Week 10

**Reflection on Backend Design for Coursework**

**Design Description**

The backend for this coursework focuses on managing data related to football teams or statistics. It uses **Express.js** as a framework and integrates with **MongoDB** for data storage, employing Mongoose for database operations. The file indicates the implementation of **4 REST API endpoints** as follows:

1. **POST /add**
   * **Purpose**: To add a new record (e.g., a football team's information or stats) to the database.
   * **MongoDB Query**:

const newData = new Football(req.body);

await newData.save();

* + This creates a new document in the Football collection with the data received in the request body.

1. **POST /update**
   * **Purpose**: To update an existing record based on the team name.
   * **MongoDB Query**:

await Football.findOneAndUpdate({ team }, updateData, { new: true });

* + Finds a document in the Football collection where the team field matches the provided name and updates it with the given data.

1. **POST /delete**
   * **Purpose**: To delete a record from the database based on the team name.
   * **MongoDB Query**:

await Football.deleteOne({ team: req.body.team });

* + Deletes a document in the Football collection where the team field matches the provided name.

1. **GET /stats/:year**
   * **Purpose**: To fetch statistics for a specific year.
   * **MongoDB Query** (assumed):

Football.find({ year: req.params.year });

* + Retrieves all documents in the Football collection that match the specified year.

**MongoDB Queries and Use Cases**

1. **Adding Data**:
   * Query: Football.save()
   * Use Case: Adds a new football team or statistic entry.
2. **Updating Data**:
   * Query: Football.findOneAndUpdate()
   * Use Case: Modifies an existing record based on the team name.
3. **Deleting Data**:
   * Query: Football.deleteOne()
   * Use Case: Removes a football team entry from the database.
4. **Fetching Data**:
   * Query: Football.find()
   * Use Case: Retrieves information based on criteria like the year.

**Observations**

* The backend is designed to perform essential CRUD operations.
* MongoDB's query methods are effectively utilized to handle operations on the Football collection.
* The endpoints are intuitive, mapping directly to typical use cases in managing football data.

**Recommendations**

* Add validation for data input to ensure consistency in database records.
* Implement error-handling middleware to manage unexpected issues gracefully.
* Secure endpoints with authentication and authorization mechanisms.
* Enhance query capabilities with filtering, sorting, and pagination for large datasets.

This backend provides a robust structure for football data management with clear REST API endpoints and well-utilized MongoDB queries. ​