HTTP API HW

This assignment is due by the date on the dropbox.

Assignment:

Fork & Download the ***HTTP-API-Assignment*** starter files from our Github organization.
https://github.com/IGM-RichMedia-at-RIT/http-api-assignment

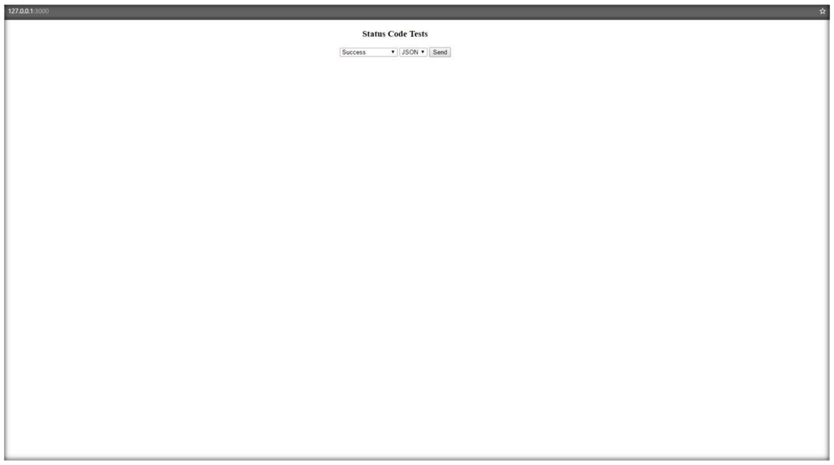
In this assignment, you will use Node.js to build a simple HTTP API complete with correct **status codes** and **accept headers**. You will also need to detect whether the client is requesting xml or json, and respond accordingly.

You will need to build an HTTP API that satisfies the following criteria:

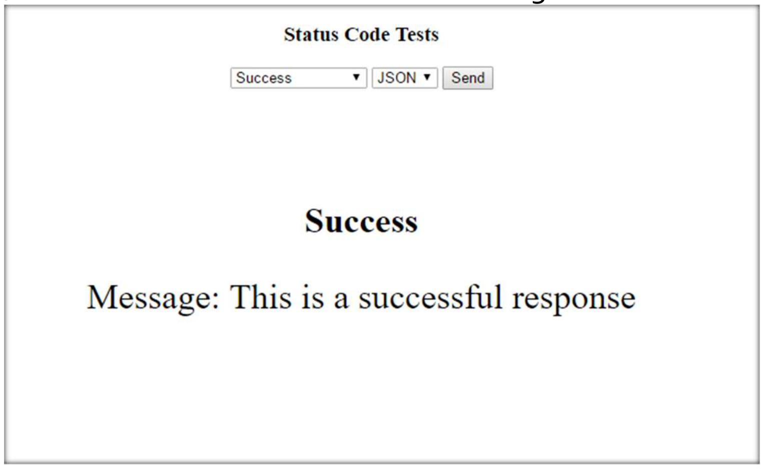
- Loads the client page provided as the home page
- Loads the css file into the client page as text/css
- Loads the following URLs both with fetch when hitting the 'send' button **and** directly to the URL providing options for either json or xml. *Should default to json.*
  - /success with a 200 status code
  - /badRequest with a 400 status code if missing the query parameter ***?valid=true***
  - /badRequest with a 200 status code if has the query parameter ***?valid=true***
  - /unauthorized with a 401 status code if missing the query parameter ***?loggedIn=yes***
  - /unauthorized with a 200 status code if has the query parameter ***?loggedIn=yes***
  - /forbidden with a 403 status code
  - /internal with a 500 status code
  - /notImplemented with a 501 status code
  - Any other page with a 404 status code
- Server should check the ***accept*** header for the correct media type.
- In fetch requests, client should send the ***accept*** header for the correct media type.
- Client should parse the returned xml or json (determined from content-type header in response) and put them onto the page.
- Responses should default to JSON as the return type if no Accept header is present.
- JSON successes should have a message field.
- JSON failures should have a message field and an id set to the status code name.
- XML successes should have a <message> tag
- XML failures should have a <message> tag and an <id> tag set to the status code name.
- Also print the raw JSON or XML text strings (before you parse them) to the console.

Examples of Correct Ouput:

The home page '/' (client.html, which also requests style.css from the server)



/success with JSON and XML after hitting the send button.



Success

Message: This is a successful response

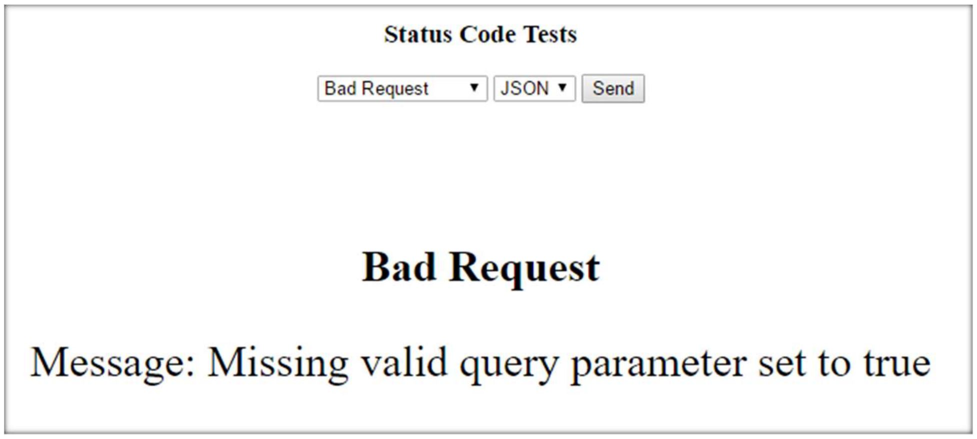
Example output of JSON response.

```
{"message":"This is a successful response"}
```

Example output of XML response.

```
<response><message>This is a successful response</message></response>
```

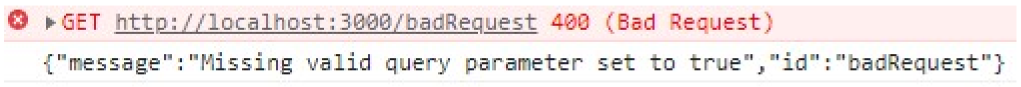
/badRequest with JSON and XML after hitting the send button.
(query parameter intentionally not included)



Bad Request

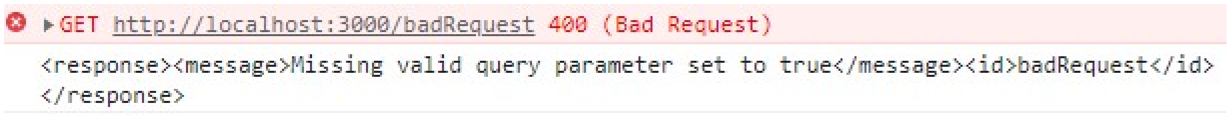
Message: Missing valid query parameter set to true

Example output of JSON response.



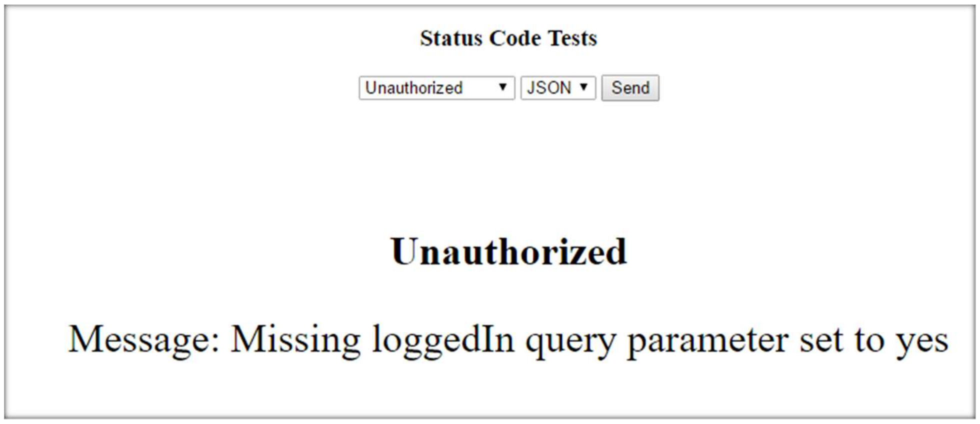
```
GET http://localhost:3000/badRequest 400 (Bad Request)
{"message":"Missing valid query parameter set to true","id":"badRequest"}
```

Example output of XML response.



```
GET http://localhost:3000/badRequest 400 (Bad Request)
<response><message>Missing valid query parameter set to true</message><id>badRequest</id>
</response>
```

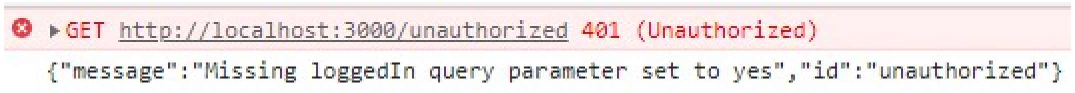
/unauthorized with JSON and XML after hitting send button
(query parameter intentionally not included)



Unauthorized

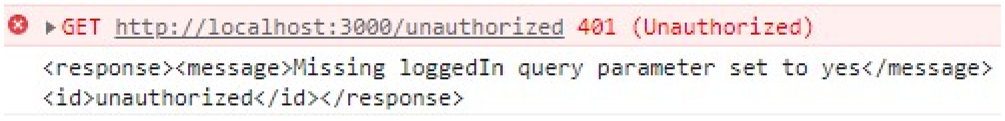
Message: Missing loggedIn query parameter set to yes

Example output of JSON response.



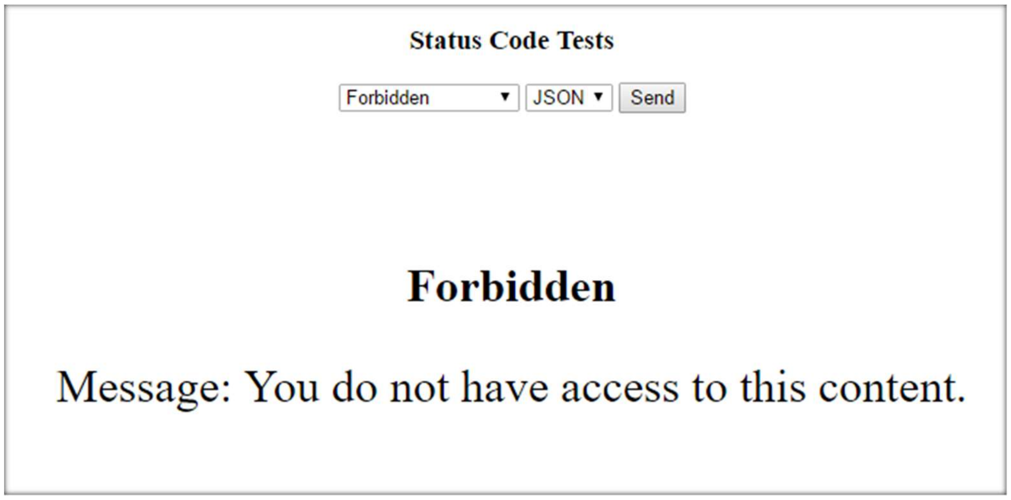
```
▶GET http://localhost:3000/unauthorized 401 (Unauthorized)
  {"message":"Missing loggedIn query parameter set to yes","id":"unauthorized"}
```

Example output of XML response.



```
▶GET http://localhost:3000/unauthorized 401 (Unauthorized)
  <response><message>Missing loggedIn query parameter set to yes</message>
  <id>unauthorized</id></response>
```

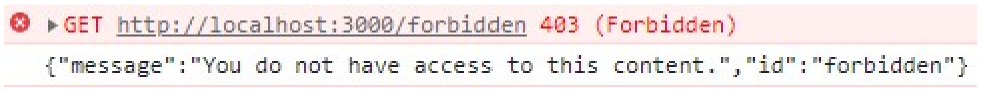
/forbidden with JSON and XML after hitting send button.



Forbidden

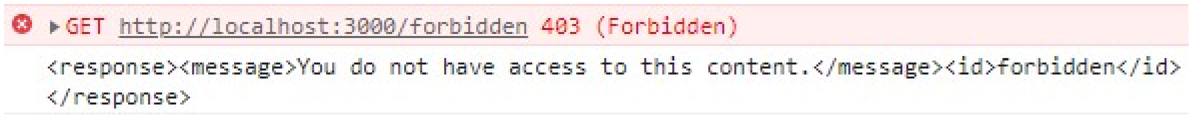
Message: You do not have access to this content.

Example output of JSON Response.



```
GET http://localhost:3000/forbidden 403 (Forbidden)
{"message":"You do not have access to this content.","id":"forbidden"}
```

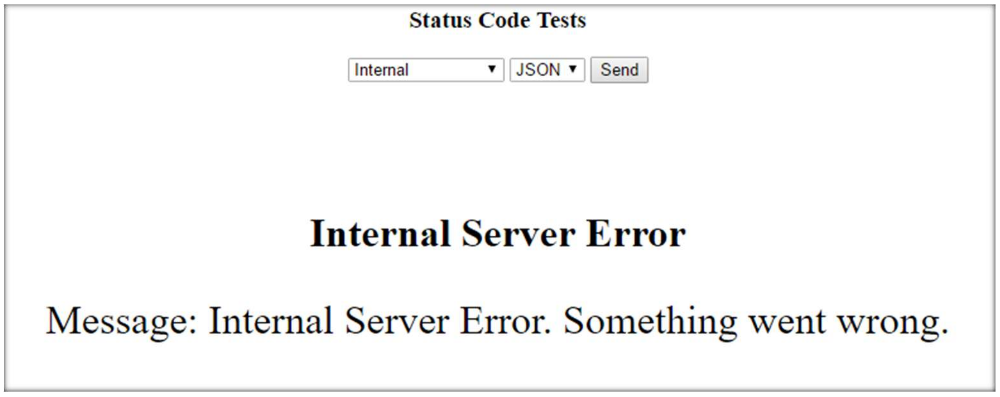
Example output of XML Response.



```
GET http://localhost:3000/forbidden 403 (Forbidden)
<response><message>You do not have access to this content.</message><id>forbidden</id>
</response>
```

HTTP API HW

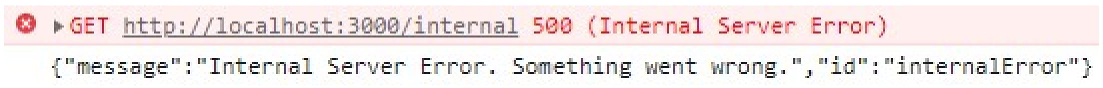
/internal with JSON and XML after hitting send button.



Internal Server Error

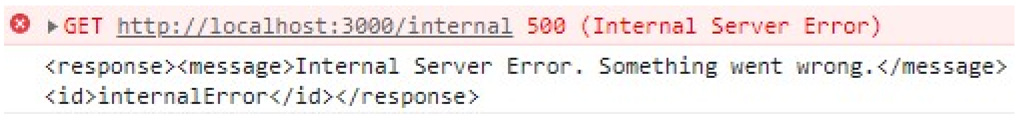
Message: Internal Server Error. Something went wrong.

Example output of JSON response.



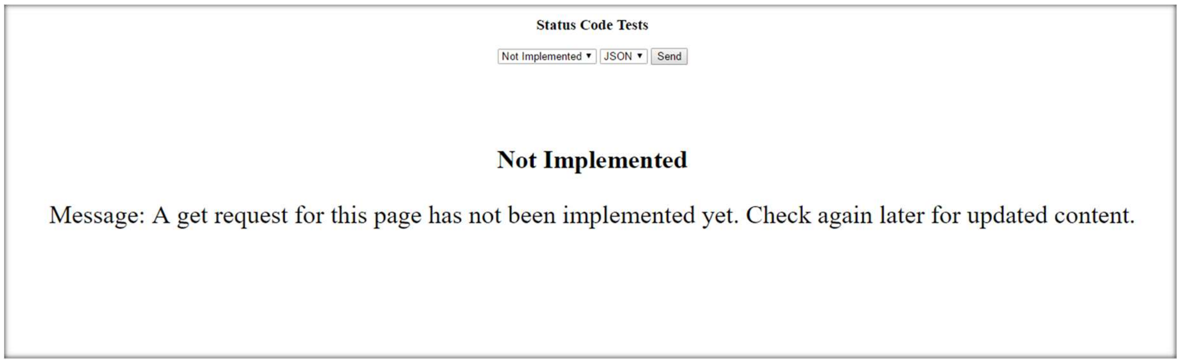
```
▶GET http://localhost:3000/internal 500 (Internal Server Error)
{"message":"Internal Server Error. Something went wrong.","id":"internalError"}
```

Example output of XML response.



```
▶GET http://localhost:3000/internal 500 (Internal Server Error)
<response><message>Internal Server Error. Something went wrong.</message>
<id>internalError</id></response>
```

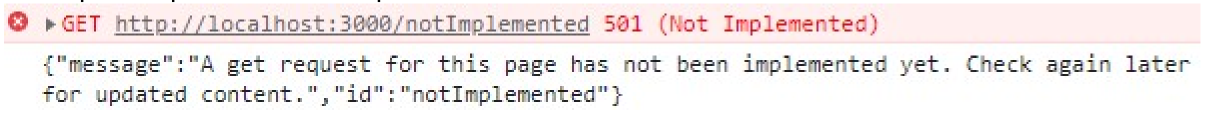
/notImplemented with JSON and XML after hitting send button.



Not Implemented

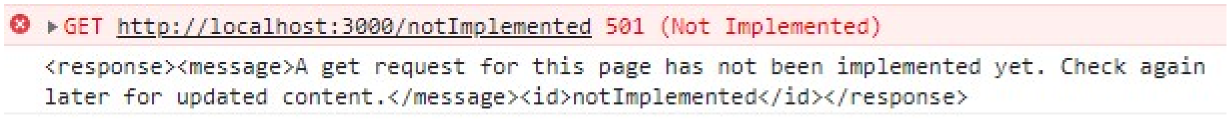
Message: A get request for this page has not been implemented yet. Check again later for updated content.

Example output of JSON response.



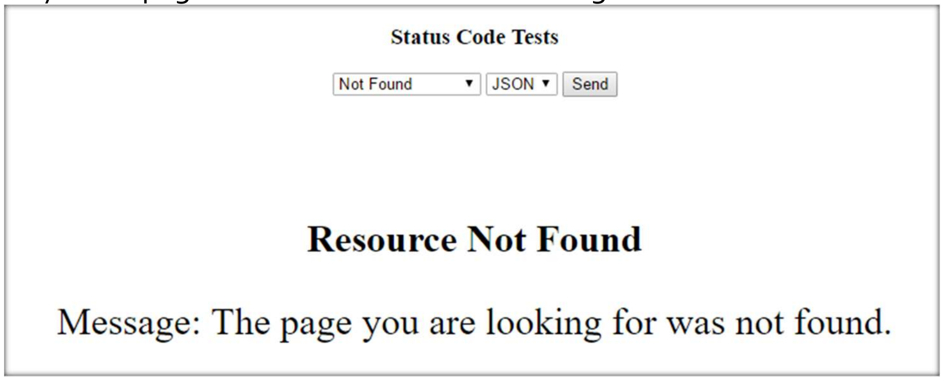
```
▶GET http://localhost:3000/notImplemented 501 (Not Implemented)
{"message":"A get request for this page has not been implemented yet. Check again later
for updated content.","id":"notImplemented"}
```

Example output of XML response.



```
▶GET http://localhost:3000/notImplemented 501 (Not Implemented)
<response><message>A get request for this page has not been implemented yet. Check again
later for updated content.</message><id>notImplemented</id></response>
```

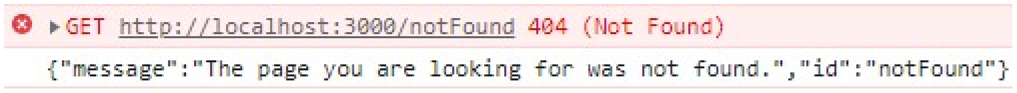
Any other page in JSON and XML after hitting the send button.



Resource Not Found

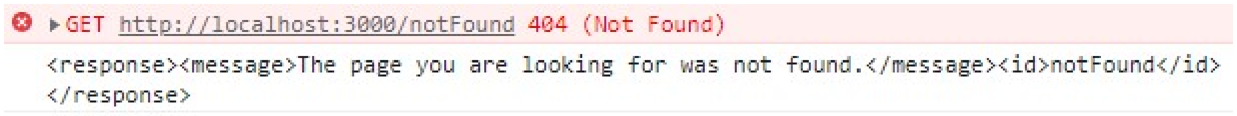
Message: The page you are looking for was not found.

Example output of JSON response.



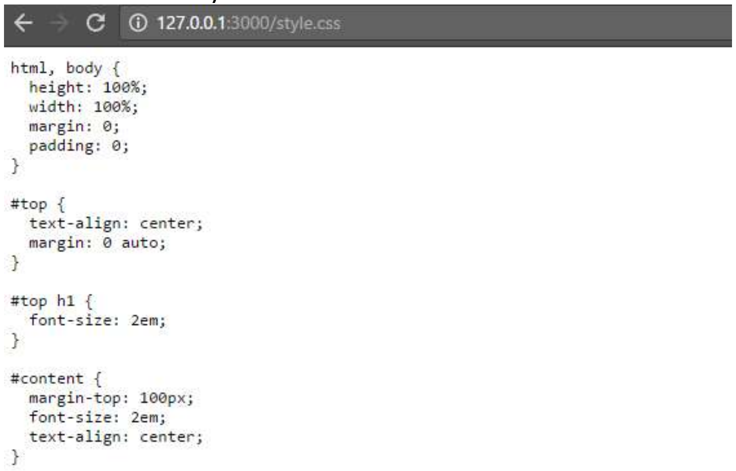
```
▶GET http://localhost:3000/notFound 404 (Not Found)
{"message":"The page you are looking for was not found.","id":"notFound"}
```

Example output of XML response.



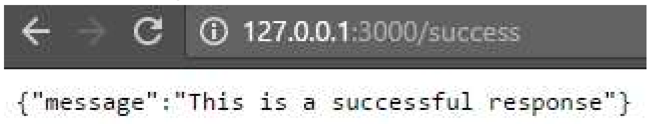
```
▶GET http://localhost:3000/notFound 404 (Not Found)
<response><message>The page you are looking for was not found.</message><id>notFound</id>
</response>
```

Direct call to /style.css



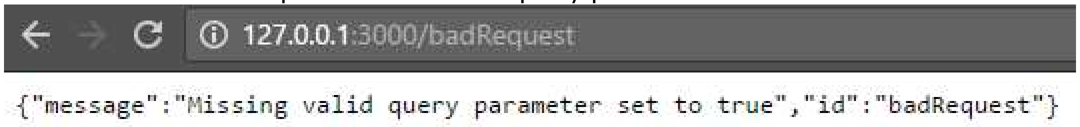
```
html, body {
  height: 100%;
  width: 100%;
  margin: 0;
  padding: 0;
}

#top {
  text-align: center;
  margin: 0 auto;
}

#top h1 {
  font-size: 2em;
}

#content {
  margin-top: 100px;
  font-size: 2em;
  text-align: center;
}
```

Direct call to /success



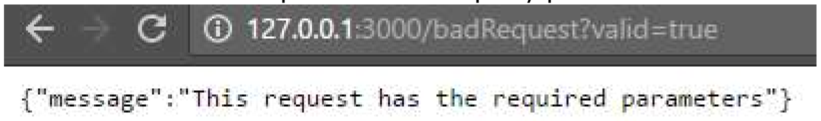
```
{"message":"This is a successful response"}
```

Direct call to /badRequest without the query parameter ***?valid=true***



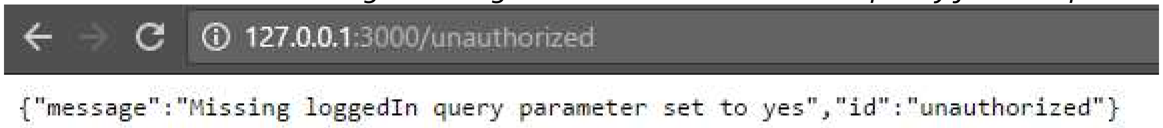
```
{"message":"Missing valid query parameter set to true","id":"badRequest"}
```

Direct call to /badRequest with the query parameter ***?valid=true***



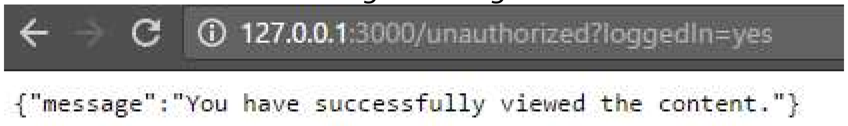
```
{"message":"This request has the required parameters"}
```

Direct call to /unauthorized without the query parameter ***?loggedIn=yes***
Note: *You would never do logins through the URL like this. This was purely for example.*



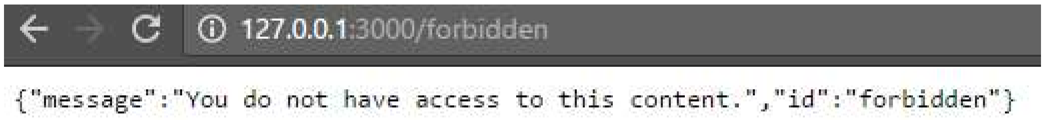
```
{"message":"Missing loggedIn query parameter set to yes","id":"unauthorized"}
```

Direct call to /unauthorized with the query parameter ***?loggedIn=yes***
Note: *You would never do logins through the URL like this. This was purely for example.*



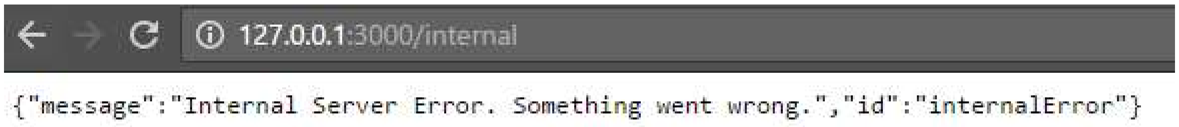
```
{"message":"You have successfully viewed the content."}
```

Direct call to /forbidden


127.0.0.1:3000/forbidden

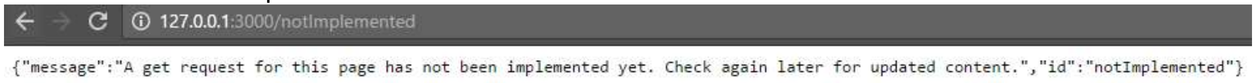
```
{"message":"You do not have access to this content.","id":"forbidden"}
```

Direct call to /internal


127.0.0.1:3000/internal

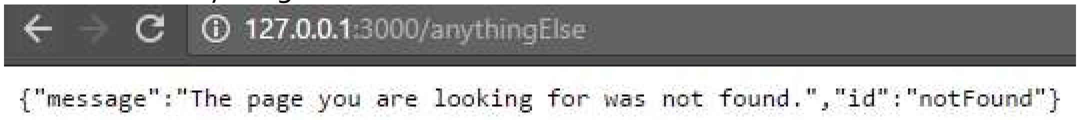
```
{"message":"Internal Server Error. Something went wrong.","id":"internalError"}
```

Direct call to /notImplemented


127.0.0.1:3000/notImplemented

```
{"message":"A get request for this page has not been implemented yet. Check again later for updated content.","id":"notImplemented"}
```

Direct call to anything else


127.0.0.1:3000/anythingElse

```
{"message":"The page you are looking for was not found.","id":"notFound"}
```

ESLint & Code Quality

You are required to use ESLint with the js/recommended configuration for code quality checks. The eslint.config.js file was given to you in the starter files. Your **server** code must pass ESLint to get credit for this part. Warnings are okay, but errors must be fixed. I won't require ESLint on the client side code in this assignment. ***If you cannot fix an error or need help, please ask us!***

You are also required to use the "pretest" script in order to run ESLint. I should be able to run "*npm test*" and have ESLint scan your code for me.

Continuous Integration

You are required to use GitHub Actions for continuous integration. Your latest commit on GitHub must show a green checkmark denoting a successful run of the npm test command. Be aware that this checkmark can take a few minutes to show up, as GitHub Actions runs the test.

All starter code for this class includes the .github folder, which contains configuration for the GitHub Action that will test your code. The test should run automatically for each commit. If you don't have the .github folder, you'll have to grab it from another project.

Submission:

By the due date please submit a zip of your project to the dropbox. **Do not** include the node_modules folder in the zip or in your git repo.

In the submission comments include the following:

- *a link to your Heroku App (not the dashboard but the working web link from 'open app')*
- *a link to your Github repo (if private, please add the TA and myself)*

HTTP API HW

Rubric

DESCRIPTION	SCORE	VALUE %
Loads the client page – Loads the client.html page as the home page.		**5**
Loads the css as text/css – Loads the style.css file as text.css for the home page and directly.		**5**
/success works correctly – The /success url sends a 200 along with the correct format from the accept header. Defaults to json. Works both directly to the url and with FETCH.		**5**
/badRequest works correctly – The /badRequest url sends a 400 if missing the query parameter ?valid=true. It sends a 200 if it has the parameter. The correct format is sent based on the accept header. Defaults to json. Works both directly to the url and with FETCH.		**10**
/unauthorized works correctly – The /unauthorized url sends a 401 if missing the query parameter ?loggedIn=yes. It sends a 200 if it has the parameter. The correct format is sent based on the accept header. Defaults to json. Works both directly to the url and with FETCH.		**10**
/forbidden works correctly – The /forbidden url sends a 403 along with the correct format from the accept header. Defaults to json. Works both directly to the url and with FETCH.		**10**
/internal works correctly – The /internal url sends a 500 along with the correct format from the accept header. Defaults to json. Works both directly to the url and with FETCH.		**10**
/notImplemented works correctly – The /notImplemented url sends a 501 along with the correct format from the accept header. Defaults to json. Works both directly to the url and with FETCH.		**10**
Any other URL works correctly – Any other url sends a 404 along with the correct format from the accept header. Defaults to json. Works both directly to the url and with FETCH.		**10**
Uses accept header – The fetch client sends the appropriate accept header, which is used appropriately by the server to determine the response type.		**10**
Client parses response correctly – The client parses the response correctly based on the content-type from the server and displays the correct data on the page.		**5**
JSON responses formatted correctly – JSON messages are formatted correctly. JSON messages include a message field in successes and an additional id field in errors.		**5**
XML responses formatted correctly – XML messages are formatted correctly. XML messages include a message tag in successes and an additional id tag in errors.		**5**
Heroku Penalty – If your app is not functional on Heroku, there will be a penalty.		**-15% (this time)**
GitHub Actions Penalty – If your build is not successful on GitHub Actions, there will be a penalty.		**-15% (this time)**
ESLint Penalties – I should be able to run 'npm test' and have ESLint scan your code without errors. Warnings are acceptable. **Check your code with ESLint often during development!** **If you cannot fix an error, please ask us!**	1 Error 2 Errors 3 Errors 4 Errors 5+ Errors	**-5%** **-10%** **-15%** **-20%** **-30%**
Additional Penalties – These are point deductions for run time errors, poorly written code or improper code. There are no set values for penalties. The more penalties you make, the more points you will lose.		
TOTAL		**100%**