CSD 3103 Assignment 5

Student Name : Ananta Poudel

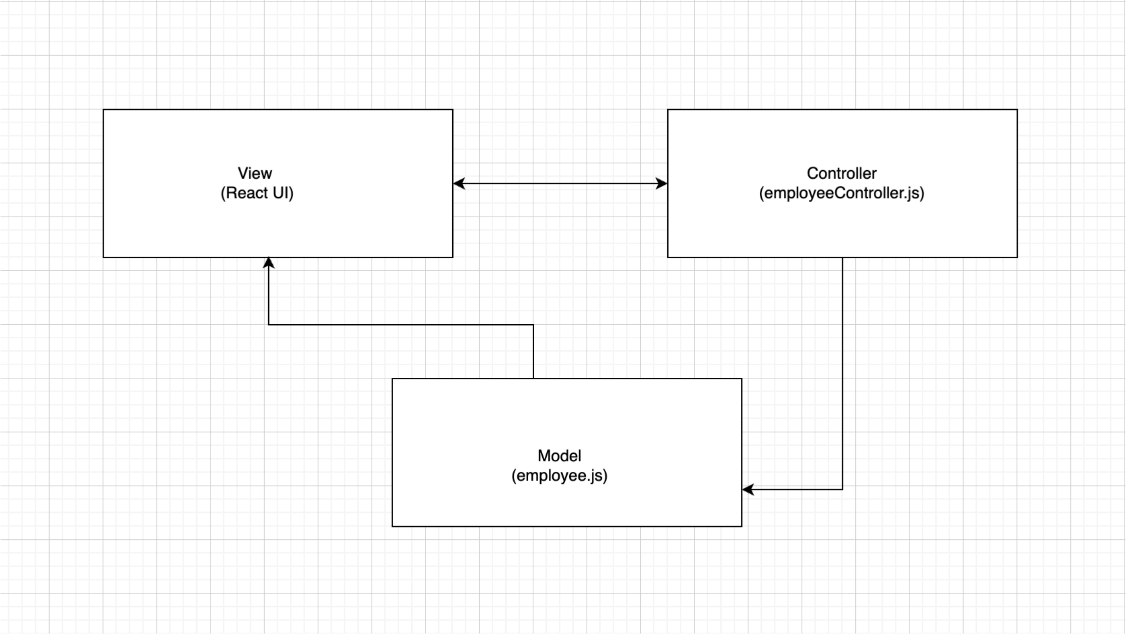
Student ID : C0913139

In this assignment, you are requested to 'Node' and 'Node API' to develop a server-side web-service that stores the information of employees. An employee is identified with first name, last name, ID, salary, working_department and email address. In this assignment, you are required to use an array of JSON objects to serve as database in order to store the information of employees. Specifically, you are asked to:

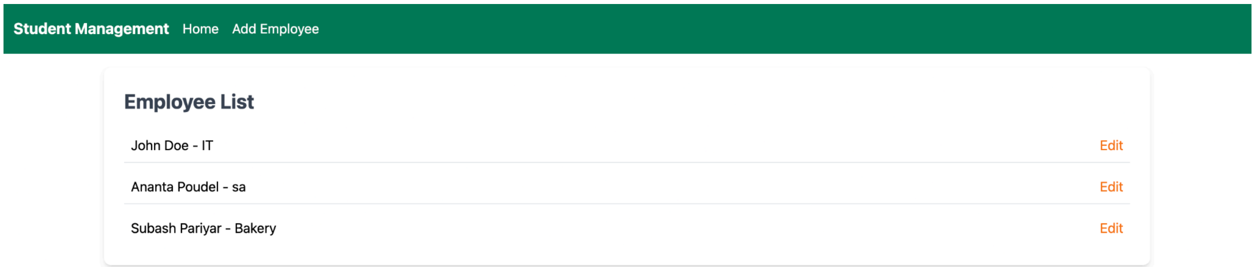
1. Use MVC to design your application and provide a brief explanation about it. Draw a diagram for your MVC model and identify the components that contribute as Model, View and Controller.

   ANSWER
   The Model-View-Controller (MVC) architecture is a design pattern used to separate the concerns of an application into three interconnected components.

- **Model**: Represents the data and the business logic. In your case, the **Model** will store and manipulate the employee information in a JSON array (acting as a simple in-memory database).
- **View**: Represents the user interface (UI), where the data is presented to the user. In this case, the View will be an HTML page with forms to input employee information.
- **Controller**: The Controller handles the incoming requests, processes the data (through the Model), and returns the response (View or data). It acts as the intermediary between the Model and the View.



2. Implement UI to input the employee information and use ‘POST’ method to transmit data from client to server

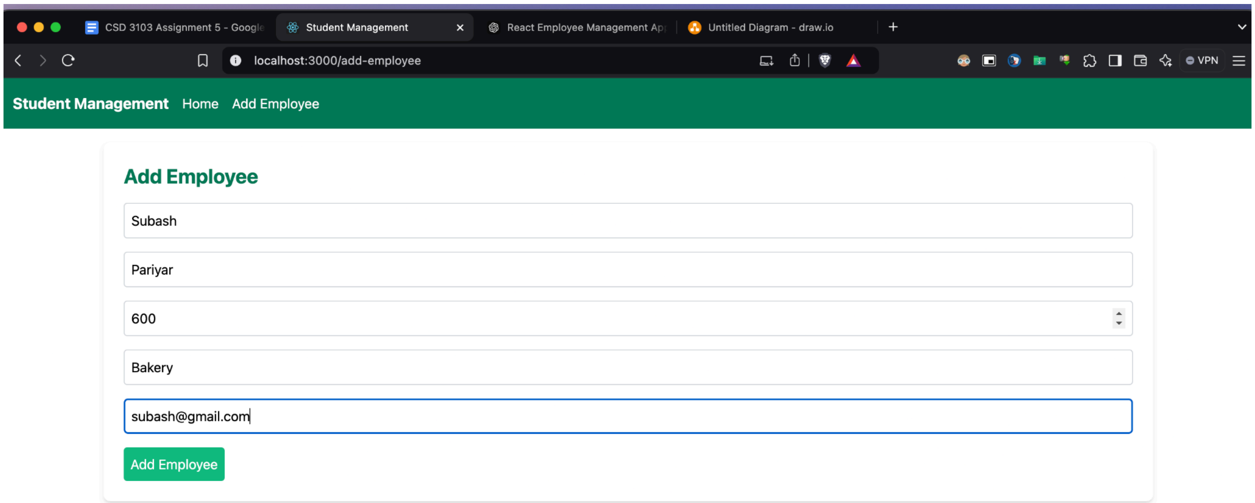


Employee List

John Doe - IT — Edit

Ananta Poudel - sa — Edit

Subash Pariyar - Bakery — Edit


CSD 3103 Assignment 5 - Google
Student Management
React Employee Management Ap
Untitled Diagram - draw.io
localhost:3000/add-employee
VPN
Student Management
Home
Add Employee

Add Employee

Subash

Pariyar

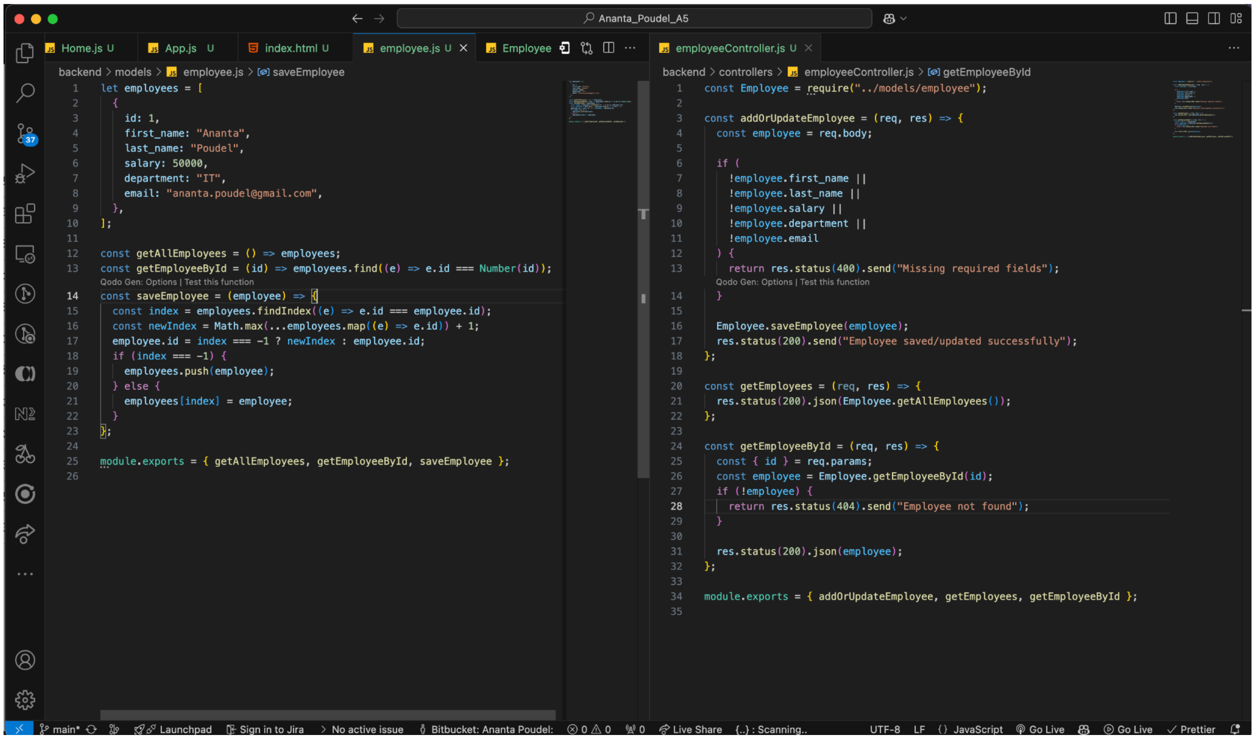
600

Bakery

subash@gmail.com

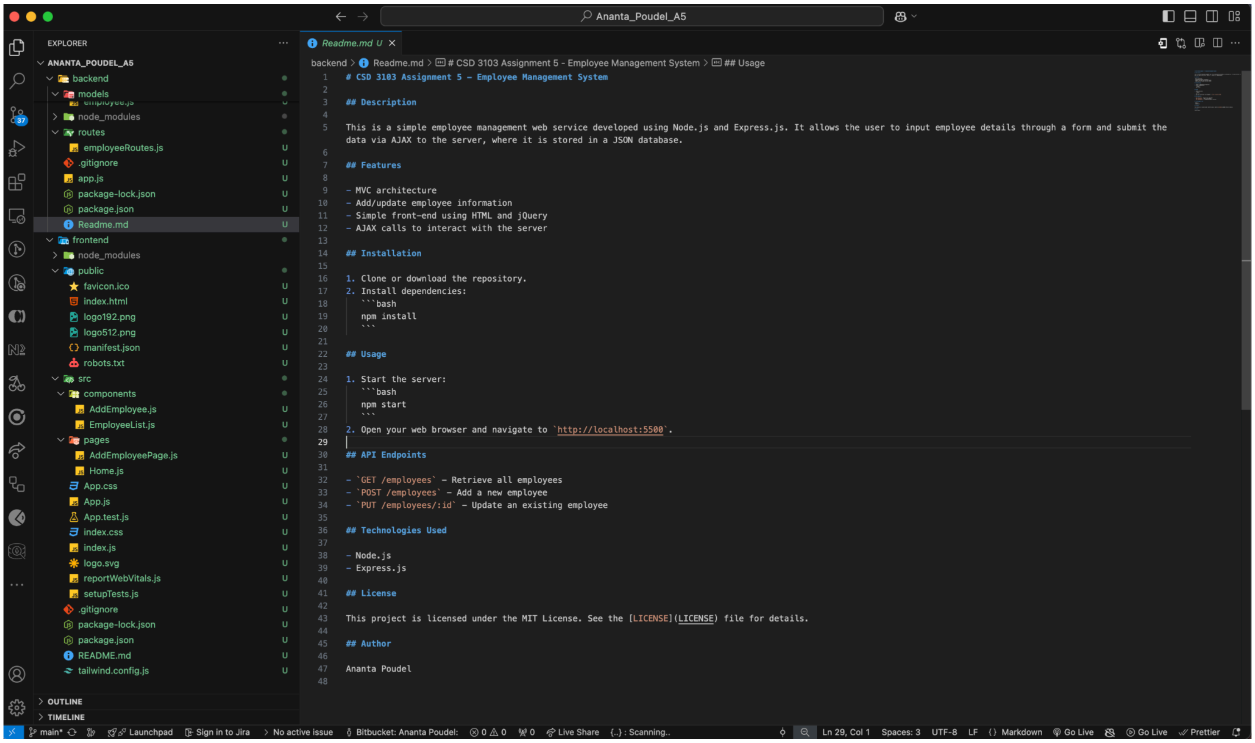
3. Develop a web-service to store the information of the employee in the JSON database. If the employee ID already exists in the database, the information of that employee should get updated in the database. Provide a document to represent the details of your web-service, e.g. end-point-interface, etc. Also, you are requested to submit the

WSDL file for your web-service.


Ananta_Poudel_A5
Home.js U
App.js U
index.html U
employee.js U
Employee
employeeController.js U
backend > models > employee.js > saveEmployee
backend > controllers > employeeController.js > getEmployeeById
main* Launchpad Sign in to Jira No active issue Bitbucket: Ananta Poudel: 0 0 0 Live Share {..} : Scanning.. UTF-8 LF {} JavaScript Go Live Go Live Prettier

```javascript
let employees = [
  {
    id: 1,
    first_name: "Ananta",
    last_name: "Poudel",
    salary: 50000,
    department: "IT",
    email: "ananta.poudel@gmail.com",
  },
];

const getAllEmployees = () => employees;
const getEmployeeById = (id) => employees.find((e) => e.id === Number(id));
Qodo Gen: Options | Test this function
const saveEmployee = (employee) => {
  const index = employees.findIndex((e) => e.id === employee.id);
  const newIndex = Math.max(...employees.map((e) => e.id)) + 1;
  employee.id = index === -1 ? newIndex : employee.id;
  if (index === -1) {
    employees.push(employee);
  } else {
    employees[index] = employee;
  }
};

module.exports = { getAllEmployees, getEmployeeById, saveEmployee };
```

```javascript
const Employee = require("../models/employee");

const addOrUpdateEmployee = (req, res) => {
  const employee = req.body;

  if (
    !employee.first_name ||
    !employee.last_name ||
    !employee.salary ||
    !employee.department ||
    !employee.email
  ) {
    return res.status(400).send("Missing required fields");
  Qodo Gen: Options | Test this function
  }

  Employee.saveEmployee(employee);
  res.status(200).send("Employee saved/updated successfully");
};

const getEmployees = (req, res) => {
  res.status(200).json(Employee.getAllEmployees());
};

const getEmployeeById = (req, res) => {
  const { id } = req.params;
  const employee = Employee.getEmployeeById(id);
  if (!employee) {
    return res.status(404).send("Employee not found");
  }

  res.status(200).json(employee);
};

module.exports = { addOrUpdateEmployee, getEmployees, getEmployeeById };
```



~~~~markdown
# CSD 3103 Assignment 5 - Employee Management System

## Description

This is a simple employee management web service developed using Node.js and Express.js. It allows the user to input employee details through a form and submit the data via AJAX to the server, where it is stored in a JSON database.

## Features

- MVC architecture
- Add/update employee information
- Simple front-end using HTML and jQuery
- AJAX calls to interact with the server

## Installation

1. Clone or download the repository.
2. Install dependencies:
   ```bash
   npm install
   ```

## Usage

1. Start the server:
   ```bash
   npm start
   ```
2. Open your web browser and navigate to `http://localhost:5500`.

## API Endpoints

- `GET /employees` - Retrieve all employees
- `POST /employees` - Add a new employee
- `PUT /employees/:id` - Update an existing employee

## Technologies Used

- Node.js
- Express.js

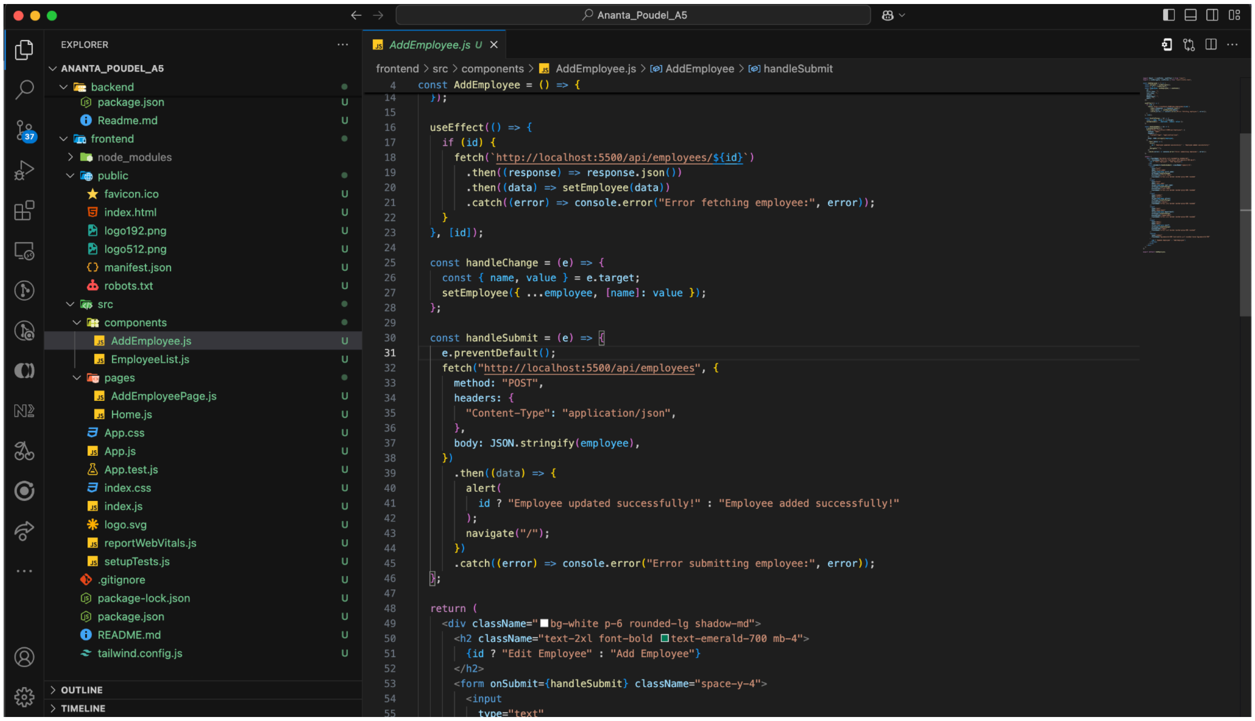
## License

This project is licensed under the MIT License. See the [LICENSE](LICENSE) file for details.

## Author

Ananta Poudel
~~~~

4. In your work, you are requested to use AJAX call in order to transmit the data to the web-service.



```javascript
const AddEmployee = () => {
  });

  useEffect(() => {
    if (id) {
      fetch(`http://localhost:5500/api/employees/${id}`)
        .then((response) => response.json())
        .then((data) => setEmployee(data))
        .catch((error) => console.error("Error fetching employee:", error));
    }
  }, [id]);

  const handleChange = (e) => {
    const { name, value } = e.target;
    setEmployee({ ...employee, [name]: value });
  };

  const handleSubmit = (e) => {
    e.preventDefault();
    fetch("http://localhost:5500/api/employees", {
      method: "POST",
      headers: {
        "Content-Type": "application/json",
      },
      body: JSON.stringify(employee),
    })
      .then((data) => {
        alert(
          id ? "Employee updated successfully!" : "Employee added successfully!"
        );
        navigate("/");
      })
      .catch((error) => console.error("Error submitting employee:", error));
  };

  return (
    <div className="bg-white p-6 rounded-lg shadow-md">
      <h2 className="text-2xl font-bold text-emerald-700 mb-4">
        {id ? "Edit Employee" : "Add Employee"}
      </h2>
      <form onSubmit={handleSubmit} className="space-y-4">
        <input
          type="text"
```

To submit:

Follow these steps to submit your work successfully:

1. Create a folder and name it as firstName_lastName_A5
2. Copy all files that you developed in this folder
3. Write down your information on top of this document and place it in the folder
4. Zip this folder and submit it through Moodle before the deadline