GitHub Copilot Video full

22 August 2024 11:29

Agenda -

1. GitHub Copilot Overview
2. Unit Testing & Test driven development (TDD) (Red, Green, Refactor)
3. Code Refactoring
4. Code Reviews
5. Performance optimization
6. Design pattern/ frameworks implementation
7. Documentation
8. Insurance Domain driven code
9. Tips, Tricks, Best Practise

1. **GitHub Copilot Overview**
   - **GitHub Copilot features**
     - You can use Copilot to:
       - Get code suggestions as you type in your IDE
       - Chat with Copilot to ask for help with your code
       - Ask Copilot for help using the command line
       - Generate a description of the changes in a pull request (Copilot Enterprise only)
       - Create and manage collections of documentation, called knowledge bases, to use as a context for chatting with Copilot (Copilo t Enterprise only)
   - **GitHub Copilot availability**:
     - In your IDE like VS Code, Visual Studio, IntelliJ
     - In GitHub Mobile, as a chat interface
     - On the command line, through the GitHub CLI
     - On GitHub.com, with a subscription to Copilot Enterprise
   - **Getting access to Copilot**
     - GitHub Copilot Individual.
     - GitHub enterprise
     - Student, teacher, or open source project contributor
   - **How to use GitHub Copilot in your editor**
     - Sign up for GitHub Copilot
     - Install latest version of editor of your choice (VS Code, Visual Studio, IntelliJ..)
     - Install GitHub Copilot extension
     - Sign in to GitHub in Editor e.g. Visual Studio Code
     - Use GitHub Copilot chat to get started with prompts
   - **Prompt engineering considerations for GitHub Copilot**
     - Start general, then get specific
     - Give examples
     - Break complex tasks into simpler tasks
     - Indicate relevant code
     - Experiment and iterate
     - Keep history relevant
     - Follow good coding practices (consistent code style and patterns, modular, scoped components, Include unit tests)
   - **Setup relevant context**
     - #editor
     - #file
     - #selection
     - @workspace agent

---

2. **Unit Testing & Test driven development (TDD) (Red, Green, Refactor)**
   - **TDD**
     - **Create TDD Unit test cases for Policy class -**
       Generate unit test case using VSTest for testing following conditions for all fields of Policy class listed here - Email should be valid expression, Account Number is 8 digit number & cannot be empty, Policy Number is 8 digit number, First Name is 50 character n on-empty string, Last Name is 50 character non-empty string, Postal Code is 7 digit number, Phone is valid phone number format, Producer Code is 5 character not empty string, Group Code is 5 character not empty string, Master Code is 5 character not empty string, City i s 20 character non-empty string, State is 20 character non-empty string, Effective Date is valid date format less than or equal to current date, Expiration Date is valid date format greater than or equal to current date, Annual Premium decimal type with 2 decimal places non zero v alue
     - **Refactor code by adding Policy validator class**
       Refactor PolicyTest class to move policy validation code for all fields into new PolicyValidator class
   - **Unit Testing**
     - **Generating test data (Policy.cs)**
       #editor Generate set of 10 sample records for AutoPolicy class
     - **XUnit 2.0 Create Positive & Negative test cases** (InsurancePolicyController.cs)

#editor Generate positive & negative unit test cases using XUnit
- **Generate Mock data using Autofaker** (InsurancePolicyController.cs)
  #editor Generate mock data using Autofaker
- **Fluent Assertions** (Unit test case assertions) (InsurancePolicyController.cs)
  #editor write unit test cases using Fluent Assertions library

---

3. **Code Refactoring -**
   - Create PolicyController to manage Policies in database
     #selection Create InsurancePolicyController web api with policy class database implementation for all create, update, delete & get operations using CQRS pattern and SQL Server queries
   - **Defect fixing**
     - **Update code for defect fix - Invalid HomeInsurancePolicy object parameter in Put method. (**InsurancePolicyController.cs)
       #editor Identify CreatePolicy method code to check invalid or null HomeInsurancePolicy object & recommend code fix
   - **Third party library integration**
     - **Generate Mapping using Automapper(**InsurancePolicyController.cs)
       #editor Generate mappings using Automapper
     - **Dependency injection - Autofac library (**InsurancePolicyController.cs)
       #editor Refactor code to implement dependency injection using Autofac library
     - **Dapr (Integration with external services)**
       Generate C# .NET code to integrate with external Paypal Payment Gateway using Dapr framework with all dependencies & sample c all
       Generate C# .NET code to invoke GoogleMap service with all dependencies & sample call
   - **Authentication & Authorization**
     - **Token based Authentication** (InsurancePolicyController.cs)
       #file:InsurancePolicyController.cs Refactor InsurancePolicyController class to web api implementing token based authentication
     - **Authorization - Add/Update existing ACL mapping** (InsurancePolicyController.cs)
       #editor implement Authorization for Admin, Reader & Writer roles using ACL mapping
   - **Fix SonarQube issues**
     - **SonarQube errors - Refactor code to fix cyclomatic complexity issue (**LongMethod.cs)
       #editor Refactor CalculateTotalPremium method to resolve cyclomatic complexity issue
   - **.NET constructs/ features**
     - **Refactor code to use right .NET constructs/ features (**LongMethod.cs)
       #editor Refactor CalculateTotalPremium method to use right .NET constructs & features to follow standardization & performance optimization
   - **Cloud native implementation**
     - **Rector to support decoupling - enterprise message broker like - azure service bus** (InsurancePolicyController.cs)
       #editor Refactor code to implement azure service bus along with message consumers
     - **Rector to support Cloud migration** (InsurancePolicyController.cs)
       Refactor #selection to cloud native implementation using Azure Function & SQL Server
   - **Data model updates**
     - **Update data model to add new field**
       @workspace refactor code to add RenewalDate field to Policy class & interfaces along with consumer code
   - **Logging**
     - **Update logging to analyse & log missing request/response/error details (**InsurancePolicyController.cs)
       #editor Update logging to analyse & log missing request, response & error details
   - **Restructuring code**
     - **Interface extraction (**Policy.cs)
       #selection Refactor Policy class to extract interface

---

4. **Code Reviews -**
   - **Code Reviews**
     - **Naming convention & coding standards** (LongMethod.cs)
       - #editor Identify coding standards issue
       - @workspace Check if #file:LongMethod.cs follows naming standards defined in naming.json
     - **Speed & performance (Performance.cs)**
       - identify performance issues in #selection
       - Check previous response for potential concurrency issues or thread contention that could impact performance
       - Multiple API calls & caching (CustomerService.cs)
         #file:CustomerService.cs find duplicate method calls & recommend refactored code
     - **Security (DrivingHistory.cs)**
       - Scan #file:DrivingHistory.cs for security issues & recommend resolutions
       - #selection Identify PII issues & recommend resolution for all such PII issue instances

---

5. **Performance optimization -**
   - **Optimize long running code- Task Parallelism (**Performance.cs)
     #file:Performance.cs optimize code for performance issues
   - **Implement Logger class to monitor method level performance (**Performance.cs)

#editor Refactor code to add logging using Logger class to monitor method level performance along with method level execution time

- **Reduce repetitive/ unwanted method calls (**Performance.cs)
  #editor Refactor code to reduce repetitive or unwanted method calls
- **Optimize internal/external service calls** (Location.cs)
  #editor Refactor GoogleMapsService code for performance optimization & caching responses using redis cache
- **Refactor code to implement pagination for big result sets (**InsurancePolicyController.cs)
  #editor Refactor GetAllPolicies() operation to optimize long resultset to implement pagination to return 100 records

---

6. **Design pattern/ frameworks implementation**
   - **Identifying anti-patterns in C# code (LongMethod.cs)**
     #editor Identify anti-patterns in Antipattern class
   - **Identifying code smells (**LongMethod.cs)
     #editor Identify codesmell issues in #file:LongMethod.cs
   - **Circuit breaker poly/retry mechanism (**DrivingHistory.cs)
     #editor Reafactor DrivingHistoryController class to implement Circuit Breaker pattern & error handling
   - **SOLID principals (**BadCode.cs)
     #selection refactor code to follow SOLID principals

7. **Documentation**
   - /doc - generate documentation
   - /explain - explains code in English

---

8. **Insurance Domain driven code-**
   - Generate PersonalAuto policy with FNOL details classes in detail

---

9. **Tips, Tricks, Best Practise**
   - @VSCode, @workspace agents for contextualized response
   - Editor features - Ghost text, yellow sparkles, chat, accept partial complete recommendations, inline chat (Ctrl+I),
   - Using # commands
     - #editor
     - #file
     - #selection
     - #terminalLastCommand
     - #terminalSelection
   - Using / commands
     - /doc - generate documentation
     - /test - create test cases
     - /fix - fix errors
     - /explain - explains code in English
     - /help - Get help on using Copilot Chat
     - /clear - Clear current conversation
     - /generate - e.g. /generate code that validates a phone number
     - /optimize - e.g. /optimize fetchPrediction method
     - /simplify - Simplify the selected code
   - @terminal - VS code terminal command help
   - Add details with # (e.g. #selection refactor code to add default values to parameters)
   - Refining code in place with Alt + /
   - GitGub command CLI (for Terminal commands or to explain commands not known)
   - @workspace agent

GitHub Copilot shortcuts -

- Accept suggestion: TAB key
- Reject suggestion: Esc key
- Show combined suggestions: Ctrl + Enter key
- Accept at word level: Ctrl + Right Arrow key
- See next suggestion: Alt +] key
- See previous suggestion: Alt + [key
- Trigger inline suggestion: Alt + / key
- Show Labs feature list: Ctrl + Shift + Alt + e key
- Toggle sidebar: Ctrl + Shift+a key