

Assignment

As a Junior Software Engineer, you've been tasked with developing a frontend application using Angular. Your goal is to research and implement data fetching via GraphQL within the Angular project. You can leverage the knowledge you gained in the COMP 3133 course at George Brown College.

The project will be maintained in a Git repository named `studentID_comp3133_assignment2`. If you choose to make the repository private, please add me as a collaborator (username: **pritamworld**).

Your evaluation will focus on how well you organize your source code, as well as how effectively you utilize Angular features such as components, data binding, services, pipes, routing, and security.

Use Assignment I as the backend to develop the frontend for Assignment II. At the end, you will deploy both the backend and frontend to a cloud platform of your choice, such as Heroku, Vercel, or Cyclic. Be sure to update the backend as needed to accommodate the frontend requirements.

Frontend with Angular

1. Application Setup and GitHub repository

- Initialize a Angular app named in the specified format (`studentID_comp3133_assignment2`) using `ng new studentID_comp3133_assignment2`. Then, *set up your GitHub repository with the same name and commit/push your changes regularly with descriptive messages.*
- *NOTE: if you are using docker-compose then make a single folder to keep the frontend and backend. For example, Organize the project structure in a single folder if using Docker:*

```
studentID_comp3133_assignment/  
├── docker-compose.yml  
├── frontend/  
└── backend/
```

2. Routing and Navigation

- Use [Router](#) for screen navigation:
 - **Login**
 - **Signup**
 - **Employee components**

3. Login and Signup Screens

- Implement login and signup screens with form validations.
- Use [Reactive or template forms](#) components to manage form input states.

4. Session Management

- Store a user session token in [service or dependency injection](#) to persist user authentication.

5. Employee Management

- After login, navigate users to an **Employee List** screen displaying all employees in a tabular format.
- Provide CRUD operations using GraphQL:
 - **Add Employee:** A form to add new employees along with profile picture.
 - **View Details:** A screen/modal displaying specific employee details.
 - **Update Information:** A form for editing employee information.
 - **Delete Employee:** A button to remove an employee record.

6. Search and File upload Functionality

Add employee search criteria for department or position and display them in the appropriate table format. This feature will allow users to search for employees based on department or position, showcasing your ability to handle dynamic queries and responsive UI. (*Add new GraphQL API to backend if needed*)

7. User Experience

- Focus on professional UI/UX design:
 - Use Material-UI, Bootstrap, or custom CSS for styling.
 - Make sure components are responsive.

8. Logout and Redirect

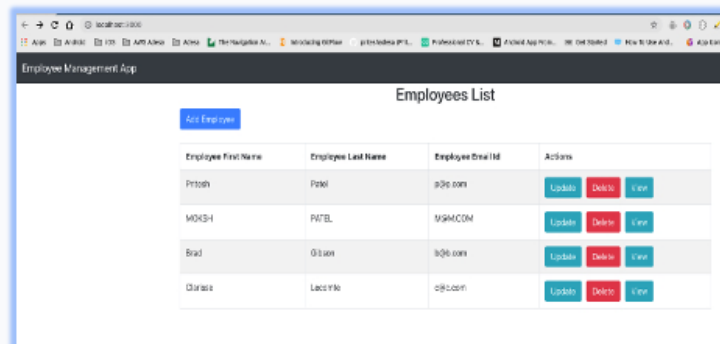
- Implement a logout button to clear user sessions and redirect users to the login page.

ASSIGNMENT

Reference Sample Screen Designs

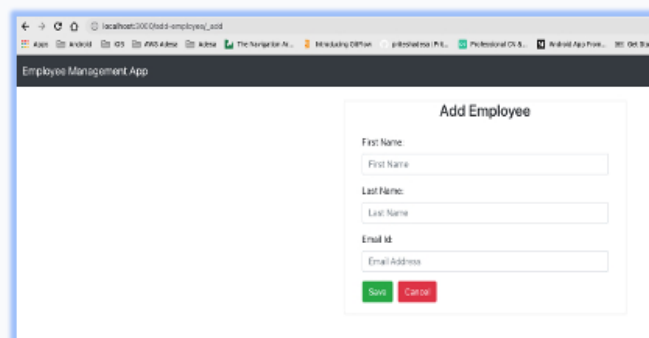
Sample Design Screens for reference only. Based on the fields of user and employee please update the screens. Make sure to design your own search result screen

1. List all employees after login



Employee First Name	Employee Last Name	Employee Email ID	Actions
Pritish	Patel	p@e.com	<button>Update</button> <button>Delete</button> <button>View</button>
MOHSHI	PATIL	MPM@COM	<button>Update</button> <button>Delete</button> <button>View</button>
Brad	Olson	bo@e.com	<button>Update</button> <button>Delete</button> <button>View</button>
Charles	Leach	cl@e.com	<button>Update</button> <button>Delete</button> <button>View</button>

2. On click of "Add Employee" add a new employee by calling API and Delete the employee record when you press the delete button



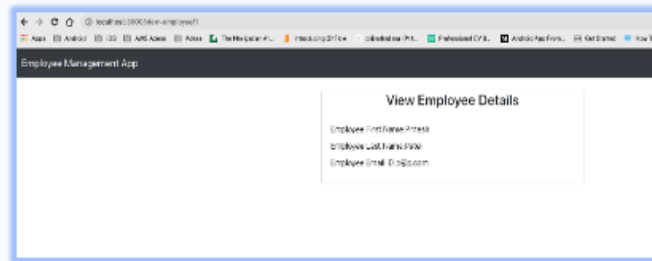
First Name

Last Name

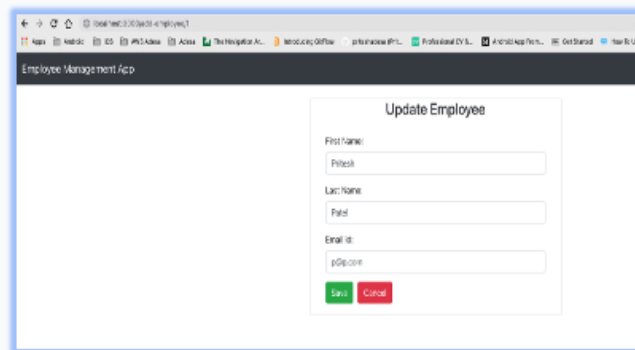
Email Address

Save Cancel

3. View Employee details when you click on the View button



4. Update employee information when you click the Update button



Evaluation Criteria

- **Part I (10%):** Successful deployment of the backend or creation of a Docker Compose file or on any cloud platform. Failure to do so results in a zero.
- **Part II (80%):** Each correctly implemented screen must follow professional design standards. Errors or incomplete features will result in a zero.
- **GitHub Repository (10%):** Proper naming conventions and valid commits are essential. Failure to submit the repository correctly will result in a zero.
- **Validation:** Ensure that your application displays appropriate error messages for all validations.

Sr. No.	Component	Points
1	Deploy backend to Heroku/Vercel/Cyclic or OTHER	05
2	Deploy frontend to Heroku/Vercel/Cyclic or OTHER	05
3	GitHub repository with valid commits and readme file. Screenshots submission for validation on D2L	05
4	Working Signup component, Login component and Logout with GraphQL	15
5	List all Employee component with good design and theme	10
6	Add New Employee screen with good design and theme and with all validation messages along with picture upload	10
7	View and Update Employee component with good design and theme and with all validation messages	10
8	Search employee by department or position	10
9	Delete Employee	10
10	Accepted UI/UX using material design OR bootstrap, etc.	10
11	Make use of Service, Pipes, directives, forms and Routing	10



Previous
Reference Sample Screen Designs

Next
Submission



Submission

Additional Requirements and Submission

- **Submit GitHub repository links:** Set up separate GitHub repositories for the backend and frontend projects, with descriptive commit messages or a single repository with both frontend and backend.

Submit links to your GitHub public repositories for both the backend and frontend applications and links to hosting on the cloud platform.

- **Validation and Error Handling:** Add appropriate error messages for input validations.
- **Put all Screenshots in one single file with an appropriate title**
 - MongoDB data (1 screenshot).
 - GraphQL API tests with Postman (5-8 screenshots).
 - Frontend CRUD operations (5-8 screenshots).
 - Search Screen (2-3 screenshots)
- **ZIP Submission:** **Remove** `node_modules` folders before zipping and upload to D2L.

No extension or last submission will be allowed. Late submission will be awarded ZERO points