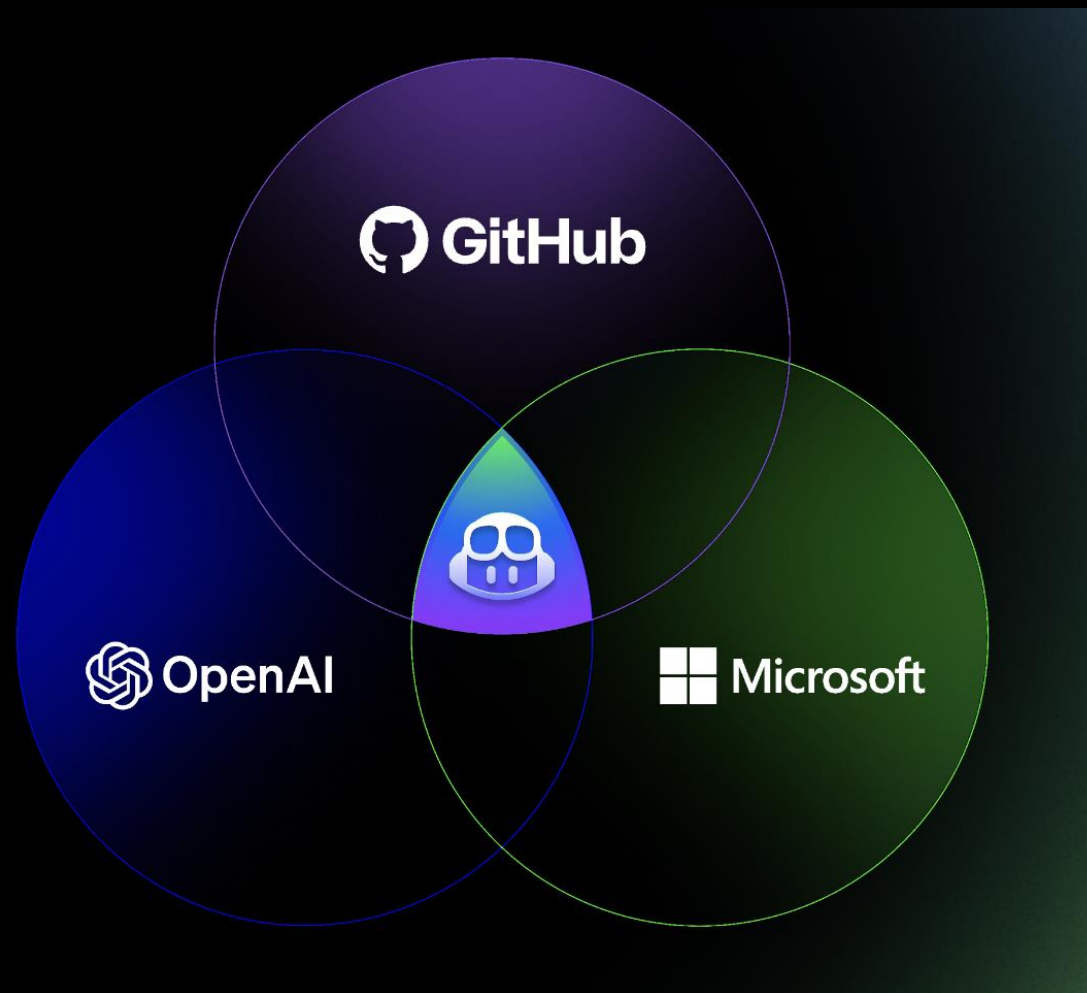




GitHub Copilot in SDLC

Vicky Israni
Senior Solutions Engineer
GitHub



Hi, I'm Vicky.

Vicky Israni

Senior Solutions Engineer, GitHub

linkedin.com/in/vicky-israni

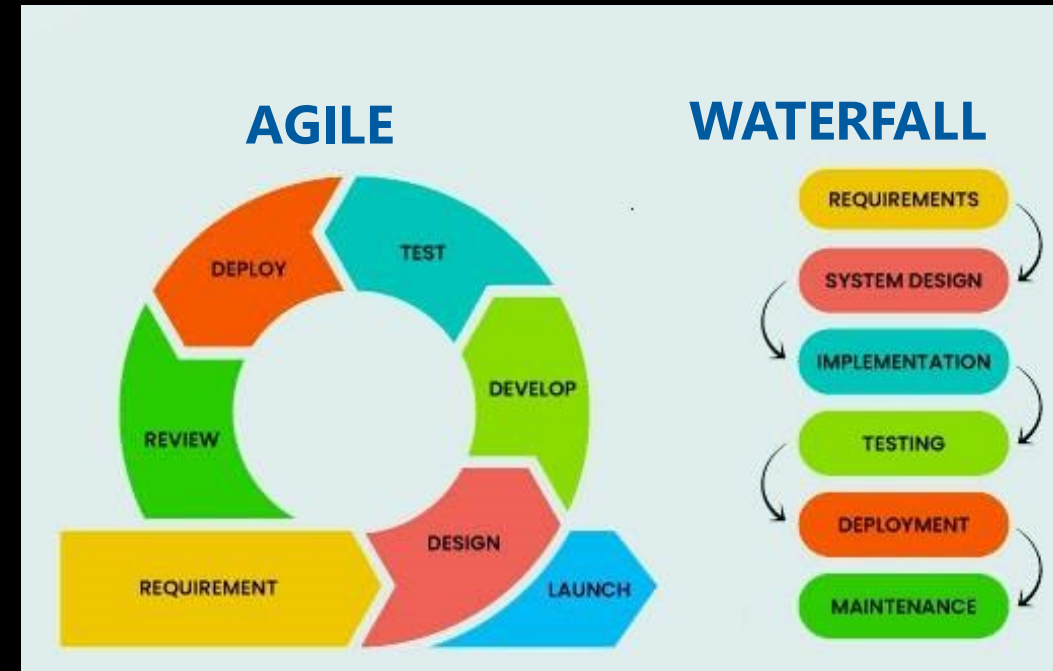
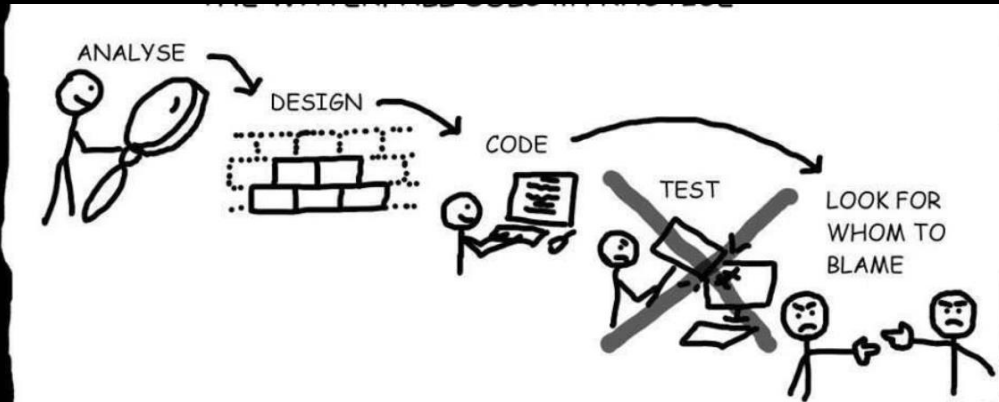


Vicky Israni

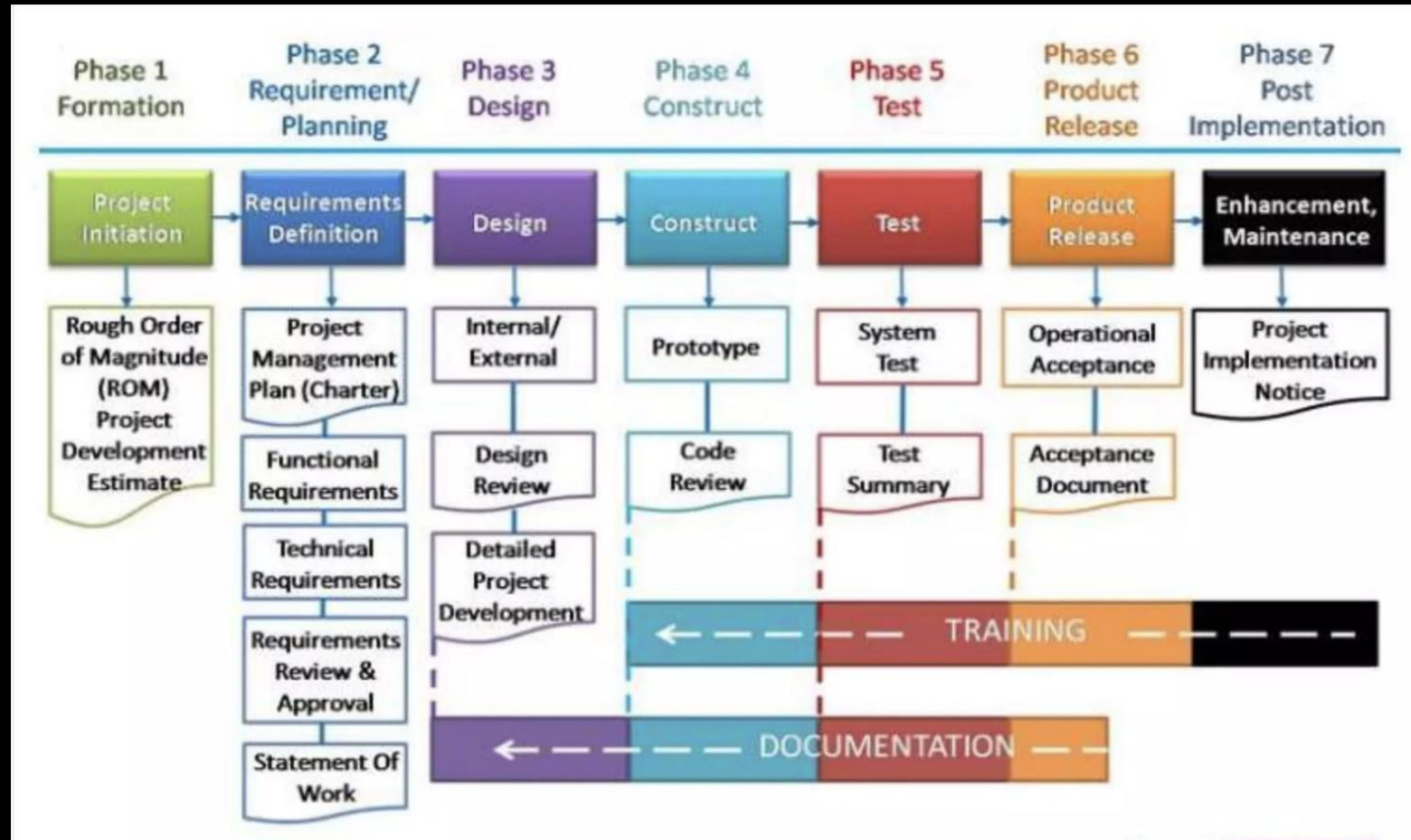
Sr. Solutions Engineer at GitHub |
Technical Consulting | Pre-Sales



SDLC – Innovate, Iterate, Integrate



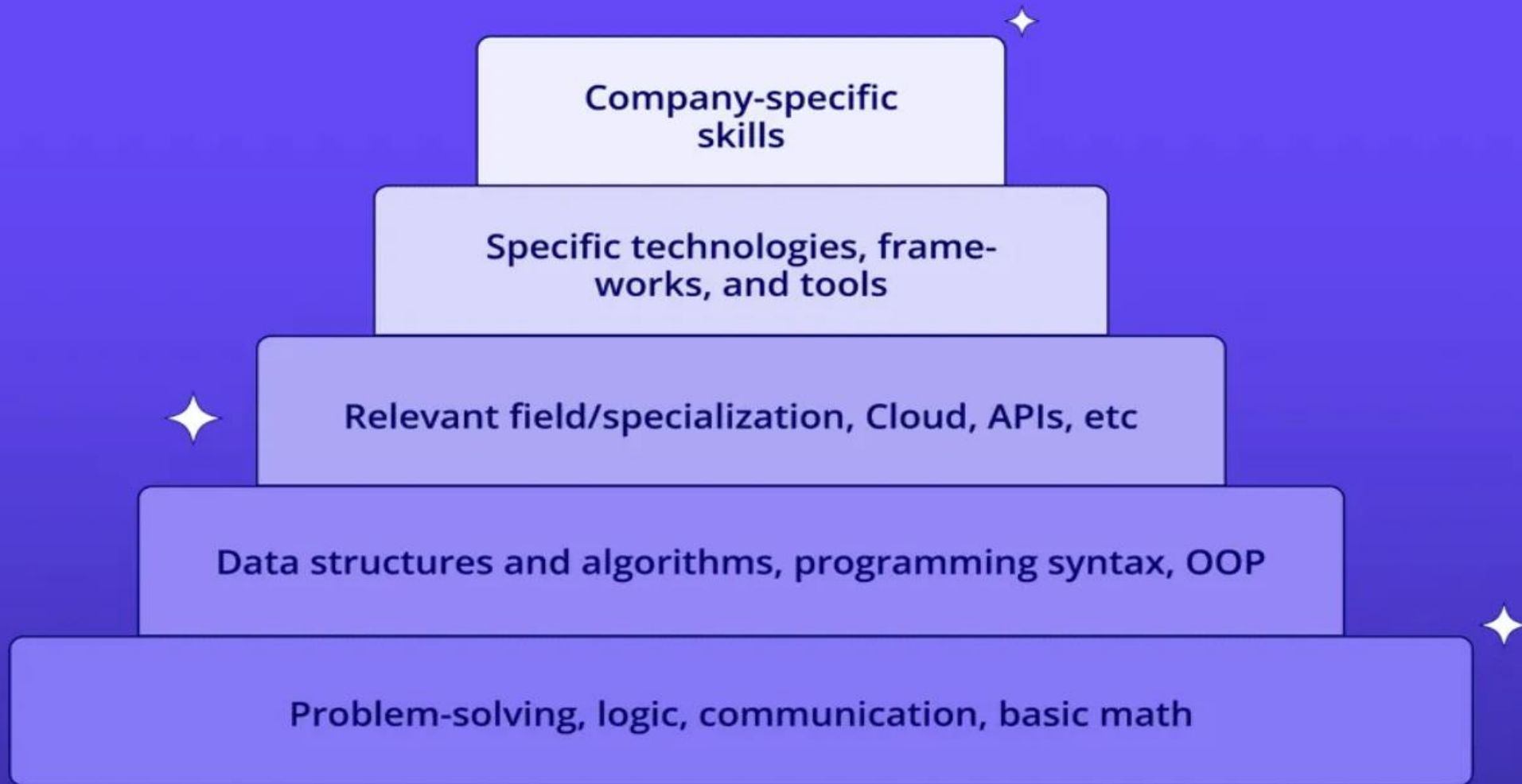
SDLC – When Planning meets Panic



Roles in SDLC



THE DEVELOPER SKILL PYRAMID



What we love...



INNOVATION



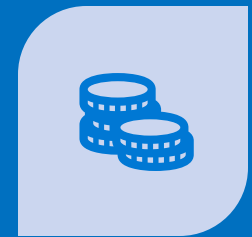
SPEED



DEVELOPER
PRODUCTIVITY

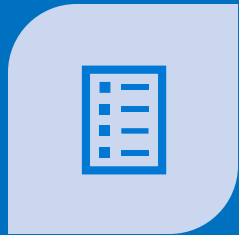


DELIVERY
EXCELLENCE



COST
REDUCTION

What loves us back...



GROWING
BACKLOG



DELAYED
DELIVERIES



EXHAUSTED
DEVELOPERS



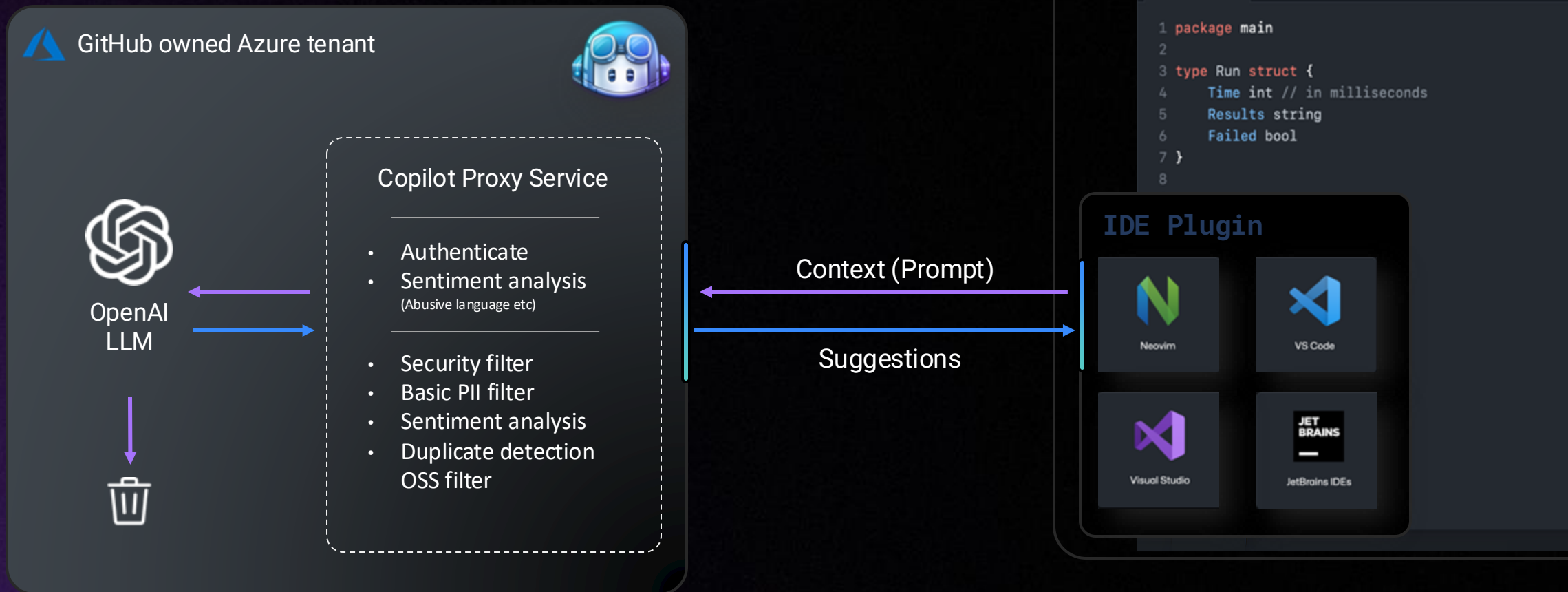
PROCESS
DETERIORATION



RISING
COSTS



Data flow





The Copilot effect.

The world's most widely adopted AI developer tool.

Maximize developer happiness

Increase developer productivity

Accelerate software development



Requirements to Code OR Code to Requirements?



He finally found the time to write down the business requirements. Too bad there wasn't a developer around to implement them.

Requirements – Lets Break Them

☰

📄

🔍

👤 4

🔗

📁

🧪

🐙

💬

...

👤

⚙️

CHAT

👤 GitHub Copilot

Hi @visrani, how can I help you?

/fix the problems in my code

/tests add unit tests for my code

/explain how the selected code works

Ask Copilot or type / for commands

📄 BookStore_Ideation.txt U ✕

docs > requirements > 📄 BookStore_Ideation.txt

```
1 Here are the high-level requirements for implementing a book store website:
2
3 ### 1. User Management
4 - **User Registration and Login:**
5   - Users should be able to create an account and log in using their credentials.
6   - Implement password recovery and account verification mechanisms.
7
8 ### 2. Book Catalog
9 - **Book Listing:**
10  - Display a comprehensive list of available books.
11  - Each book entry should include the title, author, genre, publication date, and a brief c
12  - Show book covers/thumbnails.
13
14 ### 3. Search and Filter
15 - **Search Functionality:**
16   - Allow users to search for books by title, author, ISBN, or keywords.
17 - **Filter Options:**
18   - Enable filtering by genre, author, publication date, and price range.
19 - **Sort Options:**
20   - Provide sorting options such as alphabetical order, price (low to high, high to low), pu
21
22 ### 4. Book Details
23 - **Detailed Book Page:**
24   - Display detailed information about the book including title, author, detailed descriptio
25   - Show additional information like publisher, number of pages, language, and ISBN.
26
27 ### 5. Ratings and Reviews
28 - **User Ratings:**
29   - Allow users to rate books on a scale (e.g., 1 to 5 stars).
30 - **User Reviews:**
31   - Enable users to write and read reviews for each book.
32   - Display the average rating and the total number of reviews.
```

Analysis – Deciding the Technologies

CHAT

GitHub Copilot

Hi @visrani, how can I help you?

/fix the problems in my code

/tests add unit tests for my code

/explain how the selected code works

Ask Copilot or type / for commands

BookStore_Ideation.txt

docs > requirements > BookStore_Ideation.txt

```
1 Here are the high-level requirements for implementing a book store website:
2
3 ### 1. User Management
4 - **User Registration and Login:**
5   - Users should be able to create an account and log in using their credentials.
6   - Implement password recovery and account verification mechanisms.
7
8 ### 2. Book Catalog
9 - **Book Listing:**
10  - Display a comprehensive list of available books.
11  - Each book entry should include the title, author, genre, publication date, and a brief c
12  - Show book covers/thumbnails.
13
14 ### 3. Search and Filter
15 - **Search Functionality:**
16   - Allow users to search for books by title, author, ISBN, or keywords.
17 - **Filter Options:**
18   - Enable filtering by genre, author, publication date, and price range.
19 - **Sort Options:**
20   - Provide sorting options such as alphabetical order, price (low to high, high to low), p
21
22 ### 4. Book Details
23 - **Detailed Book Page:**
24   - Display detailed information about the book including title, author, detailed descripti
25   - Show additional information like publisher, number of pages, language, and ISBN.
26
27 ### 5. Ratings and Reviews
28 - **User Ratings:**
29   - Allow users to rate books on a scale (e.g., 1 to 5 stars).
30 - **User Reviews:**
31   - Enable users to write and read reviews for each book.
32   - Display the average rating and the total number of reviews.
```

Design – From Classes to Functions



CHAT

 **GitHub Copilot**

Welcome, @visrani, I'm your Copilot and I'm here to help you get things done faster. You can also [start an inline chat session](#).

I'm powered by AI, so surprises and mistakes are possible. Make sure to verify any generated code or suggestions, and [share feedback](#) so that we can learn and improve. Check out the [Copilot documentation](#) to learn more.

/fix the problems in my code

/tests add unit tests for my code

/explain how the selected code works

Ask Copilot or type / for commands

Bookstore_Ideation.txt U X

docs > requirements > Bookstore_Ideation.txt

```
1 Here are the high-level requirements for implementing a book store website:
2
3 ### 1. User Management
4 - **User Registration and Login:**
5   - Users should be able to create an account and log in using their creden
6   - Implement password recovery and account verification mechanisms.
7
8 ### 2. Book Catalog
9 - **Book Listing:**
10  - Display a comprehensive list of available books.
11  - Each book entry should include the title, author, genre, publication da
12  - Show book covers/thumbnails.
13
14 ### 3. Search and Filter
15 - **Search Functionality:**
16   - Allow users to search for books by title, author, ISBN, or keywords.
17 - **Filter Options:**
18   - Enable filtering by genre, author, publication date, and price range.
19 - **Sort Options:**
20   - Provide sorting options such as alphabetical order, price (low to high,
21
22 ### 4. Book Details
23 - **Detailed Book Page:**
24   - Display detailed information about the book including title, author, de
25   - Show additional information like publisher, number of pages, language,
26
27 ### 5. Ratings and Reviews
28 - **User Ratings:**
29   - Allow users to rate books on a scale (e.g., 1 to 5 stars).
30 - **User Reviews:**
31   - Enable users to write and read reviews for each book.
32   - Display the average rating and the total number of reviews.
```

Design – Visualization

CHAT

visrani

Design template component diagram for this in mermaid code

GitHub Copilot

> Used 1 reference

```
classDiagram
    class BookCatalog {
        +List of Available Books
    }
    class BookListing {
        +Title
        +Author
        +Genre
        +Publication Date
        +Brief Description
        +Book Covers/Thumbnails
    }
    class SearchAndFilter {
        +Search by Title, Author, ISBN, Keywords
        +Filter by Genre, Author, Publication Date, Price
        +Sort by Alphabetical, Price, Publication Date,
    }
    class BookDetails {
        +Detailed Information
        +Publisher
        +Number of Pages
        +Language
        +ISBN
    }

    BookCatalog <|-- BookListing : Contains
    BookCatalog <|-- SearchAndFilter : Utilizes
```

What are the features of the UserManagement component?

Ask Copilot or type / for commands

Bookstore_Ideation.txt

docs > requirements > Bookstore_Ideation.txt

```
1 Here are the high-level requirements for implementing a book store website:
2
3 ### 1. User Management
4 - **User Registration and Login:**
5   - Users should be able to create an account and log in using their credentials.
6   - Implement password recovery and account verification mechanisms.
7
8 ### 2. Book Catalog
9 - **Book Listing:**
10  - Display a comprehensive list of available books.
11  - Each book entry should include the title, author, genre, publication date, and a brief description.
12  - Show book covers/thumbnails.
13
14 ### 3. Search and Filter
15 - **Search Functionality:**
16  - Allow users to search for books by title, author, ISBN, or keywords.
17 - **Filter Options:**
18  - Enable filtering by genre, author, publication date, and price range.
19 - **Sort Options:**
20  - Provide sorting options such as alphabetical order, price (low to high, high to low), publication date.
21
22 ### 4. Book Details
23 - **Detailed Book Page:**
24  - Display detailed information about the book including title, author, detailed description, publication date.
25  - Show additional information like publisher, number of pages, language, and ISBN.
26
27 ### 5. Ratings and Reviews
28 - **User Ratings:**
29  - Allow users to rate books on a scale (e.g., 1 to 5 stars).
30 - **User Reviews:**
31  - Enable users to write and read reviews for each book.
32  - Display the average rating and the total number of reviews.
33
34 ### 6. Pricing and Purchase Options
35 - **Pricing Information:**
36  - Clearly display the price of each book.
37  - Provide information on discounts and special offers.
38 - **Add to Cart:**
39  - Users should be able to add books to a shopping cart for purchase.
```

Design – Visualization

Mermaid v10.9.1 Live Editor



THEME ▾



SECURITY

DOCUMENTATION

TUTORIAL

MERMAID

CLI



Mermaid

</> Code

⚙️ Config

Auto sync



📄 DOCS

Diagram

Rough



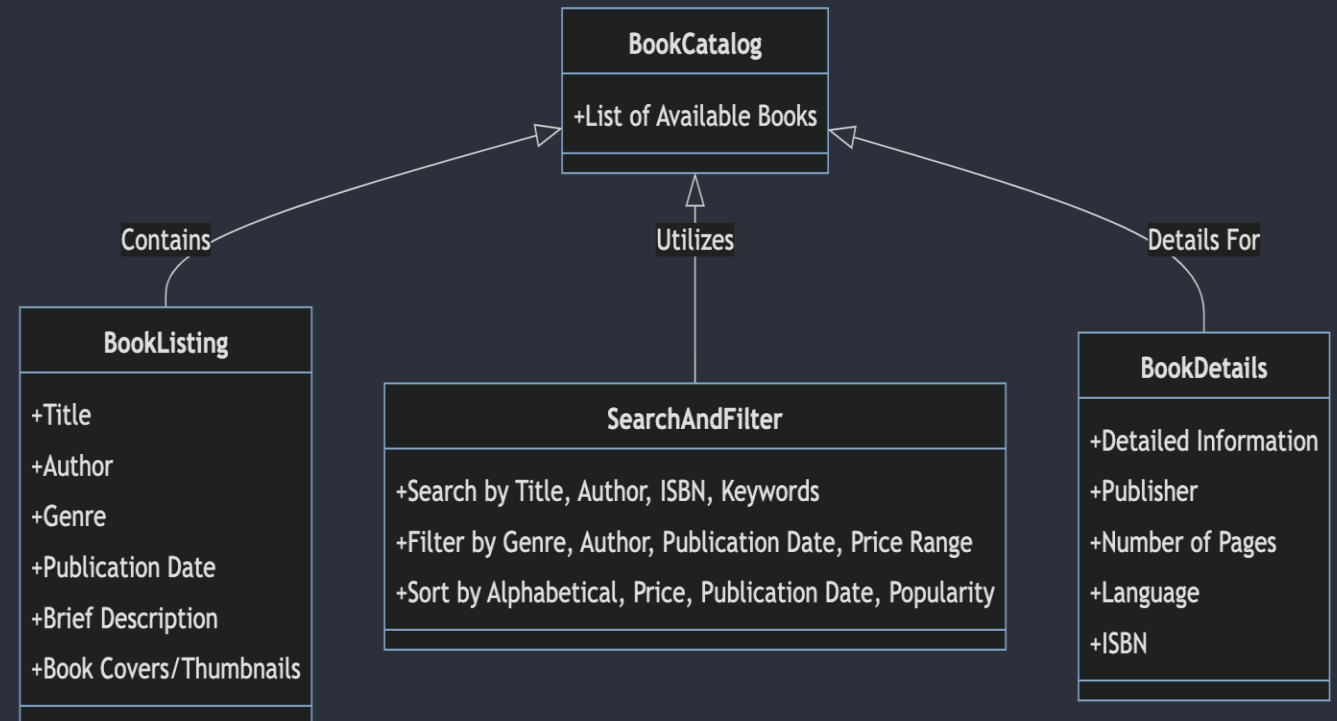
Pan & Zoom



🖥️ FULL SCREEN

💾 SAVE TO MERMAID CHART

```
1 classDiagram
2   class BookCatalog {
3     +List of Available Books
4   }
5   class BookListing {
6     +Title
7     +Author
8     +Genre
9     +Publication Date
10    +Brief Description
11    +Book Covers/Thumbnails
12  }
13  class SearchAndFilter {
14    +Search by Title, Author, ISBN, Keywords
15    +Filter by Genre, Author, Publication Date, Price Range
16    +Sort by Alphabetical, Price, Publication Date, Popularity
17  }
18  class BookDetails {
19    +Detailed Information
20    +Publisher
21    +Number of Pages
22    +Language
23    +ISBN
24  }
25
26  BookCatalog <|-- BookListing : Contains
27  BookCatalog <|-- SearchAndFilter : Utilizes
28  BookCatalog <|-- BookDetails : Details For
```



Planning – Copilot as a Scrum Master

CHAT

 **GitHub Copilot**

Hi @visrani, how can I help you?

/fix the problems in my code

/tests add unit tests for my code

/explain how the selected code works

Ask Copilot or type / for commands

Bookstore_Ideation.txt U x Technologies.txt U

docs > requirements > Bookstore_Ideation.txt

```
1 Here are the high-level requirements for implementing a book store website:
2
3 ### 1. User Management
4 - **User Registration and Login:**
5   - Users should be able to create an account and log in using their credentials.
6   - Implement password recovery and account verification mechanisms.
7
8 ### 2. Book Catalog
9 - **Book Listing:**
10  - Display a comprehensive list of available books.
11  - Each book entry should include the title, author, genre, publication date, and a br
12  - Show book covers/thumbnails.
13
14 ### 3. Search and Filter
15 - **Search Functionality:**
16   - Allow users to search for books by title, author, ISBN, or keywords.
17 - **Filter Options:**
18   - Enable filtering by genre, author, publication date, and price range.
19 - **Sort Options:**
20   - Provide sorting options such as alphabetical order, price (low to high, high to low
21
22 ### 4. Book Details
23 - **Detailed Book Page:**
24   - Display detailed information about the book including title, author, detailed descr
25   - Show additional information like publisher, number of pages, language, and ISBN.
26
27 ### 5. Ratings and Reviews
28 - **User Ratings:**
29   - Allow users to rate books on a scale (e.g., 1 to 5 stars).
30 - **User Reviews:**
31   - Enable users to write and read reviews for each book.
32   - Display the average rating and the total number of reviews.
33
34 ### 6. Pricing and Purchase Options
35 - **Pricing Information:**
36   - Clearly display the price of each book.
37   - Provide information on discounts and special offers.
38 - **Add to Cart:**
39   - Users should be able to add books to a shopping cart for purchase.
```

Coding – File Structures

CHAT

 visrani

@workspace /new Design a user registration and management system for a bookstore website considering primary technology as Java and use Postgres to store the user information

 **Workspace**

Sure, here's a proposed directory structure for a Java-based user registration and management system for a bookstore website:

src/main

java

com/bookstore

controller

UserController.java

model

User.java

repository

UserRepository.java

service

UserService.java

BookstoreApplication.java

resources

webapp

pom.xml

README.md

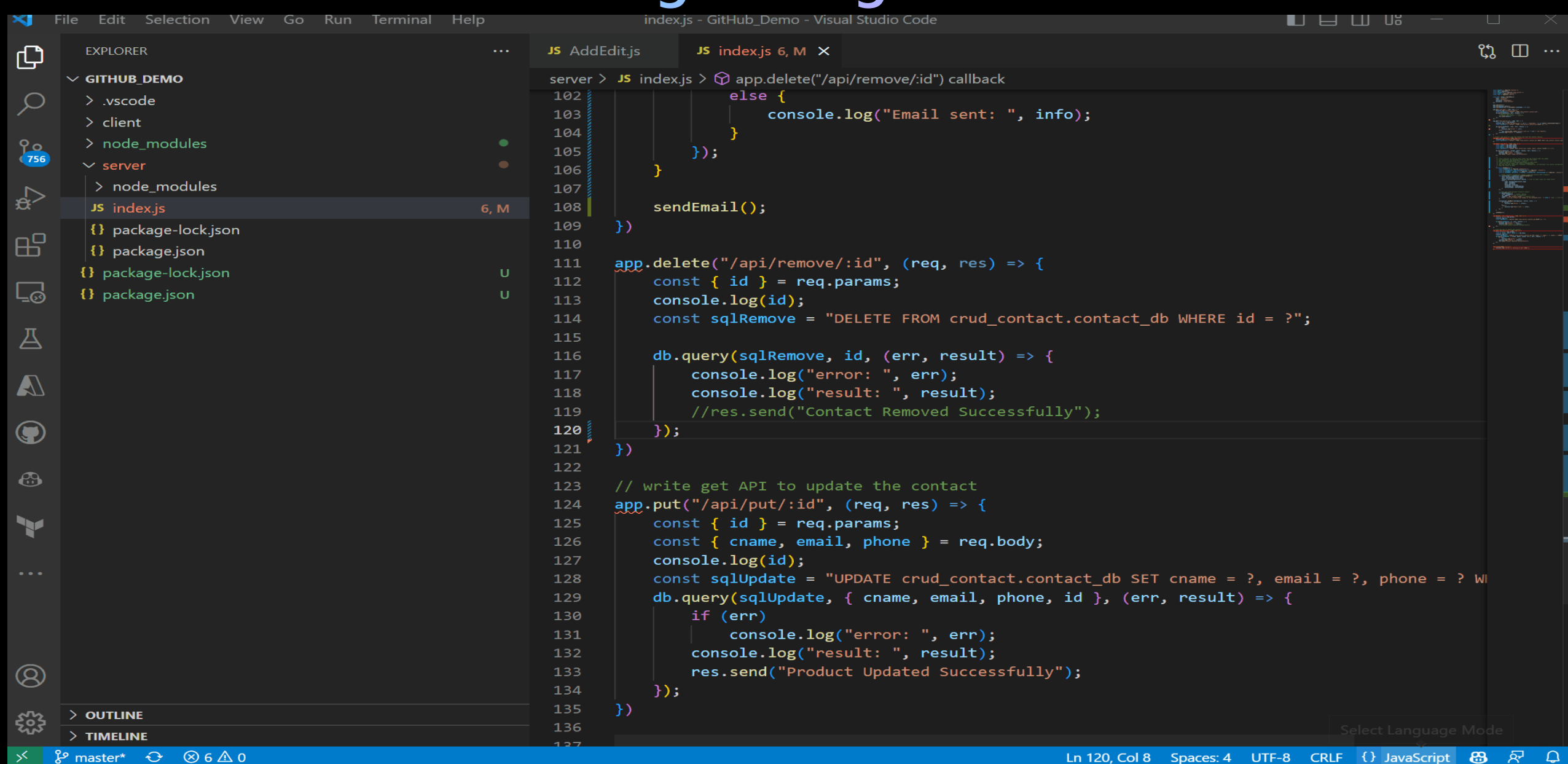
Bookstore_Ideation.txt U

UserService.java

bookstore-user-management > src > main > java > com > bookstore > service > J UserService.java

```
1 package com.bookstore.service;
2
3 import com.bookstore.model.User;
4 import com.bookstore.repository.UserRepository;
5
6 public class UserService {
7     private UserRepository userRepository;
8
9     public UserService(UserRepository userRepository) {
10         this.userRepository = userRepository;
11     }
12
13     public User registerUser(String username, String email, String password) {
14         // Validate user input
15         if (username == null || username.isEmpty() || email == null || email.isEmpty() ||
16             throw new IllegalArgumentException("Invalid user input");
17     }
18
19     // Check if the username or email already exists
20     if (userRepository.findByUsername(username) != null) {
21         throw new IllegalArgumentException("Username already exists");
22     }
23
24     if (userRepository.findByEmail(email) != null) {
25         throw new IllegalArgumentException("Email already exists");
26     }
27
28     // Create a new user object
29     User user = new User(username, email, password);
30
31     // Save the user to the database
32     userRepository.save(user);
```

Coding – Bring it on !!



Coding – Security and Bugfixes

**GitHub Copilot**
Hi @bikrade, how can I help you?

I'm powered by AI, so surprises and mistakes are possible. Make sure to verify any generated code or suggestions, and [share feedback](#) so that we can learn and improve.

Ask Copilot a question or type '/' for topics

JS index.js

```
server > JS index.js > ...  
31 // The response object is a JSON object.  
32  
33 app.get("/api/getid/:id", (req, res) => {  
34   const { id } = req.params;  
35   console.log("Inside api/getid/:id" + id + ", Timestamp: " + new Date().toLocaleStr  
36   const sqlSelect = "SELECT * FROM crud_contact.contact_db WHERE id = ?";  
37   db.query(sqlSelect, [id], (err, result) => {  
38     if (err)  
39       console.log("error: ", err);  
40     else if (result.length > 0)  
41       res.send(result);  
42     else  
43       res.status(404).send("Contact with id " + id + " not found");  
44     console.log("result: ", result);  
45   });  
46 });  
47  
48 // write a SQL query to join two tables and find the contact details  
49 app.get("/api/getjoin", (req, res) => {  
50   console.log("Inside api/getjoin");  
51   const sqlSelect = "SELECT * FROM crud_contact.contact_db INNER JOIN crud_contact.c  
52 }  
53  
54 app.post("/api/post", (req, res) => {  
55   const cname = req.body.cname;  
56   const email = req.body.email;  
57   const phone = req.body.phone;  
58   const sqlInsert = "INSERT INTO contact_db (cname, email, phone) VALUES (?, ?, ?)";  
59  
60   db.query(sqlInsert, [cname, email, phone], (err, result) => {  
61     console.log("error: ", err);  
62     console.log("result: ", result);  
63     res.send("Product Added Successfully");  
64   });  
65  
66 // write function to send an email about the new product that was added  
67 const sendEmail = () => {  
68   const nodemailer = require('nodemailer');  
69   const transporter = nodemailer.createTransport({
```

Ln 48, Col 39 Spaces: 4 UTF-8 CRLF {} JavaScript

Coding – Explain Existing Code

Files

main

Go to file

> .demo

> .devcontainer

> .github

> .vscode

> ansible

> bank

> bicep

> docs

> patches

> src

> csv

> main

> java/com/github/demo

> model

> service

BookDatabase.java

gifted-theory / src / main / java / com / github / demo / service / BookDatabaseImpl.java

Code Blame 252 lines (213 loc) · 7.92 KB

Raw Copy Download Edit

```
20 public class BookDatabaseImpl implements BookDatabase {
37     private static final int getMaxRetries() {
45
46     public BookDatabaseImpl() throws BookServiceException {
47         this(null, null, null);
48     }
49
50     public BookDatabaseImpl(String url, String username, String password) throws BookServiceException {
51         Properties props = new Properties();
52
53         if (username != null) {
54             props.setProperty("user", username);
55         }
56         if (password != null) {
57             props.setProperty("password", password);
58         }
59         // This is a postgres specific setting, but SQLite tolerates it
60         props.setProperty("ssl", "false");
61
62         // Default to a sqlite in memory database if no database url has been provided
63         if (url == null) {
64             url = "jdbc:sqlite::memory:";
65         }
66
67         connection = getConnection(url, props);
68
69         // Populate our in-memory database with data, if this is what we are
70         if (connection != null && url.indexOf(":memory:") > -1) {
71             initializeAndPopulateDatabase();
72         }
```

Reviews – Generate Summaries

 gh/queue/master/pr-...

 github/pull_requests

Feedback

Switch to GitHub:lab

Rails 7.2.0.alpha.94faed4dd4 | Ruby 3.3.1 | React 18.2.0

494ms  14  1.95KB





 octodemo / didactic-art













<> Code

Issues 1

Pull requests 6

Actions

Security 52

Insights

Settings

Implementing book rating feature #6

Edit<> Code

Open visrani wants to merge 1 commit into main from feature-book-star-rating

Conversation 2Commits 1Checks 18Files changed 4

 visrani commented on Mar 13 • edited

Some changes



 Implementing book rating feature

05a9409

 github-actions bot commented on Mar 13

 Code Coverage

Code Coverage 39.8%

Reviewers

Suggestions

 octodemobotRequest

At least 1 approving review is required to merge this pull request.

Still in progress? Convert to draft

Assignees

No one—assign yourself

Labels

None yet



Reviews – Help Review Code

 octodemo / didactic-art 

Q Type  to search

 Code  Issues 1  Pull requests 6  Actions  Security 52  Insights  Settings

Implementing book rating feature #6

[Edit](#) [Code](#)[Open](#) visrani wants to merge 1 commit into `main` from `feature-book-star-rating` [Conversation](#) 2 [Commits](#) 1 [Checks](#) 18 [Files changed](#) 4+42 -9 [Changes from all commits](#) [File filter](#) [Conversations](#) 

0 / 4 files viewed

[Ask Copilot](#)[Review in codespace](#)[Review changes](#)

- src/main
- java/com/github/demo
- model
 - Book.java
- service
 - BookDatabaseImpl.java
 - BookUtils.java
- webapp/static
 - books.html

 25  src/main/java/com/github/demo/model/Book.java ☐ Viewed   @@ -11,21 +11,36 @@ public class Book {

```
11
12     private String cover;
13
14 - public Book() {
15
```

```
16     }
17
18     public Book(String author, String title) {
```

```
11
12     private String cover;
13
14 + private int rating;
```

```
16 + public Book() {
17 +     this.title = "";
18 +     this.author = "";
19 +     this.cover = "";
20 +     this.rating = 0;
```

```
21     }
22
23     public Book(String author, String title) {
```

Testing – Test Cases and Code

CHAT

GitHub Copilot

Hi @visrani, how can I help you?

/fix the problems in my code

/tests add unit tests for my code

/explain how the selected code works

Ask Copilot or type / for commands

BookServlet.java

src > main > java > com > github > demo > servlet > BookServlet.java > BookServlet

```
23 public class BookServlet extends HttpServlet {
41     @Override
42     public void doPost(HttpServletRequest req, HttpServletResponse resp)
43         throws ServletException, IOException {
44         doGet(req, resp);
45     }
46
47     @Override
48     protected void doGet(HttpServletRequest req, HttpServletResponse resp)
49         throws ServletException, IOException {
50
51         Properties versionProperties = new Properties();
52         versionProperties.load(getClass().getResourceAsStream(name: "/version.properties"))
53
54         ServletContextTemplateResolver resolver = new ServletContextTemplateResolver(req.
55             resolver.setPrefix("/");
56             resolver.setSuffix(".html");
57
58         TemplateEngine engine = new TemplateEngine();
59         engine.setTemplateResolver(resolver);
60
61         WebContext ctx = new WebContext(req, resp, getServletContext(), req.getLocale());
62         ctx.setVariable("modified", Calendar.getInstance());
63         ctx.setVariable("version", versionProperties.getProperty(key: "version"));
64
65         resp.setContentType(type: "text/html; charset=UTF-8");
66
67         try {
68             List<Book> books = bookService.getBooks();
69             ctx.setVariable("books", books);
70             engine.process("books", ctx, resp.getWriter());
71         } catch (BookServiceException e) {
72             ctx.setVariable("error", e.getMessage());
73             engine.process("error", ctx, resp.getWriter());
74         }
75     }
76 }
77
```

Deployment – Anytime, Anywhere

CHAT

GitHub Copilot

Hi @visrani, how can I help you?

/fix the problems in my code

/tests add unit tests for my code

/explain how the selected code works

Ask Copilot or type / for commands

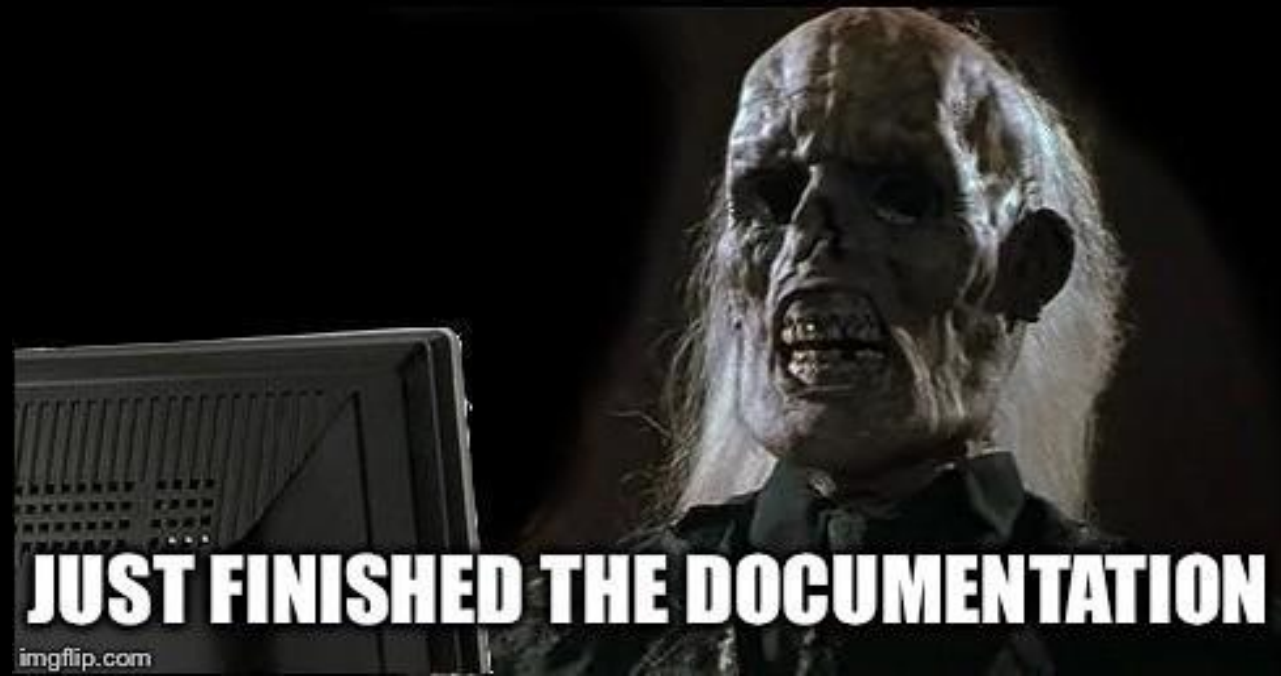
pom.xml M x

pom.xml

```
1 <project xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns="http://maven.apache.org/POM/4.0.0" xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd">
2   <modelVersion>4.0.0</modelVersion>
3
4   <groupId>com.github.octodemo</groupId>
5   <artifactId>bookstore-security</artifactId>
6   <version>${revision}${changelist}${sha1}</version>
7   <packaging>jar</packaging>
8
9   <description>A Java example project to demonstrate a Java development stack with Maven</description>
10
11   <properties>
12     <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
13     <project.reporting.outputEncoding>UTF-8</project.reporting.outputEncoding>
14
15     <java.version>11</java.version>
16
17     <!--
18      | Dependency Versions
19      -->
20     <jetty.version>10.0.0</jetty.version>
21     <log4j.version>2.17.2</log4j.version>
22
23     <!--
24      | Properties used to create a CD style version number for the Maven build
25      -->
26     <revision>1.0.0</revision>
27     <changelist></changelist>
28     <sha1>-SNAPSHOT</sha1>
29
30     <!-- GitHub Repository name -->
31     <github.repository>octodemo-resources/template-bookstore-security</github.repository>
32   </properties>
33
34   <dependencyManagement>
35     <dependencies>
36       <dependency>
37         <groupId>org.eclipse.jetty</groupId>
38         <artifactId>jetty-bom</artifactId>
39         <version>${jetty.version}</version>
```

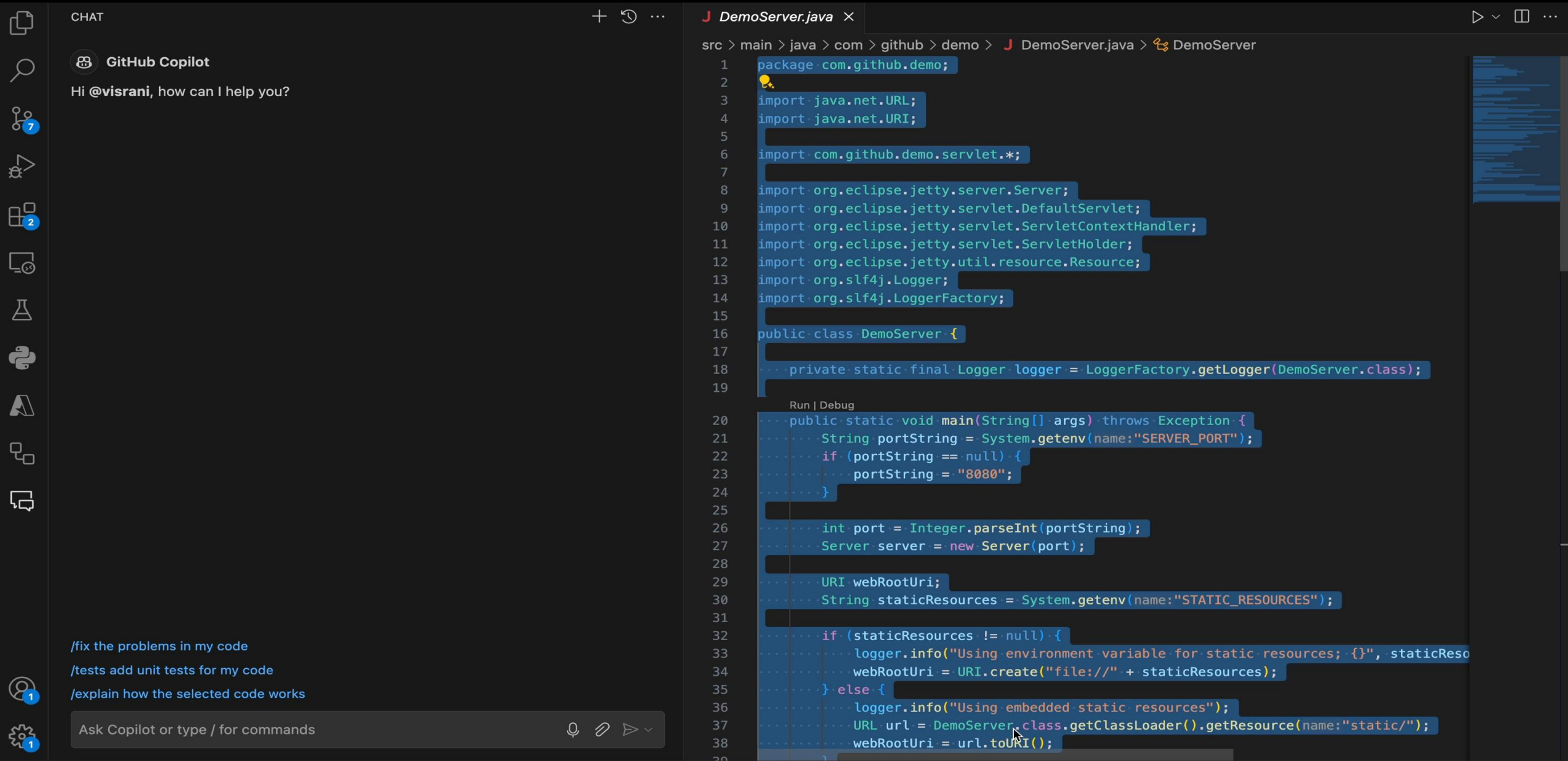


JUST FINISHED A PROJECT



JUST FINISHED THE DOCUMENTATION

Documentation – Any Help is Appreciated



The image shows a screenshot of an IDE interface. On the left, there is a sidebar with icons for various tools and a chat window. The chat window is titled "CHAT" and shows a conversation with "GitHub Copilot". The chat history includes the message "Hi @visrani, how can I help you?". Below the chat history, there are three suggested prompts: "/fix the problems in my code", "/tests add unit tests for my code", and "/explain how the selected code works". At the bottom of the chat window, there is a text input field with the placeholder "Ask Copilot or type / for commands".

The main area of the IDE displays a Java file named "DemoServer.java". The code is as follows:

```
src > main > java > com > github > demo > DemoServer.java > DemoServer
1 package com.github.demo;
2
3 import java.net.URL;
4 import java.net.URI;
5
6 import com.github.demo.servlet.*;
7
8 import org.eclipse.jetty.server.Server;
9 import org.eclipse.jetty.servlet.DefaultServlet;
10 import org.eclipse.jetty.servlet.ServletContextHandler;
11 import org.eclipse.jetty.servlet.ServletHolder;
12 import org.eclipse.jetty.util.resource.Resource;
13 import org.slf4j.Logger;
14 import org.slf4j.LoggerFactory;
15
16 public class DemoServer {
17
18     private static final Logger logger = LoggerFactory.getLogger(DemoServer.class);
19
20     public static void main(String[] args) throws Exception {
21         String portString = System.getenv(name:"SERVER_PORT");
22         if (portString == null) {
23             portString = "8080";
24         }
25
26         int port = Integer.parseInt(portString);
27         Server server = new Server(port);
28
29         URI webRootUri;
30         String staticResources = System.getenv(name:"STATIC_RESOURCES");
31
32         if (staticResources != null) {
33             logger.info("Using environment variable for static resources; {}", staticResources);
34             webRootUri = URI.create("file://" + staticResources);
35         } else {
36             logger.info("Using embedded static resources");
37             URL url = DemoServer.class.getClassLoader().getResource(name:"static/");
38             webRootUri = url.toURI();
39         }
40     }
41 }
```



Make higher-quality
software



Produce software
more efficiently



Make smarter
planning decisions



Improve
the well-being of
developers



Facilitate team
collaboration and
success