API Documentation and Code Overview

1. Project Overview

This project is a Node.js application using Express and MongoDB for managing and querying a collection of questions and answers related to university departments. The project includes endpoints for creating, reading, updating, searching, and deleting queries.

2. Project Structure

- **`index.js`:** Entry point of the application, sets up Express server and connects to the MongoDB database.
- **`routes/requestRoutes.js`:** Defines API routes and maps them to controller functions.
- **`controller/requestController.js`:** Contains the logic for handling API requests.
- **`models/Question.js`:** Defines the Mongoose schema and model for questions.
- **`config/db.js`:** Contains the function to connect to the MongoDB database.
- **`.env`:** Stores environment variables like database URL and port number.
- **`package.json`:** Lists project dependencies and scripts.

3. API Endpoints

Base URL

```
/api/queries
```

Endpoints

1. Get All Queries

- **Method:** `GET`
- **URL:** `/api/queries`
- **Description:** Retrieves all questions and answers from the database.
- **Responses:**
  - **`200 OK`:** Returns a JSON object with a list of all questions.
  - **`404 Not Found`:** No questions found.
  - **`500 Internal Server Error`:** Error fetching questions.

2. Search Queries

- **Method:** `GET`
- **URL:** `/api/queries/search`
- **Description:** Searches for questions based on a query string.
- **Query Parameters**:
  - **`query` (required):** The search string.
- **Responses:**
  - **`200 OK`:** Returns a JSON object with matching questions.
  - **`400 Bad Request`:** Missing search query parameter.
  - **`404 Not Found`:** No matching questions found.
  - **`500 Internal Server Error`:** Error performing search.

Example:

http://localhost:5000/api/queries/search?query=capital

3. Add New Query

- **Method:** `POST`

- **URL:** `/api/queries`

- **Description**: Adds a new question with variations and answers to the database.

- **Request Body (JSON):**

  - **json**

```
{
  "questionVariations": ["string"],
  "answers": [{"text": "string", "imageUrl": "string"}],
  "department": "string",
  "intent": "string",
  "entities": ["string"],
  "context": "string"
}
```

- **Responses:**

  - **`201 Created`:** Returns a JSON object with the added question.

  - **`400 Bad Request`:** Missing required fields or validation errors.

  - **`500 Internal Server Error`:** Error adding the question.

4. Update Query

- **Method:** `PUT`

- **URL:** `/api/queries/:id`

- **Description:** Updates an existing question based on the provided ID.

- **Request Body (JSON):**

  - **json**

```
{
  "questionVariations": ["string"],
  "answers": [{"text": "string", "imageUrl": "string"}],
  "department": "string",
  "intent": "string",
  "entities": ["string"],
  "context": "string"
}
```

- **URL Parameters:**
  - **`id` (required):** The ID of the question to update.
- **Responses:**
  - **`200 OK`:** Returns a JSON object with the updated question.
  - **`400 Bad Request`:** Missing ID parameter or validation errors.
  - **`404 Not Found`:** Question not found with the provided ID.
  - **`500 Internal Server Error`:** Error updating the question.

Example:

http://localhost:5000/api/queries/1234567689

5. Get Queries by Department

- **Method:** `GET`
- **URL:** `/api/queries/department/:department`
- **Description:** Retrieves questions filtered by the specified department.
- **URL Parameters:**
  - **`department` (required):** The department name.
- **Responses:**

  - **`200 OK`:** Returns a JSON object with questions for the specified department.
  - **`400 Bad Request`:** Missing department parameter.
  - **`404 Not Found`:** No questions found for this department.
  - **`500 Internal Server Error`:** Error fetching questions.

Example:

http://localhost:5000/api/queries/department/computer science

6. Delete Query

- **Method:** `DELETE`
- **URL:** `/api/queries/:id`
- **Description:** Deletes a question based on the provided ID.
- **URL Parameters:**
  - **`id` (required):** The ID of the question to delete.
- **Responses:**
  - **`200 OK`:** Confirmation message of successful deletion.
  - **`400 Bad Request`:** Missing ID parameter.
  - **`404 Not Found`:** Question not found with the provided ID.
  - **`500 Internal Server Error`:** Error deleting the question.

Example:

http://localhost:5000/api/queries/1234567689

4. Code Overview

`index.js`

- Sets up an Express server.

- Loads environment variables using `dotenv`.

- Connects to MongoDB using `connectDB`.

- Defines middleware for JSON and URL-encoded data parsing.

- Sets up routes using `queryRoutes`.

`routes/requestRoutes.js`

- Defines routes and maps them to functions in `requestController.js`.

`controller/requestController.js`

- Contains functions for handling API requests:
  - **`query_all`:** Fetches all questions.
  - **`query_department`:** Fetches questions by department.
  - **`query_add`:** Adds a new question.
  - **`query_update`:** Updates an existing question.
  - **`query_search`:** Searches questions based on query string.
  - **`query_delete`:** Deletes a question.

`models/Question.js`

- Defines the Mongoose schema for questions and answers.

- Includes fields for question variations, answers, department, intent, entities, and context.

`config/db.js`

- Contains the `connectDB` function for connecting to MongoDB.

- Handles connection errors and logs them.

`.env`

- Stores environment variables:
  - `MONGO_URL`: MongoDB connection string.
  - `PORT`: Port number for the server.