1. Create a new express app.The app should listen for requests on port 3000. Make sure you add the required dependency.
2. Install the dependency sequelize
3. In the app, initialize the database connection with Sequelize.
4. Using Sequelize, define a model called Product with the following properties (in addition to an ID):
   * title (text)
   * price(number)
   * description(text)
5. Make sure the model is synched with the database upon startup.
6. Use the model create() method to insert 3 rows of example data. This logic should happen *after* the model synchronization completes. The data should persist. Restarting the API should not cause any data to be lost.(Optional)
7. Create an express app with routes that support the following RESTful actions on the "product" resources.
   * *create* a new product resource
   * *read all* product (the collections resource)
   * *read* a single product resource
   * *update* a single product resource
   * *delete* a single product resource
8. You don't need any special logic. A standard REST implementation is ok.
9. When a request is sent to the endpoint, it should log the text property of the body to the console, and it should respond with a JSON object:

{

"title": ".....",

"price": ".....",

"description": "...."

}

In order to parse the JSON body of the request, you will need to add the middleware for it. Make sure you add the required dependency.

Perform the following validation: if the body does NOT have a text property or the string is empty, then send a "Bad Request" HTTP status code to the client.

The API should only insert data five times.

After receiving five products, sixth request should be sent a response that indicates the HTTP status for "Too Many Requests".

Make sure the correct HTTP status code is sent (Google it if you haven't seen this status message before).

Although there are libraries to implement such limits

Put the data limit logic from the previous step into a middleware function. It should behave the same.

1. Make sure that your handlers send back 404 status codes when appropriate.

Implement pagination on the "read all" collections resource end-point.

The user must be able to pass limit and offset as query parameters to correctly control what results they receive. You can access query parameters on the req.query object. Sequelize supports pagination through the findAndCountAll query method.  
The response should, in addition to the array of resources, also contain a number indicating how many results there are in *total*. So, it should look like this:  
{

data: [

{ ... },

{ ... },

...

],

total: 12

}

1. Make sure that all endpoints handle database errors . Errors should be handled by Express' built-in error handler.