CD5108 Covid Tracker Project



By: Olatunji Oguntoyinbo (U1728854) and Zeeshan Khan (U1728552)

<https://github.com/thehbk97/COVID-Tracker>

**Contents**

**Introduction ……………………………………………………………………**

**What is RESTAPI………………………………………………………………**

**Principles of RESTAPI……………………………………………………….**

**Express.js…………………………………………………………………………**

**Project Plan………………………………………………………………………**

**Screenshots………………………………………………………………………**

**Version Control…………………………………………………………………..**

**Git hub………………………………………………………………………………..**

**Project Reflection………………………………………………………………..**

**Reference…………………………………………………………………………….**

**Introduction**

For this project we was required to develop a web server application using node js express to design the RESTAPI .The RESTAPI should have endpoint to perform following task

* Adding data to the Covid collection.
* It should also have an separate endpoint for update case, death and date for a given state and country, the updated record should be displayed on the browser
* It should show separate endpoint to show total number of cases and deaths for a given state and country
* A separate endpoint for deleting a document for given state and country.
* it should have an endpoint to display first 20 documents from the covid database for a given date and state
* It should have an endpoint to display the states where Cases are more than 1 in a single day
* OS information

**What is RESTAPI ?**

An API simply means “application programming interface”. REST “Representational State Transfer”. is what defines what the API looks like. REST API creates an object, and then sends the values of an object in response to the client.

REST offer certain operations through which a client can adjust or access the resource from the server.

These operations are called as CRUD which simply means

• Create

• Read

• Update

• Delete

To perform CRUD operations, a client (the web browser) must perform a request operation which is manged by the Request object of the HTTP protocol.

The method used to make a request on HTTP are

* Get - retrieve a specific or a collection of resources
* Post – create a new resource
* Put - update a specific resource
* Delete - remove a specific resource

**Principle of RESTAPI**

Stateless: It is connectionless as it is built on a request/response mode of operation. Web browsers makes a request to the server for images, fonts, content etc. Once the request is completed the connection between the browser and server is severed.

Layered system: The layered system design allows an application to be more stable by restricting component behaviour. This type of design helps in improving the application’s security as components in each layer cannot interact beyond the next direct layer, they are in.

**Express.js**

Express.js is a web application framework that is provides a minimal interface with all the tools required to build a web application. Express.js adds flexibility to an application with a range of modules available on npm. It improves easy management of the flow of data between server and routes in the server-side applications.

**Project assumption**

For this project there are some assumption we must ensure are in place before we can begin the project.

Correct and Valid data: it is important that we valid the data we will be using and ensure that the data is accurate.

Git user account: All member of the team must ensure that they are registered and have access to the repository to ensure that we can simultaneously work on the project.

Software Installation: As a team we must ensure that we have all the required software required to develop and manage our web server application

**Design methodology**

For this project we have decided to take the Agile approach. This design methodology allows us to Focus on the code rather than the design, it is important for this application that we put an emphasis on functionality rather than design, and Agile allows us to do that in the most effective way. Agile is also very flexible allowing quick responses to changing requirements with less work. Agile also support incremental and iterative software development allowing for constant development and a structured release and update process.

**Project requirement**

For this project we needed to ensure that our web server has specific endpoints. In total we decided on 8 endpoint each with different information and functionality.

**Endpoints**

**/Covidinfo:** This endpoint will display all the data in our Coviddatabase.

**/addinfo:** This endpoint will used to add new entries unto the database

**/findtwenty:** This endpoint will return the first 20 result given the state and date

**/findmorethanone:** This endpoint will retrieve entries where cases are greater than one on a given day

**/delete:** This will delete an entry at a given county or state

**/update:** This will update an entry at a given county or state

**/osinfo:** This will retrieve all the current OS information.

**Project Body**

We started off the project by setting up a meeting and working through the specification to ensure that we both understood what we needed to do as a group. From here we discussed what our individual strengths were and what we could each work on. Zeeshan had some previous experience in coding so he opted to start working on that aspect whereas Olatunji had done some work with databases previously so he decided to start with creating the database.

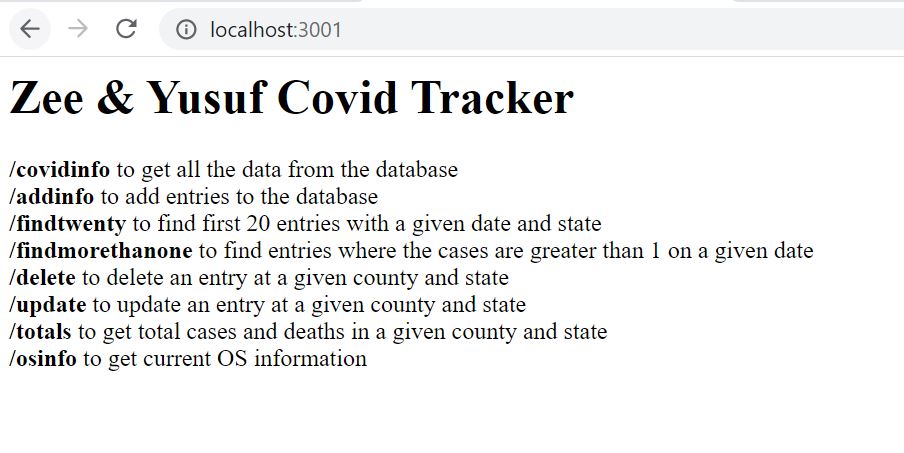
We both had past experiences of similar technologies we had never specifically worked with JavaScript, node, express or mongo DB.

Using mongo DB we created a database and added the data from this supplied CSV. We converted the CSV to JSON as we found it easier to use and we were planning to use JSON data in our programme so we wanted to use the same data type.

After this we created a simple web server and try to give it different endpoints to show different information to the user. We found that we started to do some incorrect coding by not using the correct HTTP protocols and had to rewrite the code using these protocols. We set up the different endpoints to show the user a dynamic form that they could fill in, this was done by using the get method. Once the user fills in and submits the form this posts the data to the server where we generate the reply and send that back again to the user. This is the essential dynamic that we used for all the different operations in our program.

**Project Screenshots**

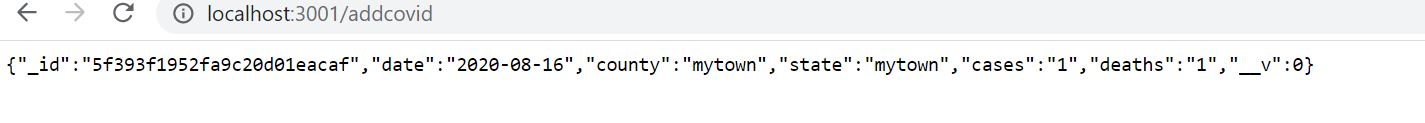
**Homepage**



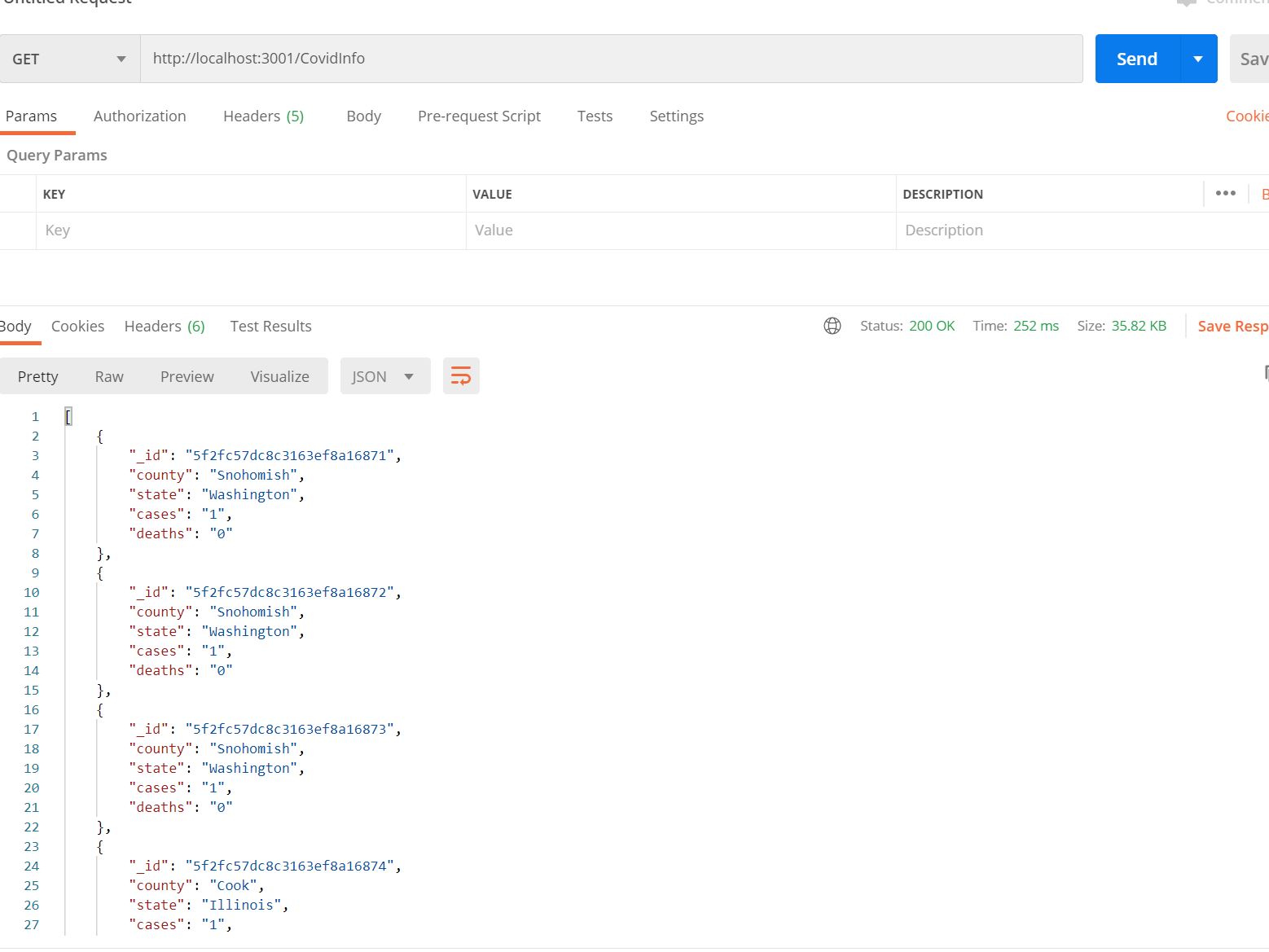
**Add data Endpoint**



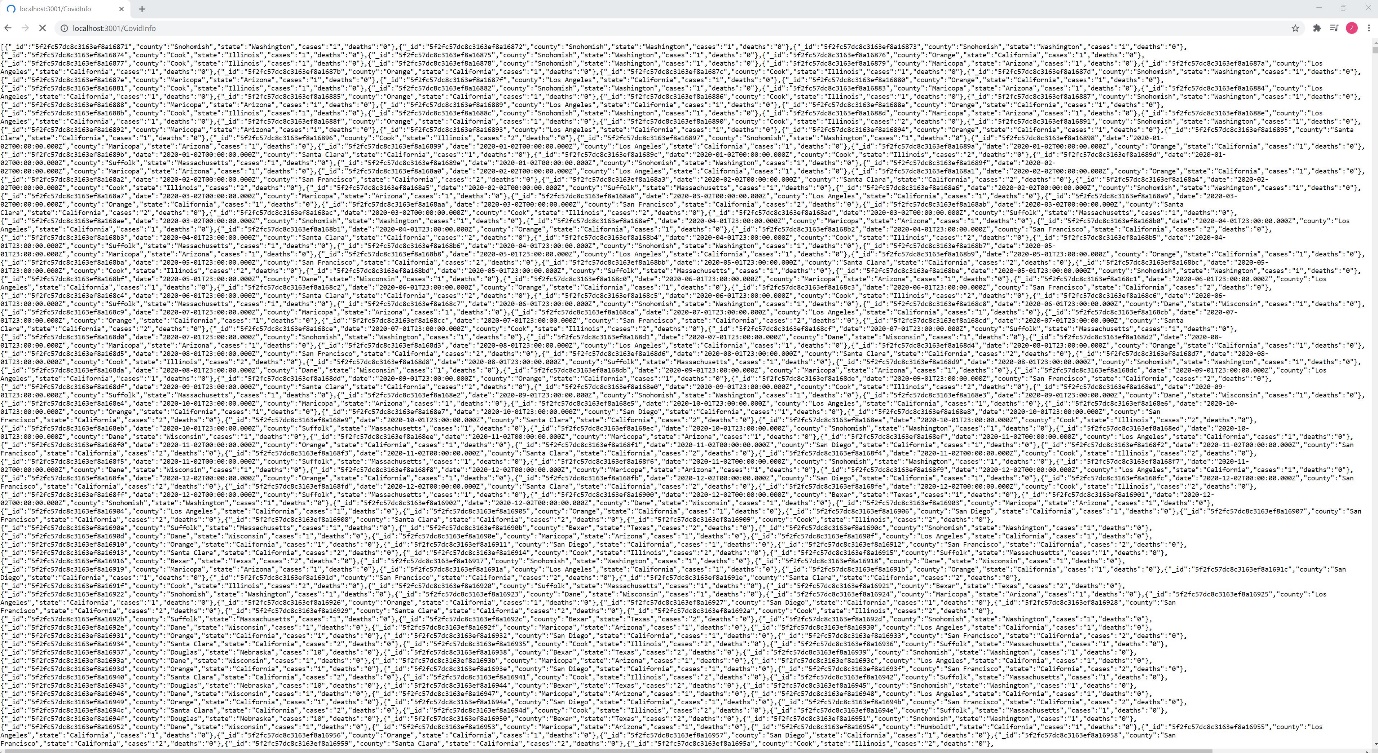
**Added data**



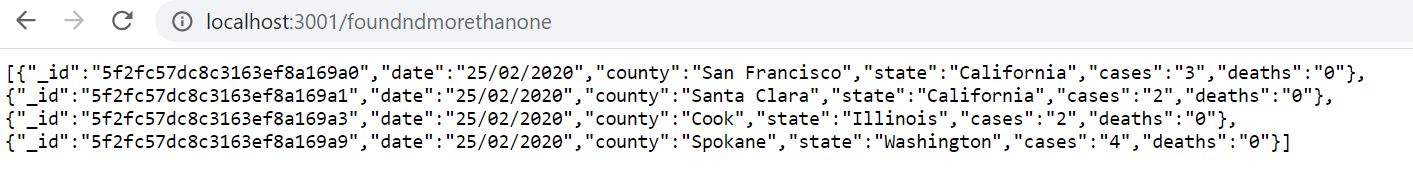
**Added data Postman**



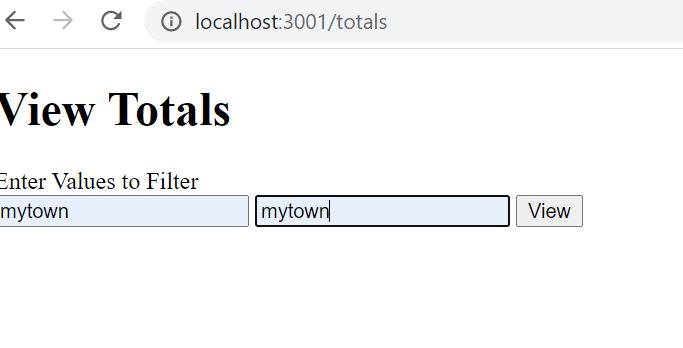
**All Data**



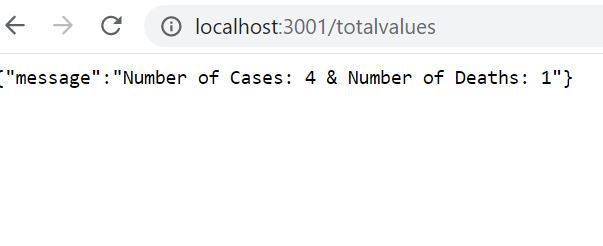
**Cases more than 1**



**Totals Page**



**Total Result**

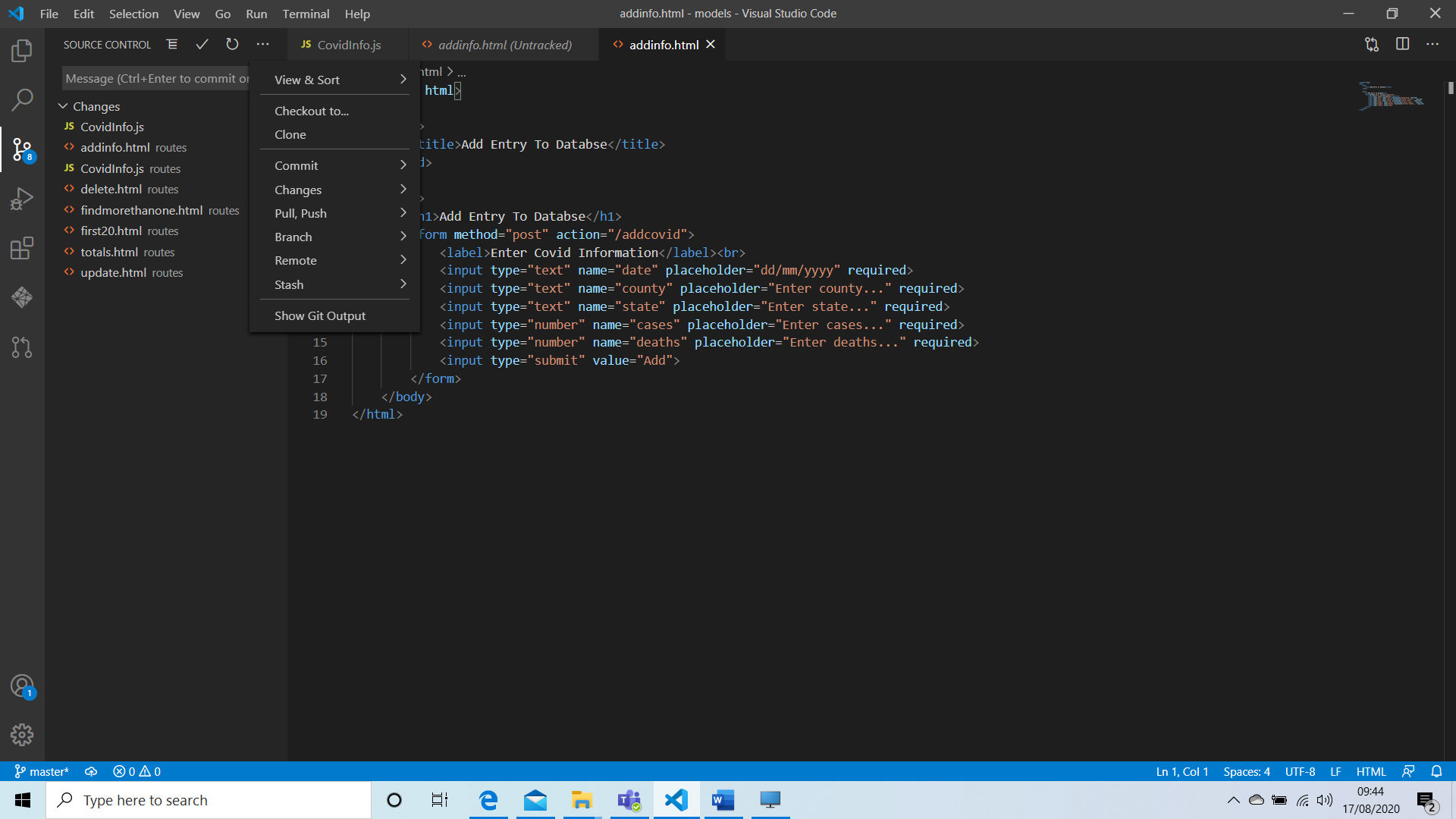


**Version Control**

For this project we decided to use GitHub for our version Control, Software developers use distribution version management systems to allow them to simultaneously work and update a project and track different version.  A master repository would be used to house the main file while users can push various update to different branches.

<https://github.com/thehbk97/Covid-Tracker>

Using this we were able to simultaneously work in parallel and it allowed to quickly review changes and apply them to the overall code. It also allowed us to maintain accurate record of new changes and who made those changes. This was important for us when reflecting as it made it easier to identify and issues and work in cohesion to resolve them.



GitHub effectively also assisted in reviewing our work. It helps us to ensure that we maintained frequent communication and also allowed us to view raw code that we were working and also allowed us to rollback any changes or commits we were not happy with as a collective and allowed us to update and fix any bugs we had.

**Mongo DB**

What is Mongo dB ?

MongoDB is an open source database that uses a document-oriented data model. MongoDB is built on an architecture of collections. Terms used in mongoDB are: Collection, Documents, Fields, schema and Models.

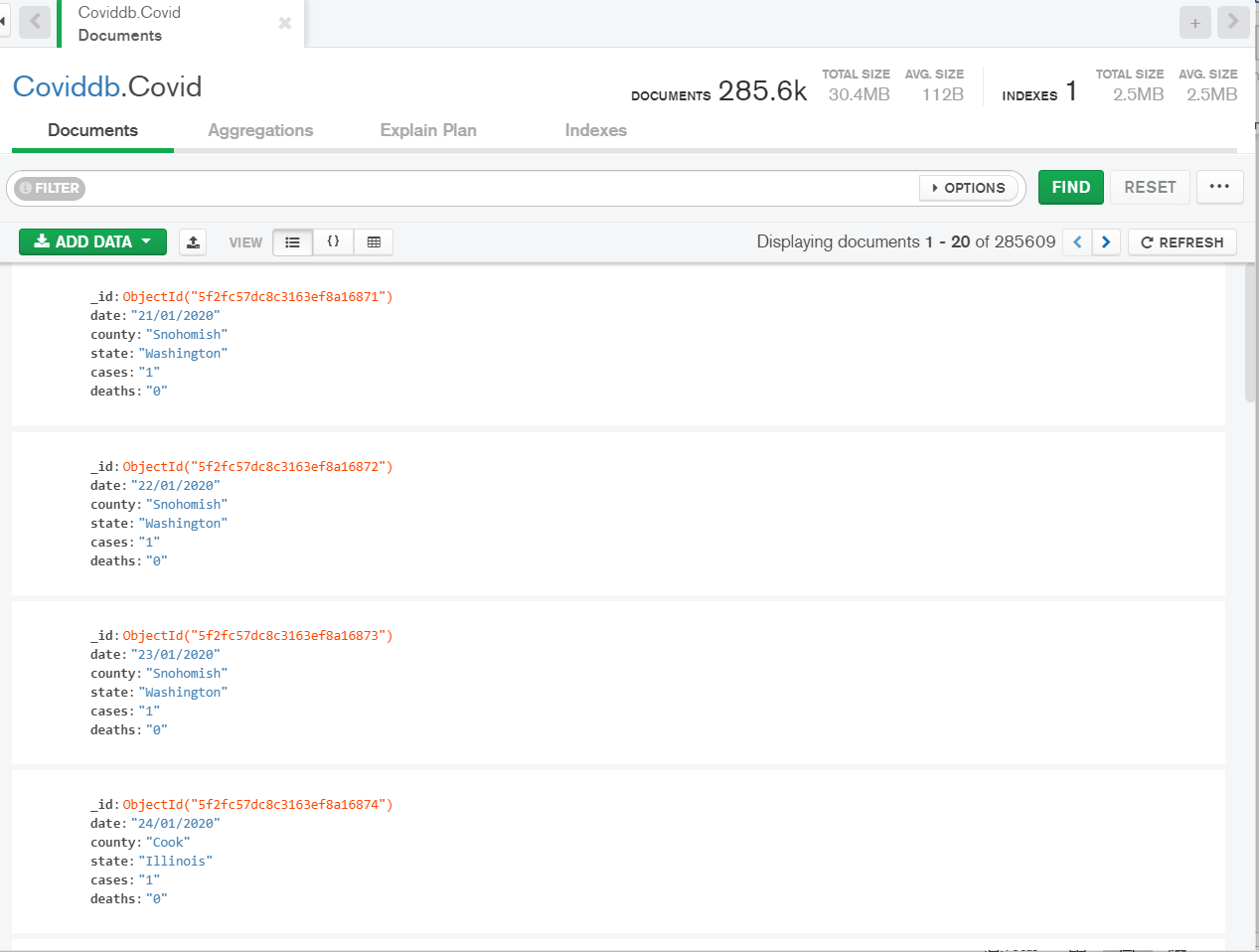
Collection: Mongo dB uses collections these are equivalent to a table in relational database

Documents: Mongo dB is a document orientated database, documents are equivalent to record or rows in a relational database.

Schema: A Mongo db ‘schema’ is a document data structure (or shape of the document) that is implemented via the application layer.

Models: Models are constructors that take a schema and create an instance of a document equivalent to records in a relational database.

Fields: These are like columns in a SQL table



Covid database on Mongo db

**Java Script**

Covidinfo.js

const mongoose = require('mongoose')

const covidSchema = new mongoose.Schema({

  date: {

    type: String,

    required: true,

  },

  county: {

    type: String,

    required: true

  },

  state: {

    type: String,

    required: true,

  },

  cases: {

    type: String,

    required: true

  },

  deaths: {

    type: String,

    required: true

  }

}, {collection: 'Covid'})

module.exports = mongoose.model('covid', covidSchema)

**CovidInfo.JS (Route)**

const express = require('express')

var os = require('os');

var util = require('util');

const router = express.Router()

const covid = require('../models/CovidInfo')

// Homepage

router.get('/', async (req, res) => {

  HomeInfo = "<html><body><h1>Zee & Yusuf Covid Tracker</h1></body>"

  HomeInfo += "<label><b>/covidinfo</b>  to get all the data from the database</label><br>"

  HomeInfo += "<label><b>/addinfo</b>  to add entries to the database</label><br>"

  HomeInfo += "<label><b>/findtwenty</b>  to find first 20 entries with a given date and state</label><br>"

  HomeInfo += "<label><b>/findmorethanone</b>  to find entries where the cases are greater than 1 on a given date</label><br>"

  HomeInfo += "<label><b>/delete</b>  to delete an entry at a given county and state</label><br>"

  HomeInfo += "<label><b>/update</b>  to update an entry at a given county and state</label><br>"

  HomeInfo += "<label><b>/totals</b>  to get total cases and deaths in a given county and state</label><br>"

  HomeInfo += "<label><b>/osinfo</b>  to get current OS information</label><br>"

  res.send(HomeInfo)

})

// Getting all data from db

router.get('/covidinfo', async (req, res) => {

    try {

      const covidData = await covid.find({})

      res.json(covidData)

    } catch (err) {

      res.status(500).json({ message: err.message })

    }

  })

//Page to Add Data

router.get('/addinfo', async (req, res) => {

  res.sendFile(\_\_dirname + "/addinfo.html")

})

//Add Data POST Method

router.post('/addcovid', async (req, res) => {

  const entry = new covid({

    date: req.body.date,

    county: req.body.county,

    state: req.body.state,

    cases: req.body.cases,

    deaths: req.body.deaths

  })

  try {

    const newEntry = await entry.save()

    res.status(201).json(newEntry)

  } catch (err) {

    res.status(400).json({ message: err.message })

  }

})

// Getting first 20

router.get('/findtwenty', async (req, res) => {

  res.sendFile(\_\_dirname + "/first20.html")

})

// Getting first 20 POST

router.post('/foundtwenty', async (req, res) => {

  try {

    const covidData = await covid.find({date: req.body.date, state: req.body.state}).limit(20)

    res.json(covidData)

  } catch (err) {

    res.status(500).json({ message: err.message })

  }

})

// Getting more than 1 case

router.get('/findmorethanone', async (req, res) => {

  res.sendFile(\_\_dirname + "/findmorethanone.html")

})

// Getting more than 1 case POST

router.post('/foundndmorethanone', async (req, res) => {

  try {

    const covidData = await covid.find({date: req.body.date, cases: {$gte:"2"}})

    res.json(covidData)

  } catch (err) {

    res.status(500).json({ message: err.message })

  }

})

// Deleting page display

router.get('/delete', async (req, res) => {

  res.sendFile(\_\_dirname + "/delete.html")

})

// Deleting Method

router.post('/deletevalues', async (req, res) => {

  try {

    values = await covid.findOneAndDelete({county: req.body.county, state: req.body.state})

    covid.save

    res.json({ message: 'Deleted One Entry', values })

  } catch (err) {

    res.status(500).json({ message: err.message })

  }

})

// Updating page display

router.get('/update', async (req, res) => {

  res.sendFile(\_\_dirname + "/update.html")

})

// Deleting Method

router.post('/updatevalues', async (req, res) => {

  try {

    values = await covid.findOneAndUpdate({county: req.body.county, state: req.body.state}, {cases: req.body.cases, deaths: req.body.deaths, date: req.body.date}, {new: true})

    covid.save

    res.json({ message: 'Updated One Entry', values })

  } catch (err) {

    res.status(500).json({ message: err.message })

  }

})

// Totals page display

router.get('/totals', async (req, res) => {

  res.sendFile(\_\_dirname + "/totals.html")

})

// Deleting Method

router.post('/totalvalues', async (req, res) => {

  try {

    var cases = 0

    var deaths = 0

    values = await covid.find({county: req.body.county, state: req.body.state})

    values.forEach(function(entry) {

      if(entry["\_doc"]["cases"]){

        cases = cases + parseInt(entry["\_doc"]["cases"])

      }

      if(entry["\_doc"]["deaths"]){

        deaths = deaths + parseInt(entry["\_doc"]["deaths"])

      }

    });

    res.json({ message: `Number of Cases: ${cases} & Number of Deaths: ${deaths}`})

  } catch (err) {

    res.status(500).json({ message: err.message })

  }

})

// Getting OS data

router.get('/osinfo', async (req, res) => {

  try {

    sysInfo = "<html><head><title>Operating System Info</title></head>"

    sysInfo += "<body><h1>Operating System Info</h1>"

    sysInfo += "<table>"

    sysInfo += "<tr><th> Tmp Dir </th><td>"

    sysInfo += os.tmpdir()

    sysInfo += "</td></tr>"

    sysInfo += "<tr><th> Host Name </th><td></td>"

    sysInfo += os.hostname()

    sysInfo += "</td></tr>"

    sysInfo += "<tr><th> OS Type</th><td>"

    sysInfo += os.type()

    sysInfo += os.platform()

    sysInfo += os.arch()

    sysInfo += os.release()

    sysInfo += '</td></tr>'

    sysInfo += '<tr><th> Uptime </th><td>'

    sysInfo += (os.uptime())/3600

    sysInfo += 'hours. userInfo'

    sysInfo += util.inspect(os.userInfo())

    sysInfo += '</td></tr>'

    sysInfo += '<tr><th> Memory </th><td> total:'

    sysInfo += os.totalmem()

    sysInfo += ' free:'+os.freemem()

    sysInfo += '</td></tr>'

    sysInfo += '<tr><th> CPU </th><td>'

    sysInfo += util.inspect(os.cpus())

    sysInfo += ' </td></tr>'

    sysInfo += '<tr><th> Network </th><td>'

    sysInfo += util.inspect(os.networkInterfaces())

    sysInfo += ' </td></tr>'

    sysInfo += '</table>'

    sysInfo += '</body></html>'

    res.send(sysInfo)

  } catch (err) {

    res.status(500).json({ message: err.message })

  }

})

module.exports = router

**Add data HTML**

<!DOCTYPE html>

<html>

    <head>

        <title>Add Entry To Databse</title>

    </head>

    <body>

        <h1>Add Entry To Databse</h1>

        <form method="post" action="/addcovid">

            <label>Enter Covid Information</label><br>

            <input type="text" name="date" placeholder="dd/mm/yyyy" required>

            <input type="text" name="county" placeholder="Enter county..." required>

            <input type="text" name="state" placeholder="Enter state..." required>

            <input type="number" name="cases" placeholder="Enter cases..." required>

            <input type="number" name="deaths" placeholder="Enter deaths..." required>

            <input type="submit" value="Add">

        </form>

    </body>

</html>

**Delete data HTML**

<!DOCTYPE html>

<html>

    <head>

        <title>Delete</title>

    </head>

    <body>

        <h1>Delete Based on State & County</h1>

        <form method="post" action="/deletevalues">

            <label>Enter Covid Information</label><br>

            <input type="text" name="county" placeholder="Enter county..." required>

            <input type="text" name="state" placeholder="Enter state..." required>

            <input type="submit" value="Delete">

        </form>

    </body>

</html>

**Findmorethanone HTML**

<!DOCTYPE html>

<html>

    <head>

        <title>Find More Than One</title>

    </head>

    <body>

        <h1>Find Cases More Than One</h1>

        <form method="post" action="/foundndmorethanone">

            <label>Enter Covid Information</label><br>

            <input type="text" name="date" placeholder="dd/mm/yyyy" required>

            <input type="submit" value="Find more than 1">

        </form>

    </body>

</html>

**First 20 HTML**

<!DOCTYPE html>

<html>

    <head>

        <title>Find 20</title>

    </head>

    <body>

        <h1>Find 20</h1>

        <form method="post" action="/foundtwenty">

            <label>Enter Covid Information</label><br>

            <input type="text" name="date" placeholder="dd/mm/yyyy" required>

            <input type="text" name="state" placeholder="Enter state..." required>

            <input type="submit" value="Find 20">

        </form>

    </body>

</html>

**Totals HTML**

<!DOCTYPE html>

<html>

    <head>

        <title>View Totals</title>

    </head>

    <body>

        <h1>View Totals</h1>

        <form method="post" action="/totalvalues">

            <label>Enter Values to Filter</label><br>

            <input type="text" name="county" placeholder="Enter county..." required>

            <input type="text" name="state" placeholder="Enter state..." required>

            <input type="submit" value="View">

        </form>

    </body>

</html>

**Update HTML**

<!DOCTYPE html>

<html>

    <head>

        <title>Update</title>

    </head>

    <body>

        <h1>Update Based on State & County</h1>

        <form method="post" action="/updatevalues">

            <label>Enter Values to Filter</label><br>

            <input type="text" name="county" placeholder="Enter county..." required>

            <input type="text" name="state" placeholder="Enter state..." required><br>

            <label>Enter Values to Update</label><br>

            <input type="number" name="cases" placeholder="Enter cases..." required>

            <input type="number" name="deahs" placeholder="Enter deaths..." required>

            <input type="text" name="date" placeholder="dd/mm/yyyy" required>

            <input type="submit" value="Update">

        </form>

    </body>

</html>

**Project Reflection**

**Overall**

Overall we worked well together as a team as we both had different skill sets that we used to advantage in the running of this project. It was initially difficult to organise meetings and time where we could both discuss the work due to the current circumstances with the coronavirus Both of us being in full time employment we were very busy there as well so it was difficult to find time where we were both free to work together. To overcome this we decided to split the work in sections that we could go off complete and then come back and review. This method worked well for us as it let us complete the work at our pace but also gave us the opportunity to get feedback on our work. The Agile approach adopted for this project meant that we held regular meetings to discuss progress and any potential blockers via Microsoft teams, this was beneficial as it allowed us to identify any issues we had and plan toward resolving those blockers.

**Olatunji:**

For this project Overall, I believe this was a successful project and we were able to cover a large majority of the project requirements. Utilising GitHub was key to the overall execution of the project, we were able to work in cohesion which saved a great amount of time. It also made it easier to help one another and to clearly highlight and address any issues we came across individually and as a team.

We had some issues with setting up the initial database on Mongodb, this was a blocker for us as we were not able to move on with the project as the database played a major role overall Project. To overcome this, we considered using another database tool to facilitate the task. However, after communication and some research we were able to resolve the issue and move past the blocker. (Olatunji U1728854)

**Zeeshan:**

I think that we were successful with this project as we developed a system that met all of the requirements of the specification. We did have some issues with the code initially where we didn’t utilise the HTTP protocols however being able to review this together helped us to quickly solve this and rework our codebase. During the course of the project I learnt how to utilise mongoDB as this was something I had never used before. Even though we did make a mistake with the code this helped me to learn a lot about using HTTP protocols and how to use them correctly.

**References**

<https://moodle.uel.ac.uk/pluginfile.php/1927344/mod_resource/content/1/Day1.pdf>

<https://moodle.uel.ac.uk/pluginfile.php/1926811/mod_resource/content/2/RESTAPI2ndday.pdf7>

<https://moodle.uel.ac.uk/pluginfile.php/2081378/mod_resource/content/2/RestAPItutorial-part1.pdf>

<https://moodle.uel.ac.uk/pluginfile.php/2259213/mod_resource/content/2/oreilly-modern-web-development-on-the-jamstack.pdf>