A Micro Project report on

MY NOTE APP USING HTML

Submitted to the CMR Institute of Technology in partial fulfillment of the requirement of the

FULL STACK WEB DEVELOPMENT

Of

III-B.Tech. II-Semester

in

Computer Science Engineering Department

Submitted by

A.HARSHITHA	**19R01A05C9**
CHAITAN GOUD	**19R01A05E0**
E. VINAY RAJ	**19R01A05E1**
G. NIKITHA	**19R01A05E2**
G. SHRAVYA	**19R01A05E3**
G.SHREYA	**19R01A05E4**

Under the guidance of

MR. NAGARAJU THANDU
(Asst. Professor)

CMR INSTITUTE OF TECHNOLOGY
(UGC AUTONOMUS)
(Approved by AICTE, Affiliated to JNTU, Kukatpally, Hyderabad)
Kandlakoya , Medchal Road ,Hyderabad

2022-2023

CMR INSTITUTE OF TECHNOLOGY
(UGC AUTONOMUS)
(Approved by AICTE, Affiliated to JNTU, Kukatpally, Hyderabad)
Kandlakoya, Medchal Road, Hyderabad.

Department of Computer Science Engineering

CMR
GROUP OF INSTITUTIONS
EXPLORE TO INVENT

CERTIFICATE

This is to certify that a Micro Project entitled with:" My Note App using HTML5" is being Submitted By

A.HARSHITHA	**19R01A05C9**
CHAITAN GOUD	**19R01A05E0**
E. VINAY RAJ	**19R01A05E1**
G. NIKITHA	**19R01A05E2**
G. SHRAVYA	**19R01A05E3**
G.SHREYA	**19R01A05E4**

In partial fulfillment of the requirement for completion of the **FULL STACK WEB DEVELOPMENT LAB** of III-B.Tech II- Semester is a record of a bonafide work carried out under guidance and supervision of MR.NAGARAJU THANDU (Asst.Professor)

Signature of Faculty **Signature of HOD**

CONTENTS

ACKNOWLEDGEMENT

We are extremely grateful to **Dr. M. Janga Reddy**, **Director**, **Dr. B. Satyanarayana**, **Principal** and **Mr. P.Pavan Kumar**, **Head of Department**, Dept of Computer Science Engineering, CMR Institute of Technology for their inspiration and valuable guidance during entire duration.

We are extremely thankful to our **Mr. Nagaraju Thandu(Asst. Professor)** ,Computer Science Engineering department, CMR Institute of Technology for his/her constant guidance, encouragement and moral support throughout the project.

We express our thanks to all staff members and friends for all the help and coordination extended in bringing out this micro project successfully in time.

Finally, we are very much thankful to our parents and relatives who guided directly or indirectly for successful completion of the project.

INTRODUCTION

My Notes - Notepad is an easy-to-use, intuitive, fast, elegant and secure notepad. You can use My Notes as a notepad, notebook, journal, agenda or diary. Notes app is used to make short text notes and update when needed and trash when you are done. It can be used for various functions as you can add your to-do list in this app, some important information for future reference etc.

This app is very useful in some cases when you want quick access to the notes. In this project, we try to build a simple My notes app using HTML, CSS

MYNOTES APP

The My Notes app provides a quick and convenient way to **save lists, thoughts, and other random items.** We capture a quick thought, create checklists, sketch ideas, and more. It is a very handy app as we can immediately write some points that we tend to forget after some time. It is a perfect way to save our thoughts so that we can look it up later when required. The app is easy to use for any user.

We just have to open it and start writing whatever we want to. One of the main advantages of this app is we can update or delete our data anytime, anywhere. We can create separate folders, organize the data accordingly. We can separate our text by giving specific heading to each note. Easy to maintain, access and store makes the app more reliable. So, let's create this simple app by following the given steps.

In this note app, users can easily add, edit, or delete their notes. The notes user has added to this app will be stored in the browser's local storage so, they won't remove on page refresh or tab close

COMPONENTS USED TO CREATE THE APP

VS CODE: Visual Studio Code combines the simplicity of a source code editor with powerful developer tooling, like IntelliSense code completion and debugging. VS Code helps you be instantly productive with syntax highlighting, bracket-matching, auto-indentation, box-selection, snippets, and more.

Installing Vs code: Go to official website of VS code and install for windows/Mac

Application shell Model:

Application shell (architecture) is the minimum HTML, CSS and JavaScript needed to build basic representational User Interface of a PWA. It is one of the main factors that provide instant loading, smooth UX and good performance on repeat visit. An app shell is cached immediately which means that the shell files are loaded once over the network and then saved to the local device. An app shell architecture focuses on speed, performance, instant loading and

Responsive web design

It is not a technology, product or a programming language. Rather, it is an approach to the design and development of web content wherein through the use of certain modern web techniques, the designs and layouts of a website or web applications are made with the ability to respond to the changes in screen size, and to layout themselves accordingly.

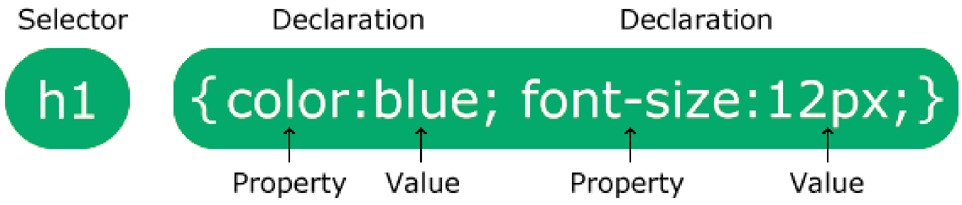
TECHNOLOGY STACK

HTML:

- HTML stands for Hyper Text Markup Language
- HTML is the standard markup language for creating Web pages
- HTML describes the structure of a Web page
- HTML consists of a series of elements
- HTML elements tell the browser how to display the content
- HTML elements label pieces of content such as "this is a heading", "this is a paragraph", "this is a link", etc.
- An HTML element is defined by a start tag, some content, and an end tag:
- <tagname> Content goes here... </tagname>
- The HTML **element** is everything from the start tag to the end tag: <h1>My First Heading</h1><p>My first paragraph. </p>

CSS:

- CSS stands for Cascading Style Sheets
- CSS describes how HTML elements are to be displayed on screen, paper, or in other media
- CSS saves a lot of work. It can control the layout of multiple web pages all at once
- External stylesheets are stored in CSS files
- CSS is used to define styles for your web pages, including the design, layout and variations in display for different devices and screen sizes.
- The style definitions are normally saved in external .CSS files.
- With an external stylesheet file, you can change the look of an entire website by changing just one file!
  CSS Syntax :



- The selector points to the HTML element you want to style.
- The declaration block contains one or more declarations separated by semicolons.
- Each declaration includes a CSS property name and a value, separated by a colon.
- Multiple CSS declarations are separated with semicolons, and declaration blocks are surrounded by curly braces.

There are three ways of inserting a style sheet:

- External CSS
- Internal CSS
- Inline CSS

External CSS

With an external style sheet, you can change the look of an entire website by changing just one file! Each HTML page must include a reference to the external style sheet file inside the <link> element, inside the head section.

Internal CSS

An internal style sheet may be used if one single HTML page has a unique style. The internal style is defined inside the <style> element, inside the head section.

Inline CSS

An inline style may be used to apply a unique style for a single element. To use inline styles, add the style attribute to the relevant element. The style attribute can contain any CSS property.

JavaScript: It is a general purpose programming language, originally developed for the purpose of providing scripting functionality for web pages, but is currently used for many different kinds of purposes, ranging from client to server to mobile development.

MongoDb: It stores data in flexible, JSON-like documents, meaning fields can vary from document to document and data structure can be changed over time

CREATING THE APP

1: Install the Visual Studio code and setup. It must support HTML, CSS

2.We need to create HTML, CSS as follows

3.HTML: We will create the basic framework of website using html

5.CSS: We use style.css to beautify our app and include it in index.html

Creating index.html:

```
<!DOCTYPE html>
 <html>
 <head>
 <title>MyNotesApp</title>
</head>
<body>
 <div id="header">
 <div id="name">Web Note</div>
 <div id="menubutton"><a id="menulink" href="#">MENU</a></div>
<div id="menu" class="hiddenmenu">
 <div class="menuitem"><a id="home" href="#">Home</a></div>
<div class="menuitem"><a id="about" href="#">About us</a></div>
</div>
<div class="clear"></div>
 </div> <div id="container">
 <textarea id="area" rows="10" cols="50"></textarea>
</div> <div class="footer"> <p>Simple Notes App</p>

</div>
</body>
</html>
```

Style.css

```
body {
    margin: 0px;
}
#header {
    background: #325232;
    color: white;
    font: arial;
    font-size: 16px;
    padding: 16px;
}
#header > #name {
    font-family: Arial, Sans-serif;
    display: inline-block;
}
#menu a, #menubutton a {
    text-decoration: none;
    color: white;
    text-transform: uppercase;
    font-family: Arial, Sans-serif;
    font-size: 13px;
}
.clear {
    clear: both;
}
#container {
    margin: 20px 20px 20px 20px;
}

#area {
    width: 100%;
    height: 100%;
    font: 1em arial;
    color: rgba(50, 82, 50, 1.0);
}
#area:focus {
    color: rgba(50, 82, 50, 1.0);
    border: 2px solid #96c56f;

 box-shadow: 0px 1px 1px #888888;
}
.footer {
    width: 100%;
    background-color: #325232;


     height: 32px;
    position: fixed;
    bottom: 0px;
}
.footer > p {
    margin: 8px;
    color: white;
    font: 14px arial;
    text-align: center;
}
```

That is called a style sheet, written in a language named "CSS". In our CSS file, as you have noticed, we have used "CSS Selectors" to reference elements used in our HTML. For example, in the above CSS file we used the following definitions:

.footer - Here we prepend a "dot" character to reference an element class name. This rule, therefore, affects all elements that have the class name "footer".

#header - Here we prepend a hash character ("#") to reference an element ID. This rule, therefore, affects the element that has the ID "header" (there should only be one; element IDs are supposed to be unique).

#menu a, #menubutton a - This selects all "a" elements (which are links) that are inside the element with ID "menu", AND all "a" elements that are inside the element with ID "menubutton"

#header > #name - Select all elements with element ID "name" where the immediately parent of that element is an element with an element ID "header"

Functionality.js

```
window.onload = function() {
    document.getElementById('menulink').onclick = function() {
        var menu = document.getElementById('menu');
        if(menu.className != 'shownmenu') {
            menu.className = 'shownmenu';
        }
        else {
            menu.className = 'hiddenmenu';
        }
    }
}
```

Linking tags:

Now we will link the style sheet and functionality in the HTML file. Add these tags inside your head element:

```
<link rel="stylesheet" type="text/css" href="style.css" />
<script src="functionality.js"></script>
```

IMPROVING MOBILE COMPATIBILITY

To make the page *mobile aware*, add the viewport configuration inside the head element of your page:

```
<meta name="viewport" content="initial-scale=1, maximum-scale=1" />
```

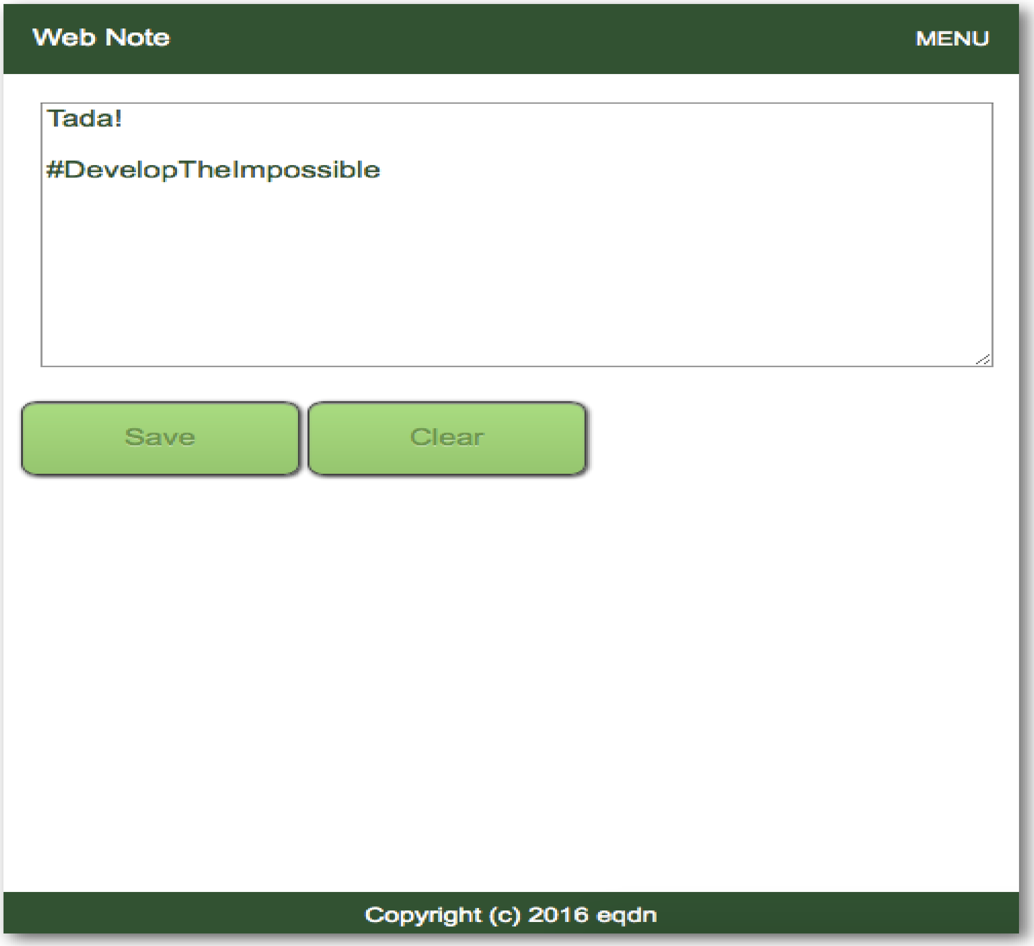
When designing modern web applications, we need to keep in mind that these applications may be accessed not just on desktop browsers, but also on laptops, tablets and smartphones, or other such devices that we may not quite expect. With that in mind, we need the make sure that our application will look good on any screen size. This kind of design is often called Responsive Web Design, and is commonly achieved through the use of Media Queries.

Since our web app is "mobile aware", now we can also configure alternative layouts depending on the screen / display size by means of Media Queries. Now we will add media queries to our style sheet by adding these lines of code in the style.css file:

```
@media only screen and (min-width: 600px) {
    #menu {
        float: right;
        display: inline-block;
    }
    .menuitem {
        display: inline;
    }
    .menuitem + .menuitem {
        border-left: 1px solid white;
        margin-left: 8px;
        padding-left: 8px;
    }
    #menubutton {
        display: none;
    }
}
@media only screen and (max-width: 599px) {
    #menubutton {
        display: inline;
        float: right;
    }
    #menu {
        position: absolute;
        top: 48px;
        right: 0px;
        background: #325232;
        padding: 16px;
    }
    .shownmenu {
        display: block;
    }
    .hiddenmenu {
        display: none;
    }
    .menuitem {
        display: block;
    }
    .menuitem + .menuitem {
        margin-top: 8px;
    }
}
```

Media query *only*

The media query keyword *only* is ignored by compliant browsers (it is treated as if it wasn't there). It also causes older browsers (that do not understand media queries) to ignore the entire media query.



Tada!

#DevelopTheImpossible

hat we have that in place, open the HTML file in a browser and try to resize the browser to make the window smaller. Your application should look something like this:

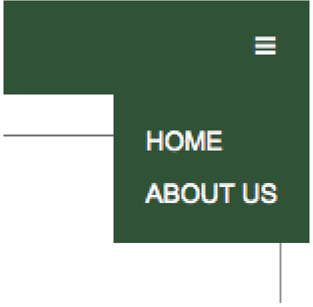
As you noticed, the menu on the top right corner adjust based on the size of the browser. Now let's change something in our code to make it more cool, replace this:

```
<div id="menubutton"><a id="menulink" href="#">MENU</a></div>
```

.. With this:

```
<div id="menubutton"><a id="menulink" href="#">&#9776</a></div>
```

That is what we call the "burger icon". The menu in the upper right corner should now look something like this:



We will use "CSS buttons" for our buttons. We will add our buttons by using the following code in a new div tag after and at the same level as the "container" div tag.

```
<div id="controls">
   <p><a href="javascript:save();" class="button">Save</a>
      <a href="javascript:clear();" class="button">Clear</a></p></div>
```

In our style sheet, we add the look of the buttons:

```
#controls {
   margin-left: 10px;
}
a.button {
   width: 15%;
   display: inline-block;
   background: -webkit-linear-gradient(top, #a9db80, #96c56f);
   background: -moz-linear-gradient(top, #a9db80, #96c56f);
   background: -ms-linear-gradient(top, #a9db80, #96c56f);
   background: -o-linear-gradient(top, #a9db80, #96c56f);
   background: linear-gradient(to bottom, #a9db80, #96c56f);
   border-radius: 8px;
   border: 1px solid #444444;
   box-shadow: 0px 1px 1px #888888 inset, 1px 1px 3px #222222;
   color: #6d954e;
   text-decoration: none;
       text-align: center;
   padding: 16px 32px 16px 32px;
   text-shadow: 0px 1px 1px rgba(255, 255, 255, 0.4);
   font-family: arial;
}
a.button:hover {
   background: rgba(120, 160, 120, 1);
   color: white;
}
a.button:active {
   background: rgba(80, 120, 80, 1); }
```

CREATING A DATABASE

We need a storage system for our notes so they can be saved even when the program ends. To add persistent storage, we'll use databases. We'll use the file database.js to connect to a database, create a collection, and name our collection Notes .

Database.js

```
const  mongooseClient = require (“mongoose”)
mongooseClient.connect(“mongodb//localhost//notepadDB”,{userNewUrlParser:true,
useUnifiedTropology:true),(err)=>
{
if(err) console.log(err); });
const NotesSchema = mongooseClient.schema({
title:String
description : String,
})
Const Notes=mongooseClient.model(“NOTES”,NotesSchema)
Module.exports=Notes;
```

Import the mongoose library
Tell Mongoose to connect to our notepadDB database. If any error occurs, then they should be logged out
We created a schema for our notes, NotesSchema. As the code suggests, the document that uses this schema will have two fields:

- A title field of type String
- A description field of type String

Update-router,js

This file will handle the routes whenever the user navigates to the directories that start with updatepage. As the path suggests, here the user will update his respective notes.

```
const Router = require('express').Router()

let id=0;

Router.get('/:__id',(req,res,next)=>{
   id= req.id = req.params.__id;
   console.log('in get middleware');
   next();
})

Router.post('/',(req,res,next)=>{
   console.log('in post middleware')
   req.id = id;
   next();
}) module.exports = Router;
```

Use express.Router() middleware to send the id data to the paths updatepage and updatepage:/__id
This __id field will be required so the database can select the note with the specified ID, which will be needed for updating purposes

Main.js

```
app.get(("/notes-add"),(req,res,next)=>{
    res.render('notes-add');
})
app.post( (req,res,next)=> {
    console.log(req.body);
    const Note = new Notes({})

    Note.title = req.body.title
    Note.description = req.body.description
        //save notes first
    Note.save((err,product)=>{
         if(err) console.log(err);
         console.log(product);
    })
res.redirect('/index')
})
```

Handling GET and POST requests for Update/Edit:

```
app.get('/updatepage/:__id',(req,res)=>{
  console.log('id for get request: ' + req.id);
  Notes.findById(req.id,(err,document)=>{
    console.log(document);
    res.render('updatepage',{data:document});
  })
})

app.post('/updatepage',(req,res,next)=>{
  console.log('id: ' + req.id);
  Notes.findByIdAndUpdate(req.id , {title : req.body.title , description: req.body.description
},{useFindAndModify:false}
    ,(err,document)=>{
console.log('updated');
})
res.redirect('/index');
})
```

Handle the POST request to the /updatepage URL.

Perform a query to find the document with the specific id. Update the title and description fields, respectively.

Redirect to the /index directory

Handling GET and POST requests for Deletion:

```
app.get("/delete/:__id", (req,res,next)=>{
    Notes.findByIdAndRemove(req.params.__id ,{useFindAndModify : false}, (err,document)=> {
       if(err) console.log(err)
       console.log(document);
    })
  res.redirect('/index');  })
```

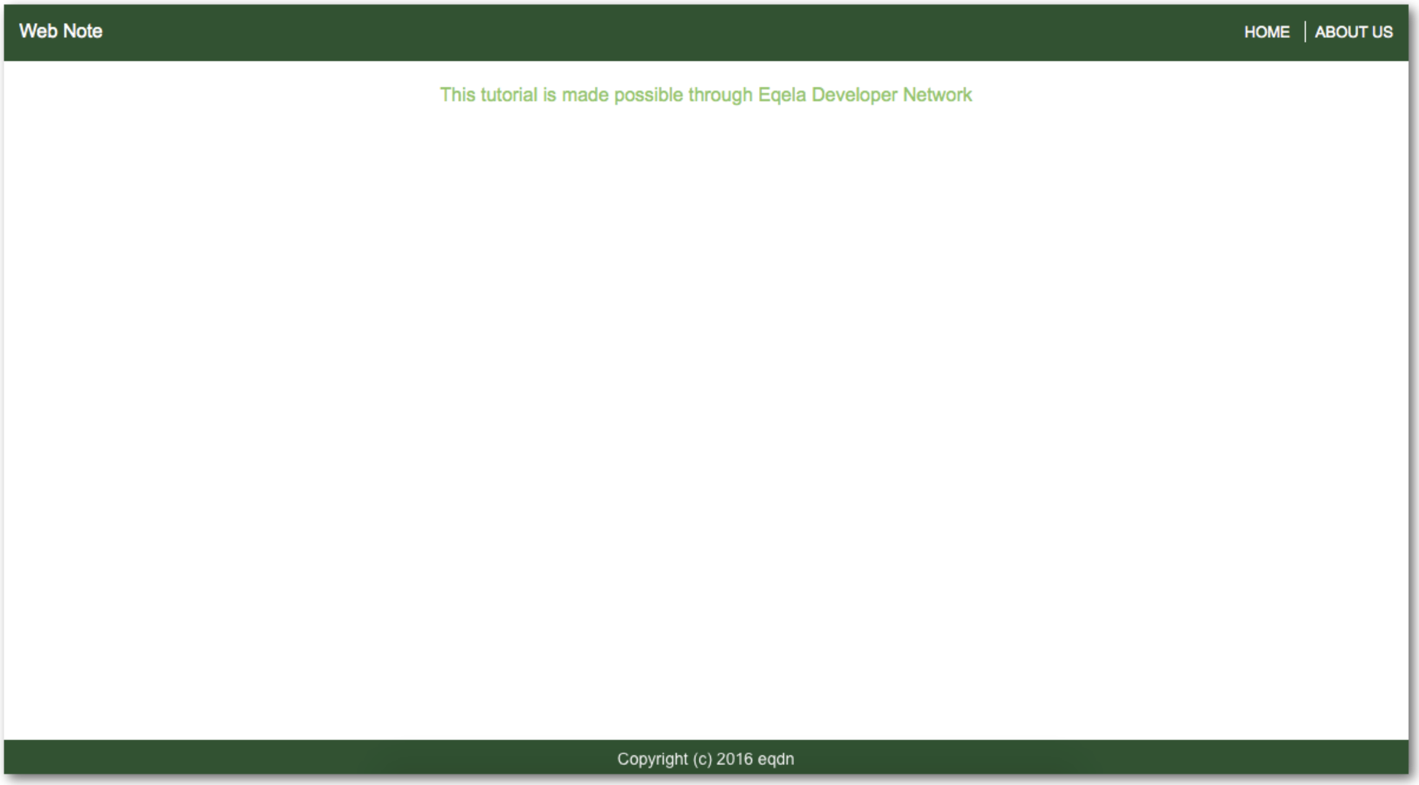
Whenever the user clicks on the delete button, the user is redirected to a /delete path with the id of the document sent as a parameter.
We can use this id field to identify the document and then delete it.

Handling GET and POST requests for Finding/reading:

```
app.get('/index' , (req,res,next)=>{
  Notes.find({}).exec((err,document)=> {
    if(err) console.log(err);
       document.forEach((value) => {
        console.log(document);
     })
   res.render('view',{data:document})
  })
})
```

the /index path will help us in viewing our notes. If the client is only receiving data from this path, what kind of HTTP request should we handle? The type of request we'll handle this time will be a GET request, as we're only getting data.

When we choose "About us", we should see this:



This tutorial is made possible through Eqela Developer Network

Copyright (c) 2016 eqdn

To do that, we will need to also add these lines of code in the "window.onload" function:

```
document.getElementById('about').onclick = function() {
   document.getElementById('container').innerHTML = "";
   document.getElementById('controls').innerHTML = "";
   document.getElementById('menu').className = 'hiddenmenu';
   var container = document.getElementById('container');
   var p = document.createElement('p');
   p.id = 'aboutus';
   container.appendChild(p);
   var text = document.createTextNode("This tutorial is made possible through Eqela Developer Network");
   p.appendChild(text);


}
```

And this to our style sheet:

```
#aboutus {
   margin: 8px;
   color: #96c56f;
   font: 16px arial;
   text-align: center;
}
```

Also add these lines of code inside the "window.onload" function:

```
var home = document.getElementById('container').innerHTML;
var controls = document.getElementById('controls').innerHTML;
display_saved_note();
document.getElementById('home').onclick = function() {
   document.getElementById('container').innerHTML = home;
   document.getElementById('controls').innerHTML = controls;
   document.getElementById('menu').className = 'hiddenmenu';
   display_saved_note();
}
```

IMPLEMENTATION

Installation of Node.js

Download Node.js installer

In a web browser, navigate to https://nodejs.org/en/download/. Click the **Windows Installer** button to download the latest default version. At the time this article was written, version 10.16.0-x64 was the latest version. The Node.js installer includes the NPM package manager.

Open a command prompt (or PowerShell), and enter the following:

```
$node -v

$npm -v
```

To Run our Application using Nodejs

```
$npm install -g serve

$serve
```

Install Mongoose - To handle Data Integration

```
npm install mongoose
```

Install Express - To handle Routing

```
npm install express
```

Install body-parser - To handle how our incoming data will be processed

```
npm install body-parser
```

STORING THE FILES

Open VS code and create a project folder. It should have index.html,style.css,main.js,database.js and functionality.js files

RUN THE FILES

To run the main.js file :

```
npm start
```

To run the get and requests code:

localhost:3000/index

To get the Notes-app:

Double click on index.html

PUSHING THE PROJECT TO GITHUB

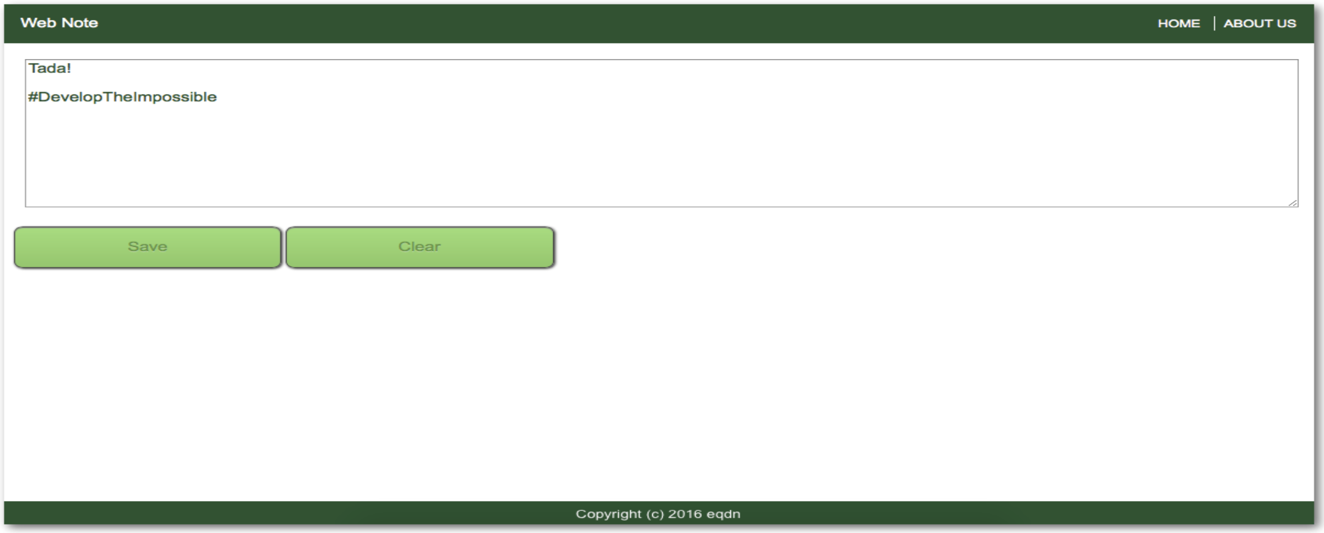
1.Install GIT

2.Run the following commands in command prompt:

```
git clone https://github.com/%7Busername%7D/FSWD.git

git config --global.username "{username}"

git config –global.useremail  " MY_NAME@example.com"

git pull origin main

git add .

git commit -m "your message"

git push origin main
```

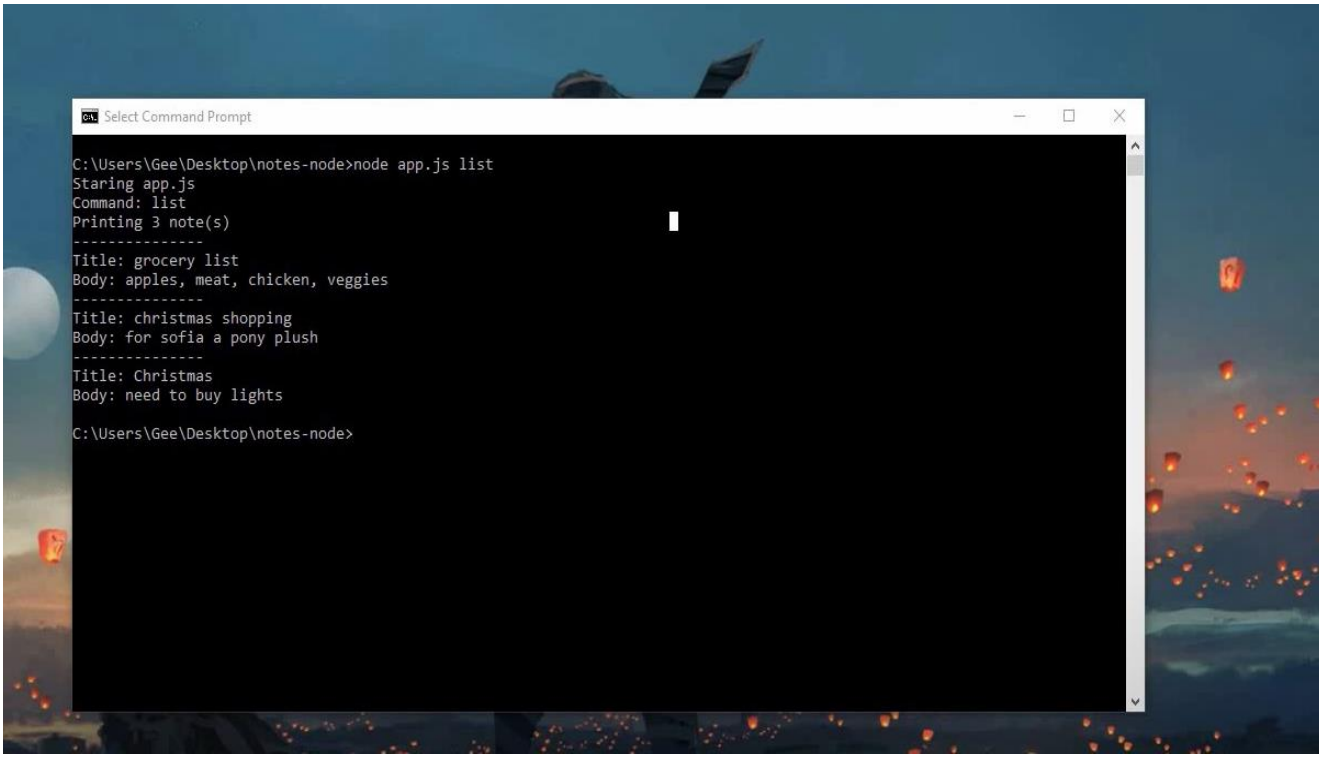
3.Open your github repository and see the changes

RESULTS

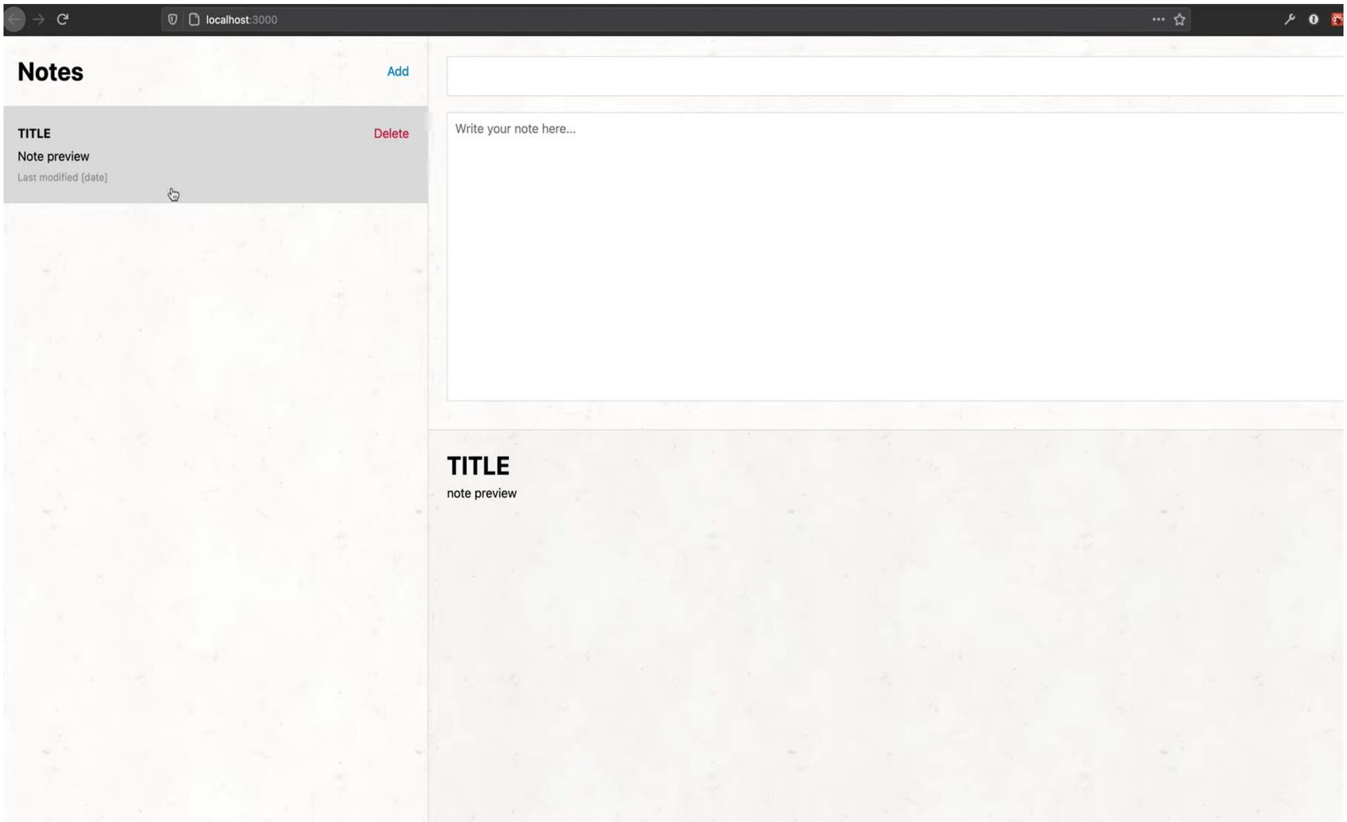


Tada!

#DevelopTheImpossible



```
C:\Users\Gee\Desktop\notes-node>node app.js list
Staring app.js
Command: list
Printing 3 note(s)
---------------
Title: grocery list
Body: apples, meat, chicken, veggies
---------------
Title: christmas shopping
Body: for sofia a pony plush
---------------
Title: Christmas
Body: need to buy lights

C:\Users\Gee\Desktop\notes-node>
```



localhost:3000
Notes
Add
TITLE
Delete
Note preview
Last modified [date]
Write your note here...
TITLE
note preview

CONCLUSION

The main goal of the this was to evaluate and implement a Notes Application. A significant amount of time was invested in obtaining insight to the topic and its components. A prototype ‘MyNoteapp’ was developed which illustrated the implementation of Notes application. To sum up, the project was successfully carried out and the goals, which were set at the beginning of the project, were achieved.

REFERENCES

https://www.codingnepalweb.com/build-a-notes-app-in-html-css-javascript/

https://www.geeksforgeeks.org/how-to-create-notes-taking-site-using-html-bootstrap-and-javascript/

https://www.w3schools.com/howto/howto_css_notes.asp

https://eqdn.tech/html5-note-app-tutorial/