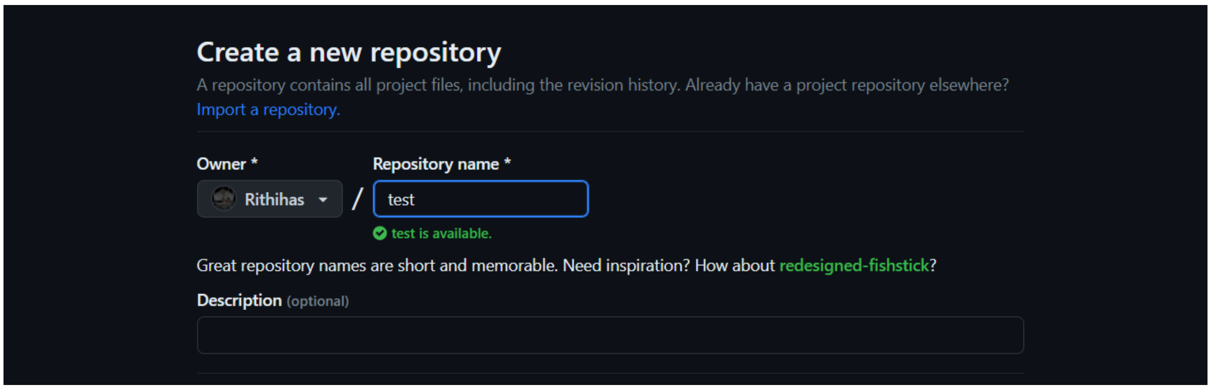
WE LAB

1.Create a git repository and clone it for changes and publish the changes using gitbash(Git commands)

A) step 1:

Create a new repository.



Create a new repository

A repository contains all project files, including the revision history. Already have a project repository elsewhere? Import a repository.

Owner * Rithihas / Repository name * test

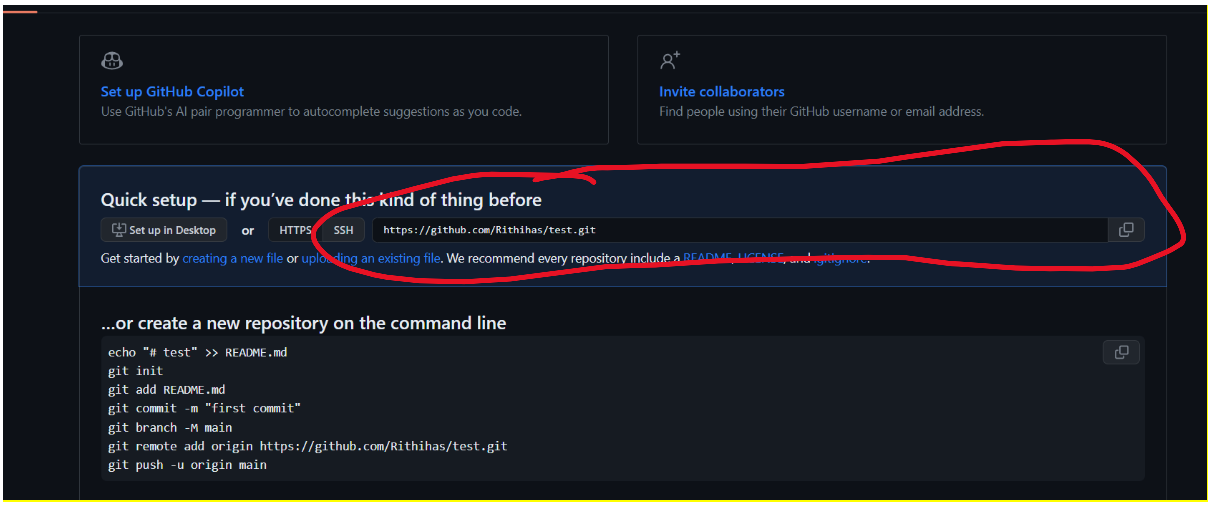
test is available.

Great repository names are short and memorable. Need inspiration? How about redesigned-fishstick?

Description (optional)

Step 2:

Copy the repository link.



Set up GitHub Copilot
Use GitHub's AI pair programmer to autocomplete suggestions as you code.

Invite collaborators
Find people using their GitHub username or email address.

Quick setup — if you've done this kind of thing before

Set up in Desktop or HTTPS SSH https://github.com/Rithihas/test.git

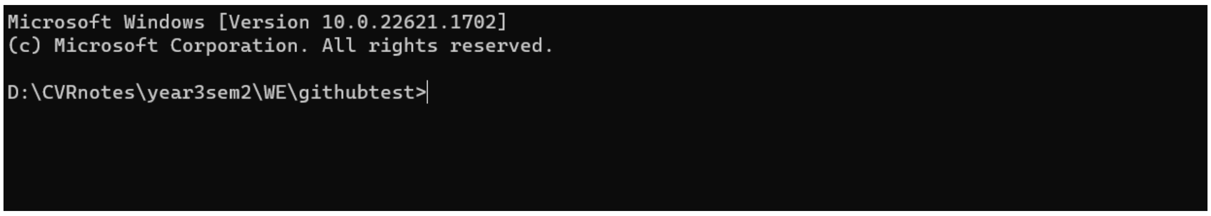
Get started by creating a new file or uploading an existing file. We recommend every repository include a README, LICENSE, and .gitignore.

...or create a new repository on the command line

```
echo "# test" >> README.md
git init
git add README.md
git commit -m "first commit"
git branch -M main
git remote add origin https://github.com/Rithihas/test.git
git push -u origin main
```

Step 3:

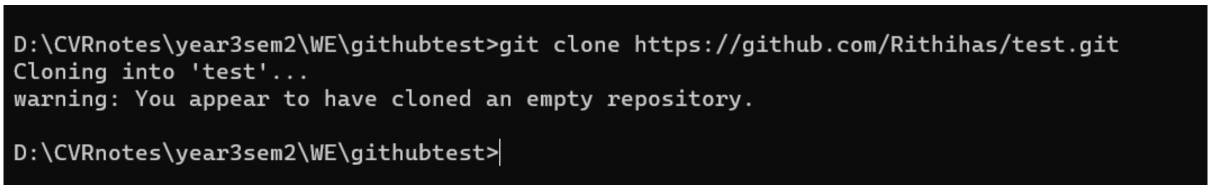
Create an empty folder on your desktop and open gitbash / command prompt in that folder. (navigate to that folder using cd command).



```
Microsoft Windows [Version 10.0.22621.1702]
(c) Microsoft Corporation. All rights reserved.

D:\CVRnotes\year3sem2\WE\githubtest>
```

Step 4:

Clone the repository into the current folder using git clone.



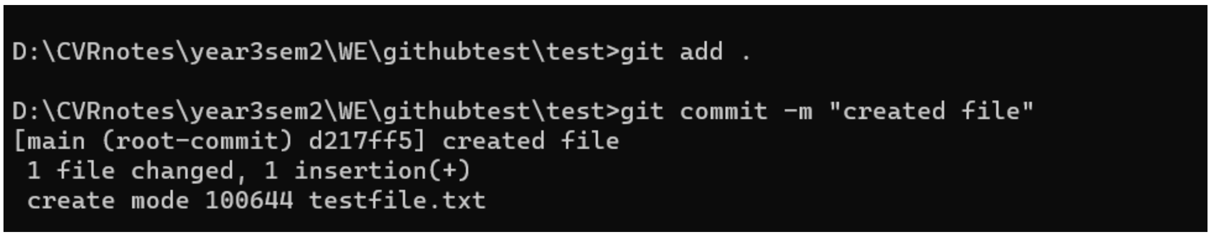
```
D:\CVRnotes\year3sem2\WE\githubtest>git clone https://github.com/Rithihas/test.git
Cloning into 'test'...
warning: You appear to have cloned an empty repository.

D:\CVRnotes\year3sem2\WE\githubtest>
```

Step 5:

Create any file and save it in the cloned folder.

Step 6:

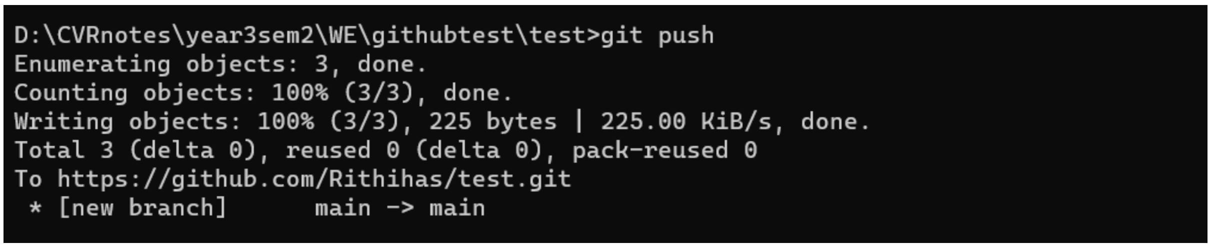
Use “ git add . “ to add the file to the staging area. And then use git commit -m “message” to commit changes.



```
D:\CVRnotes\year3sem2\WE\githubtest\test>git add .

D:\CVRnotes\year3sem2\WE\githubtest\test>git commit -m "created file"
[main (root-commit) d217ff5] created file
 1 file changed, 1 insertion(+)
 create mode 100644 testfile.txt
```

Step 7:

push changes to github using git push.



```
D:\CVRnotes\year3sem2\WE\githubtest\test>git push
Enumerating objects: 3, done.
Counting objects: 100% (3/3), done.
Writing objects: 100% (3/3), 225 bytes | 225.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
To https://github.com/Rithihas/test.git
 * [new branch]      main -> main
```

Note:

If it asks for credentials try:

$ git config --global user.name "username"

$git config --global user.email “youremail@gmail.com”

$git config --global user.password "yourpassword"

$git config –global https.proxy “proxyserver address” or $git config –global “proxyserver address”

2. Working of ES6 features like arrow functions, destructuring and function generators.

Arrow functions :

```
// Basic arrow function
const greet = () => {
  console.log("Hello, world!");
};

greet(); // Output: Hello, world!

// Arrow function with parameters
const sum = (a, b) => {
  return a + b;
};

console.log(sum(2, 3)); // Output: 5

// Arrow function with implicit return
const multiply = (a, b) => a * b;

console.log(multiply(4, 5)); // Output: 20

// Arrow function with a single parameter
const square = x => x * x;

console.log(square(3)); // Output: 9
```

destructuring:

```
// Destructuring arrays
const numbers = [1, 2, 3, 4, 5];

const [first, second, ...rest] = numbers;

console.log(first); // Output: 1
console.log(second); // Output: 2
console.log(rest); // Output: [3, 4, 5]


// Destructuring objects
const person = {
  name: "John",
  age: 30,
  address: {
    city: "New York",
    country: "USA",
  },
};

const { name:vape, age: agro, address: { city, country } } = person;

console.log(vape); // Output: John
console.log("age is : ",agro); // Output: 30
console.log(city); // Output: New York
console.log(country); // Output: USA
```

generator functions:

```
// Generator function
function* numberGenerator() {
    yield 1;
    yield 2;
    yield 3;
    yield 4;
    yield 5;
  }

  // Create an instance of the generator
  const generator = numberGenerator();

  // Iterate over the values using the generator
  console.log(generator.next().value); // Output: 1
  console.log(generator.next().value); // Output: 2
  console.log(generator.next().value); // Output: 3
  console.log(generator.next().value); // Output: 4
  console.log(generator.next().value); // Output: 5
  console.log(generator.next().value); // Output: undefined (no more yield
values)

  // Generator function with parameters
  function* rangeGenerator(start, end, step) {
    for (let i = start; i <= end; i += step) {
      yield i;
    }
  }

  const range = rangeGenerator(1, 10, 2);

  // Iterate over the range using the generator
  console.log(range.next().value); // Output: 1
  console.log(range.next().value); // Output: 3
  console.log(range.next().value); // Output: 5
  console.log(range.next().value); // Output: 7
  console.log(range.next().value); // Output: 9
console.log(range.next().value); // Output: undefined (no more yield values)
```

3. Explain the node modules: os,http,fs etc

1. os module:

```
var os = require('os');

console.log('cpu architecture: '+os.arch());

console.log('free memory :'+os.freemem());

console.log('total memory: '+os.totalmem());

console.log('os type : ' + os.type());
```

2. http module:

```
var http = require('http');

http.createServer(function(req,res){
    res.write('hello world!');
    res.end();
}).listen(8070);
```

3. fs module:

```
const fs = require('fs');

// Read from a file
fs.readFile('input.txt', 'utf8', (err, data) => {
  if (err) {
    console.error(err);
    return;
  }

  console.log('File content:');
  console.log(data);

  // Write to a file
  const content = data.toUpperCase();
  fs.writeFile('output.txt', content, 'utf8', (err) => {
    if (err) {
      console.error(err);
      return;
    }

    console.log('Data has been written to the file successfully.');
  });
});
```

NOTE

Proxy commands for typescript , react etc

npm config set proxy http://172.16.2.200:3128

npm config set http-proxy http://172.16.2.200:3128 (or try https-proxy)

npm config set registry https://registry.npmjs.org/

4. typescript classes.

Installation:

1.”npm install -g typescript”

2. “Set-ExecutionPolicy -Scope CurrentUser “ (executionpolicy value is 1)

3. create file with .ts extension.

4. “tsc filename.ts “ to compile to js file.

5. run the js file using “node filename.js”

If the above procedure doesn’t work , maybe this will , idk:

1.Create an empty folder

2. open that folder in vs terminal

3. “npm init”

4. then ”npm install -g typescript”

5. then open package . json and check scripts . add script “tsc” if missing.

6. then try npm run tsc filename.tsc

```
class Animal {
  name: string;

  constructor(name: string) {
    this.name = name;
  }

  makeSound(): void {
    console.log("The animal makes a sound");
  }
}

class Dog extends Animal {

  breed : string;

  constructor(name:string, breed:string)
  {
    super(name);
    this.breed = breed;
  }

  makeSound(): void {
    console.log("The dog barks");
  }
}

class Cat extends Animal {
  makeSound(): void {
    console.log("The cat meows");
  }
}

// Create instances of the classes
const animal = new Animal("Generic Animal");
const dog = new Dog("Bobby","sheperd");
const cat = new Cat("Whiskers");

// Call the makeSound() method on each instance
animal.makeSound(); // Output: "The animal makes a sound"
dog.makeSound(); // Output: "The dog barks"
cat.makeSound(); // Output: "The cat meows"
```

5. typescript generics.

```
// Generic class
class Box<T> {
    private item: T;

    constructor(item: T) {
      this.item = item;
    }

    public getItem(): T {
      return this.item;
    }
  }

  // Create instances of the generic class
  const box1 = new Box<number>(10);
  console.log(box1.getItem()); // Output: 10

  const box2 = new Box<string>("Hello");
  console.log(box2.getItem()); // Output: Hello

  // Generic function
  function printArray<T>(array: T[]): void {
    for (let item of array) {
      console.log(item);
    }
  }

  // Call the generic function
  const numbers: number[] = [1, 2, 3, 4, 5];
  printArray<number>(numbers); // Output: 1 2 3 4 5

  const names: string[] = ["Alice", "Bob", "Charlie"];
  printArray<string>(names); // Output: Alice Bob Charlie
```

React:

Install : npm install react

Create app : npx create-react-app myapp

6. React JSX and components.

Remove everything in index.js file and type this:

```
import React from 'react';
import ReactDOM from 'react-dom';

// functional JSX component
const MyComponent = () => {
  const name = 'John Doe';
  const age = 30;

  return (
    <div>
      <h1>Hello, using functional component : {name}!</h1>
      <p>You are {age} years old.</p>
    </div>
  );
};

class MyComponent2 extends React.Component {
  constructor(props) {
    super(props);
    this.state = {
      name: 'John Doe',
      age: 30
    };
  }

  render() {
    const { name, age } = this.state;

    return (
      <div>
        <h1>Hello, using class component : {name}!</h1>
        <p>You are {age} years old.</p>
      </div>
    );
  }
}


// Render the JSX component
ReactDOM.render(<><MyComponent />  <MyComponent2/></>,
document.getElementById('root'));
```

7. React routing / React web application:

```
React Routers Application:
Step 1 : install react router using this command: npm i -D react-router-dom
Step 2: Add the following code for all the files

Home.js

const Home = () => {
    return (
        <div>
            <img
              src={require('./CVRCollege.jpg')} alt="logo"
            />
        </div>
      );
    }


  export default Home;
-------------------------------------------------------------------------------
------------------------------------------------------------------------
Blogs.js

const Blogs = () => {
    return(<><h1>Blog Articles</h1><p> This is the Blog Articles page </p></>)
  };

  export default Blogs;

------------------------------------------------------------------------------
------------

Layout.js
import { Outlet, Link } from "react-router-dom";

const Layout = () => {
  return (
    <>
      <nav>
        <ul >
          <li>
            <Link to="/home">Home</Link>
          </li>
          <li>
            <Link to="/blogs">Blogs</Link>
          </li>
        </ul>
      </nav>

      <Outlet />
    </>
  )
};

export default Layout;

------------------------------------------------------------------------------
----------------------------------------------------------------
Step 4 : Add the following code in index.js

import ReactDOM from "react-dom/client";
import { BrowserRouter, Routes, Route } from "react-router-dom";
import Layout from "./Layout";
import Home from "./Home";
import Blogs from "./Blogs";


export default function App() {
  return (
    <center>

      <h1>React Web Application</h1>

      <BrowserRouter>
      <Routes>
        <Route path="/" element={<Layout />} />
          <Route path="home" element={<Home />} />
          <Route path="blogs" element={<Blogs />} />

      </Routes>
    </BrowserRouter>

    </center>
  );
}

const root = ReactDOM.createRoot(document.getElementById('root'));
root.render(<App />);



-------------------------------------------------------------------------------
--------------------------------------------------------------------
Step 5: Save all the files and run the command npm start
```

8. React States and Parent to child and vice-versa communication:

Create the react app , copy this code in app.js:

```
import React, { useState } from 'react';

// Child component
const ChildComponent = ({ message, onChildClick }) => {
  return (
    <div>
      <h2>Child Component</h2>
      <p>{message}</p>
      <button onClick={onChildClick}>Click me</button>
    </div>
  );
};

// Parent component
const ParentComponent = () => {
  const [parentMessage, setParentMessage] = useState('');
  const [childMessage , setChildMessage] = useState('message sent from
parent');

  const handleChildClick = () => {
    setParentMessage('Message received from Child');
  };
  return (
    <div>
      <h2>Parent Component</h2>
      <p>{parentMessage}</p>
      <ChildComponent message={childMessage} onChildClick={handleChildClick}
/>
    </div>
  );
};

// App component
const App = () => {
  return (
    <div>
      <h1>React Communication</h1>
      <ParentComponent />
    </div>
  );
};

export default App;
```

9. form validation in react:

Create a file called FormValidationExample.Js:

```
import React, { useState } from 'react';

const FormValidationExample = () => {
  const [name, setName] = useState('');
  const [email, setEmail] = useState('');
  const [message,setMessage] = useState('');

  const handleNameChange = (event) => {
    setName(event.target.value);
  };

  const handleEmailChange = (event) => {
    setEmail(event.target.value);
  };

  const handleSubmit = (event) => {
    event.preventDefault();

    if(!(email.includes('@') && email.includes('.')) || /\d/.test(name))
    {
      setMessage('invalid credentials.');
    }
    else
    setMessage('valid credentials.');

  };


  return (
    <div>
      <h2>Form Validation Example</h2>
        <div>
        <p>Name:</p>
          <input
            type="text"
            onChange={handleNameChange}
          />
        </div>
        <div>
          <p>Email:</p>
          <input
            type="email"
            onChange={handleEmailChange}
          />
        </div>
        <button type="submit" onClick={handleSubmit}>
          Submit
        </button>
        <p>{message}</p>
    </div>
  );
};

export default FormValidationExample;
```

in app.js:

```
import React from 'react';

import FormValidationExample from './FormValidationExample.js';


// App component
const App = () => {
  return (
    <div>
```

```
      <FormValidationExample></FormValidationExample>
    </div>
  );
};

export default App;
```