UPDATE

The next route we write will handle updating resources. Much like fetching single documents, when we update a resource we will pass in the dynamic route parameter. This tends to be the most complicated of the CRUD operations so we will write a basic update route method that will work and refactor later once we have a firm grasp on advanced concepts.

Part 1: Update Route Method

Step 1: Begin writing our route method

```
router.patch('/:id', (req, res) => {
```

```
// Here we will use the patch.http method, which is designed for updating an existing resource. Again we need to target specific document by id in our database so we pass in the dynamic route parameter.
```

Step 2: Locate and Update

```
  City.findByIdAndUpdate(req.params.id, req.body, {
    new: true,
    runValidators: true
  })
```

```
// The Mongoose method findByIdAndUpdate takes two arguments and we pass in an additional options object as the third argument, so that way we will get the behavior we desire.
// The first argument is object id by which to locate our document which we grab from the route parameter.
// The second argument is the request body which will contain the updates to mutate the document in our database.
// The last argument is an options object. We set a few key value pairs here:
- new: true (This way we are returned our updated document)
- runValidators:true(this way any validation we set up in our schema runs before the update.)
```

Step 3: Checking for document with conditional logic

```
    .then(city => {
      if (!city) {
        return res.status(404).send({ msg: 'City Not found' })
      }
// Remember just because our route method doesn't return a
document does not mean it failed. If the document doesn't exist
we will want to return from the function and send our custom
error message.
```

Step 4: Checking for document with conditional logic

```
      res.send(city)
    })
    .catch(err => {
      res.send(err.message)
    })
})
// If all goes well we will receive our updated document as our
response. Otherwise we will send back an error message to help
us figure out what went wrong and debug.
```

Part 1: Testing on Postman

Step 1: Fetch a single City document from your database using a GET request.

Step 2: Set the HTTP method to a "PATCH" request and set the end point to "localhost: 5000/api/cities/"

Step 3: Paste the City's object to the end of your route.

Step 4: In the "Headers" tab we need to create a key-value pair. Set the key to "Content-Type" and its value to "application/json"

Step 5: Next head to the Body tab and select the "raw" setting.

Step 6: Paste entire City object from the get request into the Body section.

Step 7: Make some distinct changes to the values of the object. i.e change the image link or the spelling of the City's name.

Step 8: Send the response. If successful we should receive our updated document in our response. If failure, use the error message to help debug.

- See Figure 1 for reference

Figure 1: