**Robust Server Structure: RESTful API's**

**Instructions**

Your goal of this checkpoint is to get the tests to pass. To do so, you will be creating an API server with three distinct routes, a "not found" handler, and an error handler.  
Your server should follow the structure you've learned in the course. Complete the following tasks to pass the tests and this assignment.

**Existing files**

| **file path** | **description** |
| --- | --- |
| src/app.js | Contains the code for the express application. Place your code here. |
| src/server.js | Contains the code to start the express application. You do not need to edit this file. |
| src/data/notes-data.js | Stores two notes used as the initial notes data for appjs. Feel free to add or remove data from the files as necessary, but keep the same shape of the data. |
| test/app.test.js | The tests your code will run against. You do not need to edit this file. |

**Existing routes**

| **route** | **description** |
| --- | --- |
| GET /notes | returns an array of notes. |
| GET /notes/:noteId | returns a single note by id. |

**Tasks**

In the app.js file, you will:

1. Create a new /notes/ route that will handle a POST request containing the notes data (i.e., {"data": {"text": "<note-text-here>"}}) to the /notes endpoint, which will in turn:
   * assign a new id to the note
   * store the note
   * return a 201 status code and the stored note as JSON on success
   * return a 400 status code if the text property is missing or empty in the incoming request body
2. Modify the /notes/:noteId handler to return an error if the :noteId does not exist
3. Create a general error handler

**NOTE:** If you are having trouble getting your tests to pass but think you've gotten it right, make sure to check for punctuation and spelling. The tests are looking for *exact* string matches.

**Routes**

To complete this project, your server must have the following routes:

**GET /notes**

This route will respond to a GET request with an array of existing notes. e.g.:

{

"data": [

{ "id": 1, "text": "REST stands for Representational State Transfer" }

]

}

* If a note is added, it should be included in the array returned by this route.

**POST /notes**

This route will respond to a POST request and create a new note of the request is valid.

* If the req.body is valid:
  + assign a unique id to the note
  + respond with a 201 status code and the new note. e.g. {"data": [{ "id": 1, "text": "REST stands for Representational State Transfer"}]}
* If the req.body *does not* contain a data property, respond with a 400 status code and an error message.
* If the req.body.data.text property is empty (""), undefined, or null, respond with a 400 status code and an error message.

**GET /notes/:noteId**

This route will respond to a GET request with the note for the provided noteId.

* If the noteId matches an existing note, respond with the note. e.g. {"data": { "id": 1, "text": "REST stands for Representational State Transfer"}}
* If the noteId *does not* match an existing note, respond with the following message.

`Note id not found: ${req.params.noteId}`;

**Error Handling**

If the user goes to a route that is not defined, respond with the following message:

`Not found: ${req.originalUrl}`;