Module–3

1. What is load testing?

Ans: Load testing is a type of performance testing conducted to evaluate how a system behaves under specific expected and peak load conditions. The purpose of load testing is to ensure that the system can handle the anticipated user load and transactions without experiencing performance degradation or failures.

1. What is stress Testing?

Ans: Stress testing is a type of software testing that evaluates how a system behaves under extreme conditions beyond normal operational limits. Unlike load testing, which focuses on testing under expected and peak loads, stress testing pushes the system to its breaking point to assess its robustness and reliability.

1. When to used Usability Testing?

Ans: Usability testing is typically used at various stages of software development and deployment to evaluate how user-friendly and intuitive a product or system is for its intended users. Here are some key scenarios and stages where usability testing is beneficial:

1. **Early Development Phases**:
   * **Prototyping**: Usability testing can be conducted on prototypes or wireframes to gather early feedback on user interactions, navigation, and interface design.
   * **Requirements Gathering**: Testing usability early helps validate user requirements and ensures they are aligned with user expectations.
2. **During Development**:
   * **Iterative Testing**: Conducting usability tests iteratively as features are developed allows for continuous improvement based on user feedback.
   * **Integration Testing**: Testing usability during integration phases ensures that different components of the system work seamlessly together from a user perspective.
3. **Before Launch or Release**:
   * **Pre-launch Testing**: Usability testing before a product launch ensures that the final version meets usability goals and addresses any last-minute issues or concerns.
   * **Beta Testing**: Involving real users in beta testing phases helps gather insights from a broader user base and validate usability across diverse user profiles.
4. **Post-Launch**:
   * **Continuous Improvement**: Usability testing post-launch helps identify areas for enhancement or refinement based on actual user usage and feedback.
   * **A/B Testing**: Comparing different versions of interfaces or features through usability testing helps optimize user experience and engagement.
5. **Major Updates or Redesigns**:
   * **Redesign Validation**: Usability testing is crucial when implementing significant design changes or updates to ensure they enhance rather than hinder user experience.
   * **Feature Addition**: Testing usability when adding new features ensures they integrate smoothly into the existing interface and are intuitive for users.
6. **Cross-platform or Device Testing**:
   * **Responsive Design**: Testing usability across different devices (desktops, tablets, smartphones) ensures consistent and optimized user experiences across all platforms.
   * **Accessibility Testing**: Ensuring usability for users with disabilities by testing adherence to accessibility standards (e.g., WCAG) and usability with assistive technologies.
7. What is the procedure for GUI Testing?

Ans: GUI (Graphical User Interface) testing is a type of software testing that focuses on ensuring the functionality, usability, and visual appearance of the graphical elements of an application's user interface. Here is a general procedure for conducting GUI testing:

**1. Understanding Requirements and Design**

* **Review Requirements**: Understand the functional and non-functional requirements related to the GUI.
* **Study Design**: Familiarize yourself with the UI design specifications, including layouts, colors, fonts, icons, and controls.

**2. Test Planning and Strategy**

* **Identify Test Scenarios**: Determine the critical user journeys, workflows, and interactions that need to be tested.
* **Define Test Coverage**: Specify which aspects of the GUI will be tested, such as layout, navigation, input fields, buttons, menus, dialogs, etc.
* **Select Tools**: Choose appropriate tools and frameworks for GUI testing, which may include automation tools for UI testing.

**3. GUI Functionality Testing**

* **Navigation Testing**: Verify navigation paths within the application (e.g., menu navigation, breadcrumbs, links).
* **Input Fields and Controls**: Test functionality of input fields (text boxes, dropdowns, checkboxes, radio buttons) and controls (buttons, sliders, date pickers).
* **Validation Checks**: Ensure proper validation messages and behaviors for input fields (e.g., required fields, format validation).
* **Error Handling**: Verify error handling and messages when incorrect inputs are provided or operations fail.

**4. Layout and Design Consistency**

* **Screen Resolution**: Test GUI elements across different screen resolutions and aspect ratios to ensure responsiveness and layout consistency.
* **Alignment and Spacing**: Verify proper alignment, spacing, and positioning of UI elements according to design specifications.
* **Color and Font Checks**: Ensure consistency in color schemes, fonts, font sizes, and styles throughout the application.

**5. Usability and Accessibility Testing**

* **User Interaction**: Test usability aspects such as ease of use, intuitiveness, and efficiency of performing common tasks.
* **Accessibility Compliance**: Verify adherence to accessibility standards (e.g., WCAG) for users with disabilities, including screen reader compatibility and keyboard navigation.

**6. Localization Testing**

* **Language Support**: Test GUI elements with different languages and character sets to ensure proper localization.
* **Cultural Sensitivity**: Check for cultural appropriateness and localization of UI elements (e.g., date formats, currency symbols).

**7. GUI Performance Testing (if applicable)**

* **Responsiveness**: Test GUI responsiveness under normal and peak load conditions to ensure smooth user interactions.
* **Resource Usage**: Monitor GUI performance metrics such as CPU and memory usage during interactions with the interface.

**8. Regression Testing**

* **Impact Analysis**: Conduct regression testing to ensure that GUI changes or updates do not introduce new issues or affect existing functionality.
* **Automate Regression Tests**: Automate GUI regression tests where possible to streamline testing efforts and ensure consistency.

**9. Documentation and Reporting**

* **Document Test Cases**: Record detailed test cases, including expected outcomes, actual results, and any defects found.
* **Report Issues**: Document and report GUI issues, defects, or inconsistencies with clear steps to reproduce and severity assessments.

**10. Review and Feedback**

* **Review Results**: Review test results with stakeholders, including developers, designers, and product owners.
* **Feedback Loop**: Provide feedback for UI improvements and collaborate on resolving identified issues.

Module-4

1.What is priority?

Ans: Priority refers to the relative importance or significance assigned to test cases, defects, or requirements within a testing process. It helps testers and teams focus their efforts on testing the most critical aspects of the software first, ensuring that high-priority items are thoroughly evaluated and any issues are identified early.

2.What is severity?

Ans: Severity in testing refers to the degree of impact that a defect or issue has on the functionality of the software application. It is a measure of how severe or critical the consequences of a defect are once it manifests in the system. Severity helps in prioritizing the order in which defects should be fixed based on their impact on the software's usability and functionality.

3.Bug categories are…

Ans: Here are the common bug categories:

1. **Functional Bugs:**
   * **Description:** These bugs occur when the software does not behave according to its functional requirements or specifications.
   * **Examples:**
     + Calculation errors
     + Incorrect data processing
     + Missing or incorrect functionalities
2. **Performance Bugs:**
   * **Description:** These bugs affect the performance of the software, causing it to operate slower or consume more resources than expected.
   * **Examples:**
     + Slow response times
     + Memory leaks
     + High CPU usage under certain conditions
3. **Compatibility Bugs:**
   * **Description:** These bugs arise when the software does not work correctly across different hardware, operating systems, browsers, or environments.
   * **Examples:**
     + Layout issues on specific browsers
     + Features not supported on certain operating systems
     + Integration issues with third-party software or APIs
4. **Usability Bugs:**
   * **Description:** These bugs affect the usability or user experience of the software, making it difficult or confusing for users to interact with.
   * **Examples:**
     + Unclear error messages
     + Inconsistent user interface behaviour
     + Poor accessibility features
5. **Security Bugs:**
   * **Description:** These bugs expose vulnerabilities in the software that could potentially be exploited by malicious actors.
   * **Examples:**
     + SQL injection vulnerabilities
     + Cross-site scripting (XSS) vulnerabilities
     + Authentication or authorization flaws
6. **Interface Bugs:**
   * **Description:** These bugs occur at the interface between different components or modules of the software.
   * **Examples:**
     + Data not transferred correctly between modules
     + Incorrect API usage or parameters
     + Integration issues with external systems
7. **Documentation Bugs:**
   * **Description:** These bugs relate to errors or inconsistencies found in the software documentation, such as user manuals, help guides, or technical specifications.
   * **Examples:**
     + Outdated information
     + Incorrect usage instructions
     + Missing documentation for new features
8. **Installation Bugs:**
   * **Description:** These bugs occur during the installation or deployment of the software.
   * **Examples:**
     + Installation failure on certain platforms
     + Incomplete installation of required components
     + Incorrect configuration settings after installation

4.Advantage of Bugzilla?

Ans: Bugzilla offers several advantages that make it a popular choice for bug tracking and issue management in software development and quality assurance teams:

1. **Open Source and Customizable:**
   * Bugzilla is an open-source tool, which means it is freely available for use and can be customized according to the specific needs of the organization. Users can modify its features, workflows, and interface to suit their project requirements.
2. **Comprehensive Bug Tracking:**
   * Bugzilla provides a robust platform for tracking bugs and issues throughout their lifecycle. It allows users to create, update, prioritize, and assign bugs, ensuring transparency and accountability in bug resolution.
3. **Advanced Search and Reporting:**
   * Bugzilla offers powerful search capabilities, allowing users to filter and find bugs based on various criteria such as status, severity, priority, assignee, and more. This helps in efficiently managing large volumes of bug reports.
   * It also supports customizable reports and charts, enabling teams to generate insightful metrics and track project progress.
4. **Collaboration and Communication:**
   * Bugzilla facilitates collaboration among team members and stakeholders by providing tools for discussions, attachments (such as screenshots or log files), and email notifications. This enhances communication and ensures that everyone involved stays informed about bug statuses and updates.
5. **Integration Capabilities:**
   * Bugzilla supports integration with other development tools and systems, such as version control systems (e.g., Git, SVN), continuous integration tools, project management software, and more. This seamless integration streamlines workflows and enhances productivity.
6. **Custom Workflow and Automation:**
   * Organizations can define custom workflows in Bugzilla to reflect their specific bug triaging and resolution processes. Automated actions, such as email notifications, status updates, or assignments, can be configured based on predefined rules, reducing manual effort and improving efficiency.
7. **Security and Access Control:**
   * Bugzilla offers robust security features, including role-based access control (RBAC) and permissions management. Administrators can control who can view, edit, or modify bugs and ensure sensitive information is protected.
8. **Scalability and Performance:**
   * Bugzilla is designed to handle large-scale projects and high volumes of bug reports effectively. It can manage thousands of bugs simultaneously without compromising performance, making it suitable for both small teams and large enterprises.

5.Difference between priority and severity

Ans:

| **Features** | **Severity** | **Priority** |
| --- | --- | --- |
|  |  |  |
| **Definition** | Severity is a parameter to denote the impact of a particular defect on the software. | Priority is a parameter to decide the order in which defects should be fixed. |
| **Purpose** | Severity means how severe the defect is affecting the functionality. | Priority means how fast the defect has to be fixed. |
| **Relation** | Severity is related to the quality standard. | Priority is related to scheduling to resolve the problem. |
| **Categories** | Severity is divided into 4 categories:   * Critical * Major * Medium * Low | Priority is divided into 3 categories:   * Low * Medium * High |
| **Who decides defects?** | The testing engineer decides the severity level of the defect. | The product manager decides the priorities of defects. |
| **Value** | Its value is objective. | Its value is subjective. |
| **Value change** | Its value doesn’t change from time to time. | Its value changes from time to time. |
| **Association** | It is associated with functionality or standards. | It is associated with scheduling. |