Extra questions

Q1. FTR (Formal Technical Review)

=>

1) A Formal Technical Review (FTR) is a way software engineers check and improve the quality of a technical document or part of the software.
2) It's a structured process to find problems, make sure standards are followed, and raise the overall quality of the product.

3) Key Goals of FTR

   i) **Detecting Issues**: Find mistakes, inconsistencies, or parts that don't follow guidelines.
   ii) **Quality Assurance**: Make sure the final product is high-quality and meets project requirements.
   iii) **Reducing Risks**: Spot and address potential risks early before they grow.
   iv) **Knowledge Sharing**: Help team members learn from each other and build shared knowledge.
   v) **Consistency**: Ensure all steps, coding standards, and policies are followed.
   vi) **Learning**: Give team members, especially newer ones, a chance to learn from experienced peers.

4) FTRs are especially useful for junior engineers to observe real examples of design, coding, and testing.
5) This process also helps with backup and continuity so that team members are familiar with all parts of the project.

6) **Types of FTRs**

   i) FTRs include different review types like **walkthroughs**, **inspections**, **round-robin reviews**, and other group assessments.
   ii) Each review is a meeting that should be well-planned, controlled, and attended by the right people to be effective.

7) **Example**

i) Imagine software development without FTR. If designing costs 10 units, coding 15 units, and testing 10 units, the total cost is 35 units.

ii) But if there's a problem with the design, fixing it could double the cost to 70 units. With FTR, issues are caught early, saving time and money.

Q2. Difference between FTR and Walkthrough

=>

Aspect	Formal Technical Review (FTR)	Walkthrough
Definition	A formal review done by software engineers and others to check quality and spot issues.	A less formal review led by the author to get feedback and help others understand the document.
Purpose	To find errors, ensure the work meets standards, and improve quality.	To present and explain the document to get feedback and make sure others understand it.
Who Leads It	Typically led by peers (other team members), not management.	Led by the author (the person who created the document).
Focus	Used for any work item, like requirements, design, code, or testing documents.	Usually for high-level documents, like requirements or initial designs.
Process	Structured, with an organized plan and checklist.	Informal; participants go through the document step-by-step with guidance from the author.
Types Included	FTR includes other review types like walkthroughs, inspections, and group assessments.	A walkthrough is one type of FTR.
Goal	To find errors, improve quality, follow standards, and help the team learn together.	To help the team understand the document better and provide informal feedback.
Best For	Any part of the project (requirements, design, coding, testing stages).	Best for early project documents that need to be clearly understood, especially by non-technical people.
Outcome	A detailed list of errors and feedback to improve the product's quality.	Suggestions and feedback to help make the document clearer.

Q. 3P's in project management

=>

1. People

- **Clear Roles and Responsibilities**:
  1. Each team member has a specific role, which helps prevent confusion and ensures everyone knows what they are responsible for.
  2. For example, in a software development project, developers focus on coding, while QA engineers focus on testing.
- **Team Collaboration**:
  1. Effective teamwork is essential. Team members should communicate openly and share updates to avoid misunderstandings.
  2. For example, in a website redesign project, designers, developers, and content creators need to work together and update each other frequently.
- **Training and Support**:
  1. Sometimes team members need new skills or support to perform better.
  2. For example, a company implementing a new software system may provide training sessions to help employees understand the new tools and workflows.

2. Product

- **Clear Product Scope**:
  1. The project manager defines what the product should do and the features it needs. This helps avoid scope creep (adding extra features later on).
  2. For example, in developing a mobile app, the scope may include features like user login, profile settings, and notifications, but not additional features like social media integration unless specified.
- **Quality Standards**:
  1. Setting quality standards helps ensure the final product meets expectations.
  2. For example, when creating an e-commerce website, the product must meet standards for loading speed, security, and user-friendliness.

3. Process

- **Well-Defined Phases**:
  1. Each project stage has clear steps, such as planning, development, testing, and deployment, which helps the team stay organized and on track.
  2. For example, in a marketing campaign project, the process could include research, design, content creation, and final launch.
- **Milestones and Deadlines**:
  1. Breaking down the project into milestones helps track progress and maintain momentum.
  2. For example, a website redesign project might have milestones like "Complete initial design by March," "Finish development by April," and "Launch by May."
- **Continuous Monitoring**:
  1. The project manager should regularly check progress to ensure the project stays on schedule and within budget.
  2. For example, in a construction project, the project manager might monitor each phase, such as laying the foundation, framing, and roofing, to ensure everything is on track.

Q4. Software Maintenance

=>

1. **Software Maintenance** is the ongoing process of updating and modifying software after it has been delivered to the customer.
2. This includes fixing bugs, adding new features, and ensuring the software remains compatible with new hardware and software environments.

3. **Key Aspects of Software Maintenance**

   i. **Bug Fixing**: Identifying and correcting errors in the software to keep it working smoothly.
      a. *Example*: Fixing a login issue where users cannot access their accounts.
   ii. **Adding Features**: Updating the software to add new features based on user feedback or changing needs.
      a. *Example*: Adding a "dark mode" to an app after users request it.
   iii. **Performance Optimization**: Improving the speed and efficiency of the software to enhance the user experience.
      a. *Example*: Reducing the loading time of a website by optimizing images and code.
   iv. **Compatibility Updates**: Adapting the software to work on new devices or operating systems.
      a. *Example*: Updating a mobile app to work on the latest iOS version.

4. **Types of Software Maintenance**

   i. **Corrective Maintenance**: Fixes bugs and issues in the software.
      a. *Example*: Patching a bug that causes the app to crash under certain conditions.
   ii. **Adaptive Maintenance**: Adapts the software to new environments (e.g., new operating systems or devices).
      a. *Example*: Updating an app to function correctly on a new Android version.
   iii. **Perfective Maintenance**: Adds new features and improves existing ones to meet evolving user needs.

    a. *Example*: Adding a new search feature to an e-commerce app to make it easier for users to find products.

iv. **Preventive Maintenance**: Addresses potential issues to avoid future problems.
    a. *Example*: Running regular security checks and backups to prevent data loss.

5. Common Challenges in Software Maintenance

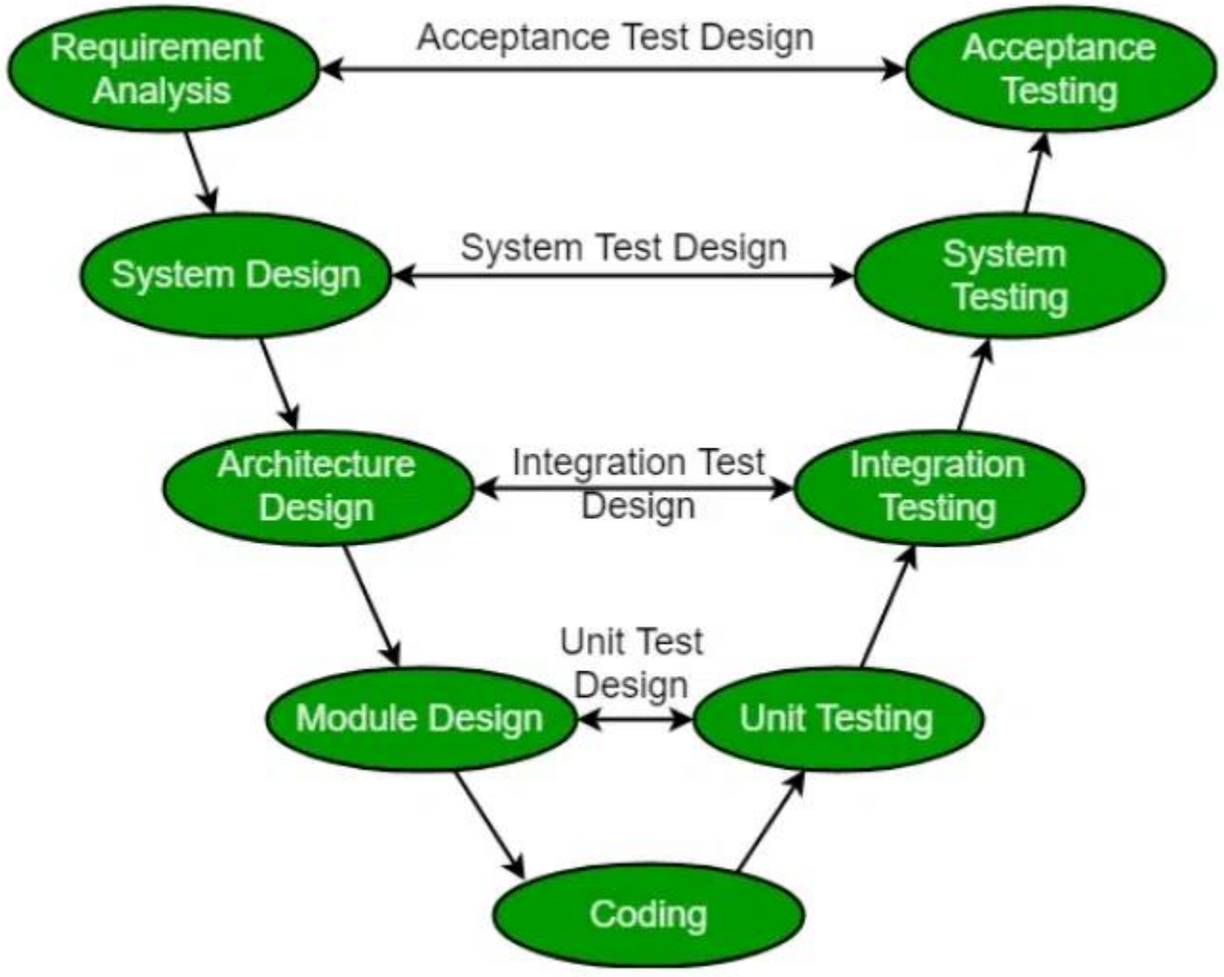
i. **High Cost**: Maintenance can be expensive, especially for large, complex software systems.
ii. **Complexity**: As software grows, it becomes harder to understand and modify, increasing the time and effort needed for updates.
iii. **Lack of Documentation**: Poorly documented software makes it difficult to find and fix problems, especially for new team members.

6. Importance of Software Maintenance

1. Maintaining software is essential for keeping it functional, secure, and up-to-date with user needs and technological advances.
2. Effective maintenance ensures the software remains reliable and continues to deliver value over time.

Q5. V-Model

=>


Requirement Analysis
Acceptance Test Design
Acceptance Testing
System Design
System Test Design
System Testing
Architecture Design
Integration Test Design
Integration Testing
Unit Test Design
Module Design
Unit Testing
Coding

Q6. Compare Scrum and Kanban

=>

S. No.	Scrum	Kanban
1	Has defined roles for team members (e.g., Scrum Master).	No specific roles; everyone can contribute equally.
2	Works in time-limited **sprints** (usually 1-4 weeks).	Work flows **continuously**, with no set time limits.
3	Breaks work into **small tasks** and handles them one by one.	Does not break work into smaller tasks; tasks are managed as they come.
4	Requires planning and estimation before starting work.	Less planning; tasks are handled as they appear.
5	Uses a **sprint backlog** to track tasks during each sprint.	Uses a **Kanban board** to visualize tasks and track progress.
6	Focuses on completing work in each sprint before moving on.	Tasks can be worked on continuously without breaks.
7	Ideal for projects with **changing priorities**.	Works best for projects with **stable priorities**.
8	Measures productivity with **velocity** (how much work is completed in a sprint).	Measures productivity with **cycle time** (how long it takes to complete a task).