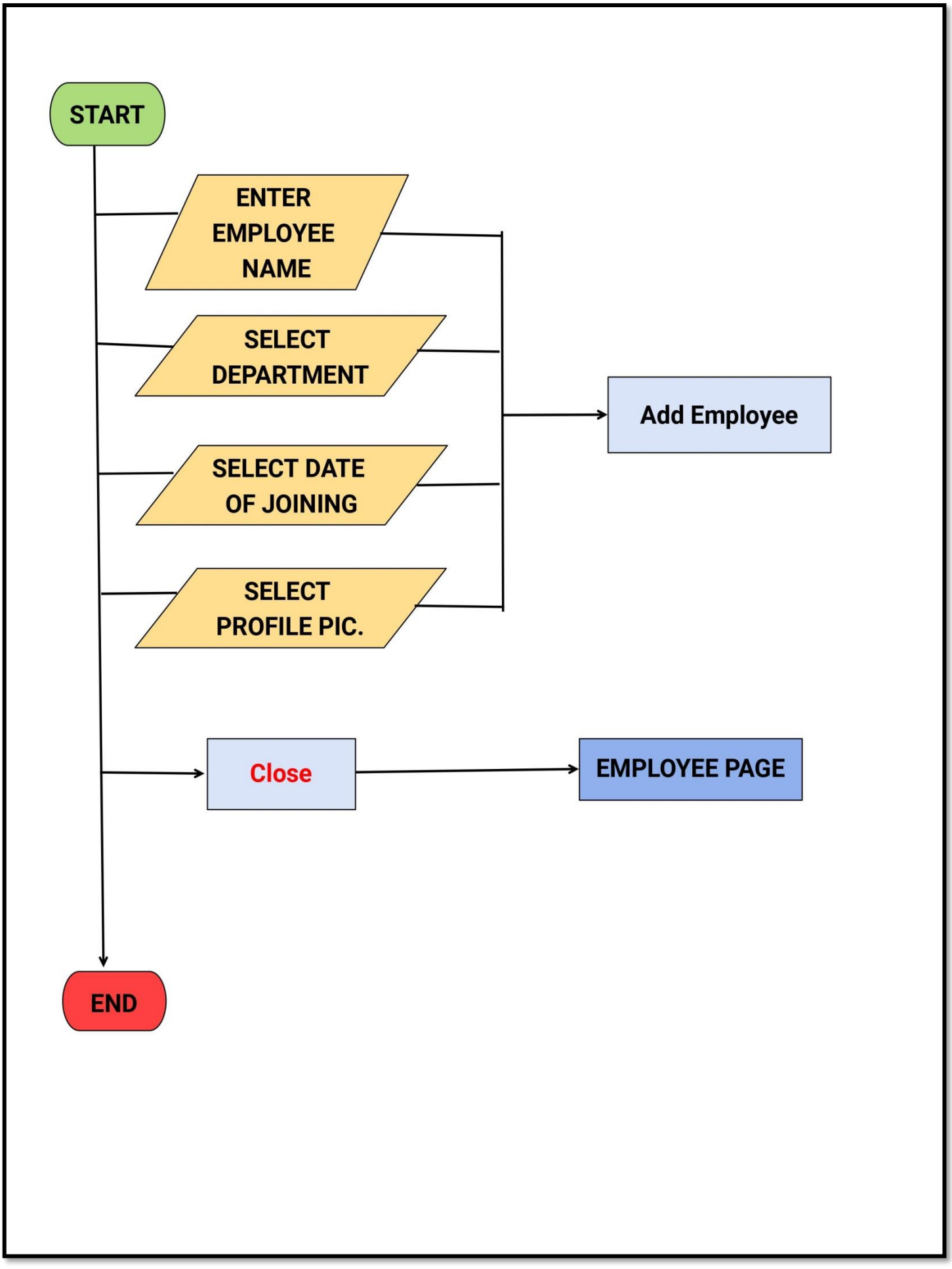
➢ System takes 4 inputs from user

I. EmployeeName (Enter Employee name)
II. Department (Select Department name)
III. DOJ (Select date of joining)
IV. Profile pic. (Add profile pic)

➢ SCREENSHOT AFTER FILLING EMPLOYEE DETAILS

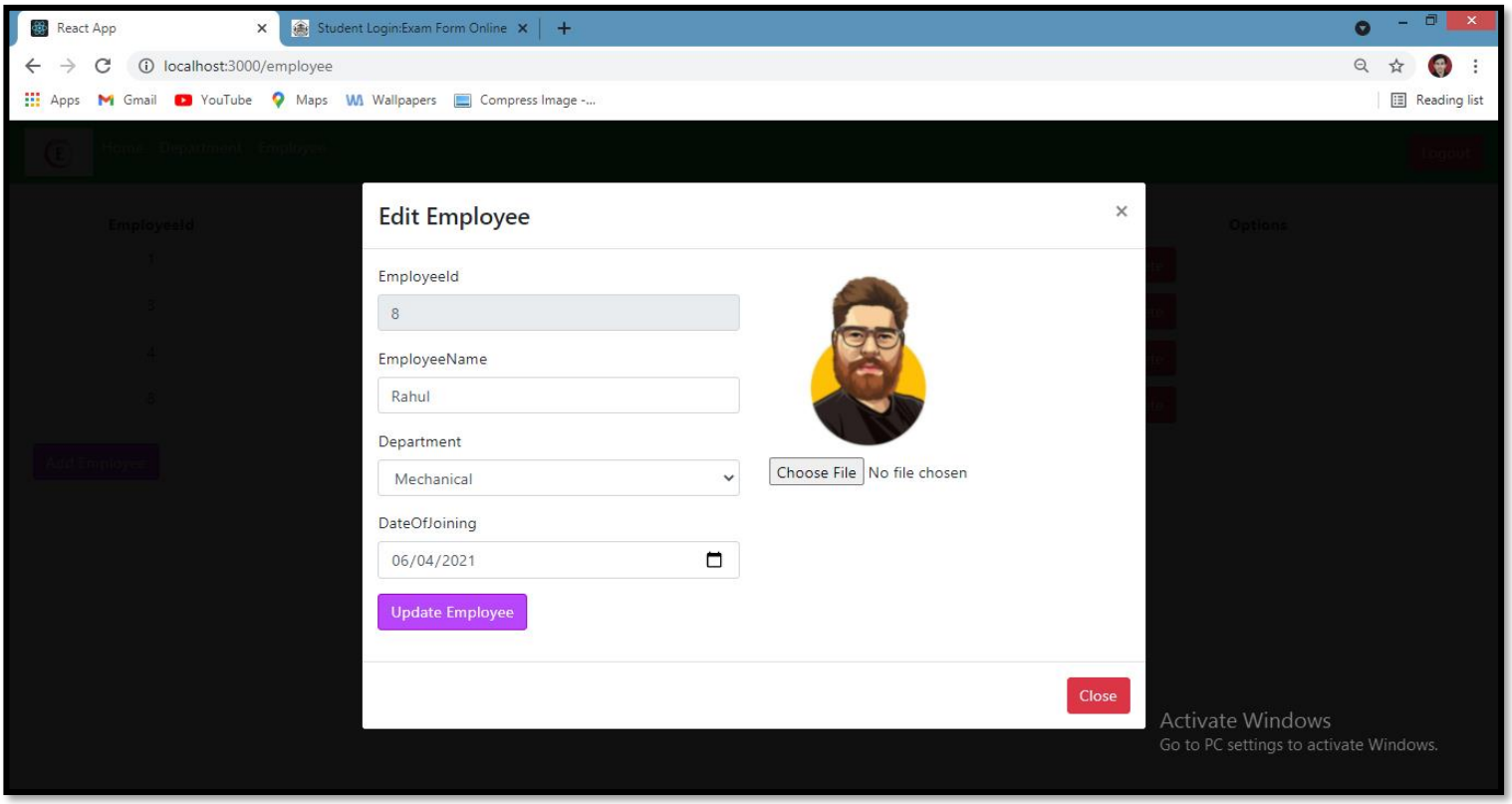


➢ FLOWCHART OF ADD EMPLOYEE MODAL

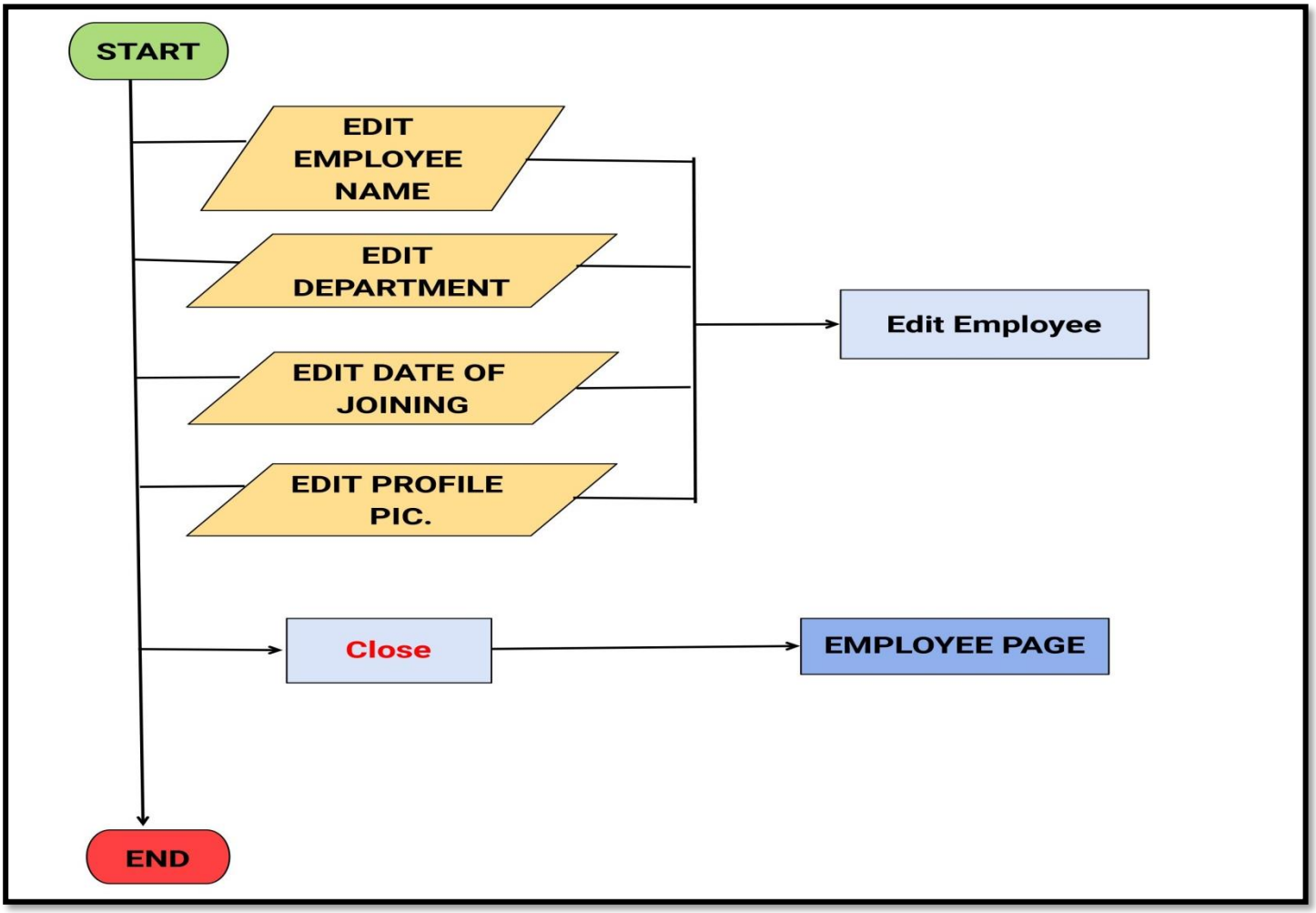


❖ EDIT EMPLOYEE MODAL

➢ SCREENSHOT OF EDIT EMPLOYEE MODAL



➢ Here user can update employee's profile



SOURCE CODE

➢ BACKEND (DJANGO) -

- models.py

```
from django.db import models

# Create your models here.

class Departments(models.Model):
    DepartmentId = models.AutoField(primary_key=True)
    DepartmentName = models.CharField(max_length=100)

class Employees(models.Model):
    EmployeeId = models.AutoField(primary_key=True)
    EmployeeName = models.CharField(max_length=100)
    Department = models.CharField(max_length=100)
    DateOfJoining = models.DateField()
    PhotoFileName = models.CharField(max_length=100)

```

- serializers.py

```
from django.db.models import fields
from django.db.models.base import Model
from rest_framework import serializers
from EmployeeApp.models import Departments, Employees

class DepartmentSerializer(serializers.ModelSerializer):
    class Meta:
        model = Departments
        fields = ('DepartmentId',
                  'DepartmentName')

class EmployeeSerializer(serializers.ModelSerializer):
    class Meta:
        model = Employees
        fields = ('EmployeeId',
                  'EmployeeName',
                  'Department',
                  'DateOfJoining',
                  'PhotoFileName')

```

- views.py

```
rom django.shortcuts import render
from django.views.decorators.csrf import csrf_exempt
from rest_framework.parsers import JSONParser
from django.http.response import JsonResponse

from EmployeeApp.models import Departments,Employees
from EmployeeApp.serializers import EmployeeSerializer,DepartmentSerializer
from django.core.files.storage import default_storage


# Create your views here.
@csrf_exempt
def departmentApi(request,id=0):
    if request.method== 'GET':
        departments = Departments.objects.all()
        departments_serializer = DepartmentSerializer(departments, many=True)
        return JsonResponse(departments_serializer.data, safe=False)

    elif request.method== 'POST':
        department_data=JSONParser().parse(request)
        department_serializer = DepartmentSerializer(data=department_data)
        if department_serializer.is_valid():
            department_serializer.save()
            return JsonResponse("Added Successfully!!",safe=False)
        return JsonResponse("Failed to Add")

    elif request.method=='PUT':
        department_data = JSONParser().parse(request)
        department=Departments.objects.get(DepartmentId=department_data['DepartmentId'])
        department_serializer=DepartmentSerializer(department,data=department_data)
        if department_serializer.is_valid():
            department_serializer.save()
            return JsonResponse("Updated Successfully!!",safe=False)
        return JsonResponse("Failed to Update.", safe=False)

    elif request.method=='DELETE':
        department=Departments.objects.get(DepartmentId=id)
        department.delete()
        return JsonResponse("Deleted Successfully!!", safe=False)

@csrf_exempt
def employeeApi(request,id=0):
    if request.method== 'GET':
        employees = Employees.objects.all()
        employees_serializer = EmployeeSerializer(employees, many=True)
        return JsonResponse(employees_serializer.data, safe=False)

    elif request.method== 'POST':
        employee_data=JSONParser().parse(request)
        employee_serializer = EmployeeSerializer(data=employee_data)
        if employee_serializer.is_valid():
            employee_serializer.save()
            return JsonResponse("Added Successfully!!",safe=False)
```

```
            return JsonResponse("Failed to Add")

        elif request.method=='PUT':
            employee_data = JSONParser().parse(request)
            employee=Employees.objects.get(EmployeeId=employee_data['EmployeeId'])
            employee_serializer=EmployeeSerializer(employee,data=employee_data)
            if employee_serializer.is_valid():
                employee_serializer.save()
                return JsonResponse("Updated Successfully!!",safe=False)
            return JsonResponse("Failed to Update.", safe=False)

        elif request.method=='DELETE':
            employee=Employees.objects.get(EmployeeId=id)
            employee.delete()
            return JsonResponse("Deleted Successfully!!", safe=False)


@csrf_exempt
def SaveFile(request):
    file=request.FILES['myFile']
    file_name = default_storage.save(file.name,file)

    return JsonResponse(file_name,safe=False)
```

- urls.py

```
from django.conf.urls import url
from EmployeeApp import views

from django.conf.urls.static import static
from django.conf import settings

urlpatterns=[
    url(r'^department$',views.departmentApi),
    url(r'^department/([0-9]+)$',views.departmentApi),

     url(r'^employee$',views.employeeApi),
    url(r'^employee/([0-9]+)$',views.employeeApi),

    url(r'^Employee/SaveFile$',views.SaveFile)

]+ static(settings.MEDIA_URL, document_root=settings.MEDIA_ROOT)
```

➢ FRONT-END (REACTJS) -

- App.js

```
import "./App.css";
import { Home } from "./Home";
import { Department } from "./Department";
import { Employee } from "./Employee";
import { Login } from "./Login";
import { Navigation } from "./Navigation";

import { BrowserRouter, Route, Switch } from "react-router-dom";

function App() {
  return (
    <div className="App">
      <BrowserRouter>
        <Switch>
          <Route exact path="/" component={Login} />
          <div>
            <Navigation />

            <Route path="/home" component={Home} />
            <Route path="/department" component={Department} />
            <Route path="/employee" component={Employee} />
          </div>
        </Switch>
      </BrowserRouter>
    </div>
  );
}

export default App;
```

- Login.js

```
import React, { Component } from "react";
import { Col, Container, Row, Form, Button } from "react-bootstrap";
import Login_icon from "./Login_icon.png";
import Login_image from "./Login_image.png";
import "./Login.css";

export class Login extends Component {
  constructor(props) {
    super(props);
    this.onSubmitHandler = this.onSubmitHandler.bind(this);
  }

  onSubmitHandler = (event) => {
    event.preventDefault();
```

```
    if (
      event.target.formBasicEmail.value === "admin" &&
      event.target.formBasicPassword.value === "admin123"
    ) {
      this.props.history.push("/home");
    } else {
      alert("Invalid Credentials..Please retry..!");
    }
  };

  render() {
    return (
      <>
        <Container className="mt-4">
          <Row>
            <Col lg={3} md={6} sm={12} className="text-center">
              <img
                className="icon-img mr-4"
                src={Login_icon}
                alt="icon"
              />
              <Form onSubmit={this.onSubmitHandler}>
                <Form.Group controlId="formBasicEmail">
                  <Form.Control
                    className="w-85 my-3 h-25"
                    type="text"
                    placeholder="Enter username"
                    autoComplete="off"
                  />
                </Form.Group>

                <Form.Group controlId="formBasicPassword">
                  <Form.Control
                    className="w-85  my-3 h-25"
                    type="password"
                    placeholder="Password"
                  />
                </Form.Group>

                <Button variant="primary" type="submit" className="">
                  Login
                </Button>
              </Form>
            </Col>
            <Col lg={9} md={6} sm={12}>
              <h1 className="ml-5 heading">Employee Management System</h1>
              <img
                className="login_image"
                src={Login_image}
                alt="sorry, img not load"
              />
            </Col>
          </Row>
        </Container>
```

```
        </>
      );
    }
  }

```

- login.css

```
.btn-primary{
    color: white;
    background-color: rgb(184, 70, 250) !important;
    border-color: rgb(167, 0, 209) !important;
    margin-right: 65px;
}

.icon-img{
    height: 90px;
    width: 90px;

}

.text-center{
    margin-top: 150px;
}

.login_image{
    width: 600px;
    margin-left: 100px;
    align-items: top;

}

.heading{
    color: rgb(184, 70, 250);

}
.home1{
    align-items: top;
}

```

- Home.js

```
import React, { Component } from "react";
import office from "./PngItem_1918871.png";
import "./MyStyles.css";


export class Home extends Component {
  render() {
    return (
      <div>
        <div className="home1" >
         <h1> This is Home page.</h1>
          <h3>Here we can add Company details, Attendance,
```

```
            important Notices, Dashboard,etc.....</h3>
          </div>
          <div className="mt-3 justify-content-center">
            <img className="Office align-items-center" src={office} alt="logo1" />
          </div>
        </div>
      );
    }
  }
```

- MyStyles.css

```
  logo1{
     width: 70px;
     height: 50px;
  }
  .Office{
     align-items: center;
     width: 600px;
     height: 500px;
      }
  .home1{
     position:relative;
     left:80px;
     top:2px;
  }
```

- Navigation.js

```
  import React, { Component } from "react";
  import { NavLink } from "react-router-dom";
  import { Navbar, Nav } from "react-bootstrap";
  import Logo1 from './Logo1.png' ;
  import './MyStyles.css';

  export class Navigation extends Component {

      render() {
          return (
              <Navbar className="bg-success " expand="lg">

                  <Navbar.Toggle aria-controls="basic-navbar-nav" />
                  <Navbar.Collapse id="basic-navbar-nav">
                      <Nav className="mr-auto">
                          <img className="logo1" src={Logo1} alt="logo" />
                          <NavLink className="d-inline p-2  text-white bg-success" to="/home">
                              Home
                       </NavLink>
                          <NavLink className="d-inline p-2  text-white bg-success" to="/department">
                              Department
                       </NavLink>
                          <NavLink className="d-inline p-2  text-white bg-success" to="/employee">
                              Employee
                       </NavLink>
```

```
                    </Nav>

                    <NavLink to="/"> <button class="btn  my-2 my-sm-0 bg-danger text-
white" type="submit">Logout</button></NavLink>


                </Navbar.Collapse>
            </Navbar>
        )
    }

}
```

- Department.js

```
import React, { Component } from "react";
import { Table } from "react-bootstrap";

import { Button,ButtonToolbar } from 'react-bootstrap';
import { AddDepModal } from './AddDepModal'
import { EditDepModal } from './EditDepModal'

export class Department extends Component {

    constructor(props) {
        super(props);
        this.state = { deps: [], addModalShow: false, editModalShow:false }
    }

    refreshList() {
        fetch(process.env.REACT_APP_API + 'department')
            .then(response => response.json())
            .then(data => {
                this.setState({ deps: data });
            });
    }

    componentDidMount() {
        this.refreshList();
        console.log('compount mount Department.js')
    }

    componentDidUpdate() {
        this.refreshList();
    }

    deleteDep(depid){
        if(window.confirm('Are you sure?')){
            fetch(process.env.REACT_APP_API+'department/'+depid,{
                method:'DELETE',
                header:{'Accept':'application/json',
            'Content-Type':'application/json'}
            })
        }
```

```
    }

    render() {
        const { deps, depid, depname } = this.state;
        let addModalClose = () => this.setState({ addModalShow: false });
        let editModalClose = () => this.setState({ editModalShow: false });
        return (
            <div className="px-4">
                <Table className="mt-4" striped bordered hover size="sm">
                    <thead>
                        <th>DepartmentId</th>
                        <th>DepartmentName</th>
                        <th>Options</th>
                    </thead>
                    <tbody>
                        {deps.map(dep =>
                            <tr key={dep.DepartmentId}>
                                <td>{dep.DepartmentId}</td>
                                <td>{dep.DepartmentName}</td>
                                <td>
<ButtonToolbar>
    <Button className="mr-2" variant="info"
    onClick={()=>this.setState({editModalShow:true,
        depid:dep.DepartmentId,depname:dep.DepartmentName})}>
            Edit
        </Button>

        <Button className="mr-2" variant="danger"
    onClick={()=>this.deleteDep(dep.DepartmentId)}>
            Delete
        </Button>

        <EditDepModal show={this.state.editModalShow}
        onHide={editModalClose}
        depid={depid}
        depname={depname} />
    </ButtonToolbar>
                                </td>
                            </tr>)}

                    </tbody>

                </Table>

                <ButtonToolbar>
                    <Button variant='primary'
                        onClick={() => this.setState({ addModalShow: true })}>
                        Add Department  </Button>

                    <AddDepModal show={this.state.addModalShow} title={'Add Department'}
                        onHide={addModalClose}></AddDepModal>
                </ButtonToolbar>
            </div>
        )
```

```
    }

}
```

- AddDepModal.js

```
import React, { Component } from 'react';
import { Modal, Button, Row, Col, Form } from 'react-bootstrap'


export class AddDepModal extends Component {
    constructor(props) {
        super(props);
        this.handleSubmit = this.handleSubmit.bind(this);
    }

    handleSubmit(event) {
        event.preventDefault();
        fetch(process.env.REACT_APP_API + "department", {
            method: 'POST',
            headers: {
                'Accept': 'application/json',
                'Content-Type': 'application/json'
            },
            body: JSON.stringify({
                DepartmentId: null,
                DepartmentName: event.target.DepartmentName.value
            })
        })
            .then(res => res.json())
            .then((result) => {
                alert(result);
            },
                (error) => {
                    alert('Failed');
                })

    }

    render() {
        return (
            <div className="container">

                <Modal
                    {...this.props}
                    size="lg"
                    aria-labelledby="contained-modal-title-vcenter"
                    centered >
                    <Modal.Header closeButton>
                        <Modal.Title id="contained-modal-title-vcenter">
                            {this.props.title}
                        </Modal.Title>
                    </Modal.Header>
                    <Modal.Body>

```

```
                        <Row>
                            <Col sm={6}>
                                <Form onSubmit={this.handleSubmit}>
                                    <Form.Group controlId="DepartmentName">
                                        <Form.Label>DepartmentName</Form.Label>
                                        <Form.Control type="text" name="DepartmentName" required
                                            placeholder="Departmentname" />
                                    </Form.Group>

                                    <Form.Group>
                                        <Button variant="primary" type="submit">
                                            Add Department
                                        </Button>
                                    </Form.Group>

                                </Form>

                            </Col>
                        </Row>


                    </Modal.Body>

                    <Modal.Footer>
                        <Button variant="danger" onClick={this.props.onHide}>Close</Button>
                    </Modal.Footer>

                </Modal>

            </div>
        )
    }
}
```

- EditDepModal.js

```
import React, { Component } from 'react';
import { Modal, Button, Row, Col, Form } from 'react-bootstrap';


export class EditDepModal extends Component {
    constructor(props) {
        super(props);
        this.handleSubmit=this.handleSubmit.bind(this);
    }

    handleSubmit(event){
        event.preventDefault();
        fetch(process.env.REACT_APP_API+ "department",{
            method:'PUT',
            headers:{
                'Accept':'application/json',
                'Content-Type':'application/json'
            },
            body:JSON.stringify({
```

```
                DepartmentId:event.target.DepartmentId.value,
                DepartmentName:event.target.DepartmentName.value
            })
        })
        .then(res=>res.json())
        .then((result)=>{
            alert(result);
        },
        (error)=>{
            alert('Failed');
        })

    }

    render() {
        return (
            <div className="container">

                <Modal
                    {...this.props}
                    size="lg"
                    aria-labelledby="contained-modal-title-vcenter"
                    centered
                >
                    <Modal.Header closeButton>
                        <Modal.Title id="contained-modal-title-vcenter">
                        Edit Department
                    </Modal.Title>
                    </Modal.Header>
                    <Modal.Body>

                        <Row>
                            <Col sm={6}>
                                <Form onSubmit={this.handleSubmit}>
                                <Form.Group controlId="DepartmentId">
                                        <Form.Label>DepartmentId</Form.Label>
                                        <Form.Control type="text" name="DepartmentId" required
                                        disabled
                                        defaultValue={this.props.depid}
                                            placeholder="DepartmentName" />
                                    </Form.Group>


                                    <Form.Group controlId="DepartmentName">
                                        <Form.Label>DepartmentName</Form.Label>
                                        <Form.Control type="text" name="DepartmentName" required
                                        defaultValue={this.props.depname}
                                            placeholder="DepartmentName" />
                                    </Form.Group>

                                    <Form.Group>
                                        <Button variant="primary" type="submit">
                                            Update Department
                                        </Button>
                                    </Form.Group>
```

```
                                    </Form>

                                </Col>
                            </Row>


                        </Modal.Body>

                        <Modal.Footer>
                            <Button variant="danger" onClick={this.props.onHide}>Close</Button>
                        </Modal.Footer>









                    </Modal>

                </div>
            )
        }
    }
```

- Employee.js

```
    import React, { Component } from "react";
    import { Table } from "react-bootstrap";

    import { Button, ButtonToolbar } from 'react-bootstrap';
    import { AddEmpModal } from './AddEmpModal';
    import { EditEmpModal } from './EditEmpModal';
    import Moment from 'moment';


    export class Employee extends Component {

        constructor(props) {
            super(props);
            this.state = { emps: [], addModalShow: false, editModalShow: false }
        }

        refreshList() {
            fetch(process.env.REACT_APP_API + 'employee')
                .then(response => response.json())
                .then(data => {
                    this.setState({ emps: data });
                });
        }

        componentDidMount() {
            this.refreshList();

        }
```

```
    componentDidUpdate() {
        this.refreshList();
    }

    deleteEmp(empid) {
        if (window.confirm('Are you sure?')) {
            fetch(process.env.REACT_APP_API + 'employee/' + empid, {
                method: 'DELETE',
                header: {
                    'Accept': 'application/json',
                    'Content-Type': 'application/json'
                }
            })
        }
    }

    getFormatedDate(doj) {
        let formatedDate = Moment(doj).format('DD-MM-YYYY');
        return formatedDate;
    }

    render() {
        const { emps, empid, empname, depmt, photofilename, doj } = this.state;
        let addModalClose = () => this.setState({ addModalShow: false });
        let editModalClose = () => this.setState({ editModalShow: false });
        return (
            <div className="px-4">
                <Table className="mt-4" striped bordered hover size="sm">
                    <thead>
                        <th>EmployeeId</th>
                        <th>EmployeeName</th>
                        <th>Department</th>
                        <th>DOJ</th>
                        <th>Options</th>
                    </thead>
                    <tbody>
                        {emps.map(emp =>
                            <tr key={emp.EmployeeId}>
                                <td>{emp.EmployeeId}</td>
                                <td>{emp.EmployeeName}</td>
                                <td>{emp.Department}</td>
                                <td>{this.getFormatedDate(emp.DateOfJoining)}</td>
                                <td>
                                    <ButtonToolbar>
                                        <Button className="mr-2" variant="info"
                                            onClick={() => this.setState({
                                                editModalShow: true,
                                                empid: emp.EmployeeId, empname: emp.EmployeeName, depmt: emp.Department,
                                                photofilename: emp.PhotoFileName, doj: emp.DateOfJoining
                                            })}>
                                            Edit
                                        </Button>
```

```
                                        <Button className="mr-2" variant="danger"
                                            onClick={() => this.deleteEmp(emp.EmployeeId)}>
                                            Delete
                                          </Button>

                                        <EditEmpModal show={this.state.editModalShow}
                                            onHide={editModalClose}
                                            empid={empid}
                                            empname={empname} />
                                    </ButtonToolbar>
                                </td>
                            </tr>)}

                        </tbody>

                    </Table>

                    <ButtonToolbar>
                        <Button variant='primary'
                            onClick={() => this.setState({ addModalShow: true })}>
                            Add Employee  </Button>

                        <AddEmpModal show={this.state.addModalShow} title={'Add Department'}
                            onHide={addModalClose}></AddEmpModal>
                    </ButtonToolbar>
                </div>
            )
        }

    }
```

- AddEmpModal.js

```
    import { data } from 'jquery';
    import React, { Component } from 'react';
    import { Modal, Button, Row, Col, Form, Image } from 'react-bootstrap'



    export class AddEmpModal extends Component {
        constructor(props) {
            super(props);
            this.state = { deps: [] }
            this.handleSubmit = this.handleSubmit.bind(this);
            this.handleFileSelected = this.handleFileSelected.bind(this);
        }

        photofilename = "smile.jpg";
        imagesrc = process.env.REACT_APP_API + this.photofilename;

        componentDidMount() {
            fetch(process.env.REACT_APP_API + 'department')
                .then(response => response.json())
                .then(data => {
                    this.setState({ deps: data });
```

```
            });
    }

    handleSubmit(event) {
        console.log(event);
        event.preventDefault();
        fetch(process.env.REACT_APP_API + "employee", {
            method: 'POST',
            headers: {
                'Accept': 'application/json',
                'Content-Type': 'application/json'
            },
            body: JSON.stringify({
                EmployeeId: null,
                EmployeeName: event.target.EmployeeName.value,
                Department: event.target.Department.value,
                DateOfJoining: event.target.DateOfJoining.value,
                PhotoFileName: this.photofilename

            })
        })
            .then(res => res.json())
            .then((result) => {
                alert(result);
            },
                (error) => {
                    alert('Failed');
                })

    }

    handleFileSelected(event) {
        event.preventDefault();
        this.photofilename = event.target.files[0].name;
        const formData = new FormData();
        formData.append(
            "myFile",
            event.target.files[0],
            event.target.files[0].name
        );
        fetch(process.env.REACT_APP_API + 'Employee/SaveFile', {
            method: 'POST',
            body: formData
        })
            .then(res => res.json())
            .then((result) => {
                this.imageSrc = process.env.REACT_APP_PHOTOPATH + result;
            },
                (error) => {
                    alert('Failed');
                })
    }
```

```
    render() {
        return (
            <div className="container">

                <Modal
                    {...this.props}
                    size="lg"
                    aria-labelledby="contained-modal-title-vcenter"
                    centered
                >
                    <Modal.Header closeButton>
                        <Modal.Title id="contained-modal-title-vcenter">
                            Add Employee
                    </Modal.Title>
                    </Modal.Header>
                    <Modal.Body>

                        <Row>
                            <Col sm={6}>
                                <Form onSubmit={this.handleSubmit}>
                                    <Form.Group controlId="EmployeeName">
                                        <Form.Label>EmployeeName</Form.Label>
                                        <Form.Control type="text" name="EmployeeName" required
                                            placeholder="EmployeeName" autoComplete="off" />
                                    </Form.Group>

                                    <Form.Group controlId="Department">
                                        <Form.Label>Department</Form.Label>
                                        <Form.Control as="select">
                                            {this.state.deps.map(dep =>
                                                <option key={dep.DepartmentId}>{dep.DepartmentName} </option>)}
                                        </Form.Control>
                                    </Form.Group>

                                    <Form.Group controlId="DateOfJoining">
                                        {this.state.dateDMY}
                                        <Form.Label>DateOfJoining</Form.Label>
                                        <Form.Control
                                            type="date"
                                            name="DateOfJoining"
                                            required
                                            placeholder="DateOfJoining"
                                        />
                                    </Form.Group>

                                    <Form.Group>
                                        <Button variant="primary" type="submit">
                                            Add Employee
                                        </Button>
                                    </Form.Group>
```

```
                                </Form>

                            </Col>

                            <Col sm={6}>
                                <Image width="200px" height="200px" src={this.imageSrc} border-radius="50%" />
                                <input onChange={this.handleFileSelected} type="file" />
                            </Col>
                        </Row>


                    </Modal.Body>

                    <Modal.Footer>
                        <Button variant="danger" onClick={this.props.onHide}>Close</Button>
                    </Modal.Footer>
                </Modal>

            </div>
        )
    }
}
```

- EditEmpModal.js

```
import { data } from 'jquery';
import React, { Component } from 'react';
import { Modal, Button, Row, Col, Form, Image } from 'react-bootstrap'
import Moment from 'moment';


export class EditEmpModal extends Component {
    constructor(props) {
        super(props);
        this.state = { deps: [] }
        this.handleSubmit = this.handleSubmit.bind(this);
        this.handleFileSelected = this.handleFileSelected.bind(this);
    }

     photofilename = "smile.jpg";
    imagesrc = process.env.REACT_APP_API + this.photofilename;

    componentDidMount() {
        fetch(process.env.REACT_APP_API + 'department')
            .then(response => response.json())
            .then(data => {
                this.setState({ deps: data });
            });
    }

    handleSubmit(event) {
        console.log(event);
        event.preventDefault();
        fetch(process.env.REACT_APP_API + "employee", {
            method: 'PUT',
```

```
            headers: {
                'Accept': 'application/json',
                'Content-Type': 'application/json'
            },
            body: JSON.stringify({
                EmployeeId:  event.target.EmployeeId.value,
                EmployeeName: event.target.EmployeeName.value,
                Department: event.target.Department.value,
                DateOfJoining: event.target.DateOfJoining.value,
                PhotoFileName: this.photofilename

            })
        })
            .then(res => res.json())
            .then((result) => {
                alert(result);
            },
                (error) => {
                    alert('Failed');
                })

    }

    handleFileSelected(event) {
        event.preventDefault();
        this.photofilename = event.target.files[0].name;
        const formData = new FormData();
        formData.append(
            "myFile",
            event.target.files[0],
            event.target.files[0].name
        );
        fetch(process.env.REACT_APP_API + 'Employee/SaveFile', {
            method: 'POST',
            body: formData
        })
            .then(res => res.json())
           .then((result) => {
               this.imageSrc = process.env.REACT_APP_PHOTOPATH + result;
           },
                (error) => {
                    alert('Failed');
                })
    }



    render() {
        return (
            <div className="container">

                <Modal
                    {...this.props}
                    size="lg"
```

```
                    aria-labelledby="contained-modal-title-vcenter"
                    centered
                >
                    <Modal.Header closeButton>
                        <Modal.Title id="contained-modal-title-vcenter">
                            Edit Employee
                    </Modal.Title>
                    </Modal.Header>
                    <Modal.Body>

                        <Row>
                            <Col sm={6}>
                                <Form onSubmit={this.handleSubmit}>
                                <Form.Group controlId="EmployeeId">
                                        <Form.Label>EmployeeId</Form.Label>
                                        <Form.Control type="text" name="EmployeeName" required
                                            placeholder="EmployeeName" autocomplete="off"
                                            disabled
                                            defaultValue={this.props.empid} />
                                    </Form.Group>

                                    <Form.Group controlId="EmployeeName">
                                        <Form.Label>EmployeeName</Form.Label>
                                        <Form.Control type="text" name="EmployeeName" required
                                            placeholder="EmployeeName" autocomplete="off"
                                            defaultValue={this.props.empname} />
                                    </Form.Group>

                                    <Form.Group controlId="Department">
                                        <Form.Label>Department</Form.Label>
                                        <Form.Control as="select" defaultValue={this.props.dempt}>
                                            {this.state.deps.map(dep =>
                                                <option key={dep.DepartmentId}>{dep.DepartmentName} </option>)}
                                        </Form.Control>
                                    </Form.Group>

                                    <Form.Group controlId="DateOfJoining">
                                        {this.state.dateDMY}
                                        <Form.Label>DateOfJoining</Form.Label>
                                        <Form.Control
                                            type="date"
                                            name="DateOfJoining"
                                            required
                                            placeholder="DateOfJoining"
                                            defaultValue={this.props.doj}
                                        />
                                    </Form.Group>

                                    <Form.Group>
                                        <Button variant="primary" type="submit">
                                            Update Employee
                                        </Button>
                                    </Form.Group>

```

```
                                </Form>

                            </Col>

                            <Col sm={6}>
                                <Image width="200px" height="200px"
                                 src={process.env.REACT_APP_PHOTOPATH + this.props.photofilename} />
                                <input onChange={this.handleFileSelected} type="file" />
                            </Col>
                        </Row>


                    </Modal.Body>

                    <Modal.Footer>
                        <Button variant="danger" onClick={this.props.onHide}>Close</Button>
                    </Modal.Footer>









                </Modal>

            </div>
        )
    }
}
```

❖ CONCLUSION -

- In this report, an information systems development has been presented. It was emphasized on the basic steps, consequently taken during the project's development course as a particular attention was turned to the basic operative functions performed upon the data into the database.
- The report's content comprises the whole task solution, starting from the programming environments have been selected, going through the database, the application's analyze and construction, and finishing with the code-implementation and test-samples, shown separately in Appendix chapters.
- As a future work, some additional stuff could be implemented and integrated into the application code making it much more reliable and flexible; especially what concerns a pay-roll module, for instance.

➢ FUTURE ENHANCEMENT

- Filter employees based on different parameters.
- Trace all employess progress and attendence.
- Grouping employees department wise.
- Creating graph of progress.
- Computerized events & requests management.
- Employee Leave Entry Form.
- Login Authenticate User.
- Create new users to the system accordingly.
- Creating Admin & Employee Dashboard.

❖ BIBLIOGRAPHY

- **https://www.djangoproject.com/**
- **https://reactjs.org/**
- **https://www.sqlite.org/index.html**
- **https://www.django-rest-framework.org/**
- **https://youtu.be/f5ygXQKF6M8**
- **https://youtu.be/RGKi6LSPDLU**
- **https://youtu.be/JxzZxdht-XY**
- **https://youtu.be/5bFxbwjN-Gk**