Incident API Design

1. Create Incident

- **Method**: `POST`
- **Endpoint**: `/api/incidents`

Request Body:

```
{
  "accidentId": "integer",
  "title": "string",
  "description": "string",
  "location": "string",
  "severity": 1,
  "status": "string"
}
```

- **Required Fields**: `accidentId`, `title`, `description`, `location`, `severity`.

Response:

- **201 Created**:

  ```
  {
    "accidentId": "integer",
    "message": "Incident created successfully."
  }
  ```

- **400 Bad Request** (invalid body data):

  ```
  {
    "error": "Bad Request",
    "message": "Missing required fields or invalid data."
  }
  ```

- **409 Conflict** (duplicate `AccidentID`):

  ```
  {
    "error": "Conflict",
    "message": "Incident with the given AccidentID already exists."
  }
  ```

2. Delete Incident

- **Method**: `DELETE`
- **Endpoint**: `/api/incidents/{accidentId}`

Path Parameter:

- `accidentId`: Unique ID of the incident to delete.

Response:

- **200 OK**:

```
{
  "message": "Incident deleted successfully."
}
```

- **404 Not Found**:

```
{
  "error": "Not Found",
  "message": "Incident with the given AccidentID not found."
}
```

3. Modify Incident

- **Method**: `PUT`
- **Endpoint**: `/api/incidents/{accidentId}`

Path Parameter:

- `accidentId`: Unique ID of the incident to modify.

Request Body:

```
{
  "title": "Updated title",
  "description": "Updated description",
  "location": "Updated location",
  "severity": 3,
  "status": "Closed"
}
```

- **Explanation**: Fields to update, without the need for `accidentId` in the body.

Response:

- **200 OK**:

```
{
  "message": "Incident updated successfully."
}
```

- **400 Bad Request** (invalid data):

```
{
  "error": "Bad Request",
  "message": "Invalid data provided for updating the incident."
}
```

- **404 Not Found**:

```
{
  "error": "Not Found",
  "message": "Incident with the given AccidentID not found."
}
```

---

4. List All Incidents

- **Method**: `GET`
- **Endpoint**: `/api/incidents`

Query Parameters (optional):

- `status`: Filter by status (e.g., `Open`, `Closed`).
- `severity`: Filter by severity (e.g., `1-5`).
- `page`: Pagination: page number.
- `limit`: Pagination: number of records per page.

Response:

- **200 OK**:

```
{
  "incidents": [
    {
```

```
      "accidentId": 1,
      "title": "Incident title 1",
      "description": "Detailed description 1",
      "location": "Location 1",
      "severity": 3,
      "status": "Open",
      "createdTime": "datetime",
      "updatedTime": "datetime"
    },
    {
      "accidentId": 2,
      "title": "Incident title 2",
      "description": "Detailed description 2",
      "location": "Location 2",
      "severity": 2,
      "status": "Closed",
      "createdTime": "datetime",
      "updatedTime": "datetime"
    }
  ],
  "pagination": {
    "currentPage": 1,
    "totalPages": 2,
    "totalRecords": 5
  }
}
```

- **404 Not Found**:

```
{
  "error": "Not Found",
  "message": "No incidents found."
}
```

Exception Handling

1. **400 Bad Request**:
   - Triggered by missing or invalid required data (e.g., `title`, `description`, or an invalid `severity` value).
2. **404 Not Found**:
   - Triggered when trying to access, modify, or delete an incident that doesn't exist (invalid `accidentId`).
3. **409 Conflict**:
   - Triggered when attempting to create an incident with an already existing `accidentId`.

4. **500 Internal Server Error**:
    - Triggered by unexpected server-side issues
