

Software Testing Viva Questions

chatgpt.com/c/672ed8ad-e7dc-8006-ac0f-bbb2707ea3ed

1. Test Scenarios for Gmail Login Page

Question 1: What is the importance of writing test scenarios for the Gmail login page?

Answer: Writing test scenarios for the Gmail login page is crucial to ensure that all functionalities, such as entering correct and incorrect credentials, password recovery, account locking after multiple failed attempts, and CAPTCHA validation, work as expected. These test scenarios help identify potential issues that could affect user experience or security.

Question 2: What test cases will you cover for the Gmail login page?

- Answer:** I will cover test cases such as:
- Valid login with correct credentials.
 - Invalid login with incorrect username/password.
 - Login with empty fields.
 - Password reset functionality.
 - Account lockout after multiple failed login attempts.
 - CAPTCHA validation.
 - Session timeout after inactivity.

Question 3: How will you validate the security of the login page?

Answer: Security validation can be done by checking for SQL injection, cross-site scripting (XSS) vulnerabilities, ensuring encrypted password transmission (SSL/TLS), and verifying login attempts are logged for abnormal activity.

2. Test Cases for a Social Media Website

Question 1: Why is it important to write test cases for a social media website?

Answer: Writing test cases for a social media website is essential to ensure that the features, such as user registration, login, profile management, posting, commenting, notifications, and privacy settings, function properly and meet user expectations. It helps in identifying defects before deployment.

Question 2: What major features would you test for in a social media application?

- Answer:** Some of the major features I would test include:
- User sign-up and login.
 - Profile creation and management (updating profile picture, bio, etc.).
 - Posting updates (text, images, videos).
 - Commenting and liking posts.
 - Following/unfollowing users.
 - Privacy settings.
 - Notifications.
 - Search functionality.
 - Friend requests and acceptance/rejection.

Question 3: What types of testing would you perform for the social media application?

- Answer:** The types of testing would include:
- Functional testing to ensure that all features work as expected.
 - Usability testing to check if the app is user-friendly.
 - Security testing for data protection and authentication.
 - Performance testing to ensure the app performs well under load.
 - Compatibility testing for different devices and browsers.

3. Defect Report for an Application

Question 1: What is a defect report, and why is it important?

Answer: A defect report is a document that describes a bug or issue found during testing, including its severity, steps to reproduce, expected vs. actual results, and any relevant screenshots or logs. It is important because it helps the development team track, understand, and fix defects effectively.

Question 2: What information should be included in a defect report?

- Answer:** A defect report should include:
- Title/ID of the defect.
 - Description of the defect.
 - Steps to reproduce the issue.
 - Expected and actual results.
 - Severity and priority levels.
 - Environment details (browser, OS, version).
 - Screenshots or error logs (if applicable).
 - Status of the defect (open, in-progress, resolved).

Question 3: How would you classify defects in terms of severity?

- Answer:** Defects can be classified as:
- **Critical:** Major functionality or security issue causing the system to fail.
 - **High:** A significant issue affecting a key feature but with a workaround.
 - **Medium:** A moderate issue that does not critically affect functionality.
 - **Low:** Minor issues, such as UI glitches or cosmetic problems.

4. Installation of Selenium Grid, WebDriver, and Java with Eclipse

Question 1: What is Selenium Grid, and why is it used?

Answer: Selenium Grid is a tool that allows you to run tests on different machines, browsers, and environments simultaneously. It is used to perform parallel test execution, reducing the test execution time and making it possible to test on multiple configurations.

Question 2: How do you set up Selenium WebDriver and Grid with Eclipse?

- Answer:** To set up Selenium WebDriver and Grid with Eclipse:
1. Download and install the Selenium WebDriver and Grid jars.
 2. Set up Java and Eclipse IDE.
 3. Configure WebDriver with required browser drivers (like ChromeDriver, GeckoDriver).
 4. Start the Selenium Grid hub and nodes by running specific commands.
 5. Create and execute the test scripts in Eclipse using WebDriver for automation.

Question 3: What are the advantages of using Selenium WebDriver for automation testing?

- Answer:** Some advantages include:
- Supports multiple programming languages (Java, Python, etc.).
 - Allows testing across multiple browsers and operating systems.
 - Enables execution of tests in parallel, saving time.
 - Easy integration with test frameworks like TestNG and JUnit.

5. Software Requirement Specification (SRS) for a Project

Question 1: What is a Software Requirement Specification (SRS), and why is it important?

Answer: SRS is a detailed document that describes the system's functional and non-functional requirements. It acts as a blueprint for developers, testers, and stakeholders to understand what the system should do. It helps in ensuring clear expectations and serves as a reference for future updates.

Question 2: What sections are typically included in an SRS document?

- Answer:** Common sections include:
- Introduction (overview of the system, objectives).
 - Functional Requirements (features and functions).
 - Non-Functional Requirements (performance, security, etc.).
 - System and Software Design Constraints.
 - Use Cases or User Stories.
 - System Architecture and Interfaces.

Question 3: How do you ensure that the requirements in the SRS are clear and testable?

- Answer:** To ensure the requirements are clear and testable:
- Write requirements in a simple, unambiguous language.
 - Break down complex requirements into smaller, manageable tasks.
 - Ensure that each requirement has acceptance criteria to make it verifiable through testing.

Automation Testing: What is automation testing, Automated Testing Process, Automation Frameworks, Benefits of automation testing, how to choose automation testing tools. Selenium Automation Tools: Selenium's Tool Suite- Selenium IDE, Selenium RC, Selenium Web driver, Selenium Grid. Automation Tools: SoapUI, Robotic Process Automation (RPA), Tosca, Appium.

Automation Testing

Question 1: What is automation testing?

Answer: Automation testing is the process of using specialized software tools to perform tests on software applications automatically. It involves writing test scripts that can execute predefined test cases, compare actual outcomes with expected results, and report any discrepancies, reducing manual intervention and increasing testing efficiency.

Question 2: What are the key benefits of automation testing?

- Answer:** The key benefits include:
- Faster execution of tests, especially for repetitive tasks.
 - Increased test coverage.
 - Enhanced accuracy, as tests are executed consistently without human error.
 - Reusability of test scripts across multiple test cycles.
 - Cost-effective in the long run by reducing manual testing efforts.
 - Suitable for regression testing and continuous integration environments.

Question 3: What are the steps involved in the automated testing process?

Answer: The automated testing process typically involves:

1. **Test Planning:** Identify the scope and decide which tests should be automated.
2. **Test Design:** Write test scripts based on the identified test cases.
3. **Test Execution:** Run the automated tests using the selected automation tool.
4. **Result Analysis:** Check the results and compare actual outcomes with expected outcomes.
5. **Reporting:** Generate reports on the test results.
6. **Maintenance:** Update the automation scripts as the application evolves.

Question 4: What are the different types of automation frameworks?

Answer: Common automation frameworks include:

- **Linear Scripting:** Simple script-based approach without reusable components.
- **Modular Testing Framework:** Divides test scripts into smaller, manageable modules.
- **Data-Driven Framework:** Uses external data sources (e.g., Excel, CSV) for input data and allows testing with multiple data sets.
- **Keyword-Driven Framework:** Involves writing scripts using keywords representing specific actions, improving reusability.
- **Hybrid Framework:** Combines the features of modular, data-driven, and keyword-driven frameworks.

Question 5: How do you choose automation testing tools?

Answer: To choose an automation testing tool, consider:

- **Compatibility:** The tool should support the application’s platform, browsers, and technologies.
- **Ease of Use:** Choose tools that are easy to use, with a user-friendly interface and support for various scripting languages.
- **Cost:** Consider both initial and long-term costs, including tool licensing.
- **Integration:** The tool should integrate well with existing development, testing, and CI/CD tools.
- **Community and Support:** The tool should have a strong community and available documentation for support.

Selenium Automation Tools

Question 1: What is Selenium, and why is it so popular in automation testing?

Answer: Selenium is an open-source suite of tools designed for automating web browsers. It is popular because it supports multiple programming languages (Java, Python, C#, etc.), works across various browsers and platforms, and allows for both functional and regression testing in a cost-effective way.

Question 2: Describe the different tools in the Selenium suite.

Answer: Selenium offers the following tools:

- **Selenium IDE:** A simple, record-and-playback tool for automating web applications. It's mainly used for quick prototyping and is browser-based (available as a Firefox and Chrome extension).
- **Selenium RC (Remote Control):** An older tool that allows for automation using different programming languages. It involves setting up a server that acts as a middleman between the test script and the browser. It is now largely deprecated in favor of Selenium WebDriver.
- **Selenium WebDriver:** The most commonly used tool in the Selenium suite. It directly controls the browser by interacting with its native events, making it faster and more powerful than Selenium RC.
- **Selenium Grid:** A tool that allows the parallel execution of tests on multiple machines and browsers simultaneously. It helps speed up the testing process by running tests across different environments in parallel.

Question 3: How does Selenium WebDriver work?

Answer: Selenium WebDriver works by directly communicating with the browser, simulating user actions such as clicking buttons, entering text, or verifying page elements. It uses browser-specific drivers (e.g., ChromeDriver for Chrome, GeckoDriver for Firefox) to interact with the browser.

Question 4: What are the limitations of Selenium WebDriver?

Answer: Some limitations of Selenium WebDriver include:

- Limited support for desktop applications.
- No built-in support for image-based testing or testing flash applications.
- Limited reporting features, requiring third-party integration for comprehensive test reports.
- Handling of pop-ups, alerts, and other advanced browser interactions can sometimes be tricky.

Other Automation Tools

Question 1: What is SoapUI, and when would you use it?

Answer: SoapUI is a popular open-source tool for testing web services, including SOAP and REST APIs. It is used for functional testing, security testing, and load testing of APIs, providing an easy-to-use interface to send requests, validate responses, and analyze API behavior.

Question 2: What is Robotic Process Automation (RPA), and how is it different from automation testing?

Answer: RPA is a technology that uses software robots or “bots” to automate repetitive tasks usually performed by humans, such as data entry, invoice processing, or copying data between systems. Unlike automation testing, which focuses on testing software applications, RPA automates business processes and workflows.

Question 3: What is Tosca, and how is it used in automation testing?

Answer: Tosca is a commercial automation testing tool used for functional and regression testing. It supports test automation without the need for programming skills and provides a model-based testing approach, making it useful for both technical and non-technical testers.

Question 4: What is Appium, and why is it used?

Answer: Appium is an open-source tool for automating mobile applications. It supports testing for both Android and iOS apps and allows testers to write tests using any programming language that supports WebDriver, such as Java, Python, or JavaScript. It enables cross-platform mobile testing with a single codebase.

Testing Framework: Software Quality, Software Quality Dilemma, Achieving Software Quality, Software Quality Assurance Elements of SQA, SQA Tasks, Goals and Metrics, Formal Approaches to SQA, Statistical Software Quality Assurance, Six Sigma for Software Engineering, ISO 9000 Quality Standards, SQA Plan, Total Quality Management, Product Quality Metrics, In process Quality Metrics, Software maintenance, Ishikawa's 7 basic tools, Flow Chart, Checklists, Pareto diagrams, Histogram, Run Charts, Scatter diagrams, Control chart, Cause Effect diagram. Defect Removal Effectiveness and Process

Testing Framework and Software Quality Concepts

Question 1: What is a **testing framework**?

Answer: A testing framework is a set of guidelines or rules used to automate the testing process. It defines the structure and approach for organizing and executing test scripts. Common types of frameworks include data-driven, keyword-driven, and hybrid frameworks.

Question 2: What do you understand by **software quality**?

Answer: Software quality refers to the degree to which a software product meets the specified requirements and satisfies user needs. It encompasses various attributes, including reliability, performance, security, maintainability, and usability.

Software Quality Dilemma

Question 3: What is the **software quality dilemma**?

Answer: The software quality dilemma arises from the trade-offs between time, cost, and quality. Delivering software on time and within budget can sometimes compromise its quality. Quality assurance practices aim to balance these factors by ensuring that the software meets predefined quality standards while staying within project constraints.

Achieving Software Quality

Question 4: How can **software quality** be achieved?

Answer: Achieving software quality involves implementing good software development practices such as:

- Requirement analysis and clear specification.
- Design reviews and quality assurance throughout the development lifecycle.
- Continuous testing (unit testing, integration testing, regression testing).
- Use of quality metrics.
- Adhering to coding standards and performing code reviews.
- Continuous improvement based on feedback.

Software Quality Assurance (SQA)

Question 5: What is **Software Quality Assurance (SQA)**?

Answer: SQA refers to a set of activities designed to ensure that software meets specified requirements and is of high quality. It involves planning, monitoring, and improving the software development process to ensure that the final product is defect-free and meets user expectations.

Question 6: What are the **elements of SQA**?

Answer: The elements of SQA include:

- **Quality Assurance Planning:** Defining processes, standards, and metrics.
- **Reviews and Audits:** Continuous examination of processes and products.
- **Testing:** Verifying that the product behaves as expected.
- **Training and Education:** Ensuring that all team members understand quality practices.
- **Process Improvement:** Ongoing analysis and improvement of development processes.

Question 7: What are the **tasks of SQA**?

Answer: The main tasks of SQA include:

- Defining quality standards and metrics.
- Conducting regular reviews and audits.
- Performing tests at different levels (unit, integration, system, acceptance).
- Managing defects and monitoring process performance.
- Ensuring adherence to standards, guidelines, and best practices.

Question 8: What are the **goals and metrics of SQA**?

Answer: The goals of SQA include:

- Ensuring the software meets customer requirements.
- Minimizing defects and quality issues.
- Improving the development process to increase efficiency and effectiveness.
- Enhancing customer satisfaction.

Metrics can include:

- **Defect Density:** Number of defects per size unit (e.g., lines of code).
- **Test Coverage:** Percentage of requirements or code covered by tests.
- **Mean Time to Failure (MTTF):** Time between failures in the product lifecycle.

Formal Approaches to SQA

Question 9: What are **formal approaches to SQA**?

Answer: Formal approaches to SQA involve structured methodologies such as:

- **CMMI (Capability Maturity Model Integration):** A process improvement approach to improve quality and performance.
- **ISO 9001:** An international standard specifying requirements for a quality management system.
- **Six Sigma:** A method aimed at improving process quality by identifying and removing causes of defects.

Statistical Software Quality Assurance (SSQA)

Question 10: What is **statistical software quality assurance (SSQA)**?

Answer: SSQA uses statistical methods to measure and analyze software quality. It includes techniques such as sampling, control charts, and process behavior analysis to understand and improve the software development process and ensure quality outcomes.

Question 11: How does **Six Sigma** apply to software engineering?

Answer: Six Sigma in software engineering aims to improve the development process by identifying defects and minimizing variability. It uses DMAIC (Define, Measure, Analyze, Improve, Control) methodology to streamline processes and ensure high-quality software with fewer defects.

Question 12: What is the role of **ISO 9000** quality standards in SQA?

Answer: ISO 9000 provides a framework for establishing, maintaining, and improving quality management systems. Adherence to ISO 9000 ensures that organizations follow standardized processes and continuously monitor and improve their software quality.

SQA Plan

Question 13: What is a **Software Quality Assurance (SQA) plan**?

Answer: An SQA plan is a document that outlines the policies, procedures, and resources necessary to ensure software quality throughout the development process. It includes guidelines for quality control, standards to be followed, test strategies, and metrics to measure success.

Total Quality Management (TQM)

Question 14: What is **Total Quality Management (TQM)** in the context of software development?

Answer: TQM is a management approach focused on improving the quality of products and services by involving all employees in the continuous improvement process. In software development, TQM emphasizes collaboration, customer feedback, process optimization, and defect prevention to ensure software quality.

Product Quality and Process Quality Metrics

Question 15: What are **product quality metrics**?

Answer: Product quality metrics evaluate the software's features and functionality from the user's perspective. Common product quality metrics include:

- **Defect Density**
- **Code Complexity**
- **Performance (response time, resource utilization)**

Question 16: What are **in-process quality metrics**?

Answer: In-process quality metrics measure the efficiency and effectiveness of the development process itself, such as:

- **Defect Arrival Rate**
- **Test Coverage**
- **Code Review Effectiveness**

Software Maintenance

Question 17: What is **software maintenance**?

Answer: Software maintenance involves modifying and updating software after its initial release to correct defects, improve performance, or add new features. It includes activities such as bug fixing, patching, performance optimization, and updating software to comply with new regulations.

Ishikawa's 7 Basic Tools

Question 18: What are **Ishikawa's 7 basic tools** used in software quality assurance?

- Answer:** The 7 basic tools are:
- **Flow Chart:** Visual representation of processes or workflows.
 - **Checklists:** Lists of items or steps to ensure tasks are completed.
 - **Pareto Diagrams:** Visuals based on the Pareto principle, showing the most significant problems.
 - **Histograms:** Graphs showing the frequency distribution of data.
 - **Run Charts:** Line graphs that display data trends over time.
 - **Scatter Diagrams:** Graphs showing relationships between two variables.
 - **Control Charts:** Graphs used to track process behavior over time.

Question 19: How is a **Cause and Effect (Fishbone) diagram** used in software quality assurance?

Answer: The Cause and Effect diagram (also known as Ishikawa or Fishbone diagram) is used to identify the root causes of problems. It helps break down the causes of defects into categories such as people, processes, equipment, materials, environment, and management, which can then be analyzed and addressed.

Defect Removal Effectiveness (DRE) and Process

Question 20: What is **Defect Removal Effectiveness (DRE)**, and why is it important?

Answer: DRE is a metric used to measure the effectiveness of defect detection and removal in the software development process. It is calculated by dividing the number of defects found before release by the total number of defects (found before and after release). A higher DRE indicates a more effective testing and quality assurance process.

These questions and answers cover various aspects of software quality, software quality assurance (SQA), and the associated tools and metrics, helping you prepare for your viva in this area. Let me know if you need further details!