SE/COM S 3190 - Construction Of User Interfaces
Assignment 4

Total Points: 100
Published On: April 28, 2025, 1:00 PM CST
Due Date: May 4, 2025, 11:59 PM CST

Contents

Prepared by: ***Pranava Sai Maganti***

1 Overview

The goal of this assignment is to develop the **backend functionality** for the *CycloneHR* application, whose user interface (UI) has been inspired by Workday at Iowa State University. Students will implement backend services using **Node.js** (Express framework) and **MongoDB** as the database.

2 Provided Materials

- Complete **frontend** code (already developed and provided).
- Backend folder structure with empty files:
  - `server.js` (empty file)
  - `routes/leaveRequests.js` (empty file)
  - `routes/jobApplications.js` (empty file)
- Two JSON files containing sample data:
  - `LeaveRequests.json`
  - `JobApplications.json`

Important: You must use the exact **collection names** and **field names** as listed in the provided JSON files. TA grading will be based on matching field and collection names exactly.

3 Setup and Configuration

- **Database Name:** `CycloneHR`
- **Collections to Create:**
  - `LeaveRequests`
  - `JobApplications`

4 Tasks to Complete

1. Set up a MongoDB database named `CycloneHR`.
2. Implement the following APIs inside the empty backend structure:
   - Leave Requests API (CRUD operations)
   - Job Applications API (CRUD operations)
3. Create a working `server.js` file that:
   - Imports both feature routes.
   - Mounts them under a common base path (`/api`).
   - Listens on `http://localhost:8080`.

5 Endpoints to Implement

Leave Requests (`routes/leaveRequests.js`)

Endpoint	**Description**
`GET /leaveRequests`	Retrieve all leave requests.
`GET /leaveRequests/:employeeId`	Retrieve leave requests for a specific employee.
`POST /leaveRequests`	Create a new leave request.
`PUT /leaveRequests/:leaveId`	Update an existing leave request.
`DELETE /leaveRequests/:leaveId`	Delete a leave request.

Job Applications (`routes/jobApplications.js`)

Endpoint	**Description**
`GET /jobApplications`	Retrieve all job applications.
`GET /jobApplications/:jobId`	Retrieve job applications for a specific job ID.
`POST /jobApplications`	Create a new job application.
`PUT /jobApplications/:applicationId`	Update an existing job application.
`DELETE /jobApplications/:applicationId`	Delete a job application.

6 How Mounting Works (Simple Explanation)

In Express, **mounting** refers to attaching a group of routes (known as a router) to a specific base URL path. Once mounted, all the endpoints defined in that router will automatically inherit the base path.

For example, if we mount the leave requests router at `/api/leaveRequests`, then any route defined inside the router (such as `GET /`) will actually be available at `/api/leaveRequests/` on the server.

Similarly, if we mount the job applications router at `/api/jobApplications`, its routes will be available at `/api/jobApplications/`.

Mounting helps in keeping the project **organized and modular**. Instead of defining all routes directly inside the `server.js` file, we separate them into feature-specific files (routers) and attach them at a logical base path.

7 Technical Details

- Use **Express.js** as the server framework.
- Use **MongoDB** to store data for `LeaveRequests` and `JobApplications` collections.
- Use the exact field names and collection names provided in the JSON files.
- CORS and Body-Parser middleware must be properly configured.

- Server must run at `http://localhost:8080`.

8 Task Distribution

Each team must divide the backend implementation work as follows:

- **Team Member 1:** Implement the `LeaveRequests` feature.
- **Team Member 2:** Implement the `JobApplications` feature.

Both team members are responsible for ensuring that the final backend integrates correctly through `server.js` and can run successfully without errors.

Important: While tasks are divided for development, grading will consider the overall functionality of the combined backend.

9 GitLab Instructions

Branch Naming Conventions

- Each team member must work on a separate feature branch.
- Branch names must follow the convention: `feature/<feature-name>/<your-first-name>`.
- Examples:
  - `feature/leaveRequests/pranava`
  - `feature/jobApplications/jabir`

Code Review and Merging Process

- Each member must push their code regularly to their respective feature branch.
- Before merging into the `main` branch:
  - Create a **Merge Request** (MR) on GitLab.
  - Assign the Merge Request to your teammate for review.
  - Your teammate must review and approve your code before it can be merged.
  - After approval, your teammate will merge the feature branch into `main`.
- Final backend integration should happen on the `main` branch.

Important Guidelines

- Ensure your `main` branch is always in a runnable state.
- Avoid direct commits to `main`; always use Merge Requests.
- Keep your feature branches updated with the latest changes from `main` by periodically pulling and merging.
- Once a branch is merged, **do not delete the branch**.

10 Submission Instructions

- Push your complete backend code to the `backend/` folder in your team's GitLab repository.
- **Do not modify** the provided frontend code.
- Your GitLab repository must include the following files:
  - `server.js`
  - `routes/leaveRequests.js`
  - `routes/jobApplications.js`
- TAs will pull your backend code directly from GitLab for grading.
- No separate ZIP file submission is required.
- Submit the GitLab repository URL on Canvas to complete your submission.

11 Grading

Criteria	**Points**
Correct setup of MongoDB database (`CycloneHR` with `LeaveRequests` and `JobApplications` collections)	10
Feature 1: LeaveRequests (5 endpoints, 8 points each)	40
Feature 2: JobApplications (5 endpoints, 8 points each)	40
Proper integration of both routes into `server.js` (mounting at `/api`)	10
Total	**100 Points**

12 Important Notes

- **No late submissions** will be accepted.
- Students are expected to manually seed their database if necessary.
- Only **Node.js (Express)** and **MongoDB** are allowed.
- Ensure that your server runs without crashes and all API endpoints are functional.

Good luck!

Prepared by: ***Pranava Sai Maganti***