

Demo : Working with GitHub Copilot Edits and GitHub Copilot chat

GitHub Copilot Edits suggests code completions and edits directly within your editor. Works in the background as you type, providing real-time suggestions for completing lines of code, fixing bugs, or improving code snippets. Its focus is on automated code-writing and editing assistance to make your work faster and more efficient.

GitHub Copilot Chat is a conversational tool that allows you to interact with Copilot via a chat interface. You can ask questions, troubleshoot issues, or request guidance in a more interactive way. It is ideal for debugging, learning new concepts, or understanding unfamiliar code as it lets you have a dialogue with Copilot for deeper context.

Prerequisites


- VS Code
- GitHub Copilot extension for VS Code
- GitHub Copilot subscription
- NodeJS 18 or later
- Book API project

Using Copilot Chat

1. Open the `book-api` project in VS Code.
2. Use the Copilot Chat to review and validate the `bookModel.js` file. Open the Copilot Chat and write the following prompt


```
Does the code in #file:bookModel.js follows best practices?
```

3. The Copilot Chat responds as follows:

 sonusathyadas

Does the code in `#file:bookModel.js` follows best practices?

`JS bookModel.js` `JS bookModel.js:1-59`

 GitHub Copilot

> Used 1 reference

The code in `JS bookModel.js` follows many best practices, but there are a few areas where improvements can be made:

1. **Error Handling:** The current error handling is minimal. It would be better to handle errors more explicitly and provide meaningful messages.
2. **SQL Injection Prevention:** The code uses parameterized queries, which is good for preventing SQL injection.
3. **Code Duplication:** The SQL queries are repeated in multiple methods. Consider using a helper function to reduce duplication.
4. **Asynchronous Operations:** The code uses callbacks for asynchronous operations. Using Promises or `async/await` can make the code cleaner and easier to read.

Here is an improved version of the code:

```
const sqlite3 = require('sqlite3').verbose();
const db = require('../database');

class Book {
  constructor(id, title, author, publishedYear, category, la
    this.id = id;
    this.title = title;
    this.author = author;
```

What specific improvements should I prioritize?

Ask Copilot

Demo_14-Using_GHCP_Edits_and_GHCP_chat.md Current file


@ 📎 🗣️

GPT-4o ▾ ▶ ▾

4. Try the following prompt to add JWT authentication for the web application.


Generate code to enable JWT authentication for the routes defined in `#file:bookRoutes.js`

5. This will generate the code and steps required to add JWT authentication for the `book-api` application. You need to execute the commands and copy the codes to appropriate files manually.

 sonusathyadas

Generate code to enable JWT authentication for the routes defined in `#file:bookRoutes.js`

`JS` bookRoutes.js `JS` bookController.js:1-91

 GitHub Copilot

> Used 2 references

To enable JWT authentication for the routes defined in `JS bookRoutes.js`, you need to create a middleware function that verifies the JWT token and then apply this middleware to the routes.

First, install the necessary packages:

```
npm install jsonwebtoken
```

Next, create a middleware function for JWT authentication:



```
const jwt = require('jsonwebtoken');



const authMiddleware = (req, res, next) => {
  const token = req.header('Authorization')?.split(' ')[1];
  if (!token) {
    return res.status(401).json({ message: 'Access denied.' });
  }


  try {
    const decoded = jwt.verify(token, process.env.JWT_SECRET);
    req.user = decoded;
    next();
  } catch (error) {
    return res.status(401).json({ message: 'Access denied.' });
  }
}
```

How do I test this setup?

Ask Copilot

 Demo_14-Using_GHCP_Edits_and_GHCP_chat.md Current file 

@  

GPT-4o 

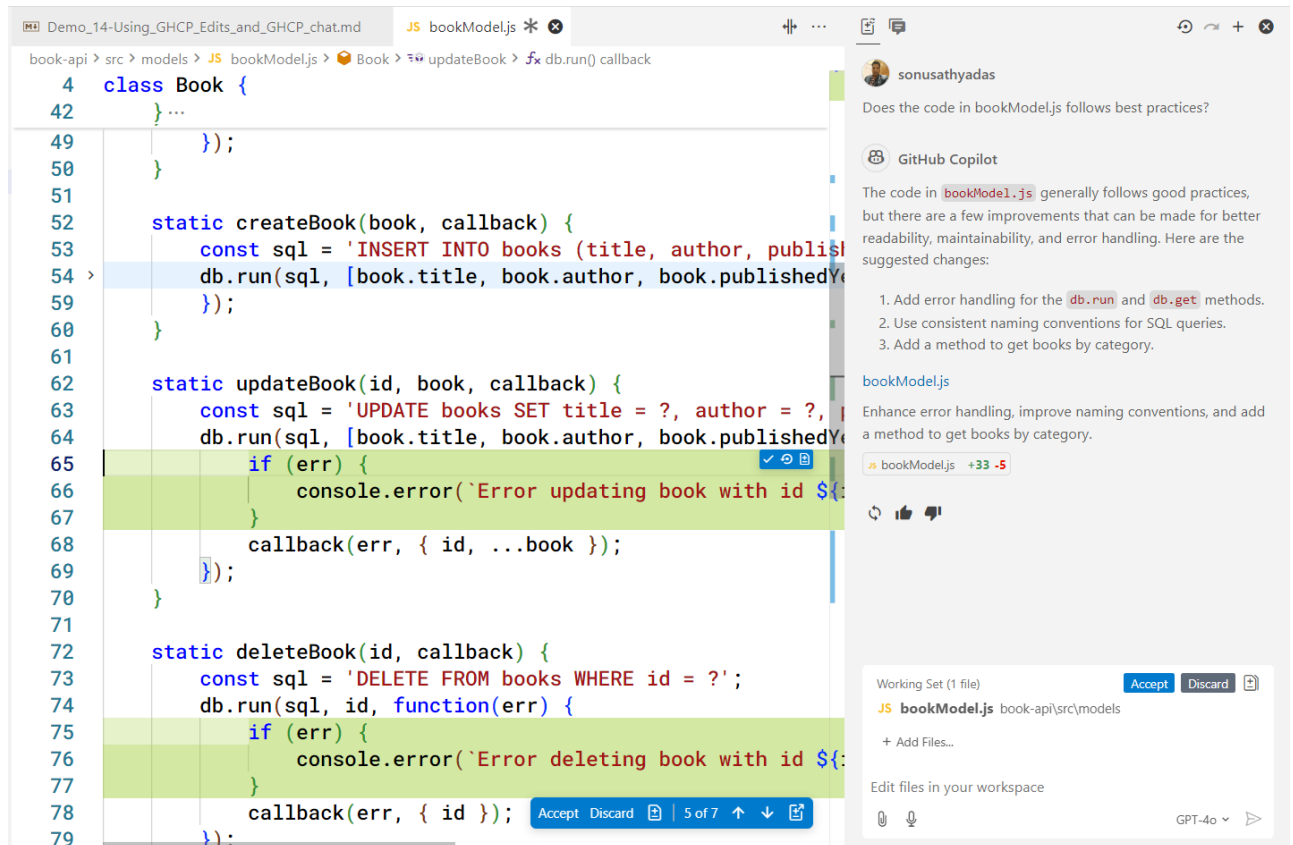
DO NOT update the code in the editor. Discard the suggestions in the chat. We will be updating the code using the Copilot Edits in the next Task

Using Copilot Edits

1. Open the `book-api` project in VS Code.
2. Use the Copilot Edits to review and validate the `bookModel.js` file. Open the Copilot Edits and write the following prompt. Ensure the `bookModel.js` is opened in the editor and it is added in the working set of GitHub Copilot Edits.

Does the code in `bookModel.js` follows best practices?

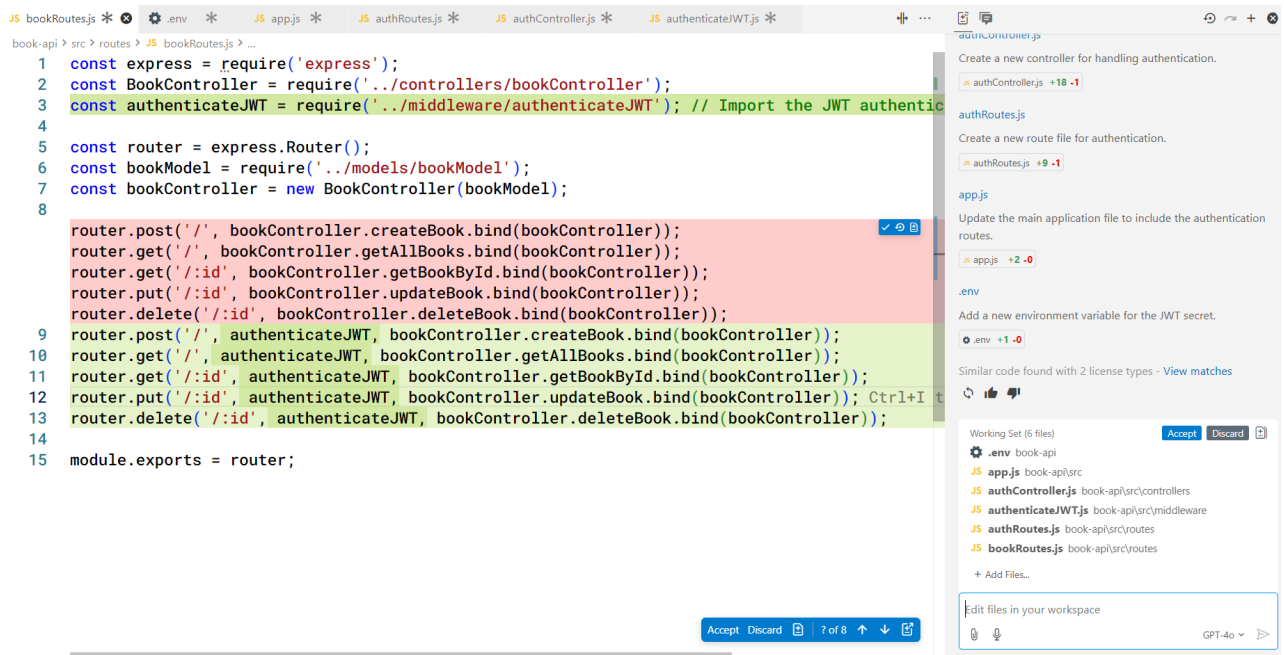
3. The Copilot Edit directly updates the `bookModel.js` and wait for you to accept the changes.



4. Click on the Accept button to accept the changes made on the file.
5. Try the following prompt to add JWT authentication for the web application in the Copilot Edits. Add the `bookRoutes.js`, `bookModel.js`, `bookController.js` files in the working set. Alternatively you can select the entire code base by clicking in the `@` icon in the Copilot Edits.

Generate code to enable JWT authentication for the routes defined in `bookRoutes.js`

6. This will update the exits files and create a new set of files that are required to implement the JWT authentication.



6. Review the suggestions in each file and accept the suggestions in all files.