2-Mark Questions (Fully Expanded)

1. Define a software project with an example.
A **software project** is a structured and organized effort undertaken to create, modify, or maintain a software product or system within a fixed scope, budget, and schedule. It involves a series of planned activities such as requirement gathering, designing, coding, testing, deployment, and maintenance. Every software project is unique in terms of its objectives, deliverables, and constraints.
Example: Creating a **student attendance management system** for a college that allows teachers to mark attendance, generate reports, and notify parents. This project would involve front-end design, database creation, backend logic, and integration with SMS/email services.

---

2. What is the purpose of estimation in project planning?
The main purpose of **estimation** is to predict and calculate the resources, effort, cost, and time needed to successfully complete a software project. Estimation helps in:

- **Setting realistic deadlines** so that the team can deliver the product without unnecessary pressure.
- **Allocating resources effectively**, ensuring each team member has clear and achievable targets.
- **Budget planning** so that financial constraints are respected and unexpected costs are minimized.
- **Risk management**, since knowing how long and how much a project might take helps anticipate challenges early.

---

3. List any two attributes of a good software quality measure.

1. **Reliability:** The ability of a software system to perform its required functions under specified conditions for a given period without failure. For example, an online banking application should be able to process transactions accurately at all times.
2. **Maintainability:** The ease with which a software product can be modified to correct defects, improve performance, or adapt to a new environment. A well-documented and modular codebase is easier to maintain.

---

4. What do you mean by software roles and responsibilities?
In a software project, **roles** refer to the specific job positions such as developer, tester, project manager, business analyst, etc. **Responsibilities** are the tasks and duties assigned to these roles. Clearly defining roles and responsibilities:

- Ensures **accountability** so each person knows exactly what is expected of them.
- Prevents **overlap and confusion** between tasks.
- Improves **efficiency** by utilizing each team member's skills effectively.
  For example, a tester's role is to find defects, while their responsibility is to ensure that the final product meets quality standards.

---

5. Mention two important components of a Software Development Life Cycle (SDLC).

1. **Requirement Analysis:** This phase involves gathering, documenting, and analyzing the needs of the client or stakeholders. The outcome is a clear set of functional and non-functional requirements.
2. **Testing:** This phase verifies that the developed software meets the specified requirements and is free from major defects before deployment. Testing ensures quality and reduces post-release issues.

3-Mark Questions (Fully Expanded)

6. Explain the importance of a capability baseline in project planning.
A **capability baseline** is a documented, approved version of the project scope, schedule, and cost that serves as a reference for measuring and monitoring performance. It plays a crucial role in project planning because:

1. **Performance Tracking:** It allows project managers to compare actual results against planned values to determine whether the project is on track.
2. **Change Control:** Any deviation from the baseline requires formal approval, ensuring that changes are controlled and justified.
3. **Stakeholder Alignment:** It provides a shared understanding among all stakeholders regarding the agreed project objectives and deliverables.
   Example: If a project baseline says the login module should be completed in 2 weeks, but the team takes 3 weeks, managers can investigate causes such as scope changes or resource delays.

7. Briefly describe the term "Project Management Plan".

A **Project Management Plan (PMP)** is a comprehensive, formal document that outlines how the project will be initiated, executed, monitored, controlled, and closed. It acts as a **roadmap** for project execution and includes:

- **Scope Statement:** What will and will not be included in the project.
- **Schedule:** Timelines, milestones, and deadlines.
- **Budget:** Estimated costs and resource allocation.
- **Risk Management:** Identifying potential risks and mitigation strategies.
- **Communication Plan:** How stakeholders will be informed about progress and issues.
  A PMP ensures that everyone involved in the project works towards common goals.

8. Differentiate between functional and non-functional requirements in software projects.

Aspect	**Functional Requirements**	**Non-Functional Requirements**
Definition	Specify what the system should do (specific behaviors or functions).	Specify how well the system should perform and constraints on the system.
Example	“The system must allow users to reset their password via email verification.”	“The system must load within 2 seconds and support 1000 concurrent users.”
Purpose	Describe the intended features and capabilities.	Define performance, usability, and reliability standards.

9. Describe any three quality attributes of a software product.

1. **Reliability:** The probability of software operating without failure under given conditions.
   Example: A railway ticket booking system must process bookings correctly during peak hours without crashing.
2. **Usability:** The extent to which users can easily understand and operate the system.
   Example: A mobile payment app with simple navigation and intuitive design.

3. **Efficiency:** The capability of software to use resources optimally while delivering maximum performance. Example: A search engine that provides accurate results in milliseconds.

---

10. Why is team structure important in software project planning?
A **well-defined team structure** determines how team members are organized and how work flows between them. It is important because:

- It assigns **clear roles** to avoid duplication of work.
- It ensures **effective communication** by defining reporting relationships.
- It helps in **resource optimization**, ensuring the right people are assigned to the right tasks.
- It supports **conflict resolution** by defining decision-making authority.
  Without a structured team, projects can face confusion, inefficiency, and poor coordination.

Q11. Discuss any two software development models and their use cases.

A **software development model** is a structured approach that defines how software projects are planned, executed, and delivered. Different models are suitable for different types of projects depending on the complexity, clarity of requirements, and flexibility needed.

---

1. Waterfall Model
The Waterfall Model is one of the earliest and most widely recognized models. It follows a **linear sequence** where one phase must be completed before moving to the next.

Phases in Waterfall:

1. Requirement gathering and analysis
2. System design
3. Implementation (coding)
4. Testing
5. Deployment
6. Maintenance

Use Cases:

- Banking applications where security is critical and requirements are fixed.
- Government projects where documentation is mandatory.

Advantages:

- Easy to understand and manage.
- Clearly defined milestones and deliverables.

Disadvantages:

- Not flexible to changes.
- Errors found late in testing can be expensive to fix.

---

2. Agile Model
The Agile Model focuses on **iterative and incremental** development, delivering working software in short time frames called **sprints** (usually 2–4 weeks).

Principles of Agile:

- Customer collaboration over contract negotiation.
- Responding to change over following a rigid plan.
- Delivering small, functional parts of the software quickly.

Use Cases:

- Mobile app development where user feedback is frequent.
- Startups where requirements may change every few weeks.

Advantages:

- High flexibility and adaptability.
- Early delivery of usable features.

Disadvantages:

- Requires experienced teams.
- May lead to scope creep if not managed well.

Conclusion:
The **Waterfall Model** is best for projects with stable requirements, while **Agile** suits dynamic and fast-changing environments.

Q12. Explain the testing strategy with respect to project planning.

A **testing strategy** is a high-level document that defines the approach to be followed to ensure the software meets its quality objectives. It is created during project planning to guide the testing activities throughout the development process.

Importance in Project Planning:

- Ensures defects are detected early.
- Reduces rework and delays.
- Provides a roadmap for testing phases and responsibilities.

Components of a Testing Strategy:

1. **Scope of Testing**
    - Functional testing (verifying features)
    - Non-functional testing (performance, security, usability)
2. **Testing Levels**
    - *Unit Testing:* Checking individual modules.
    - *Integration Testing:* Verifying data flow between modules.
    - *System Testing:* Checking the complete system.
    - *User Acceptance Testing (UAT):* Final approval from end-users.
3. **Resource Planning**
    - Assigning testers, automation engineers, and test leads.
4. **Schedule Planning**
    - Aligning test activities with development sprints or milestones.
5. **Defining Quality Metrics**
    - Defect density, test coverage percentage, and pass/fail ratio.

Example:
In a food delivery app, payment gateway and cart checkout testing would be prioritized first as they are most critical to business success.

Conclusion:
A well-planned testing strategy ensures quality, reduces risks, and increases customer satisfaction.

Q13. Describe various roles in a typical software project team.

A **software project team** is a group of professionals with specific skills and responsibilities, working together to achieve the objectives of a software project. The composition and structure of the team are crucial for delivering the project successfully, within time, budget, and quality constraints.

Key Roles and Responsibilities:

1. **Project Manager (PM):**
    - Oversees the project from initiation to closure.
    - Manages scope, budget, and timelines.
    - Acts as a communication bridge between stakeholders and the development team.
    - Handles risk management and conflict resolution.
2. **Business Analyst (BA):**
    - Works closely with clients to gather and document requirements.
    - Ensures requirements are clear, complete, and feasible.
    - Acts as a translator between business needs and technical solutions.
3. **Software Developers (Frontend, Backend, Full-stack):**
    - Write, test, and maintain the code.
    - Implement functionalities as per the specifications.
    - Collaborate with testers to fix bugs.
4. **Quality Assurance Engineers / Testers:**

    - Design test cases and execute them.
    - Ensure the software is defect-free and meets the requirements.
    - Perform functional, performance, and security testing.

5. **UI/UX Designers:**
    - Create visually appealing, user-friendly interfaces.
    - Ensure the design aligns with usability and accessibility standards.
6. **System Administrators / DevOps Engineers:**
    - Manage servers, deployment pipelines, and cloud resources.
    - Ensure smooth deployment and system stability.

Importance of Defined Roles:

- Prevents confusion and duplication of work.
- Improves accountability and productivity.
- Ensures that all aspects of the project — technical, managerial, and quality-related — are covered.

Example:
In a hospital management software project, the PM handles overall delivery, the BA defines requirements like patient registration and appointment booking, developers build the system, testers check for bugs, and UI/UX designers ensure doctors and nurses can use it easily.

Q14. Illustrate the estimation process with a simple example.

Definition:
Estimation is the process of predicting the amount of time, cost, and effort required to complete a software project. Accurate estimation ensures that the project runs smoothly without over-commitment or resource shortage.

Steps in Estimation:

1. **Requirement Analysis:**

- Understand the project scope in detail.
- Identify deliverables and functionalities.

2. **Break Down the Work (WBS):**
   - Divide the project into small, manageable tasks or modules.
3. **Choose an Estimation Technique:**
   - *Function Point Analysis (FPA):* Measures functionality based on inputs, outputs, files, and interfaces.
   - *Lines of Code (LOC):* Based on expected lines of code to be written.
   - *Expert Judgment:* Based on previous experience with similar projects.
4. **Calculate Effort for Each Task:**
   - Assign hours or days needed for each activity.
5. **Add Contingency Buffer:**
   - Usually 10–20% extra time and cost to cover unexpected issues.

Example:
For an **Online Library Management System**:

- UI Design: 40 hours
- Backend Development: 100 hours
- Testing: 60 hours
  Total = 200 hours
  With a 10% buffer = 220 hours.

If the team has 4 developers working 5 hours/day, the project will take **220 / (4 × 5) = 11 working days** approximately.

Importance:

- Prevents underestimation and overestimation.
- Helps in resource allocation and budgeting.

- Builds trust with clients by setting realistic deadlines.

Q15. How are metrics useful for managing different types of software projects?

Definition:
Software metrics are quantitative measurements that help track, assess, and improve software development processes and products. They are essential for making informed project management decisions.

Uses of Metrics in Project Management:

1. **Tracking Productivity:**
   - *Example:* Lines of Code (LOC) written per developer per week.
   - Helps assess efficiency and workload distribution.
2. **Monitoring Quality:**
   - *Example:* Defect density (defects per 1,000 LOC).
   - Helps ensure high-quality standards are maintained.
3. **Measuring Schedule Adherence:**
   - *Example:* Planned vs. actual completion dates.
   - Detects delays early so corrective measures can be taken.
4. **Risk Management:**
   - Metrics highlight performance gaps and potential risks, enabling timely intervention.
5. **Improving Future Projects:**
   - Historical data from metrics helps in better estimation and planning for future projects.

Example:
In an Agile project, metrics like *velocity* (story points completed per sprint) and *burn-down charts* visually show progress and help identify delays before they become critical.

Conclusion:
Without metrics, project management becomes guesswork; with metrics, decisions are based on facts and measurable evidence.

Q16. Draw and explain the Software Development Life Cycle (SDLC) with phases.

Definition:
The SDLC is a structured process for developing high-quality software in a systematic way. It includes phases that guide the project from conception to maintenance.

Phases of SDLC:

1. **Requirement Analysis:**
   - Gather requirements from stakeholders.
   - Document functional and non-functional requirements.
2. **System Design:**
   - Create architecture, data flow diagrams, and database designs.
3. **Implementation (Coding):**
   - Developers write the code according to the design documents.
4. **Testing:**
   - Testers check for functionality, performance, and security issues.
5. **Deployment:**
   - The software is released for use in the production environment.
6. **Maintenance:**
   - Fix bugs, update features, and adapt to changing environments.

Diagram to Draw in Exam:
(A simple circle or waterfall diagram showing 6 phases with arrows between them.)

Example:
In a food ordering app, requirement analysis involves understanding menu features, design

involves creating the layout, coding builds the app, testing ensures no bugs, deployment makes it live, and maintenance keeps it updated.

Q17. Develop a basic project management plan outline for a student record management system.

1. Project Overview:
Developing a system to digitally manage student profiles, grades, and attendance.

2. Scope:

- Store student personal and academic details.
- Generate performance reports.
- Track attendance.

3. Timeline:

- Requirement Gathering: 2 weeks
- Design: 3 weeks
- Development: 8 weeks
- Testing: 3 weeks
- Deployment: 2 weeks

4. Budget:
₹1,50,000 for software licenses, hardware, and manpower.

5. Team Structure:

- Project Manager (1)
- Developers (2)
- Tester (1)

6. Quality Standards:

- Data accuracy ≥ 99%.
- Uptime ≥ 99.5%.

7. Risks & Mitigation:

- Data loss → regular backups.
- Server downtime → redundant hosting.

8. Communication Plan:
Weekly status meetings and monthly client reviews.

9. Closure:
Final testing, documentation, and staff training.

Q18. Explain the relationship between software quality attributes and software metrics with examples.

Relationship:
Software quality attributes define the properties that make software effective, while software metrics measure these attributes in numerical terms.

Examples:

1. **Performance → Metric:** Average response time in seconds.
2. **Reliability → Metric:** Mean Time Between Failures (MTBF).
3. **Usability → Metric:** Time taken by a new user to complete a task.

Conclusion:
Metrics make quality attributes measurable, ensuring objective evaluation and continuous improvement.

Q19. Analyze the impact of poor planning on a software project's success.

Impacts:

1. **Scope Creep:** Uncontrolled changes lead to delays and cost overruns.
2. **Budget Overruns:** Misallocation of resources increases costs.
3. **Missed Deadlines:** Unrealistic timelines cause delivery delays.
4. **Low Quality:** Inadequate testing time results in defects.
5. **Team Burnout:** Overloaded teams become less productive.

Example:
An e-commerce site launched without proper planning might crash during peak sales, causing loss of revenue and reputation.

Q20. Compare three software development models and identify which one is best suited for a large-scale enterprise project.

1. Waterfall:

- Rigid, sequential approach.
- Best for stable, fixed requirements.

2. Agile:

- Flexible, iterative approach.
- Best for projects with changing requirements.

3. Spiral:

- Combines iterative development with risk analysis.
- Best for large, complex projects with high uncertainty.

Best Choice:
For a large-scale enterprise project, the **Spiral Model** is most suitable due to its strong risk management, stakeholder feedback loops, and adaptability to complexity.