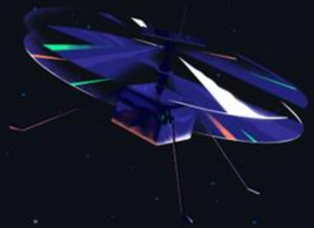
GitHub Copilot Developers Training

Level: Intermediate



GitHub Copilot

Resources

- Getting started with GitHub Copilot
- Configuring GitHub Copilot in your environment
- Insider newsletter digest: How to use GitHub Copilot
- Video - Get Started with the Future of Coding: GitHub Copilot
- Tutorial: GitHub Copilot and VS Code
- Copilot Exercises

Choose Workshop Track

Workshop for those who want to see how to build web application

Primary IDE: VS Code
Primary OS: Independent

Workshop for those who want to see how to build Java applications

Primary IDE: IntelliJ
Primary OS: Independent

Workshop for those who want to see how to build .NET applications

Primary IDE: Visual Studio IDE
Primary OS: Windows

AGENDA

- GitHub Copilot - Introduction
- Coding
- Best practices & prompt engineering
- Workshop (1 - 2 hours long)
- Secure coding
- Wrap-up, Q&A

Outcome of this training

You will achieve...

✓ Get answers to specific use case scenario questions

✓ Increase existing Copilot skills by following a specific workshop tutorial catered to your needs

✓ Learn in-depth tips and tricks and best practices on how to best utilize GitHub Copilot

Covered in Copilot Fundamentals

We will not talk about...

- ✗ Statistics around Copilot usage and satisfaction
- ✗ Successful customer case study
- ✗ Enterprise & Organization administrator interface

GitHub Copilot Fundamentals Recap

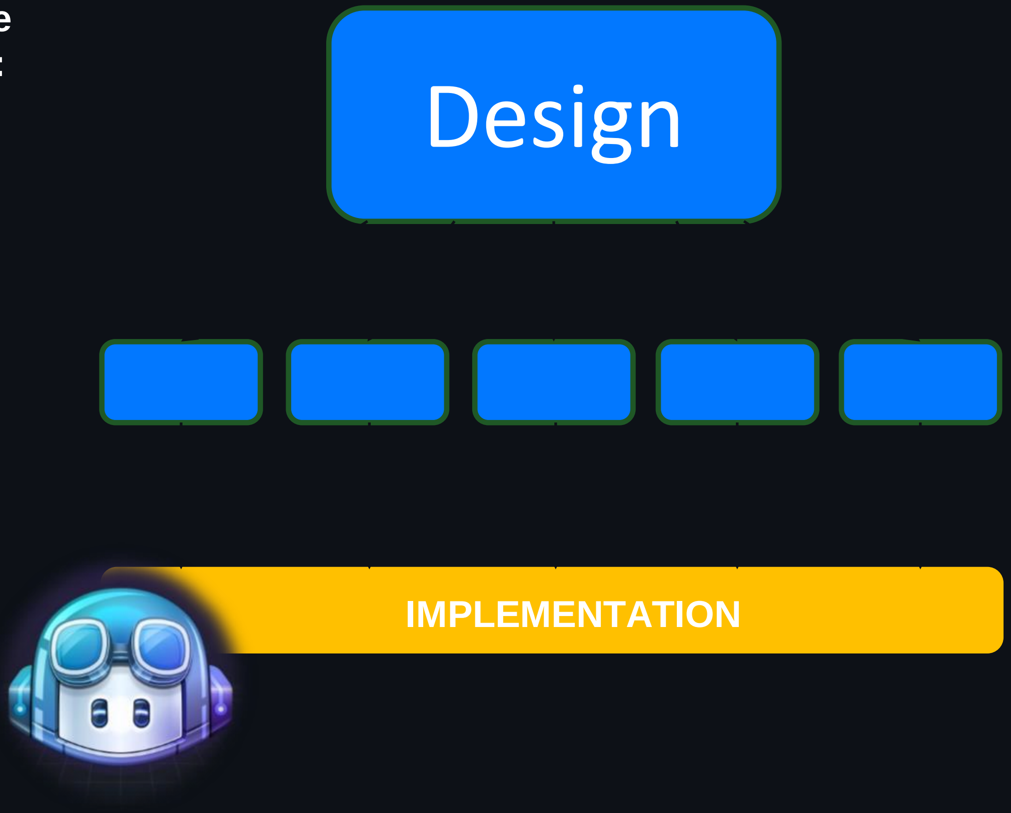
The software process can be broken down into two steps:

1) Design

2) Implementation

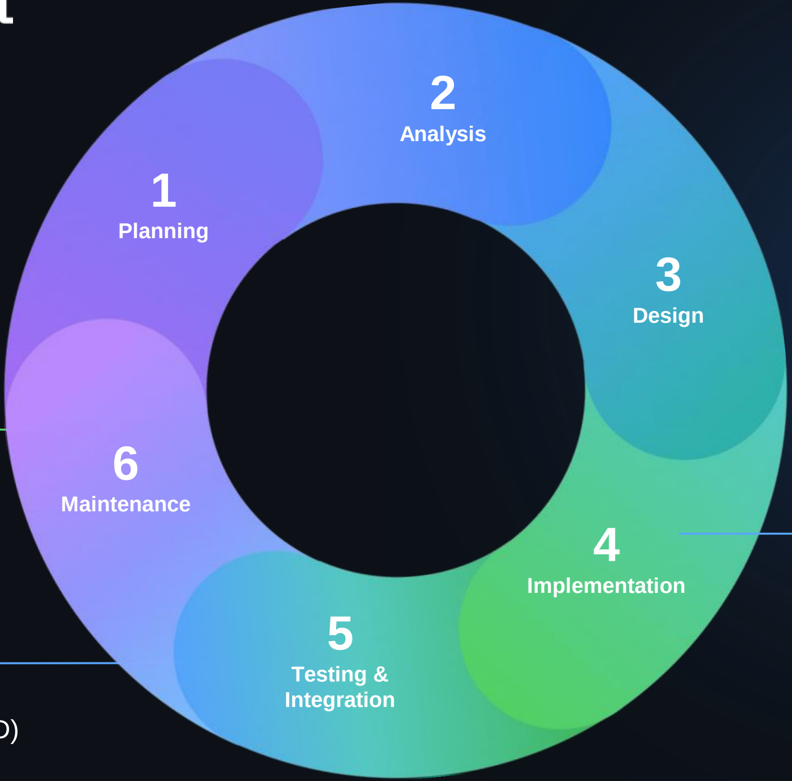
The first step is driven by you.

The second step is where Copilot can assist you with the development effort.



GitHub Copilot

Helps developers stay in the flow throughout the entire SDLC



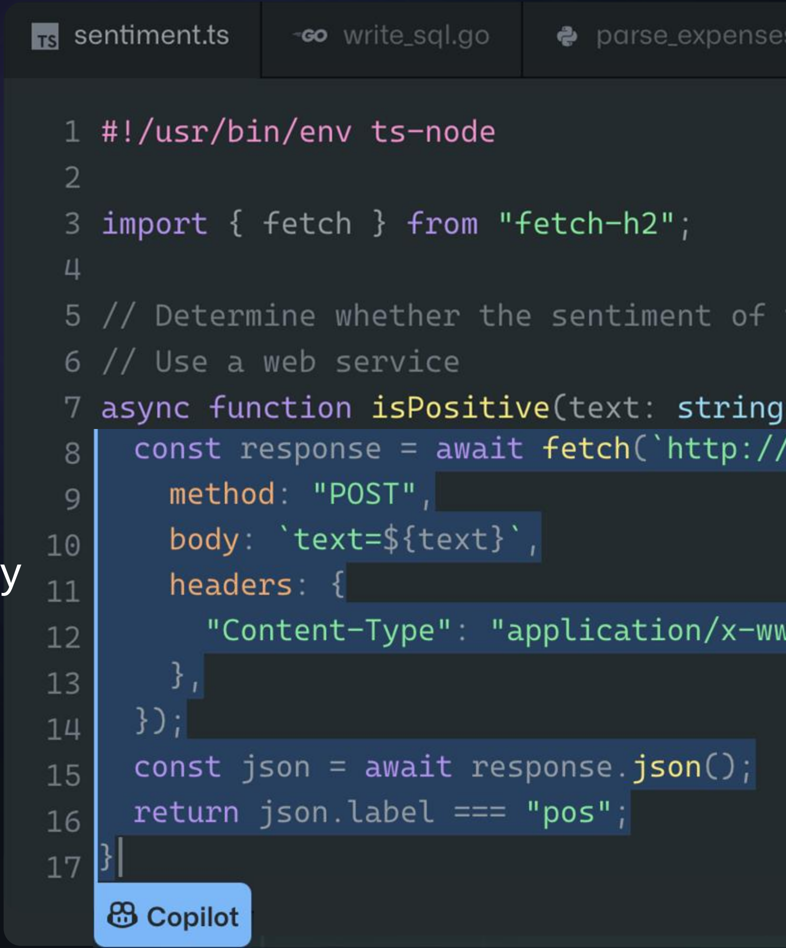
Refactoring code (code translate)
Reviewing code (code explain)
Documentation

Unit testing (TDD and BDD)
Finding code errors
Debugging
Code review
AI Pull Requests

Convert comments to code
Autofill for repetitive code
Show alternatives

GitHub Copilot

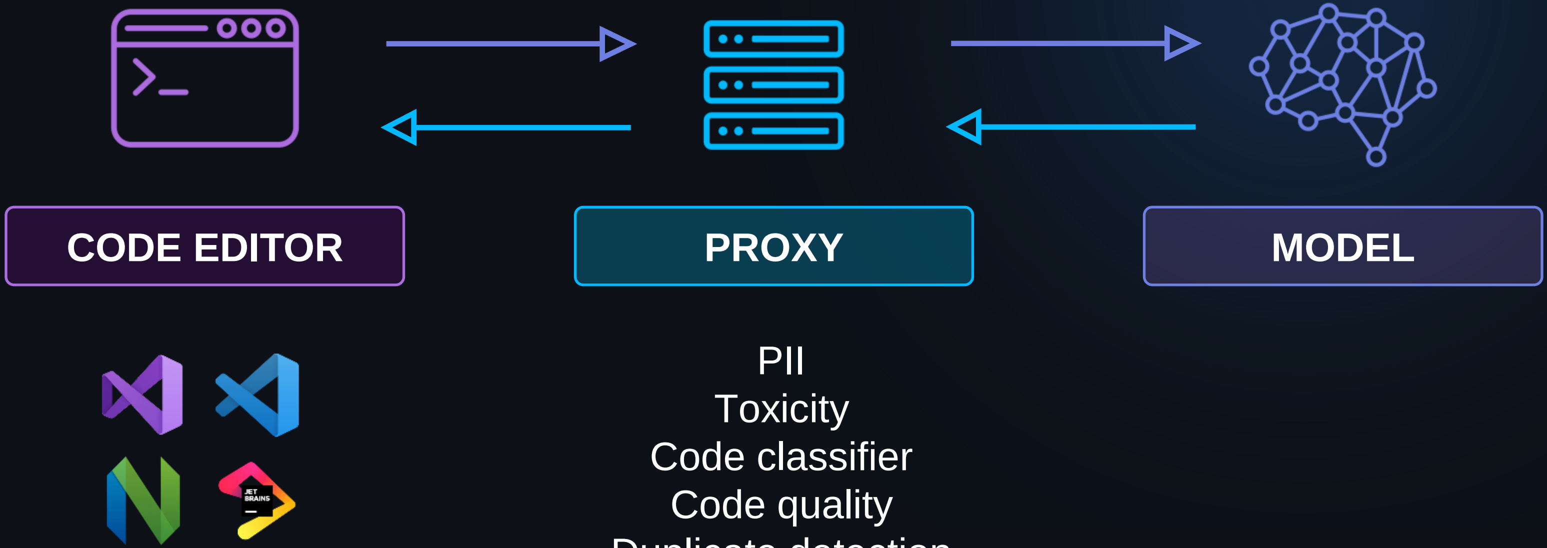
- An intelligent pair programmer
- Draws context from comments & code in open tabs to suggest individual lines and whole functions
- Powered by OpenAI Codex
  - Copilot uses a transformative model
  - Trained on large datasets to ensure accuracy
- Available as extensions to popular IDEs
- Programming Languages and Technology available in Public code base all are supported



```
#!/usr/bin/env ts-node

import { fetch } from "fetch-h2";

// Determine whether the sentiment of
// Use a web service
async function isPositive(text: string
  const response = await fetch(`http://
    method: "POST",
    body: `text=${text}`,
    headers: {
      "Content-Type": "application/x-ww
    },
  });
  const json = await response.json();
  return json.label === "pos";
}
```

February 2023

Data flow through the Copilot ecosystem



Coding

Copilot vs Copilot Chat

Copilot

Direct Code Writing

Seamless IDE Integration

Solo Development

Copilot Chat

In-Depth Assistance

Learning & Teaching

Collaborative Scenarios

Using Copilot

GitHub Copilot

- Code Driven Development
- Multilingual
- Offering Alternative Results
- Create Unit Tests
- Documentation

Guide Copilot

Good code techniques

Use good names

GitHub Copilot understands natural language

Spell out variable names

Single letter variables and abbreviations are ambiguous

Keep functions functional

Follow strong principles when creating named blocks

Be consistent

Generated code follows contextual patterns

Helpful Patterns

Variable names

Use descriptive variable names to make your intentions clear.

```
total_attendees = 5
```

Method Signatures

Define method signatures with unambiguous parameter names and types.

```
calculateAverage(int num)
```

Naming Conventions

Maintain consistent naming conventions for variables and functions

i.e. using `camelCase` for variable names consistently

Helpful Patterns (Cont.)

Input/Output Format

Describe the expected input and output formats.

"Write a function that takes an array of strings as input and returns true if a palindrome is found."

Error Handling

Specify error handling scenarios

"Exit where the integer input is empty and throw an error if the input is not an integer at all"

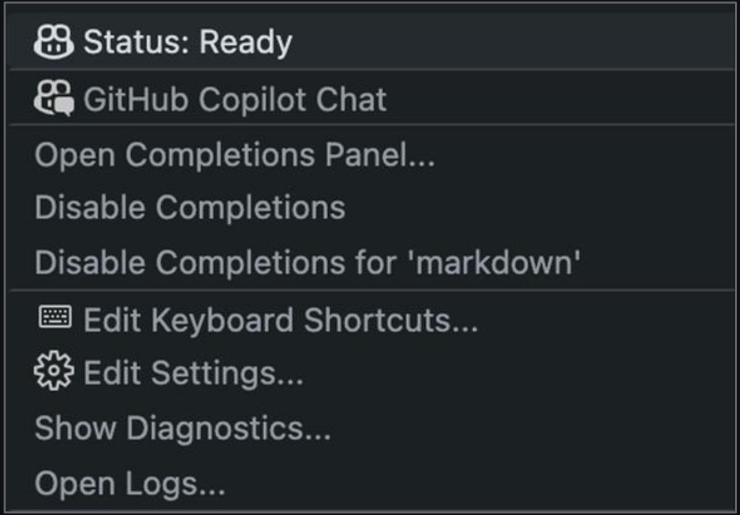
Control Structures

Describe control flow structures.

"Write a while loop to find the first prime number given a list of integers. If no such number exists, the loop should exit."

Advanced Settings

Configuration Options


Status: Ready
GitHub Copilot Chat
Open Completions Panel...
Disable Completions
Disable Completions for 'markdown'
Edit Keyboard Shortcuts...
Edit Settings...
Show Diagnostics...
Open Logs...

[Configuring GitHub Copilot in your environment]

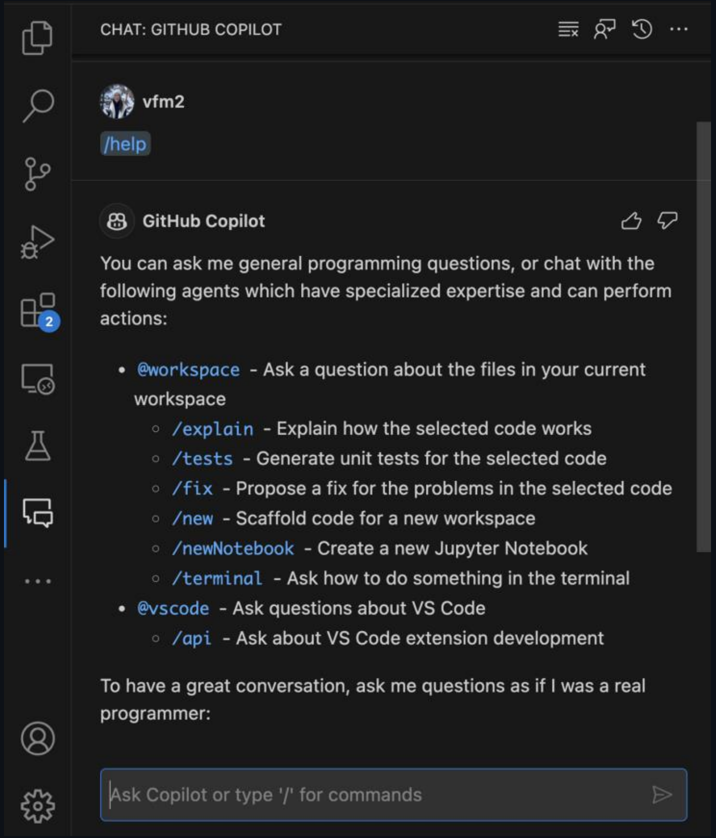
Using Copilot Chat

GitHub Copilot Chat

- ○ Refactor Code
- ○ Generate Tests
- ○ Debug
- ○ Create Workspace
- ○ Documentation

Copilot Chat: Slash Commands

/help to find available commands in your IDE



vfm2

/help

GitHub Copilot

You can ask me general programming questions, or chat with the following agents which have specialized expertise and can perform actions:

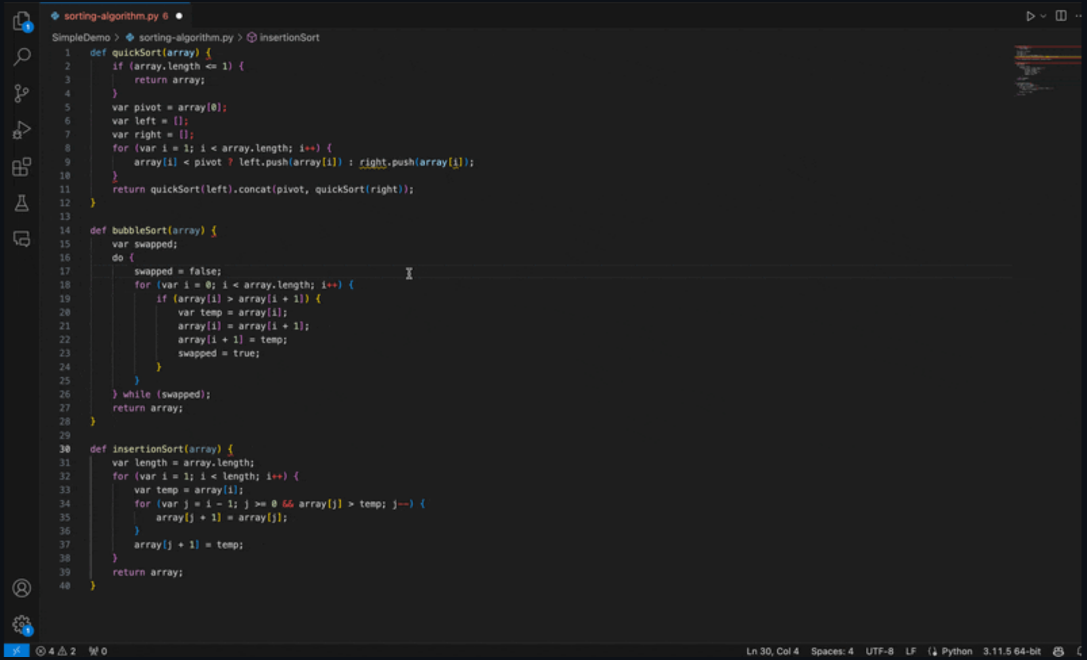
- `@workspace` - Ask a question about the files in your current workspace
  - `/explain` - Explain how the selected code works
  - `/tests` - Generate unit tests for the selected code
  - `/fix` - Propose a fix for the problems in the selected code
  - `/new` - Scaffold code for a new workspace
  - `/newNotebook` - Create a new Jupyter Notebook
  - `/terminal` - Ask how to do something in the terminal
- `@vscode` - Ask questions about VS Code
  - `/api` - Ask about VS Code extension development

To have a great conversation, ask me questions as if I was a real programmer:

Ask Copilot or type '/' for commands

In-file Copilot Chat

Copilot offers in-file Copilot feature to selectively improve



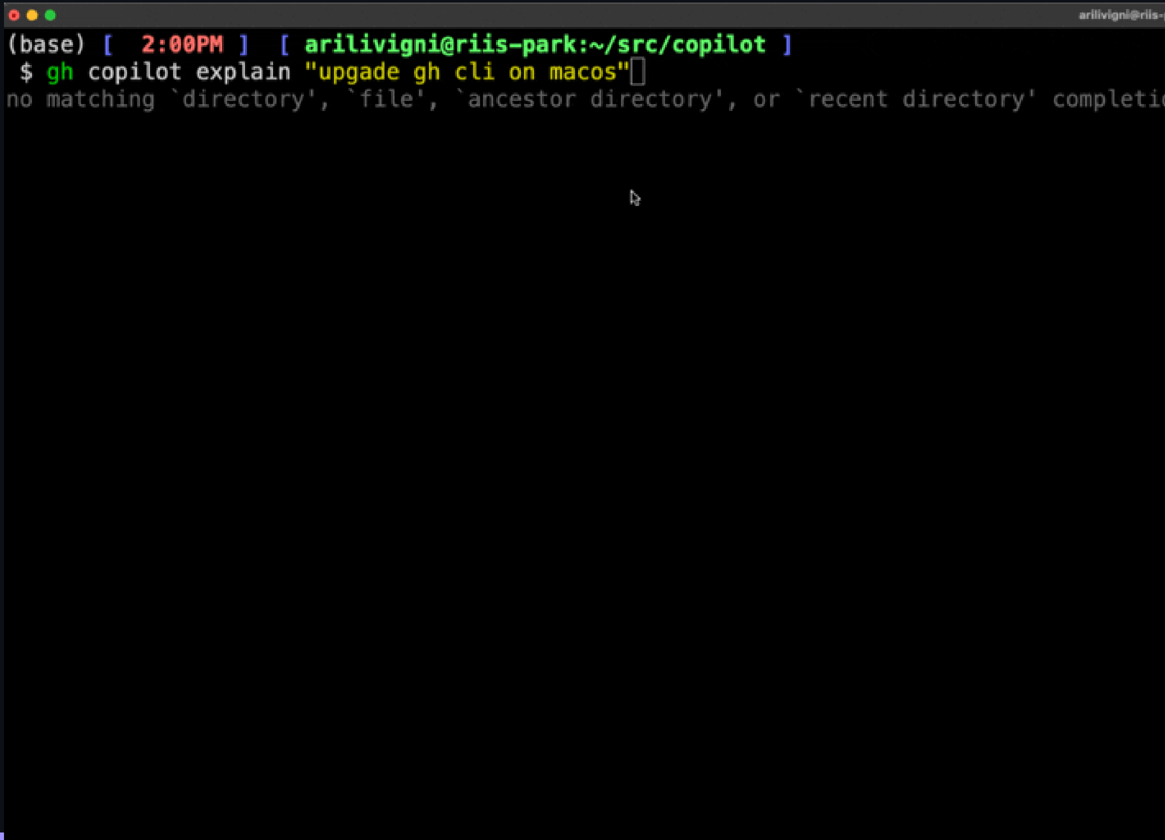
```
def quickSort(array) {
    if (array.length <= 1) {
        return array;
    }
    var pivot = array[0];
    var left = [];
    var right = [];
    for (var i = 1; i < array.length; i++) {
        array[i] < pivot ? left.push(array[i]) : right.push(array[i]);
    }
    return quickSort(left).concat(pivot, quickSort(right));
}

def bubbleSort(array) {
    var swapped;
    do {
        swapped = false;
        for (var i = 0; i < array.length; i++) {
            if (array[i] > array[i + 1]) {
                var temp = array[i];
                array[i] = array[i + 1];
                array[i + 1] = temp;
                swapped = true;
            }
        }
    } while (swapped);
    return array;
}

def insertionSort(array) {
    var length = array.length;
    for (var i = 1; i < length; i++) {
        var temp = array[i];
        for (var j = i - 1; j >= 0 && array[j] > temp; j--) {
            array[j + 1] = array[j];
        }
        array[j + 1] = temp;
    }
    return array;
}
```

GitHub Copilot Chat (default)

Run in Terminal

Ask in GitHub Copilot CLI

GitHub Copilot Chat (default)

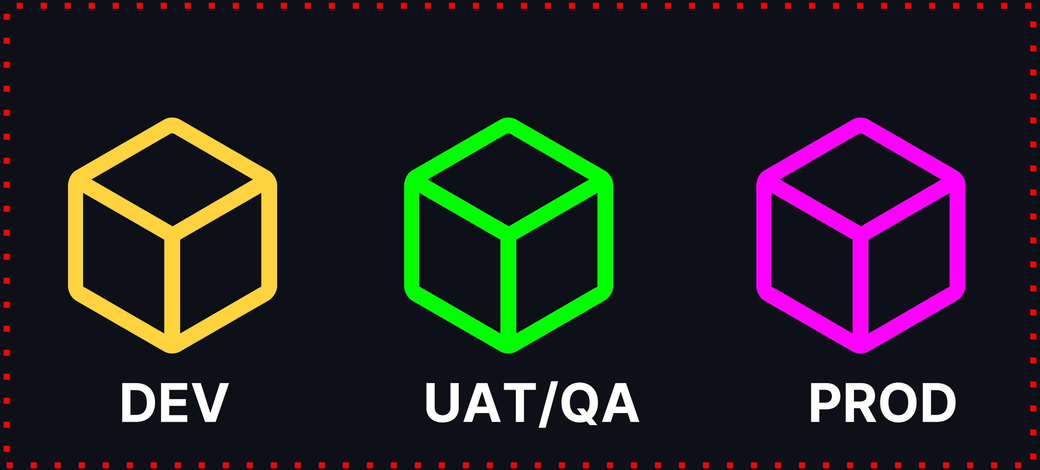


```
(base) [  2:00PM ]  [ arilivigni@riis-park:~/src/copilot ]
 $ gh copilot explain "upgade gh cli on macos"
no matching `directory', `file', `ancestor directory', or `recent directory' completi
```

What about my production layer?

Local

Cloud/Server



Current Copilot

Copilot Enterprise

Best practices

Getting accuracy closer to expectation

Problems

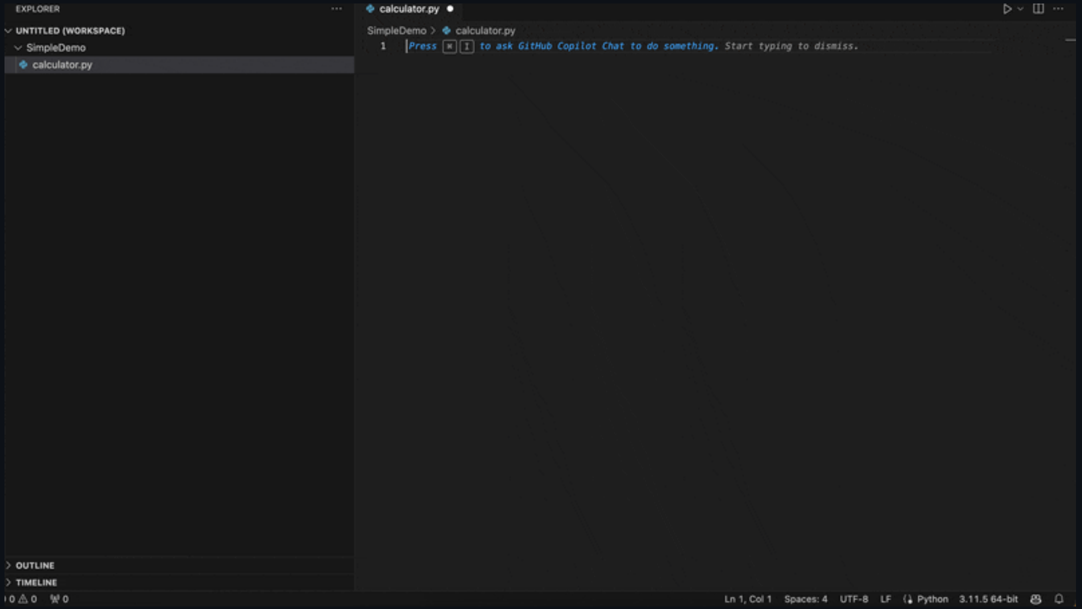
- ⚠ Copilot fails to produce answer or to keep repeating
- ⚠ Copilot generates incorrect result
- ⚠ Library/module version discrepancies issue
- ⚠ Copilot suggests non-optimal solution

Problems #1:

Copilot fails to produce answers or will keep repeating

Some problems

- Fails to produce answer
- Hallucinations
- Keeps repeating



```
Press ⌘ I to ask GitHub Copilot Chat to do something. Start typing to dismiss.
```

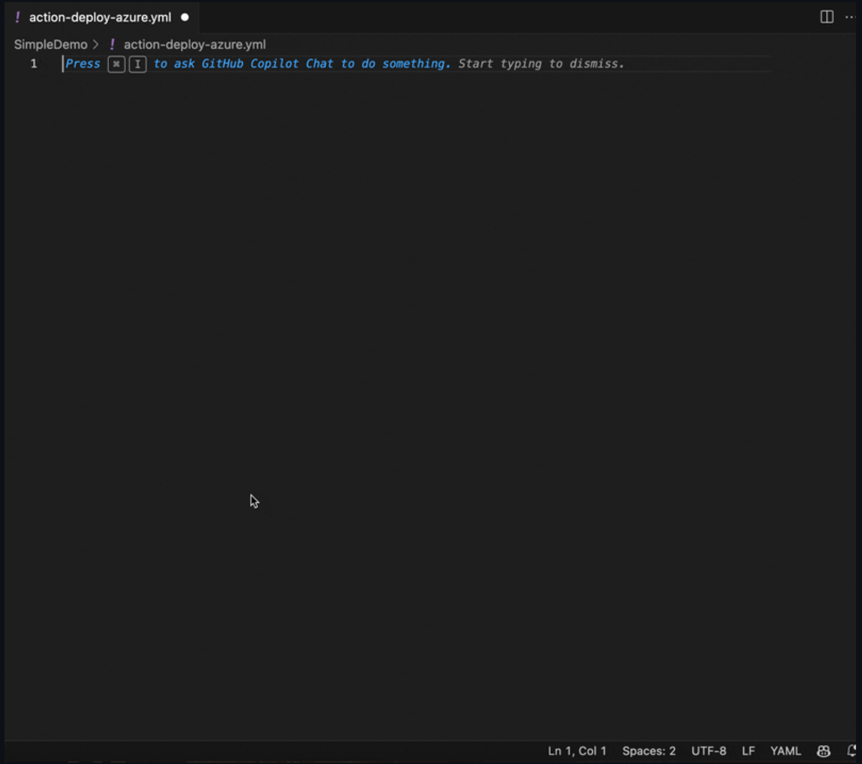
Ln 1, Col 1   Spaces: 4   UTF-8   LF   Python   3.11.5 64-bit

Problems #2:

Library/module version discrepancy

Old trained data

- While packages go through frequent updates, Copilot does not use latest data


action-deploy-azure.yml

SimpleDemo > action-deploy-azure.yml

```
Press ⌘ I to ask GitHub Copilot Chat to do something. Start typing to dismiss.
```

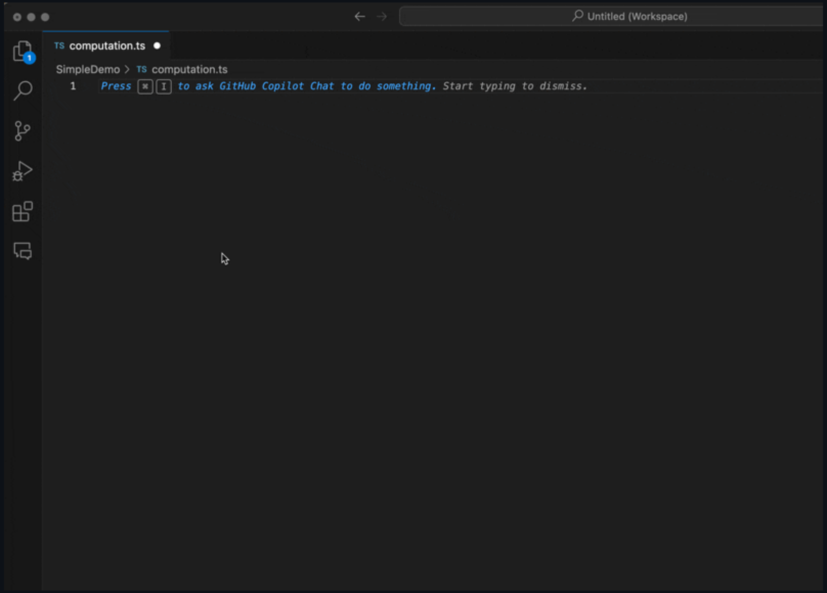
Ln 1, Col 1 Spaces: 2 UTF-8 LF YAML

Problems #3:

Copilot suggests non-optimal solution

Although solution works...

It is not optimal space complexity



Press ⌘ I to ask GitHub Copilot Chat to do something. Start typing to dismiss.

Problems #4:

Copilot Token Limit

Strategies to overcome the Token Limit

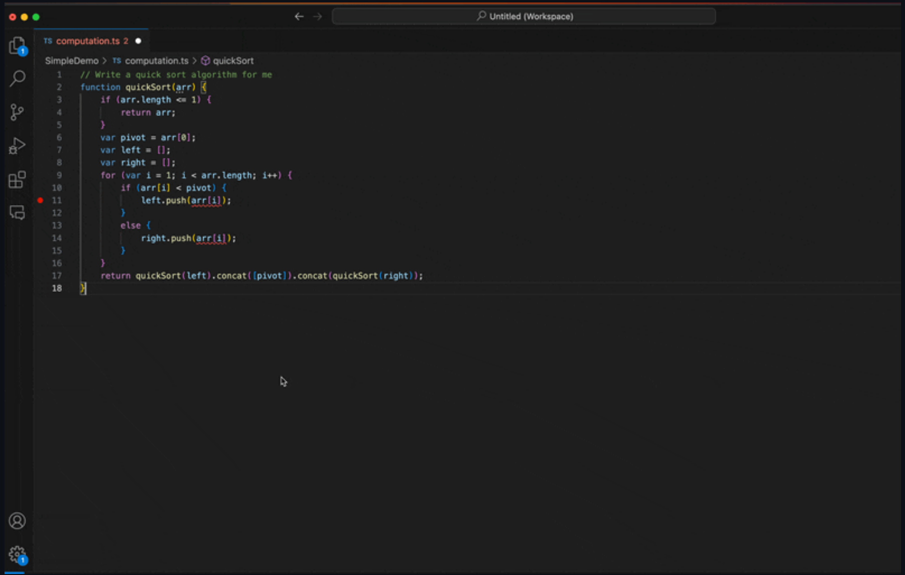
- Break Down Tasks
- Be Concise
- Iterative Development
- Post-Processing
- Stay Updated
- Feedback Loop

Solution: Prompt Engineering

Ways to improve Copilot Results

How to improve

- Ask in Copilot Chat
- Use in-line suggestion from Copilot



```
// Write a quick sort algorithm for me
function quickSort(arr) {
    if (arr.length <= 1) {
        return arr;
    }
    var pivot = arr[0];
    var left = [];
    var right = [];
    for (var i = 1; i < arr.length; i++) {
        if (arr[i] < pivot) {
            left.push(arr[i]);
        }
        else {
            right.push(arr[i]);
        }
    }
    return quickSort(left).concat([pivot]).concat(quickSort(right));
}
```

Copilot Prompt Engineering

What

Prompt engineering is the process of designing and creating high-quality prompts that can be used to generate accurate and useful code suggestions with Copilot.

Why

- Maximizes utility of Copilot
- Minimizes repeated iterations
- Determines code suggestion quality
- Skill prompts guide Copilot to understand context & nuances
- Refined interactions reduces misunderstandings

How

- Neighboring Tabs
- Zero-Shot Prompting
- One-Shot Prompting
- Few-Shot Prompting
- Let's Think Step by Step

Prompting Best Practices

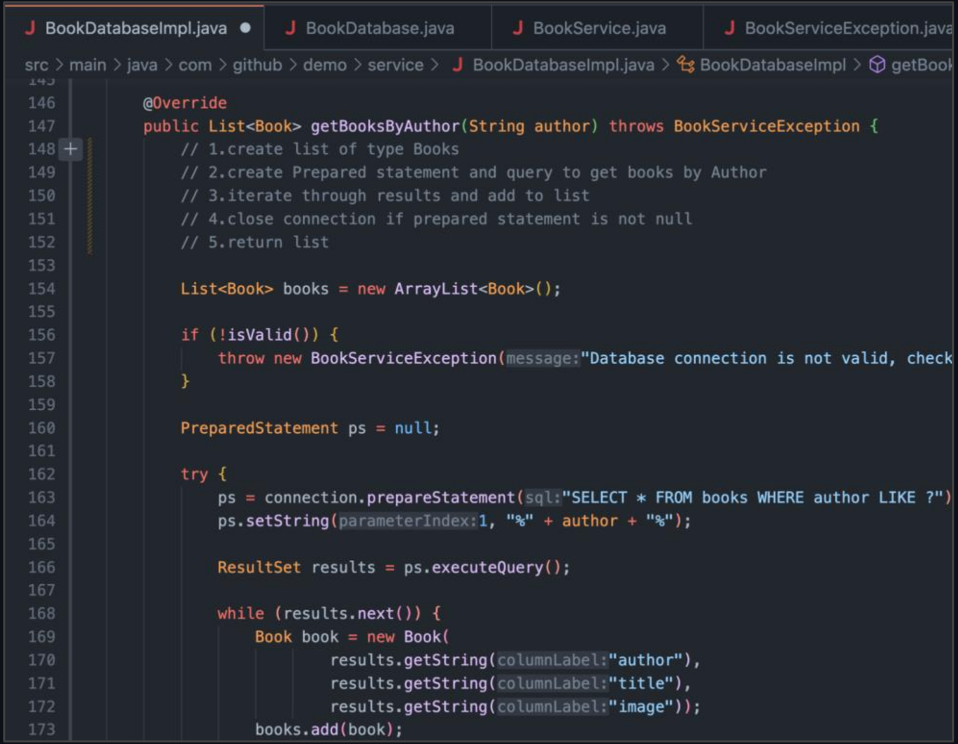
Improving results

These are best practices, now let's see how those can be used in different ways

- **Provide references**
  Improve relevance of the response by providing an example and context
- **Write clear instructions**
  Refine your prompt, provide context, write clearly, and give Copilot ample input for better results
- **Split up big tasks**
  Breaking down complex tasks minimizes errors and utilizes previous outcomes for efficiency
- **Allow time to think**
  Requesting Copilot's thought process will enhance Copilot accuracy, but it may prolong wait times.
- **Test changes systematically**
  Measure performance, watch for prompt changes' side effects, and use test suites for implementation

Neighboring Tabs

Keep relevant files open, things related to your current code


BookDatabaseImpl.java
BookDatabase.java
BookService.java
BookServiceException.java
src > main > java > com > github > demo > service > BookDatabaseImpl.java > BookDatabaseImpl > getBook

```java
    @Override
    public List<Book> getBooksByAuthor(String author) throws BookServiceException {
        // 1.create list of type Books
        // 2.create Prepared statement and query to get books by Author
        // 3.iterate through results and add to list
        // 4.close connection if prepared statement is not null
        // 5.return list

        List<Book> books = new ArrayList<Book>();

        if (!isValid()) {
            throw new BookServiceException(message:"Database connection is not valid, check
        }

        PreparedStatement ps = null;

        try {
            ps = connection.prepareStatement(sql:"SELECT * FROM books WHERE author LIKE ?")
            ps.setString(parameterIndex:1, "%" + author + "%");

            ResultSet results = ps.executeQuery();

            while (results.next()) {
                Book book = new Book(
                        results.getString(columnLabel:"author"),
                        results.getString(columnLabel:"title"),
                        results.getString(columnLabel:"image"));
                books.add(book);
```

If at first you don’t succeed, iterate!

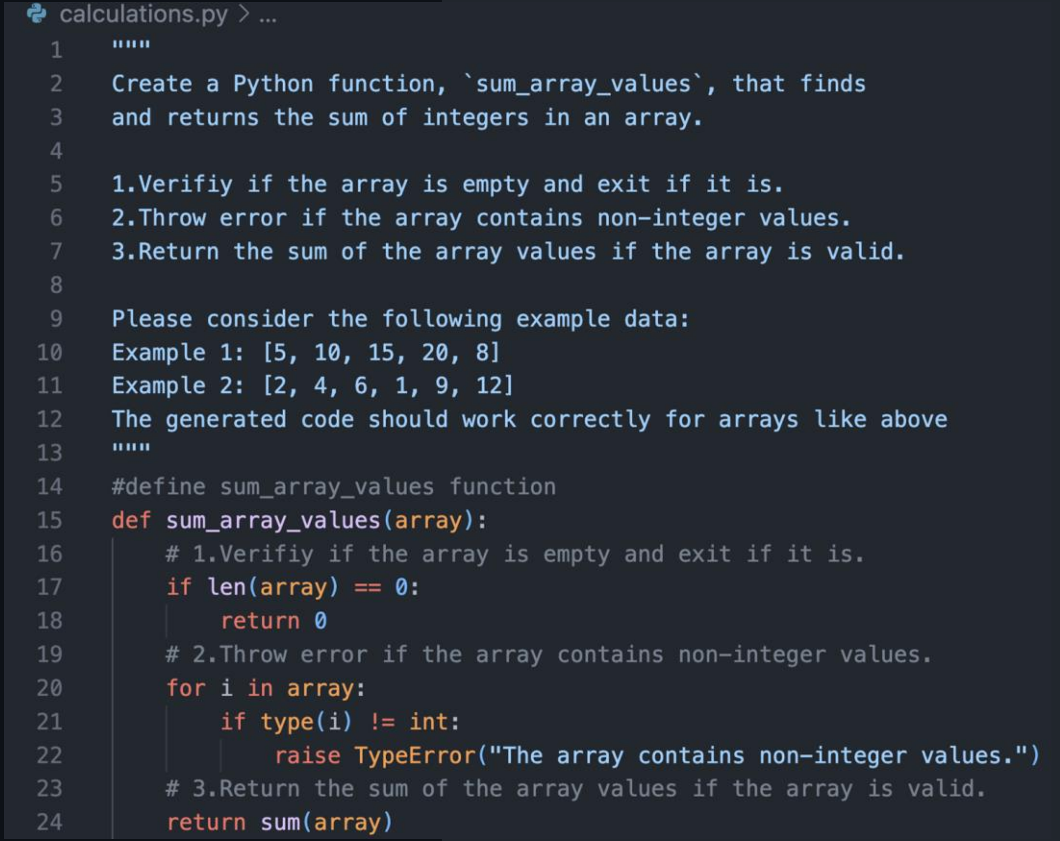
Zero Shot Prompt

```javascript
//Write a JavaScript function that finds
//the maximum value in an array.
function max(array) {
    return Math.max.apply(null, array);
}
```

One-Shot Prompt

```javascript
//Create a JavaScript function, `findMax`, that takes an array of
//and returns the string with the maximum length.
//Ensure that the function works correctly for arrays that are em
//as well as arrays that contain both strings and numbers.
function findMax(array) {
    var max = array[0].length;
    array.map(v => max = Math.max(max, v.length));
    result = array.filter(v => v.length == max);
    return result;
}
```

Provide a high level description to clarify expectations



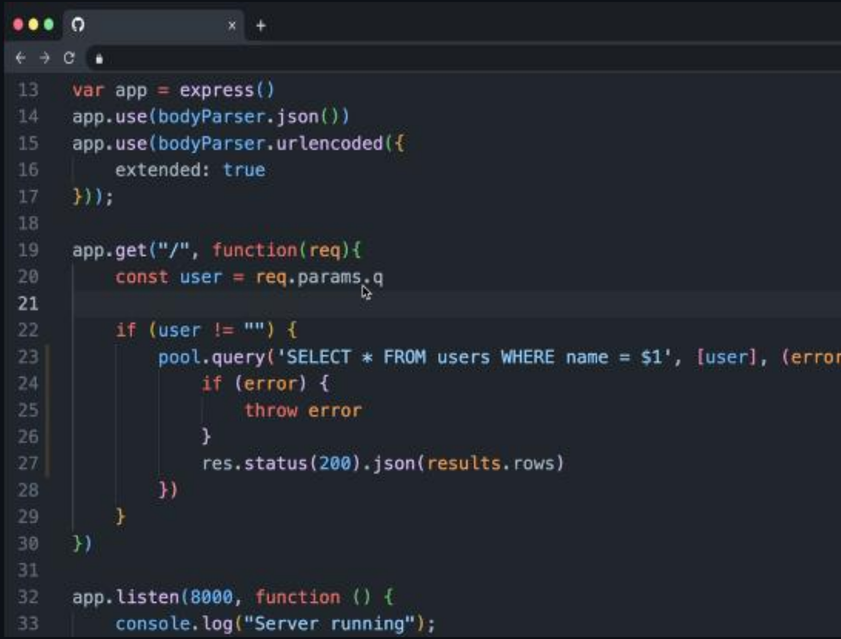
calculations.py > ...

```python
"""
Create a Python function, `sum_array_values`, that finds
and returns the sum of integers in an array.

1.Verifiy if the array is empty and exit if it is.
2.Throw error if the array contains non-integer values.
3.Return the sum of the array values if the array is valid.

Please consider the following example data:
Example 1: [5, 10, 15, 20, 8]
Example 2: [2, 4, 6, 1, 9, 12]
The generated code should work correctly for arrays like above
"""
#define sum_array_values function
def sum_array_values(array):
    # 1.Verifiy if the array is empty and exit if it is.
    if len(array) == 0:
        return 0
    # 2.Throw error if the array contains non-integer values.
    for i in array:
        if type(i) != int:
            raise TypeError("The array contains non-integer values.")
    # 3.Return the sum of the array values if the array is valid.
    return sum(array)
```

Secure coding

Copilot and Secure Coding

- AI-based vulnerability system that helps prevent insecure coding patterns (e.g. SQL script injection)
- Vulnerability filters are applied only to the Copilot-generated suggestion itself
  - ⊗ It cannot detect downstream vulnerabilities introduced by the code e.g. on deployment infrastructure
  - ⓘ **We recommend taking the same precautions you take with code written by engineers (linting, code scanning, etc.)**
- Copilot Chat can be used to query code for known vulnerabilities



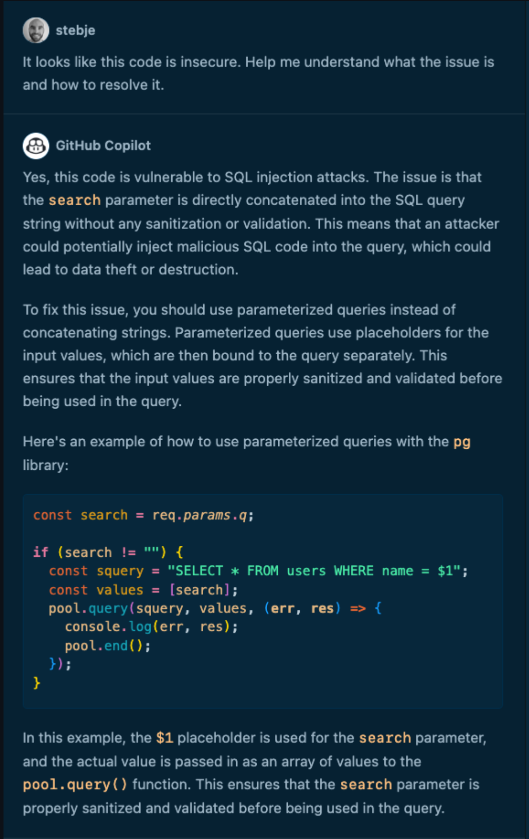
```
var app = express()
app.use(bodyParser.json())
app.use(bodyParser.urlencoded({
    extended: true
}));

app.get("/", function(req){
    const user = req.params.q

    if (user != "") {
        pool.query('SELECT * FROM users WHERE name = $1', [user], (error
            if (error) {
                throw error
            }
            res.status(200).json(results.rows)
        })
    }
})

app.listen(8000, function () {
    console.log("Server running");
```

Copilot + GHAS

- Copilot is not a replacement of GHAS features
- Copilot can be used in tandem with GHAS features to detect and remediate vulnerabilities earlier during the SDLC
  - GHAS Code scanning results
  - GHAS Secret scanning

Detection & Remediation

- Prompt for insecure coding patterns
- Use GHAS Code Scanning results
- Write custom CodeQL queries
- Increase your knowledge of secure coding patterns
- Create custom Secret Scanning patterns



stebje

It looks like this code is insecure. Help me understand what the issue is and how to resolve it.

GitHub Copilot

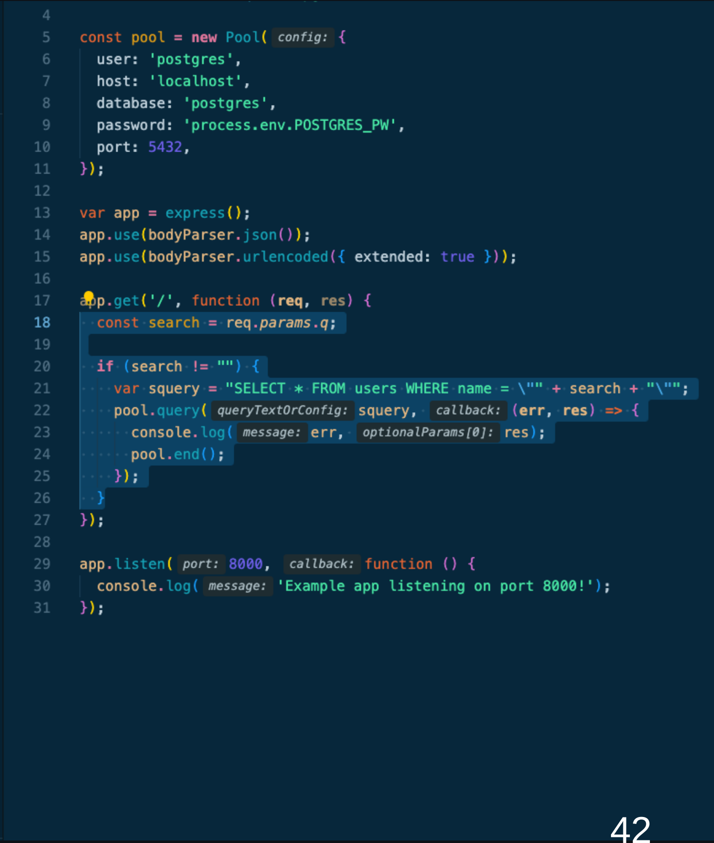
Yes, this code is vulnerable to SQL injection attacks. The issue is that the `search` parameter is directly concatenated into the SQL query string without any sanitization or validation. This means that an attacker could potentially inject malicious SQL code into the query, which could lead to data theft or destruction.

To fix this issue, you should use parameterized queries instead of concatenating strings. Parameterized queries use placeholders for the input values, which are then bound to the query separately. This ensures that the input values are properly sanitized and validated before being used in the query.

Here's an example of how to use parameterized queries with the `pg` library:

```
const search = req.params.q;

if (search != "") {
  const squery = "SELECT * FROM users WHERE name = $1";
  const values = [search];
  pool.query(squery, values, (err, res) => {
    console.log(err, res);
    pool.end();
  });
}
```

In this example, the `$1` placeholder is used for the `search` parameter, and the actual value is passed in as an array of values to the `pool.query()` function. This ensures that the `search` parameter is properly sanitized and validated before being used in the query.

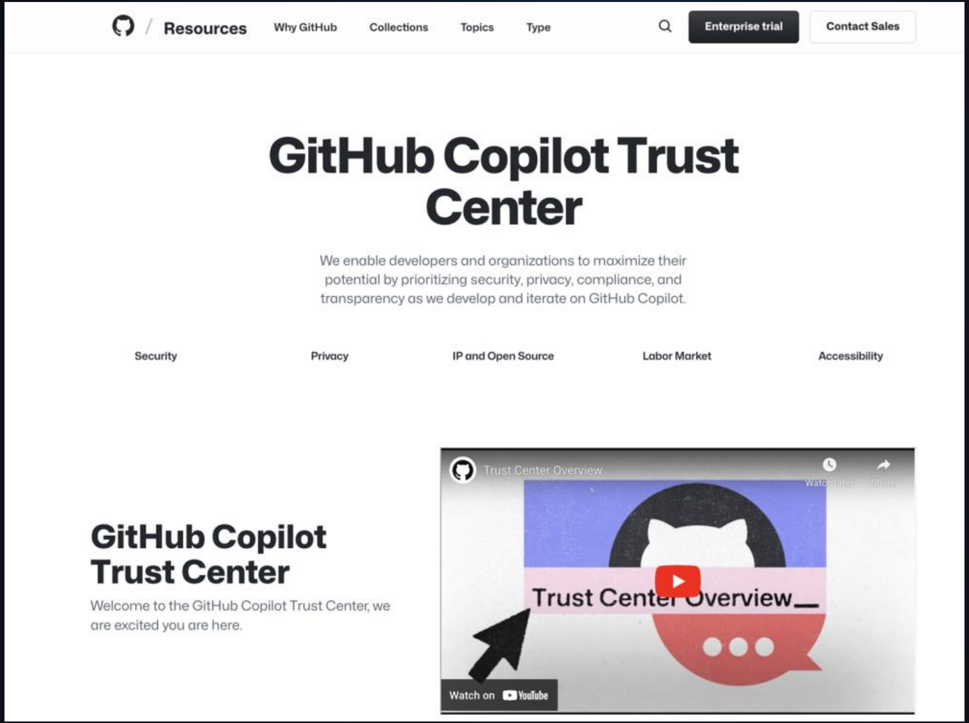


```

const pool = new Pool( config: {
  user: 'postgres',
  host: 'localhost',
  database: 'postgres',
  password: 'process.env.POSTGRES_PW',
  port: 5432,
});

var app = express();
app.use(bodyParser.json());
app.use(bodyParser.urlencoded({ extended: true }));

app.get('/', function (req, res) {
  const search = req.params.q;

  if (search != "") {
    var squery = "SELECT * FROM users WHERE name = \"" + search + "\"";
    pool.query( queryTextOrConfig: squery,  callback: (err, res) => {
      console.log( message: err,  optionalParams[0]: res);
      pool.end();
    });
  }
});

app.listen( port: 8000,  callback: function () {
  console.log( message: 'Example app listening on port 8000!');
});
```

Security & Trust

Copilot Trust Center

- Security
- Privacy
- Data flow
- Copyright
- Labor market
- Accessibility
- Contracting


Resources Why GitHub Collections Topics Type Enterprise trial Contact Sales

GitHub Copilot Trust Center

We enable developers and organizations to maximize their potential by prioritizing security, privacy, compliance, and transparency as we develop and iterate on GitHub Copilot.

Security Privacy IP and Open Source Labor Market Accessibility

GitHub Copilot Trust Center

Welcome to the GitHub Copilot Trust Center, we are excited you are here.

GitHub Copilot Developers

Wrap Up

Thank you