CS6471: Advanced Software Engineering Lab

TEST CASES

For

Road Repair and Tracking System

Prepared by: **Archisman Chakraborty(224CS1004)**
Submitted to: **Dr. Judhistir Mahapatro**

Date: **17th November, 2024**

National Institute of Technology, Rourkela

Contents

1. Introduction

The *Road Repair and Tracking System* is designed to streamline the process of tracking and managing road repairs within a city. This system caters to multiple user roles, including Clerks, Supervisors, City Administrators, and the Mayor. Each role has specific responsibilities and access permissions, ensuring the effective allocation and utilization of city resources. The system facilitates the efficient registration of complaints, assignment of tasks, management of resources, and oversight of repair work, while also providing comprehensive statistics and utilization metrics for better decision-making.

2. Endpoints Overview

The API is organized to serve various functionalities for different user roles:

- **Clerk Endpoints:**
  - **Register Complaint:** Allows clerks to register a new road repair complaint.
  - **Complaint List By Area:** Fetches a list of complaints specific to an area.
  - **All Complaints:** Retrieves all registered complaints.
- **Supervisor Endpoints:**
  - **Review a Request:** Enables supervisors to review and update a complaint's status.
  - **Add Estimate:** Supervisors can add resource and material estimates for a complaint.
  - **Get Schedule By ID:** Fetches the schedule details for a specific complaint.
  - **Update Schedule:** Allows supervisors to update the status of a schedule.
- **City Administrator Endpoints:**
  - **View All Resources:** Lists all available resources.
  - **View Resource By ID:** Fetches details of a specific resource.
  - **Add New Resource:** Adds a new resource to the system.
  - **Delete Resource By ID:** Deletes a specific resource.
  - **Update Resource By ID:** Updates information about a specific resource.
  - **View All Materials:** Lists all available materials.
  - **View Material By ID:** Fetches details of a specific material.
  - **Add New Material:** Adds a new material to the system.
  - **Delete Material By ID:** Deletes a specific material.
  - **Update Material By ID:** Updates information about a specific material.
- **Mayor Endpoints:**

- **Repairs In a Period:** Retrieves statistics for repairs completed within a specified period.
- **Outstanding Repairs:** Lists all outstanding repair tasks.
- **Resource Utilizations:** Provides statistics on resource utilization over a given time frame.

- **Authentication:**
    - **Login:** Authenticates users and returns a token that includes role information for role-based access.

3. Postman Collection and Setup

To facilitate testing and interaction with the API, a Postman collection is provided, which includes pre-configured requests for all the endpoints mentioned above.

1. **Import the Collection:**
    - Download and import the provided Postman collection Road Repair and Tracking System.postman_collection.json.
    - Open Postman and use the "Import" button to load the collection.
2. **Environment Setup:**
    - Configure environment variables such as baseURL, authPath, complaintPath, resourcesPath, materialsPath, estimatePath, and statisticsPath.
    - Ensure that authToken is set to receive the token dynamically after a successful login.
3. **Using the Collection:**
    - **Authentication:** Start by using the *Login* request to authenticate and generate a token. The token will automatically be set as the authToken environment variable.
    - **Subsequent Requests:** Use other requests such as *Register Complaint*, *Add Estimate*, or *Repairs In a Period* as needed. The collection includes test scripts to validate the responses.

4. Postman Test Scripts

4.1. Register Complaint

Actual Script:

```
pm.test("Status code is 200", function () {
   pm.response.to.have.status(200);
});

pm.test("Response contains status and message", function () {
   const jsonData = pm.response.json();
   pm.expect(jsonData).to.have.property("status");
   pm.expect(jsonData).to.have.property("message");
});
```

Negative Check Script:

```
pm.test("Status code is not 200", function () {
   pm.response.to.not.have.status(200);
});

pm.test("Response does not contain unexpected properties", function () {
   const jsonData = pm.response.json();
   pm.expect(jsonData).to.not.have.property("unexpectedProperty");
});
```

4.2. Complaint List By Area

Actual Script:

```
pm.test("Status code is 200", function () {
   pm.response.to.have.status(200);
});

pm.test("Response is an array", function () {
   const jsonData = pm.response.json();
   pm.expect(jsonData).to.be.an("array");
});
```

<u>Negative Check Script:</u>

```
pm.test("Status code is not 200", function () {
   pm.response.to.not.have.status(200);
});

pm.test("Response is not an empty array", function () {
   const jsonData = pm.response.json();
   pm.expect(jsonData).to.not.be.empty;
});
```

4.3. All Complaints

<u>Actual Script:</u>

```
pm.test("Status code is 200", function () {
   pm.response.to.have.status(200);
});

pm.test("Response is an array", function () {
   const jsonData = pm.response.json();
   pm.expect(jsonData).to.be.an("array");
});
```

<u>Negative Check Script:</u>

```
pm.test("Status code is not 200", function () {
   pm.response.to.not.have.status(200);
});

pm.test("Response is not an empty array", function () {
   const jsonData = pm.response.json();
   pm.expect(jsonData).to.not.be.empty;
});
```

4.4. Add New Material

Actual Script:

```
pm.test("Status code is 200", function () {
    pm.response.to.have.status(200);
});

pm.test("Response contains correct properties", function () {
    const jsonData = pm.response.json();
    pm.expect(jsonData).to.have.property("id");
    pm.expect(jsonData).to.have.property("type");
    pm.expect(jsonData).to.have.property("description");
    pm.expect(jsonData).to.have.property("inventory");
    pm.expect(jsonData).to.have.property("cost");
});

pm.environment.set("materialId", pm.response.json().id);
```

Negative Check Script:

```
pm.test("Status code is not 200", function () {
    pm.response.to.not.have.status(200);
});

pm.test("Response does not contain missing properties", function () {
    const jsonData = pm.response.json();
    pm.expect(jsonData).to.not.have.property("missingProperty");
});
```

4.5. Delete Material By Id

Actual Script:

```
pm.test("Status code is 204", function () {
   pm.response.to.have.status(204);
});
```

Negative Check Script:

```
pm.test("Status code is not 204", function () {
   pm.response.to.not.have.status(204);
});

pm.test("Response should not have body", function () {
   pm.expect(pm.response.text()).to.equal("");
});
```

4.6. Update Material By Id

Actual Script:

```
pm.test("Status code is 200", function () {
   pm.response.to.have.status(200);
});

pm.test("Response properties are updated", function () {
   const jsonData = pm.response.json();
   pm.expect(jsonData.inventory).to.equal(100); // Example check
   pm.expect(jsonData.cost).to.equal(50.0); // Example check
});
```

Negative Check Script:

```
pm.test("Status code is not 200", function () {
   pm.response.to.not.have.status(200);
});

pm.test("Response properties are not incorrect", function () {
   const jsonData = pm.response.json();
   pm.expect(jsonData.inventory).to.not.equal(-1);
   pm.expect(jsonData.cost).to.not.equal(-50.0);
});
```

4.7. Authentication

Actual Script:

```
pm.test("Status code is 200", function () {
   pm.response.to.have.status(200);
});

pm.test("Token is present in response", function () {
   const jsonData = pm.response.json();
   pm.expect(jsonData).to.have.property("token");
});

pm.environment.set("token", pm.response.json().token);
```

Negative Check Script:

```
pm.test("Status code is not 200", function () {
   pm.response.to.not.have.status(200);
});

pm.test("Token should not be missing or empty", function () {
   const jsonData = pm.response.json();
   pm.expect(jsonData.token).to.not.be.undefined;
   pm.expect(jsonData.token).to.not.be.empty;
});
```

4.8. Repairs In a Period

Actual Script:

```
pm.test("Status code is 200", function () {
   pm.response.to.have.status(200);
});

pm.test("Response is an array", function () {
   const jsonData = pm.response.json();
   pm.expect(jsonData).to.be.an("array");
});
```

Negative Check Script:

```
pm.test("Status code is not 200", function () {
   pm.response.to.not.have.status(200);
});

pm.test("Response is not empty", function () {
   const jsonData = pm.response.json();
   pm.expect(jsonData).to.not.be.empty;
});
```

4.9. Outstanding Repairs

Actual Script:

```
pm.test("Status code is 200", function () {
   pm.response.to.have.status(200);
});

pm.test("Response is an array", function () {
   const jsonData = pm.response.json();
   pm.expect(jsonData).to.be.an("array");
});
```

<u>Negative Check Script:</u>

```
pm.test("Status code is not 200", function () {
    pm.response.to.not.have.status(200);
});

pm.test("Response is not empty", function () {
    const jsonData = pm.response.json();
    pm.expect(jsonData).to.not.be.empty;
});
```

4.10. Resource Utilizations

<u>Actual Script:</u>

```
pm.test("Status code is 200", function () {
    pm.response.to.have.status(200);
});

pm.test("Response contains utilization statistics", function () {
    const jsonData = pm.response.json();
    pm.expect(jsonData).to.have.property("personnelUtilization");
    pm.expect(jsonData).to.have.property("machineUtilization");
});
```

<u>Negative Check Script:</u>

```
pm.test("Status code is not 200", function () {
    pm.response.to.not.have.status(200);
});
```

```
pm.test("Response should not be missing utilization data", function () {
    const jsonData = pm.response.json();
    pm.expect(jsonData).to.not.have.property("missingProperty");
});
```

4.11. Update Schedule

Actual Script:

```
pm.test("Status code is 200", function () {
    pm.response.to.have.status(200);
});

pm.test("Response indicates success", function () {
    const jsonData = pm.response.json();
    pm.expect(jsonData.message).to.equal("Schedule updated successfully");
});
```

Negative Check Script:

```
pm.test("Status code is not 200", function () {
    pm.response.to.not.have.status(200);
});

pm.test("Response message should not indicate failure", function () {
    const jsonData = pm.response.json();
    pm.expect(jsonData.message).to.not.equal("Failed to update schedule");
});
```

4.12. Review a Request

Actual Script:

```
pm.test("Status code is 200", function () {
    pm.response.to.have.status(200);
});

pm.test("Response indicates a successful review", function () {
    const jsonData = pm.response.json();
    pm.expect(jsonData.message).to.equal("Request reviewed successfully");
});
```

Negative Check Script:

```
pm.test("Status code is not 200", function () {
    pm.response.to.not.have.status(200);
});

pm.test("Response message should not indicate failure", function () {
    const jsonData = pm.response.json();
    pm.expect(jsonData.message).to.not.equal("Failed to review request");
});
```

4.13. Add Estimate

Actual Script:

```
pm.test("Status code is 200", function () {
    pm.response.to.have.status(200);
});

pm.test("Estimate added successfully", function () {
    const jsonData = pm.response.json();
    pm.expect(jsonData.message).to.equal("Estimate added successfully");
});
```

Negative Check Script:

```
pm.test("Status code is not 200", function () {
    pm.response.to.not.have.status(200);
});

pm.test("Response message should not indicate failure", function () {
    const jsonData = pm.response.json();
    pm.expect(jsonData.message).to.not.equal("Failed to add estimate");
});
```

5. Test Cases

TEST CASE	INPUT(I)	STATE(S)	RESULT(R)
Register Complaint	Complaint details: road ID, description, type	Complaint registration endpoint is accessible	Status code 200, Response contains "status" and "message" fields
	Invalid complaint details (missing fields)	Complaint registration endpoint is accessible	Status code not 200, Error message returned
Complaint List By Area	Area ID	Complaints endpoint is accessible	Status code 200, Response is an array
	Invalid Area ID (non-existent)	Complaints endpoint is accessible	Status code not 200, Error or empty response
All Complaints	No input needed	Complaints endpoint is accessible	Status code 200, Response is an array
	No complaints in the system	Complaints endpoint is accessible	Status code 200, Empty array
Add New Material	Material details: type, description, inventory, cost	Material creation endpoint is accessible	Status code 200, Response contains material properties
	Missing required fields	Material creation endpoint is accessible	Status code not 200, Error message
Delete Material By Id	Material ID	Material exists in the database	Status code 204, No content in response
	Non-existent Material ID	Material deletion endpoint is accessible	Status code not 204, Error message
Update Material By Id	Material ID and updated properties	Material exists in the database	Status code 200, Updated

			material properties returned
	Invalid Material ID or properties	Material update endpoint is accessible	Status code not 200, Error message
Authentication	Username and password	User exists and credentials are correct	Status code 200, Response contains a "token"
	Incorrect username or password	User exists in the database	Status code not 200, Authentication failed message
Repairs In a Period	Start and end dates	Valid date range	Status code 200, Response is an array
	Invalid date range (start > end)	Valid date range provided	Status code not 200, Error message
Outstanding Repairs	No input needed	Repairs endpoint is accessible	Status code 200, Response is an array
	No outstanding repairs	Repairs endpoint is accessible	Status code 200, Empty array
Resource Utilizations	Start and end dates	Utilization endpoint is accessible	Status code 200, Response contains utilization statistics
	Invalid date range	Utilization endpoint is accessible	Status code not 200, Error message
Update Schedule	Schedule ID and updated properties	Schedule exists in the database	Status code 200, Success message
	Non-existent Schedule ID	Schedule update endpoint is accessible	Status code not 200, Error message
Review a Request	Request ID and review details	Request exists in the system	Status code 200, Success message
	Invalid or non-existent Request ID	Request review endpoint is accessible	Status code not 200, Error message
Add Estimate	Estimate details: complaint ID, resource ID, quantity	Estimate creation endpoint is accessible	Status code 200, Success message
	Missing or invalid estimate details	Estimate creation endpoint is accessible	Status code not 200, Error message