DUMPSARENA

GitHub CopilotCertification Exam

GitHub GitHub-Copilot

Version Demo

Total Demo Questions: 8

Total Premium Questions: 81

Buy Premium PDF

https://dumpsarena.co

sales@dumpsarena.co

sales@dumpsarena.co
dumpsarena.co

Topic Break Down

Topic	No. of Questions
Topic 1, Responsible AI	20
Topic 2, GitHub Copilot plans and features	21
Topic 3, How GitHub Copilot works and handles data	5
Topic 4, Prompt crafting and prompt engineering	9
Topic 5, Developer use cases for AI	7
Topic 6, Testing with GitHub Copilot	7
Topic 7, Privacy fundamentals and context exclusions	11
Topic 8, Mix Questions	1
Total	81

QUESTION NO: 1

How long does it take content exclusion to add or be updated?

A. Up to 30 minutes

B. 45 - 60 minutes

C. 60 - 90 minutes

D. 24 hours

ANSWER: A

Explanation:

Up to 30 minutes is correct. GitHub Copilot content exclusion settings are not necessarily applied instantly across all Copilot services after an administrator creates, changes, or removes an exclusion. GitHub documents that it can take up to 30 minutes for a content exclusion to be added or updated. This propagation period allows the revised repository, file-path, or other configured exclusion rules to be distributed and enforced by the Copilot infrastructure. During that window, users may temporarily continue to receive suggestions that reflect the prior configuration, so organizations handling sensitive code should account for the delay when planning policy changes or access-control procedures. Content exclusions are intended to prevent Copilot from using matching content as context for code-completion suggestions, helping organizations control which material is available to Copilot. Administrators should verify that their content-exclusion configuration is correctly scoped and allow the documented propagation interval before testing whether the policy is taking effect. For the current behavior and administrative setup details, see GitHub's content exclusions documentation and the GitHub Copilot documentation.

QUESTION NO: 2

What is a key consideration when relying on GitHub Copilot Chat's explanations of code functionality and proposed improvements?

A. The explanations are dynamically updated based on user feedback.

B. Reviewing and validating the generated output for accuracy and completeness.

C. GitHub Copilot Chat uses a static database for generating explanations.

D. The explanations are primarily derived from user-provided documentation.

ANSWER: B

Explanation:

Reviewing and validating the generated output for accuracy and completeness is essential when using GitHub Copilot Chat. Copilot can explain code, identify potential issues, and propose improvements, but its responses are generated by an AI model and can contain mistakes, omissions, insecure patterns, or recommendations that do not fit the project's requirements and context. Developers remain responsible for understanding the code they accept and for applying normal engineering controls, including code review, testing, security analysis, and verification against applicable specifications and documentation. This is particularly important when a suggestion affects authentication, authorization, data handling, dependencies, or production behavior. GitHub explicitly advises users to review and validate Copilot's generated suggestions before accepting them, rather than treating output as authoritative. Copilot Chat is most valuable as an assistive tool that accelerates investigation and drafting while human expertise provides the final assessment of correctness, quality, and suitability. See GitHub's guidance on responsible use of GitHub Copilot Chat and its code review guidance.

QUESTION NO: 3

A pull request author uses GitHub Copilot to generate a pull request summary before requesting review. Which actions should the author take before relying on the generated summary? (Select two.)

A. Compare the generated summary with the actual changeset.

B. Edit the summary to correct omissions and add relevant context.

C. Automatically merge the pull request when the summary is generated.

D. Treat the generated summary as confirmation that the code is correct.

E. Remove the pull request description because Copilot has created one.

ANSWER: A B

Explanation:

The author should compare the summary with the actual changeset to ensure that it accurately describes the purpose, important behavior changes, and affected areas. The author should also edit the summary to correct omissions or misleading statements and to add relevant context that is not apparent from the diff.

A Copilot-generated pull request summary is intended to help reviewers orient themselves; it does not replace author accountability or human code review. Automatically merging the pull request or treating the generated summary as proof that the change is correct would be inappropriate.

QUESTION NO: 4

How does GitHub Copilot Enterprise assist in code reviews during the pull request process? (Select two.)

A. It automatically merges pull requests after an automated review.

B. It generates a prose summary and a bulleted list of key changes for pull requests.

C. It can validate the accuracy of the changes in the pull request.

D. It can answer questions about the changeset of the pull request.

ANSWER: B D

Explanation:

GitHub Copilot Enterprise can streamline pull-request review by generating a concise prose summary together with a bulleted list of the key changes. This gives reviewers a quick, structured overview of the pull request's purpose and the files or areas most affected, reducing the time required to orient themselves before examining the diff in detail. GitHub documents that Copilot can generate these pull-request summaries from the pull request's changed files and commits, and authors can review or edit the generated content before using it.

It can also answer questions about the changeset of the pull request through Copilot Chat. A reviewer can ask for clarification about particular changes, request an explanation of relevant code, or ask questions that use the pull request's context. This conversational assistance helps reviewers investigate changes and understand implementation details without manually piecing together all context first. These capabilities support human review by making the pull request easier to understand and discuss. See GitHub's documentation on generating pull request summaries and using Copilot Chat in GitHub.

QUESTION NO: 5

What configuration needs to be set to get help from Microsoft and GitHub protecting against IP infringement while using GitHub Copilot?

A. Suggestions matching public code to 'blocked'

B. Enforce blocking of MIT or GPL licensed code

C. You need to check code suggestions yourself before accepting

D. Enable GitHub Copilot license checking

ANSWER: A

Explanation:

Suggestions matching public code to 'blocked' is the required configuration. GitHub Copilot can use a code-referencing filter that compares a suggestion with public code on GitHub. When the setting is configured to block matching suggestions, Copilot does not present suggestions that meet the filter's match criteria. This gives developers an important technical safeguard against accepting output that substantially resembles publicly available code and is the setting associated with GitHub's IP-indemnity commitment for eligible Copilot offerings.

The setting should be applied by the organization or enterprise administrator where Copilot Business or Copilot Enterprise is managed, so that it consistently governs suggestions for covered users. It is not merely a developer preference to review output manually: the protection depends on enabling the public-code matching filter in blocking mode. Developers should still review, test, and validate generated code before use, including considering applicable licensing and other project obligations. GitHub documents how administrators configure the public-code matching setting in its organization Copilot settings documentation; the applicable coverage conditions are described in the GitHub Copilot Product Specific Terms.

QUESTION NO: 6

An organization configures content exclusions for a directory containing generated client code. Which effects should the organization expect from this configuration? (Select two.)

A. The generated client code is not available to Copilot as context when the exclusion applies.

B. Copilot code-completion suggestions are unavailable in files that match the exclusion.

C. The excluded files are deleted from the repository after the policy takes effect.

D. Developers lose their GitHub permissions to view the excluded files.

E. The exclusion encrypts the directory in every developer's local workspace.

ANSWER: A B

Explanation:

When content is covered by a content exclusion, GitHub Copilot does not use that content as context for supported Copilot features. Copilot code-completion suggestions are also unavailable in files that match the exclusion. These effects help an organization limit Copilot interaction with designated repository content.

Content exclusions do not delete files, change repository permissions, or prevent developers from viewing the files in their IDE. They are a Copilot content-control setting and should not be treated as a replacement for source-code access controls or other security measures.

QUESTION NO: 7

What method can a developer use to generate sample data with GitHub Copilot? (Each correct answer presents part of the solution. Choose two.)

A. Utilizing GitHub Copilot's ability to create fictitious information from patterns in training data.

B. Leveraging GitHub Copilot's ability to independently initiate and manage data storage services.

C. Utilize GitHub Copilot's capability to directly access and use databases to create sample data.

D. Leveraging GitHub Copilot's suggestions to create data based on API documentation in the repository.

ANSWER: A D

Explanation:

Utilizing GitHub Copilot's ability to create fictitious information from patterns in training data is a valid way to produce sample data. A developer can describe the desired structure and constraints in a natural-language prompt or code comment, such as requesting realistic but fictional customer records, product objects, test users, dates, or JSON payloads. Copilot then proposes content that conforms to the surrounding code and requested format, allowing the developer to review, edit, and use it as non-production test data. GitHub explains that Copilot uses the context supplied in the editor and the prompt to generate suggestions, while developers remain responsible for reviewing those suggestions before use. See GitHub's documentation on responsible use of code completion.

Leveraging GitHub Copilot's suggestions to create data based on API documentation in the repository is also correct. Documentation, type definitions, schemas, example requests, and endpoint descriptions available in the repository can provide relevant context for Copilot. This enables suggestions for sample request bodies, response objects, fixtures, or mock payloads that follow the API's documented fields and shapes. GitHub documents that Copilot can use context from the current file and relevant repository content to improve the relevance of its responses; see asking Copilot questions in an IDE.

QUESTION NO: 8

An independent contractor develops applications for a variety of different customers. Assuming no concerns from their customers, which GitHub Copilot plan is best suited? Â

A. GitHub Copilot Individual

B. GitHub Copilot Business

C. GitHub Copilot Business for non-GHE Customers

D. GitHub Copilot Enterprise

E. GitHub Copilot Teams

ANSWER: A

Explanation:

GitHub Copilot Individual is the best fit because an independent contractor generally uses Copilot through their own personal GitHub account and is responsible for their own development environment, subscription, and workflow. The scenario explicitly removes customer concerns, so there is no stated need for an organization to centrally purchase seats, apply organization-wide policy controls, manage members, or provide enterprise-specific knowledge and administration features. An individual plan is designed for a single developer and provides the Copilot assistance needed for coding tasks across the contractor's client projects without requiring the customer to establish or administer a GitHub organization for that contractor. This makes it the appropriate and cost-conscious choice when the contractor is working independently. GitHub's documentation distinguishes Copilot access for individual users from organization-managed offerings and explains that individual subscriptions are assigned to a user's personal account. See Managing GitHub Copilot for yourself and Subscription plans for GitHub Copilot.