VIT®

Vellore Institute of Technology

(Deemed to be University under section 3 of UGC Act, 1956)

Vellore-632014, Tamil Nadu, India

Biological Databases

Course Code: BIT2002

(J-Component)

Analysis of mutation in breast cancer using COSMIC Database

Submitted by

Kirit Kumar (19BCB0001)

Prakhar Soni (19BCB0006)

Submitted to

Dr. SATARUPA BANERJEE

SBST

June 2021

Introduction:

COSMIC – the Catalogue of Somatic Mutations in Cancer – is the world's largest source of expert manually curated somatic mutation information relating to human cancers. Currently, it is the broadest database of mutations in cancer, the information in COSMIC is curated by expert scientists, primarily by scrutinizing large numbers of scientific publications. It covers all the genetic mechanisms by which somatic mutations promote cancer, including non-coding mutations, gene fusions, copy-number variants, and drug-resistance mutations. Being hand-curated means that COSMIC ensures quality, accuracy, and descriptive data capture.

For this project, we will be using the latest version of COSMIC available online, COSMIC v94, released on 28th May 2021, which has nearly 1.5 million samples available, from more than 28 thousand papers. The Cancer Browser available in COSMIC will allow us to browse COSMIC by tissue type and histology, allowing us to sort the data more efficiently.

Background:

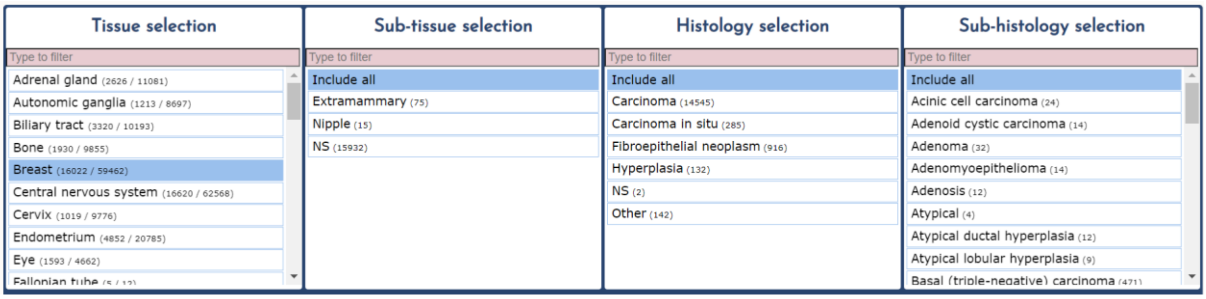
Next-generation sequencing has enabled the detection of thousands of mutations in single samples, in large quantities. However, not all mutations are malignant and cause cancer, as mutation is a regular occurrence that affects cellular functions on the whole. Mutations that provide a selective growth advantage, and promote the development of the cancer, are called driver mutations. The mutations that do not do so are known as passenger mutations. Those genes which have been identified as drivers in at least one cancer type are known as cancer genes.

Oncogenes (a gene that has the potential to cause cancer) are defined as driver genes in which driver mutations are activating or result in new functions. Tumor suppressors are driver genes in which driver mutations are inactivating. Oncogenes tend to be affected by focal amplifications or missense mutations at a limited number of codons, whereas tumor suppressors tend to be affected by focal deletions or nonsense, frameshift, and splice-site mutations dispersed across the gene.

The specific type of mutation (activating or inactivating) varies between driver genes, for example, MYC oncogene contributes to non-Hodgkin lymphoma (a cancer that starts in white blood cells called lymphocytes), whereas the BRAF oncogene is affected by the V600E mutation (a driver mutation which causes melanoma, hairy cell leukemia, papillary thyroid, etc.)

Method:

For this project, we will be using the Cancer Browser, provided by COSMIC. From this, we will be using the Selenium WebDriver in Python, to scrape COSMIC, obtain the mutations for the first 100 genes available in the cancer browser (with scope for expansion later on), then show these genes in the form of a database which we have made ourselves. This allows us to narrow down the genes and better refer the genes with the largest number of mutated samples, and have it available for easy access and browsing.



Tissue selection

Type to filter

- Adrenal gland (2626 / 11081)
- Autonomic ganglia (1213 / 8697)
- Biliary tract (3320 / 10193)
- Bone (1930 / 9855)
- Breast (16022 / 59462)
- Central nervous system (16620 / 62568)
- Cervix (1019 / 9776)
- Endometrium (4852 / 20785)
- Eye (1593 / 4662)
- Fallopian tube (5 / 12)

Sub-tissue selection

Type to filter

- Include all
- Extramammary (75)
- Nipple (15)
- NS (15932)

Histology selection

Type to filter

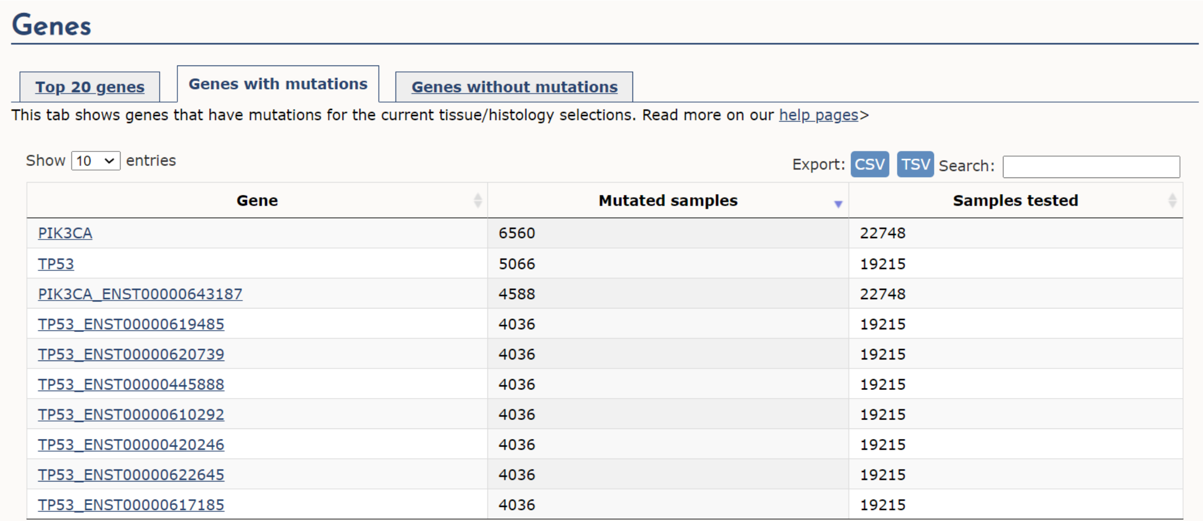
- Include all
- Carcinoma (14545)
- Carcinoma in situ (285)
- Fibroepithelial neoplasm (916)
- Hyperplasia (132)
- NS (2)
- Other (142)

Sub-histology selection

Type to filter

- Include all
- Acinic cell carcinoma (24)
- Adenoid cystic carcinoma (14)
- Adenoma (32)
- Adenomyoepithelioma (14)
- Adenosis (12)
- Atypical (4)
- Atypical ductal hyperplasia (12)
- Atypical lobular hyperplasia (9)
- Basal (triple-negative) carcinoma (471)

After selecting the appropriate options for the other required fields and submitting them, the Cancer Browser provides us with a list of genes that have been found to be linked to the selected tissue type and histology along with other relevant information.



Genes

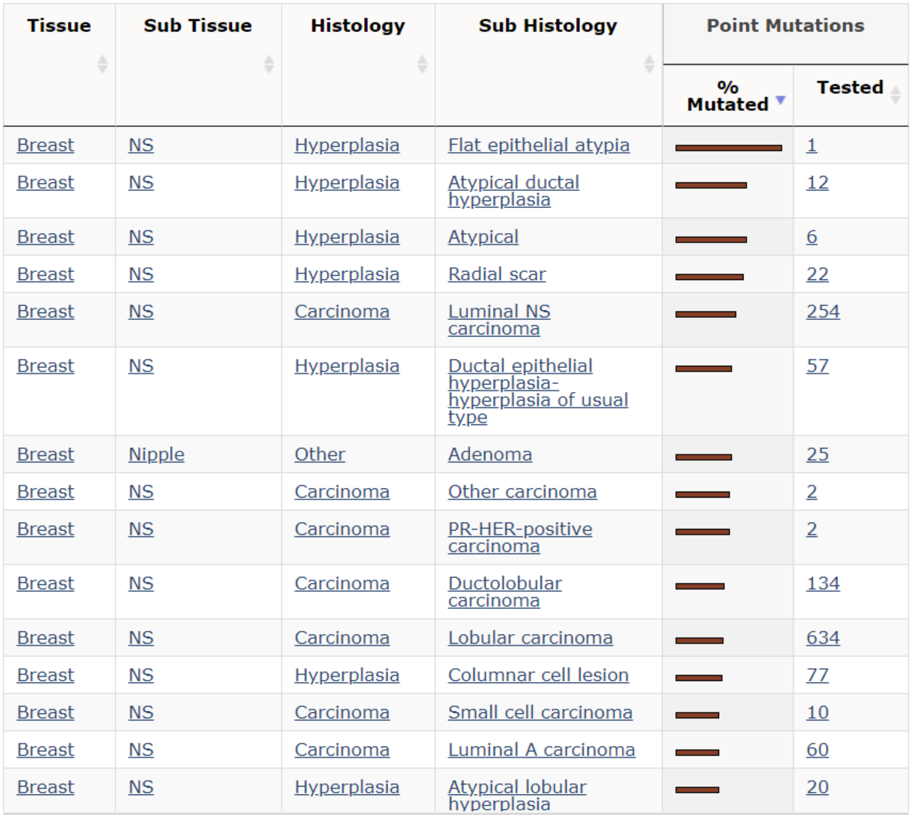
This tab shows genes that have mutations for the current tissue/histology selections. Read more on our help pages>

Show 10 entries

Export: CSV TSV Search:

Gene	Mutated samples	Samples tested
PIK3CA	6560	22748
TP53	5066	19215
PIK3CA_ENST00000643187	4588	22748
TP53_ENST00000619485	4036	19215
TP53_ENST00000620739	4036	19215
TP53_ENST00000445888	4036	19215
TP53_ENST00000610292	4036	19215
TP53_ENST00000420246	4036	19215
TP53_ENST00000622645	4036	19215
TP53_ENST00000617185	4036	19215

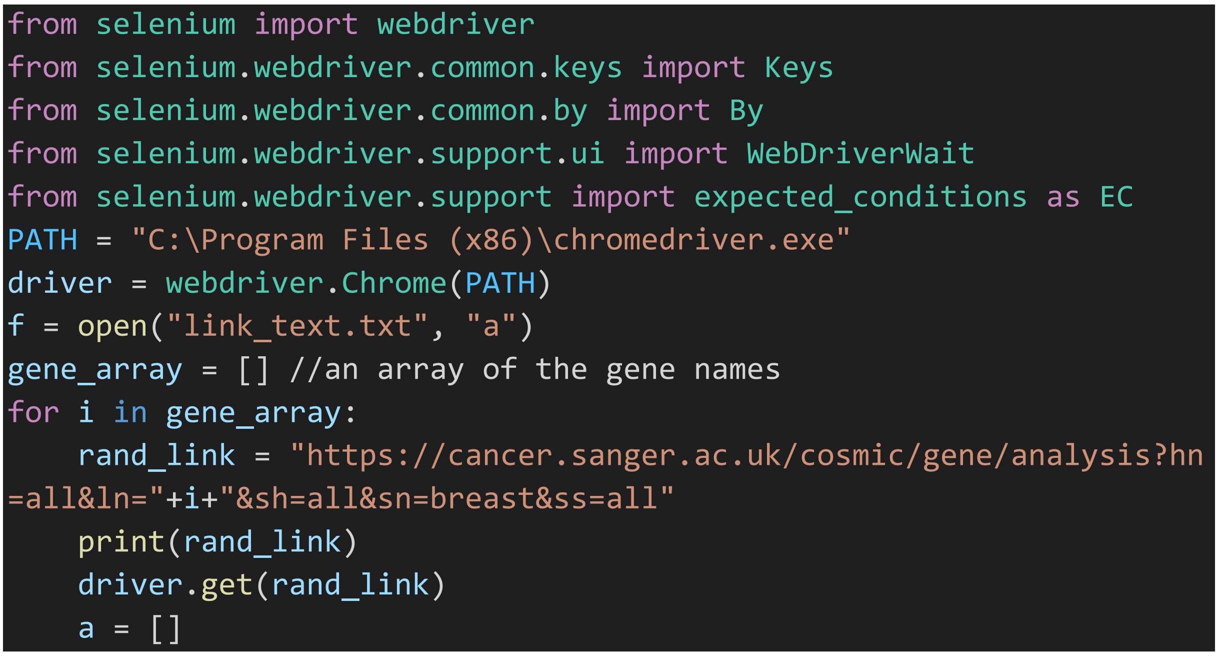
Using the example of PIK3CA gene, we will be obtaining the .csv file for mutations of this gene, from this table:



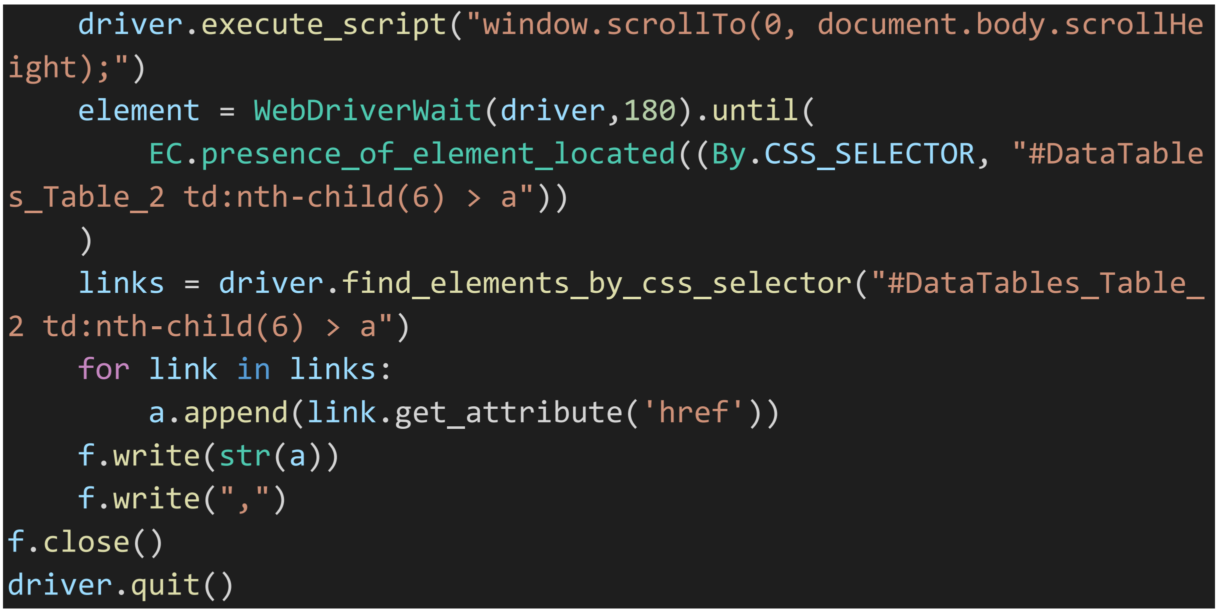
Tissue	Sub Tissue	Histology	Sub Histology	Point Mutations	
				% Mutated	Tested
Breast	NS	Hyperplasia	Flat epithelial atypia	▬	1
Breast	NS	Hyperplasia	Atypical ductal hyperplasia	▬	12
Breast	NS	Hyperplasia	Atypical	▬	6
Breast	NS	Hyperplasia	Radial scar	▬	22
Breast	NS	Carcinoma	Luminal NS carcinoma	▬	254
Breast	NS	Hyperplasia	Ductal epithelial hyperplasia-hyperplasia of usual type	▬	57
Breast	Nipple	Other	Adenoma	▬	25
Breast	NS	Carcinoma	Other carcinoma	▬	2
Breast	NS	Carcinoma	PR-HER-positive carcinoma	▬	2
Breast	NS	Carcinoma	Ductolobular carcinoma	▬	134
Breast	NS	Carcinoma	Lobular carcinoma	▬	634
Breast	NS	Hyperplasia	Columnar cell lesion	▬	77
Breast	NS	Carcinoma	Small cell carcinoma	▬	10
Breast	NS	Carcinoma	Luminal A carcinoma	▬	60
Breast	NS	Hyperplasia	Atypical lobular hyperplasia	▬	20

Implementation:

The links are then scraped from each of the links of the point mutations (in %mutations) using Selenium, and the code is given below:



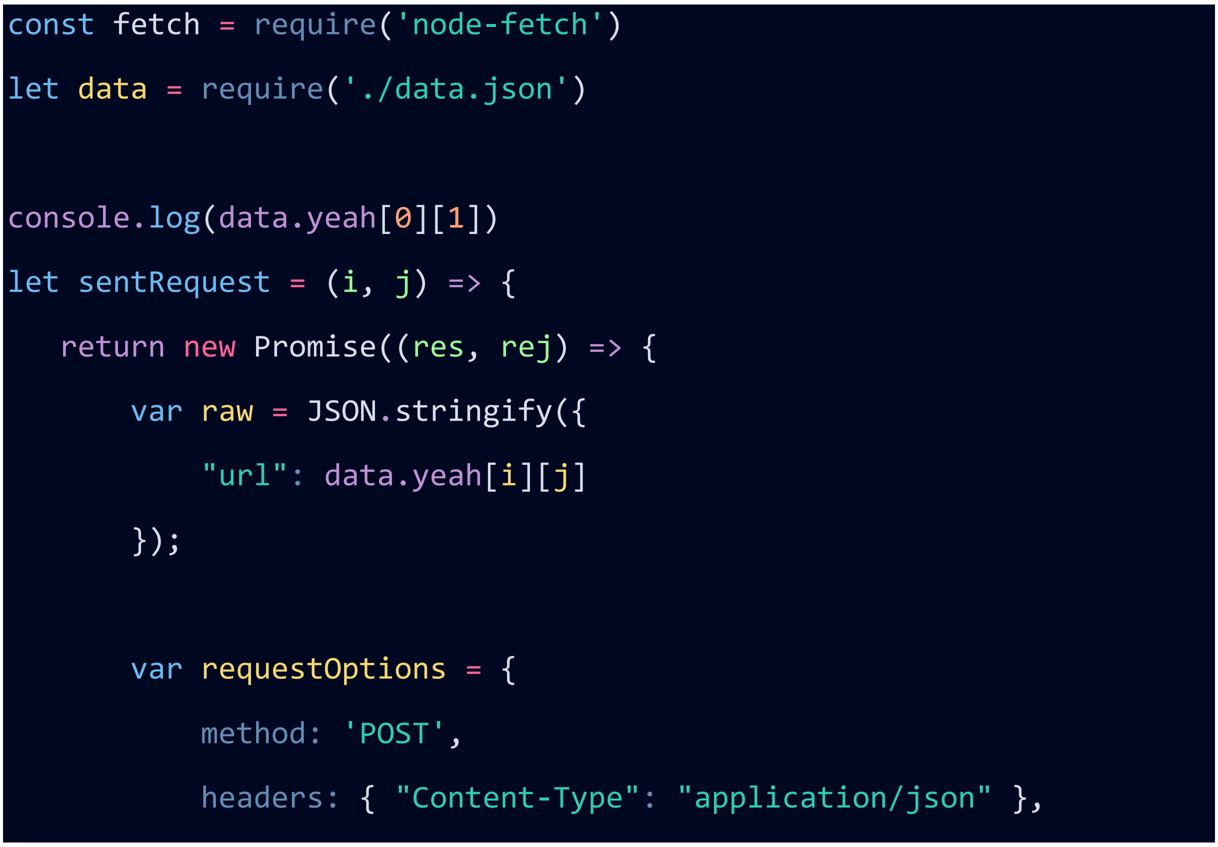
```
from selenium import webdriver
from selenium.webdriver.common.keys import Keys
from selenium.webdriver.common.by import By
from selenium.webdriver.support.ui import WebDriverWait
from selenium.webdriver.support import expected_conditions as EC
PATH = "C:\Program Files (x86)\chromedriver.exe"
driver = webdriver.Chrome(PATH)
f = open("link_text.txt", "a")
gene_array = [] //an array of the gene names
for i in gene_array:
    rand_link = "https://cancer.sanger.ac.uk/cosmic/gene/analysis?hn=all&ln="+i+"&sh=all&sn=breast&ss=all"
    print(rand_link)
    driver.get(rand_link)
    a = []
```



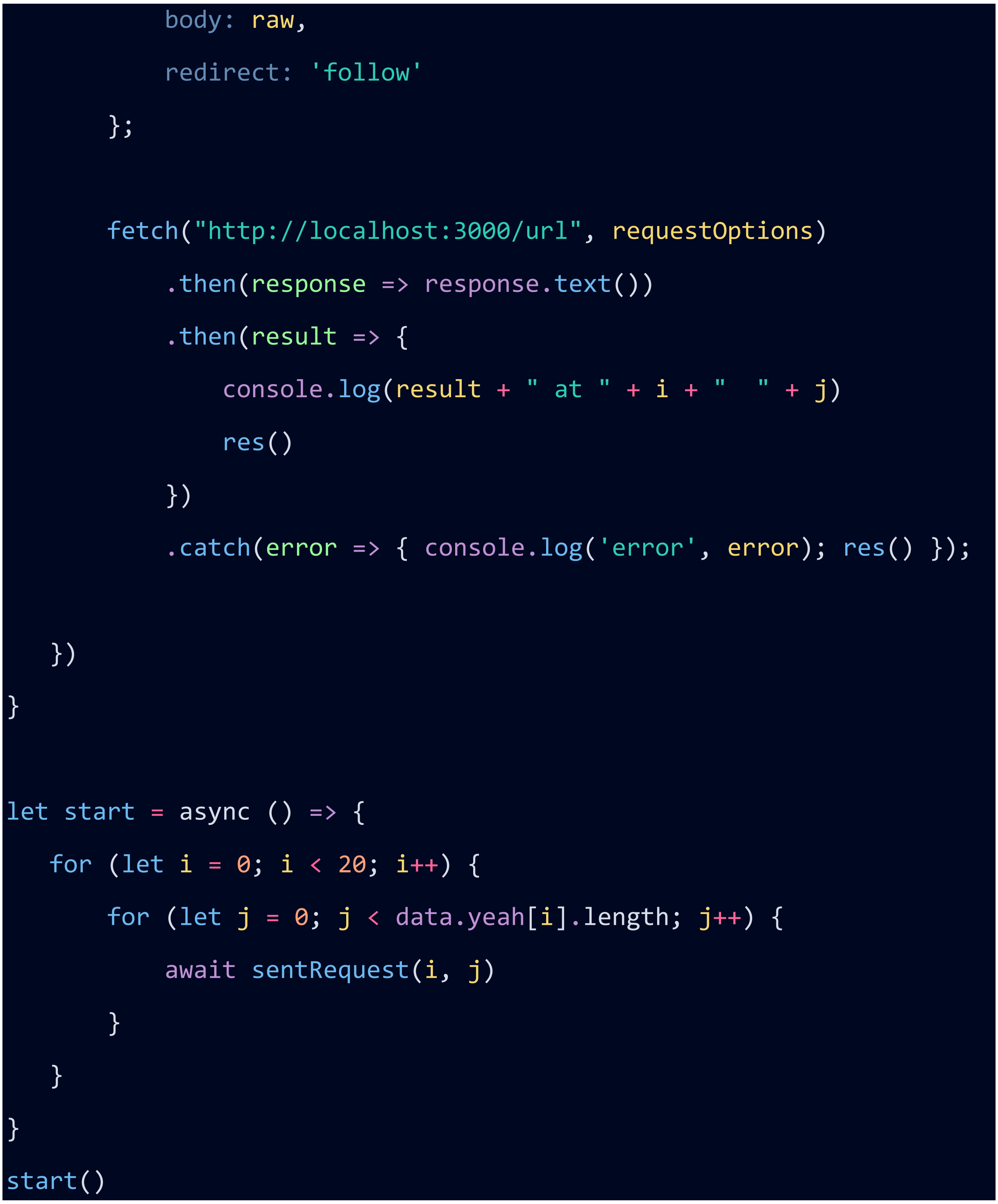
```python
    driver.execute_script("window.scrollTo(0, document.body.scrollHeight);")
    element = WebDriverWait(driver,180).until(
        EC.presence_of_element_located((By.CSS_SELECTOR, "#DataTables_Table_2 td:nth-child(6) > a"))
    )
    links = driver.find_elements_by_css_selector("#DataTables_Table_2 td:nth-child(6) > a")
    for link in links:
        a.append(link.get_attribute('href'))
    f.write(str(a))
    f.write(",")
f.close()
driver.quit()
```

Once the code is run, the links from COSMIC are scraped and put in a text file named “link_text.txt”, which holds the links in the form of an array for each of the genes mentioned in the “gene_array” variable. This allows us to conveniently have the links for the .csv files available to us, and from there we put the data obtained into another database, with approximately 100,000 entries using only 20% of the links.

These links are added to the database using this script that runs using NodeJS :



```javascript
const fetch = require('node-fetch')
let data = require('./data.json')


console.log(data.yeah[0][1])
let sentRequest = (i, j) => {
   return new Promise((res, rej) => {
       var raw = JSON.stringify({
           "url": data.yeah[i][j]
       });


       var requestOptions = {
           method: 'POST',
           headers: { "Content-Type": "application/json" },
```



```
           body: raw,
           redirect: 'follow'
       };

       fetch("http://localhost:3000/url", requestOptions)
           .then(response => response.text())
           .then(result => {
               console.log(result + " at " + i + "  " + j)
               res()
           })
           .catch(error => { console.log('error', error); res() });

   })
}

let start = async () => {
   for (let i = 0; i < 20; i++) {
       for (let j = 0; j < data.yeah[i].length; j++) {
           await sentRequest(i, j)
       }
   }
}
start()
```

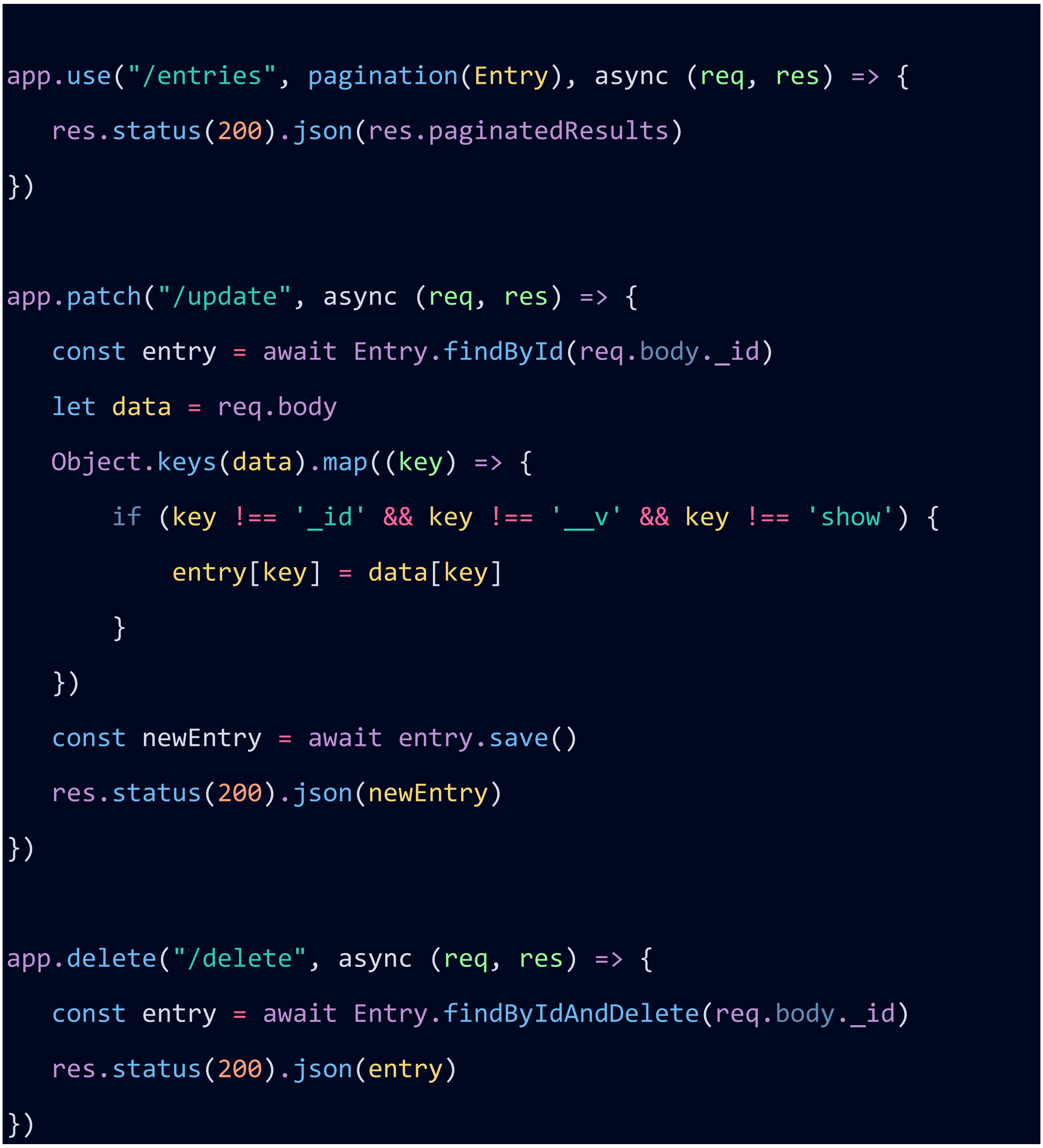
The Backend:

The backend consist of a Rest API made using Node js and a database using MongoDB

Rest API: Node.js is an open-source, cross-platform, back-end JavaScript runtime environment that runs on the V8 engine and executes JavaScript code outside a web browser. Node.js lets developers use JavaScript to write command line tools and for server-side scripting—running scripts server-side to produce dynamic web page content before the page is sent to the user's web browser.

This Rest API is available to anyone who wants to perform various actions with the Database.

A small code snippet of the backend:

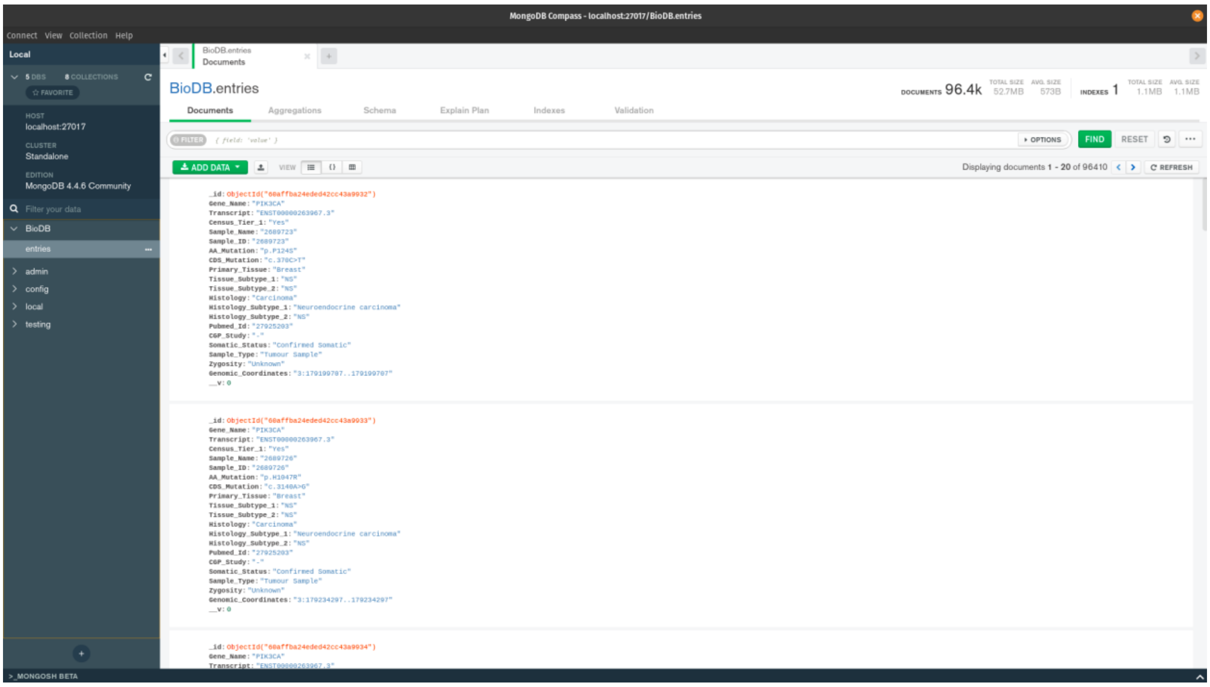


```
app.use("/entries", pagination(Entry), async (req, res) => {
    res.status(200).json(res.paginatedResults)
})

app.patch("/update", async (req, res) => {
    const entry = await Entry.findById(req.body._id)
    let data = req.body
    Object.keys(data).map((key) => {
        if (key !== '_id' && key !== '__v' && key !== 'show') {
            entry[key] = data[key]
        }
    })
    const newEntry = await entry.save()
    res.status(200).json(newEntry)
})

app.delete("/delete", async (req, res) => {
    const entry = await Entry.findByIdAndDelete(req.body._id)
    res.status(200).json(entry)
})
```

Github link to the Backend: https://github.com/prakhar0912/BioDB-JComp-Backend

Database: The database is a MongoDB database which is a document database, which means it stores data in JSON-like documents. We believe this is the most natural way to think about data, and is much more expressive and powerful than the traditional row/column model making it ideal for this Biological project.

A visual representation of the database:

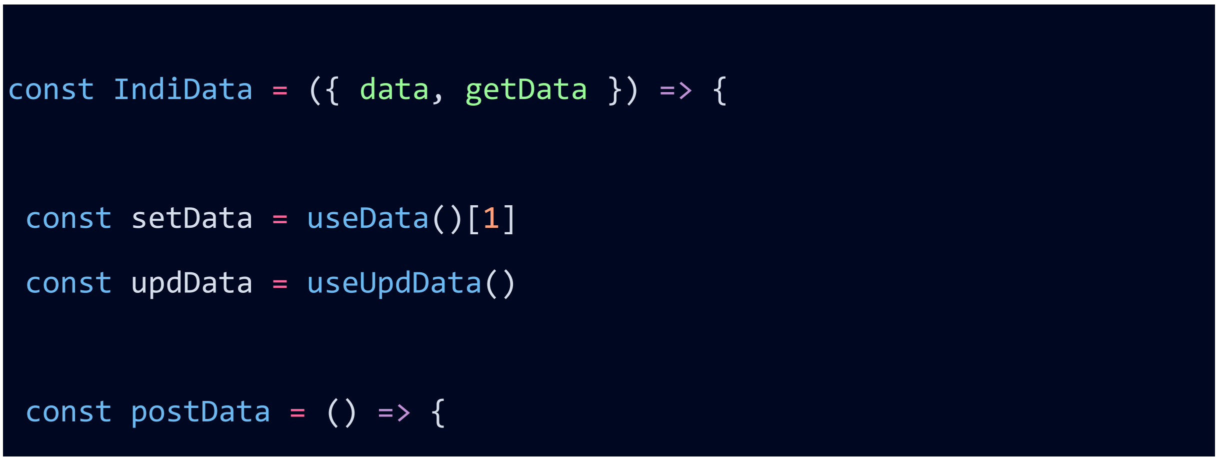


```
_id: ObjectId("60affba24eded42cc43a9932")
Gene_Name: "PIK3CA"
Transcript: "ENST00000263967.3"
Census_Tier_1: "Yes"
Sample_Name: "2689723"
Sample_ID: "2689723"
AA_Mutation: "p.P124S"
CDS_Mutation: "c.370C>T"
Primary_Tissue: "Breast"
Tissue_Subtype_1: "NS"
Tissue_Subtype_2: "NS"
Histology: "Carcinoma"
Histology_Subtype_1: "Neuroendocrine carcinoma"
Histology_Subtype_2: "NS"
Pubmed_Id: "27925203"
CGP_Study: "-"
Somatic_Status: "Confirmed Somatic"
Sample_Type: "Tumour Sample"
Zygosity: "Unknown"
Genomic_Coordinates: "3:179199707..179199707"
__v: 0

_id: ObjectId("60affba24eded42cc43a9933")
Gene_Name: "PIK3CA"
Transcript: "ENST00000263967.3"
Census_Tier_1: "Yes"
Sample_Name: "2689726"
Sample_ID: "2689726"
AA_Mutation: "p.H1047R"
CDS_Mutation: "c.3140A>G"
Primary_Tissue: "Breast"
Tissue_Subtype_1: "NS"
Tissue_Subtype_2: "NS"
Histology: "Carcinoma"
Histology_Subtype_1: "Neuroendocrine carcinoma"
Histology_Subtype_2: "NS"
Pubmed_Id: "27925203"
CGP_Study: "-"
Somatic_Status: "Confirmed Somatic"
Sample_Type: "Tumour Sample"
Zygosity: "Unknown"
Genomic_Coordinates: "3:179234297..179234297"
__v: 0

_id: ObjectId("60affba24eded42cc43a9934")
Gene_Name: "PIK3CA"
Transcript: "ENST00000263967.3"
```

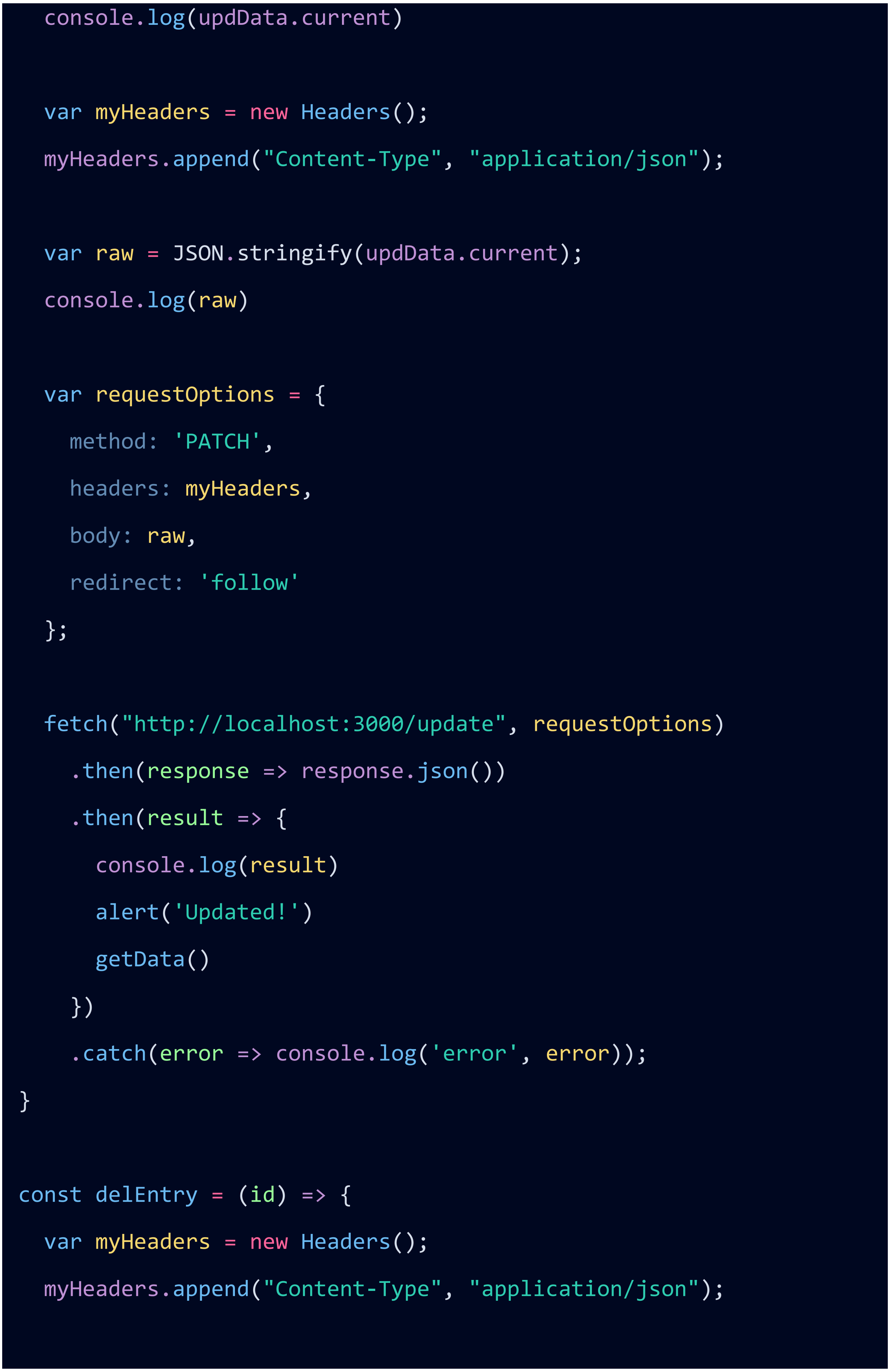
The Frontend:

The frontend to interact in a meaningful way with the database.

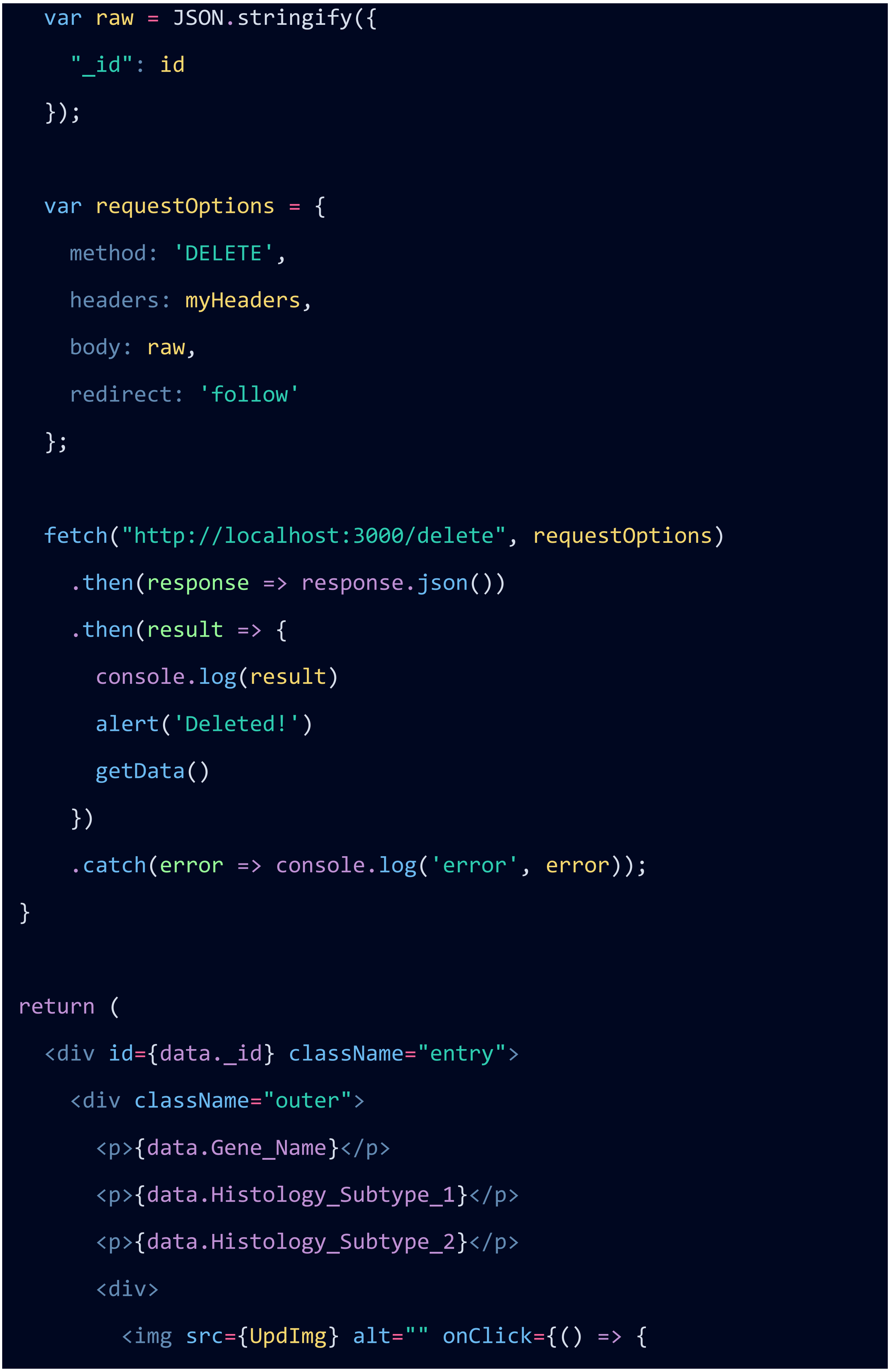
The frontend was made using the popular JavaScript Frontend Framework React, which is a declarative, efficient, and flexible JavaScript library for building user interfaces. It lets you compose complex UIs from small and isolated pieces of code called "components". One such component that we have used is a functional component with the following code snippet:



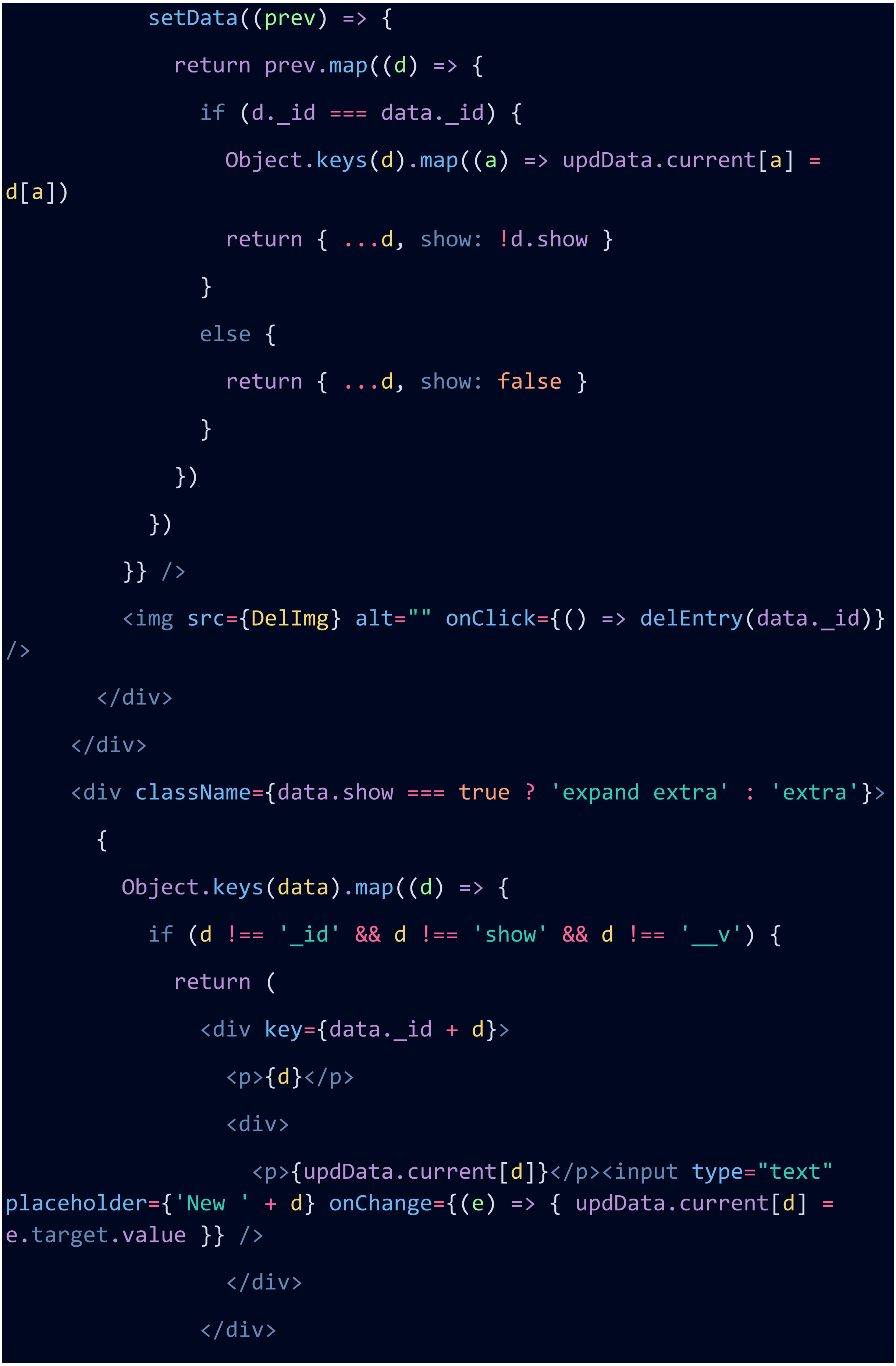
```
const IndiData = ({ data, getData }) => {

 const setData = useData()[1]
 const updData = useUpdData()

 const postData = () => {
```



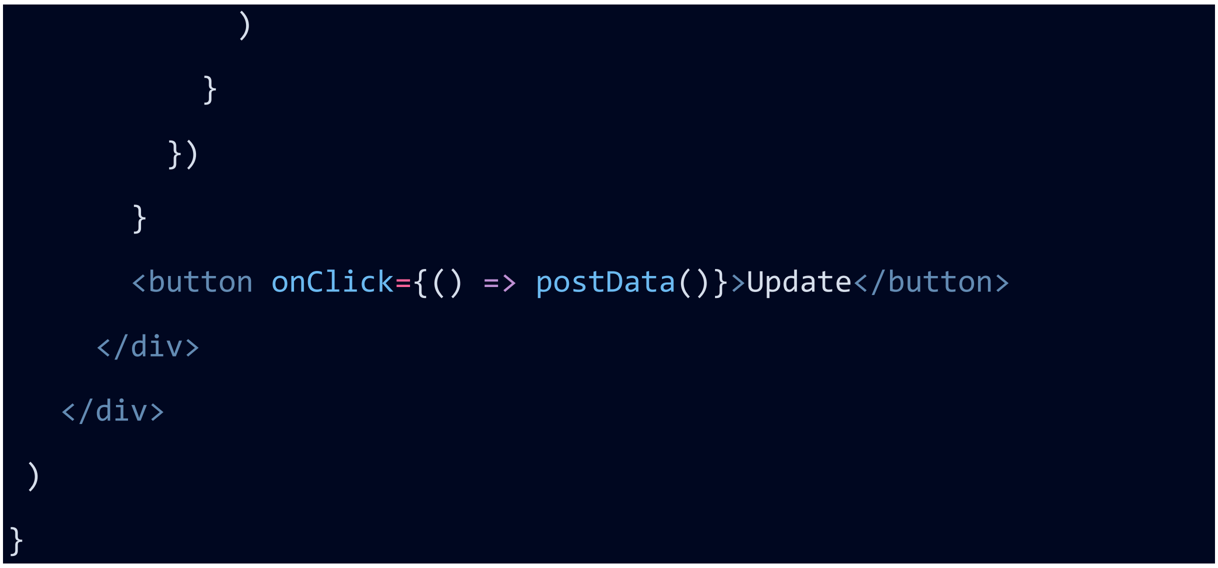
```
  console.log(updData.current)

  var myHeaders = new Headers();
  myHeaders.append("Content-Type", "application/json");

  var raw = JSON.stringify(updData.current);
  console.log(raw)

  var requestOptions = {
    method: 'PATCH',
    headers: myHeaders,
    body: raw,
    redirect: 'follow'
  };

  fetch("http://localhost:3000/update", requestOptions)
    .then(response => response.json())
    .then(result => {
      console.log(result)
      alert('Updated!')
      getData()
    })
    .catch(error => console.log('error', error));
}

const delEntry = (id) => {
  var myHeaders = new Headers();
  myHeaders.append("Content-Type", "application/json");
```



```
  var raw = JSON.stringify({
    "_id": id
  });

  var requestOptions = {
    method: 'DELETE',
    headers: myHeaders,
    body: raw,
    redirect: 'follow'
  };

  fetch("http://localhost:3000/delete", requestOptions)
    .then(response => response.json())
    .then(result => {
      console.log(result)
      alert('Deleted!')
      getData()
    })
    .catch(error => console.log('error', error));
}

return (
  <div id={data._id} className="entry">
    <div className="outer">
      <p>{data.Gene_Name}</p>
      <p>{data.Histology_Subtype_1}</p>
      <p>{data.Histology_Subtype_2}</p>
      <div>
        <img src={UpdImg} alt="" onClick={() => {
```



```
           setData((prev) => {
             return prev.map((d) => {
               if (d._id === data._id) {
                 Object.keys(d).map((a) => updData.current[a] =
d[a])
                 return { ...d, show: !d.show }
               }
               else {
                 return { ...d, show: false }
               }
             })
           })
         }} />
         <img src={DelImg} alt="" onClick={() => delEntry(data._id)}
/>
       </div>
     </div>
     <div className={data.show === true ? 'expand extra' : 'extra'}>
       {
         Object.keys(data).map((d) => {
           if (d !== '_id' && d !== 'show' && d !== '__v') {
             return (
               <div key={data._id + d}>
                 <p>{d}</p>
                 <div>
                   <p>{updData.current[d]}</p><input type="text"
placeholder={'New ' + d} onChange={(e) => { updData.current[d] =
e.target.value }} />
                 </div>
               </div>
```



```
             )
           }
         })
       }
       <button onClick={() => postData()}>Update</button>
     </div>
   </div>
 )
}
```

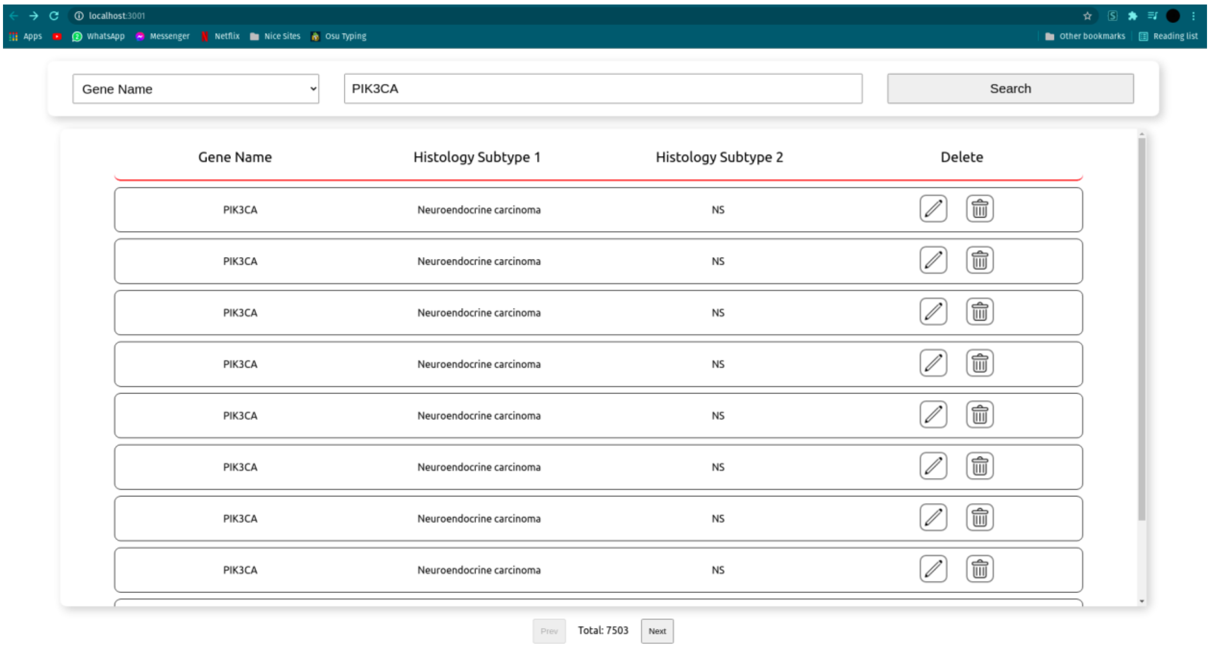
Functions provided via the frontend and backend that have been integrated :

1. Append data to the database using raw data
2. Append data to the database using a COSMIC Url
3. Alter/Update the data of any entry
4. Search for data based on any parameter such as
   a. Gene_Name
   b. Transcript
   c. Census_Tier_1
   d. Sample_Name
   e. Sample_ID
   f. AA_Mutation
   g. CDS_Mutation
   h. Primary_Tissue
   i. Tissue_Subtype_1
   j. Tissue_Subtype_2
   k. Histology
   l. Histology_Subtype_1
   m. Histology_Subtype_2
   n. Pubmed_Id
   o. CGP_Study
   p. Somatic_Status
   q. Sample_Type
   r. Zygosity
   s. Genomic_Coordinates
5. Paginated View of the data
6. Delete any data entry

Github Link to the frontend: https://github.com/prakhar0912/BioDB-JComp-Frontend

Screenshots of the Frontend have been provided below:

Frontend to View Data:

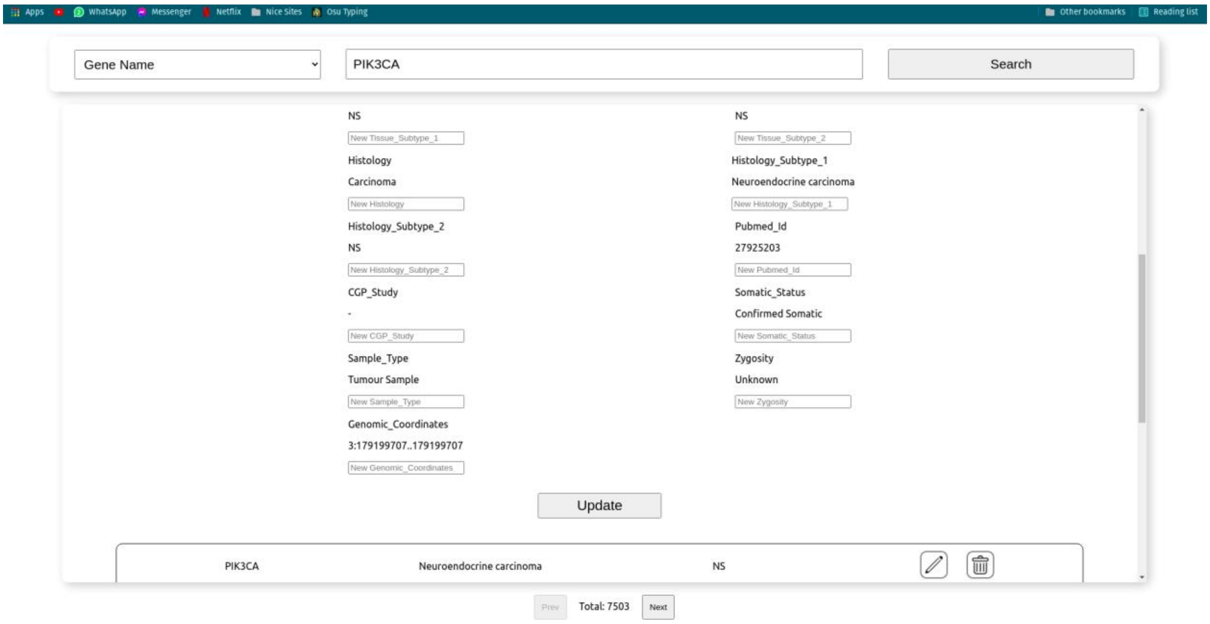


Gene Name | PIK3CA | Search

Gene Name	Histology Subtype 1	Histology Subtype 2	Delete
PIK3CA	Neuroendocrine carcinoma	NS	
PIK3CA	Neuroendocrine carcinoma	NS	
PIK3CA	Neuroendocrine carcinoma	NS	
PIK3CA	Neuroendocrine carcinoma	NS	
PIK3CA	Neuroendocrine carcinoma	NS	
PIK3CA	Neuroendocrine carcinoma	NS	
PIK3CA	Neuroendocrine carcinoma	NS	
PIK3CA	Neuroendocrine carcinoma	NS	

Prev Total: 7503 Next

Altering Data Using Frontend:



Gene Name | PIK3CA | Search

NS
New Tissue_Subtype_1
Histology
Carcinoma
New Histology
Histology_Subtype_2
NS
New Histology_Subtype_2
CGP_Study
-
New CGP_Study
Sample_Type
Tumour Sample
New Sample_Type
Genomic_Coordinates
3:179199707..179199707
New Genomic_Coordinates

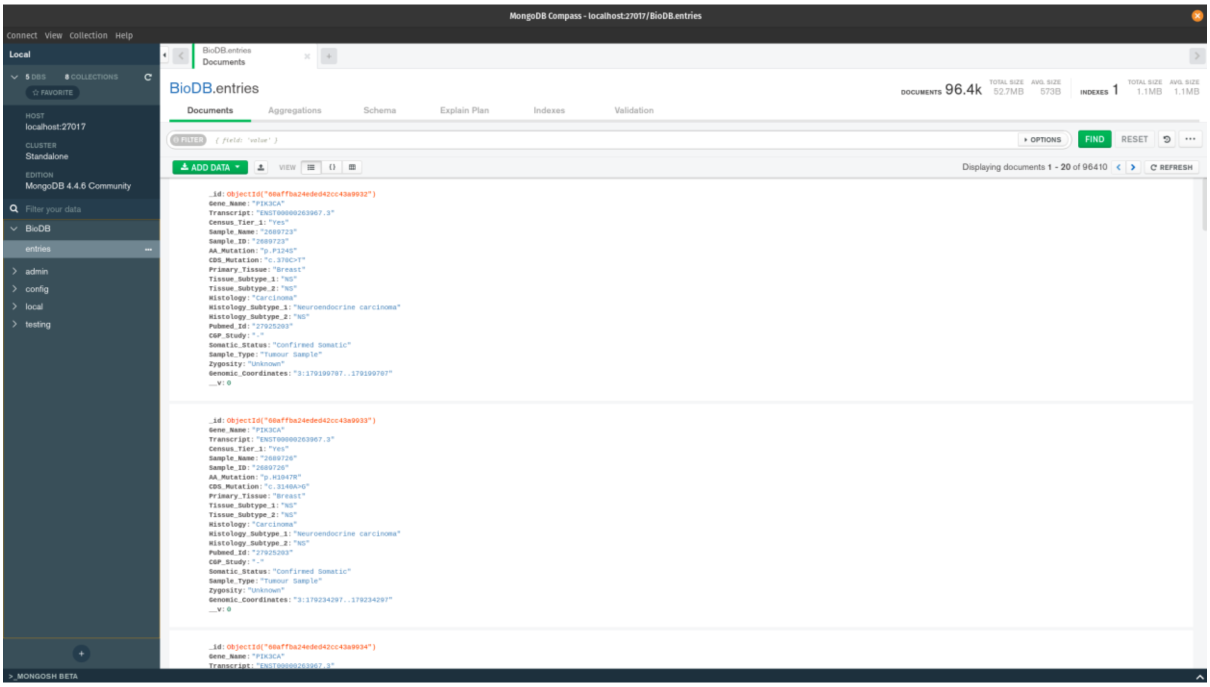
NS
New Tissue_Subtype_2
Histology_Subtype_1
Neuroendocrine carcinoma
New Histology_Subtype_1
Pubmed_Id
27925203
New Pubmed_Id
Somatic_Status
Confirmed Somatic
New Somatic_Status
Zygosity
Unknown
New Zygosity

Update

PIK3CA Neuroendocrine carcinoma NS

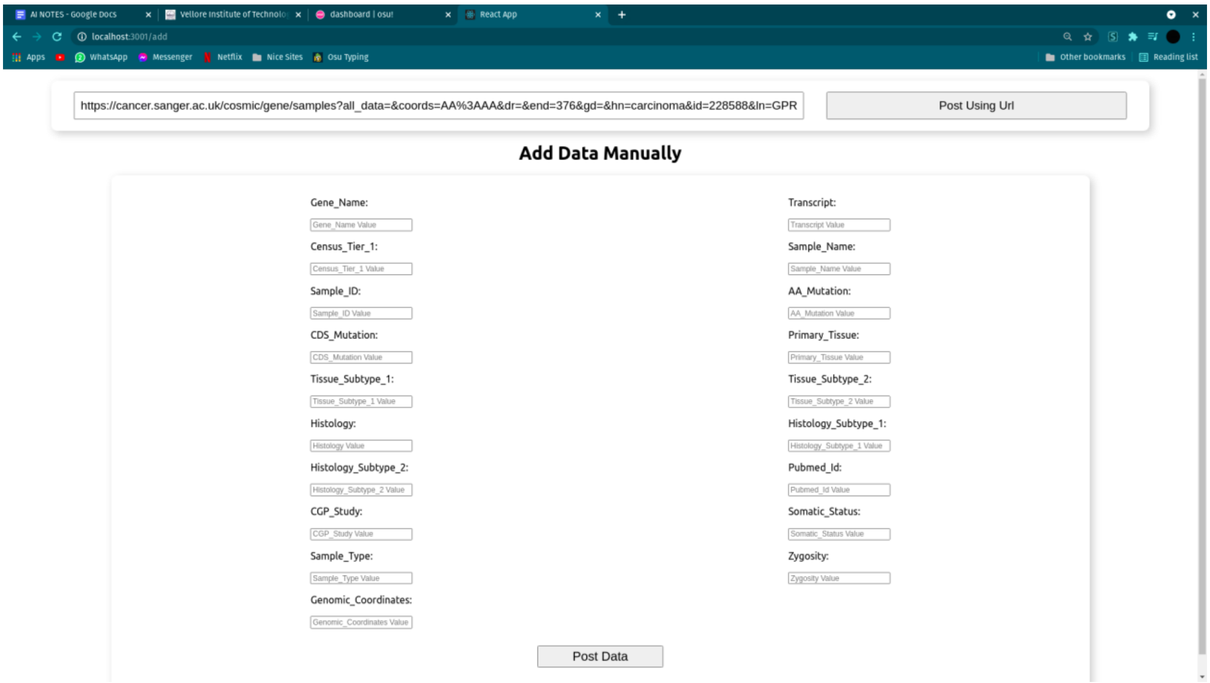
Prev Total: 7503 Next

Database:



```
_id: ObjectId("60affba24eded42cc43a9932")
Gene_Name: "PIK3CA"
Transcript: "ENST00000263967.3"
Census_Tier_1: "Yes"
Sample_Name: "2689723"
Sample_ID: "2689723"
AA_Mutation: "p.P124S"
CDS_Mutation: "c.370C>T"
Primary_Tissue: "Breast"
Tissue_Subtype_1: "NS"
Tissue_Subtype_2: "NS"
Histology: "Carcinoma"
Histology_Subtype_1: "Neuroendocrine carcinoma"
Histology_Subtype_2: "NS"
Pubmed_Id: "27925203"
CGP_Study: "-"
Somatic_Status: "Confirmed Somatic"
Sample_Type: "Tumour Sample"
Zygosity: "Unknown"
Genomic_Coordinates: "3:179199707..179199707"
__v: 0

_id: ObjectId("60affba24eded42cc43a9933")
Gene_Name: "PIK3CA"
Transcript: "ENST00000263967.3"
Census_Tier_1: "Yes"
Sample_Name: "2689726"
Sample_ID: "2689726"
AA_Mutation: "p.H1047R"
CDS_Mutation: "c.3140A>G"
Primary_Tissue: "Breast"
Tissue_Subtype_1: "NS"
Tissue_Subtype_2: "NS"
Histology: "Carcinoma"
Histology_Subtype_1: "Neuroendocrine carcinoma"
Histology_Subtype_2: "NS"
Pubmed_Id: "27925203"
CGP_Study: "-"
Somatic_Status: "Confirmed Somatic"
Sample_Type: "Tumour Sample"
Zygosity: "Unknown"
Genomic_Coordinates: "3:179234297..179234297"
__v: 0

_id: ObjectId("60affba24eded42cc43a9934")
Gene_Name: "PIK3CA"
Transcript: "ENST00000263967.3"
```

Adding Data Using Frontend:



https://cancer.sanger.ac.uk/cosmic/gene/samples?all_data=&coords=AA%3AAA&dr=&end=376&gd=&hn=carcinoma&id=228588&ln=GPR

Add Data Manually

Gene_Name: Gene_Name Value

Transcript: Transcript Value

Census_Tier_1: Census_Tier_1 Value

Sample_Name: Sample_Name Value

Sample_ID: Sample_ID Value

AA_Mutation: AA_Mutation Value

CDS_Mutation: CDS_Mutation Value

Primary_Tissue: Primary_Tissue Value

Tissue_Subtype_1: Tissue_Subtype_1 Value

Tissue_Subtype_2: Tissue_Subtype_2 Value

Histology: Histology Value

Histology_Subtype_1: Histology_Subtype_1 Value

Histology_Subtype_2: Histology_Subtype_2 Value

Pubmed_Id: Pubmed_Id Value

CGP_Study: CGP_Study Value

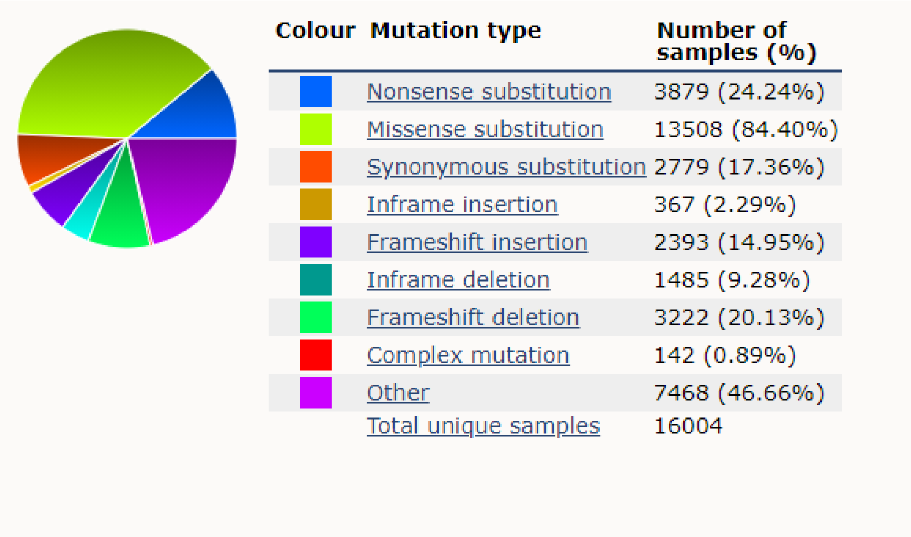
Somatic_Status: Somatic_Status Value

Sample_Type: Sample_Type Value

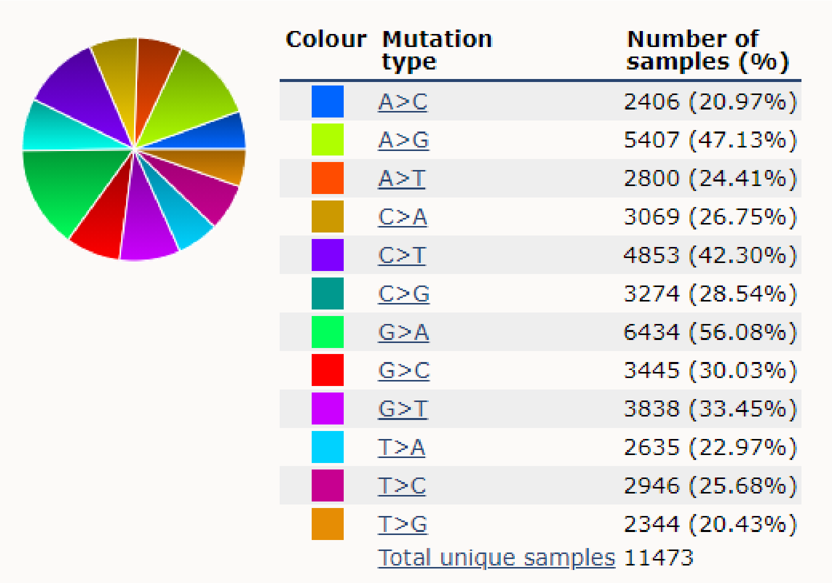
Zygosity: Zygosity Value

Genomic_Coordinates: Genomic_Coordinates Value

Distribution of Substitutions and Mutations:



Colour	Mutation type	Number of samples (%)
🟦	Nonsense substitution	3879 (24.24%)
🟩	Missense substitution	13508 (84.40%)
🟧	Synonymous substitution	2779 (17.36%)
🟨	Inframe insertion	367 (2.29%)
🟪	Frameshift insertion	2393 (14.95%)
🟩	Inframe deletion	1485 (9.28%)
🟩	Frameshift deletion	3222 (20.13%)
🟥	Complex mutation	142 (0.89%)
🟪	Other	7468 (46.66%)
	Total unique samples	16004



Colour	Mutation type	Number of samples (%)
🟦	A>C	2406 (20.97%)
🟩	A>G	5407 (47.13%)
🟧	A>T	2800 (24.41%)
🟨	C>A	3069 (26.75%)
🟪	C>T	4853 (42.30%)
🟩	C>G	3274 (28.54%)
🟩	G>A	6434 (56.08%)
🟥	G>C	3445 (30.03%)
🟪	G>T	3838 (33.45%)
🟦	T>A	2635 (22.97%)
🟪	T>C	2946 (25.68%)
🟧	T>G	2344 (20.43%)
	Total unique samples	11473

As seen in the above example, majority of the mutations are of the missense substitution type (substitution resulting in an alternate codon, altering the amino acid at this position only), caused by an SNP (single nucleotide polymorphism) from Guanine to Adenine.

Result:

From the data present in COSMIC, we were able to analyse as well as scrape the data to have it more easily accessible. The data given to us mentions the gene and sample name, as well as the tissue, the histology, and their subtypes. It also includes the Pubmed ID for easy access, as well as the zygosity and the genomic coordinates. Most importantly, the AA mutations (change in peptide sequence) and CDS mutations (change in the nucleotide sequence) are mentioned in the .csv file we obtain. The Cancer Browser is a powerful tool that allows us to find many genes and their mutations with one search, and we have created a Database to harness this aspect of COSMIC with an accompanying frontend to accommodate various functions that can be performed onto the created Database.

References:

1. https://stackoverflow.com/questions/20986631/how-can-i-scroll-a-web-page-using-selenium-webdriver-in-python
2. https://www.tutorialspoint.com/fetch-all-href-link-using-selenium-in-python#:~:text=We%20can%20fetch%20href%20links,use%20the%20method%20find_elements_by_tag_name().
3. https://stackoverflow.com/questions/33939198/selenium-webdriver-finding-elements-using-cssselector-and-nth-child
4. https://stackoverflow.com/questions/18600391/selenium-python-selecting-via-css-selector
5. https://www.geeksforgeeks.org/find_elements_by_class_name-driver-method-selenium-python/
6. https://www.youtube.com/watch?v=J82SxHP5SWY
7. https://cancer.sanger.ac.uk/cosmic/gene/samples?all_data=&coords=AA%3AAA&dr=&end=596&gd=&hn=carcinoma&id=272048&ln=ESR1_ENST00000206249&seqlen=596&sh=luminal_NS_carcinoma&sn=breast&src=gene&ss=NS&start=1#complete
8. https://cancer.sanger.ac.uk/cosmic/gene/analysis?hn=all&ln=PIK3CA_ENST00000643187&sh=all&sn=breast&ss=all
9. https://cancer.sanger.ac.uk/cosmic/gene/analysis?hn=all&ln=PIK3CA&sh=all&sn=breast&ss=all
10. Github Link to the frontend: https://github.com/prakhar0912/BioDB-JComp-Frontend
11. Github link to the Backend: https://github.com/prakhar0912/BioDB-JComp-Backend