Fundamentals of Software Development (4331604) - Winter 2023 Solution

Milav Dabgar

January 20, 2024

Question 1(a) [3 marks]

Define Software and explain its characteristics.

Solution

Software is a collection of programs, instructions, and documentation that performs tasks on a computer system.
Key Characteristics:

Table 1. Software Characteristics

Characteristic	Description
Intangible	Cannot be touched physically
Logical	Created through systematic approach
Manufactured	Developed, not produced traditionally
Complex	Has intricate internal structure

Mnemonic

"In Logic, Manufacturing Creates: Intangible, Logical, Manufactured, Complex"

Question 1(b) [4 marks]

Write a note on Software engineering – A layered technology.

Solution

Software engineering is structured as a layered technology with each layer supporting the next.
Layered Structure:

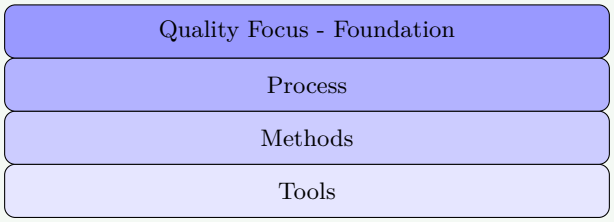

Figure 1. Software Engineering Layers

Table 2. Layer Descriptions

Layer	Purpose	Description
Quality Focus	Foundation	Emphasis on delivering quality products
Process	Framework	Defines how software development is done
Methods	Techniques	Specific ways to perform activities
Tools	Automation	Software that supports methods

Mnemonic

"Tools Make Process Quality: Tools, Methods, Process, Quality"

Question 1(c) [7 marks]

Explain Software Process framework and umbrella activities.

Solution

Software Process Framework provides structure for software development with core activities and umbrella activities.

Framework Activities:

Table 3. Framework Activities

Activity	Purpose	Key Tasks
Communication	Understand requirements	Stakeholder interaction, requirement gathering
Planning	Create roadmap	Estimation, scheduling, risk assessment
Modeling	Create blueprints	Analysis and design models
Construction	Build software	Coding and testing
Deployment	Deliver to users	Installation, support, feedback

Umbrella Activities:

- **Software project tracking**: Monitor progress and control quality
- **Risk management**: Identify and mitigate potential problems
- **Quality assurance**: Ensure standards are met
- **Configuration management**: Control changes systematically
- **Work product preparation**: Create deliverable documents

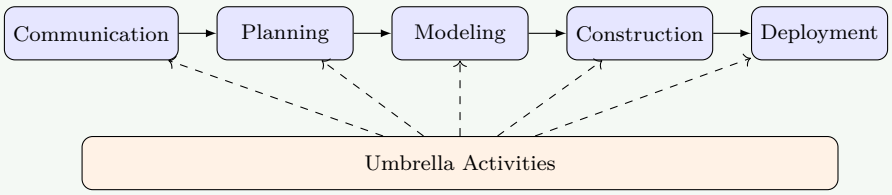

Figure 2. Process Framework

Mnemonic

"Can People Model Construction Daily (Framework)" "Track Risk Quality Configuration Work (Umbrella)"

Question 1(c OR) [7 marks]

Define SDLC and explain each phase.

Solution

SDLC (Software Development Life Cycle) is a systematic process for developing software applications.
SDLC Phases:

Table 4. SDLC Phases

Phase	Purpose	Key Activities	Deliverables
Planning	Define scope	Feasibility study, resource allocation	Project plan
Analysis	Gather requirements	Requirement collection, documentation	SRS document
Design	Create architecture	System design, database design	Design documents
Implementation	Write code	Programming, unit testing	Source code
Testing	Verify quality	System testing, bug fixing	Test reports
Deployment	Release software	Installation, user training	Live system
Maintenance	Ongoing support	Bug fixes, enhancements	Updated system

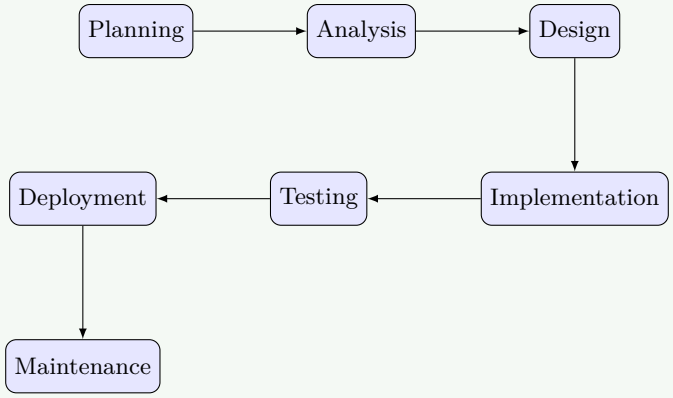

Figure 3. SDLC Phases

Mnemonic

"Please Analyze Design Implementation Testing Deployment Maintenance"

Question 2(a) [3 marks]

Describe advantage disadvantage of prototype model.

Solution

Prototype Model Analysis:

Table 5. Prototype Model Pros and Cons

Advantages	Disadvantages
Early feedback from users	**Time consuming** development process
Reduced risk of failure	**Cost increase** due to iterations
Better understanding of requirements	**Scope creep** may occur

Mnemonic

"Early Reduced Better vs Time Cost Scope"

Question 2(b) [4 marks]

Explain Prototyping Model and justify when to use with example.

Solution

Prototyping Model creates working model of software early in development process.
When to Use:

Table 6. Usage Scenarios

Situation	Example	Justification
Unclear requirements	Online shopping cart	User interface needs refinement
New technology	Mobile banking app	Feasibility testing required
User interaction critical	Gaming application	User experience validation needed

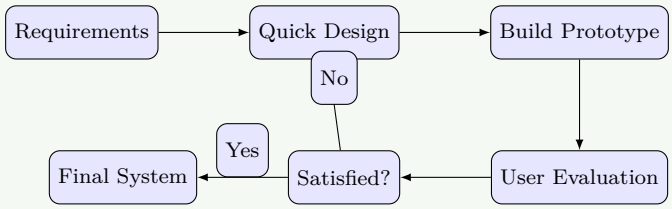

Figure 4. Prototyping Process

Mnemonic

"Requirements Quick Build User Satisfied Final"

Question 2(c) [7 marks]

Sketch and discuss (I) Waterfall model & (II) Incremental Model.

Solution

(I) Waterfall Model: Linear sequential approach where each phase must complete before next begins.

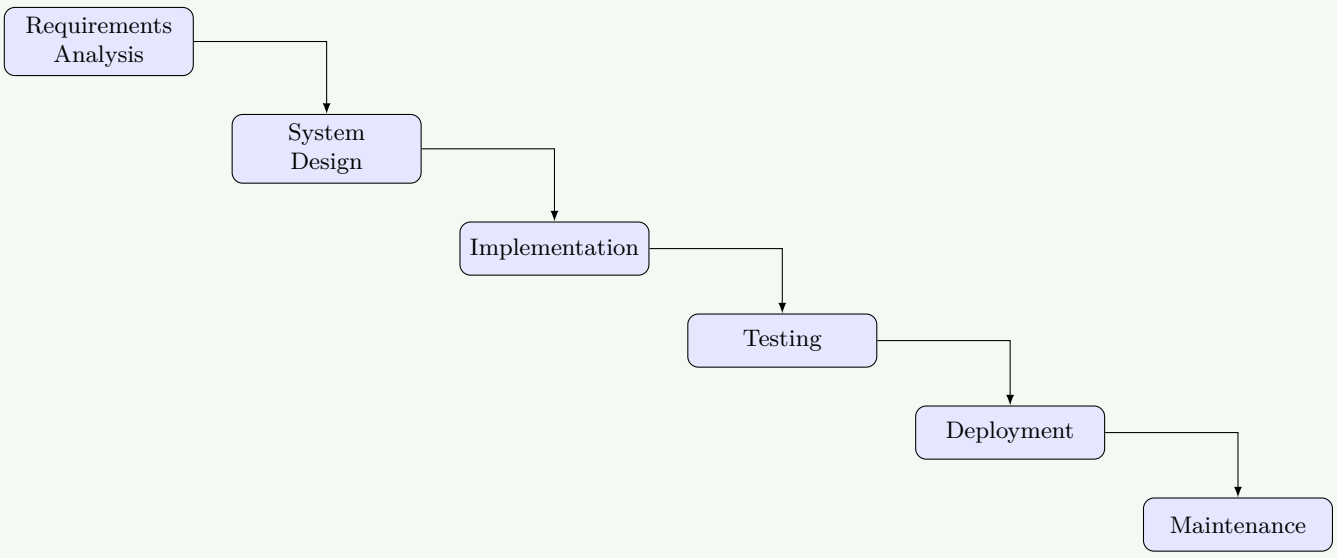

Figure 5. Waterfall Model

Table 7. Waterfall Characteristics

Characteristics	Description
Sequential	One phase at a time
Documentation driven	Heavy documentation
Suitable for	Well-defined requirements

(II) Incremental Model: Development in small increments with each increment adding functionality.

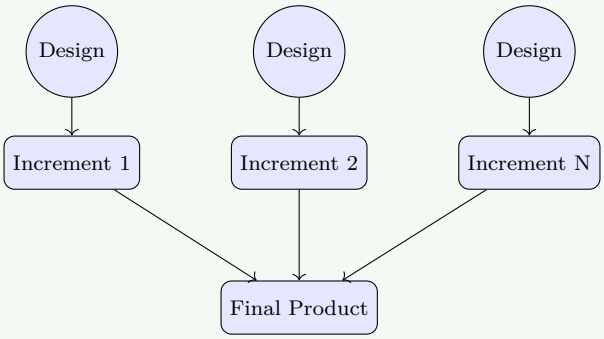

Figure 6. Incremental Model Concept

Table 8. Use Comparison

Feature	Waterfall	Incremental
Flexibility	Low	High
Risk	High	Low
Delivery	End of project	Multiple deliveries

Mnemonic

"Water Falls Once, Increments Build Multiple"

Question 2(a OR) [3 marks]

Describe advantage and disadvantage of Incremental Model.

Solution

Incremental Model Analysis:

Table 9. Incremental Model Pros and Cons

Advantages	Disadvantages
Early delivery of working software	**Total cost** may be higher
Easier testing of small increments	**System architecture** issues
Reduced risk through early feedback	**Management complexity** increases

Mnemonic

"Early Easier Reduced vs Total System Management"

Question 2(b OR) [4 marks]

Write concept of Rapid Application Development (RAD) and explain it.

Solution

RAD emphasizes rapid prototyping and quick feedback over planning and testing.
RAD Components:

Table 10. RAD Process

Phase	Duration	Activities	Output
Business Modeling	Short	Define information flow	Business requirements
Data Modeling	Short	Define data objects	Data models
Process Modeling	Short	Define processing functions	Process descriptions
Application Generation	Short	Use tools to create	Working application
Testing & Turnover	Short	Test and deliver	Final system

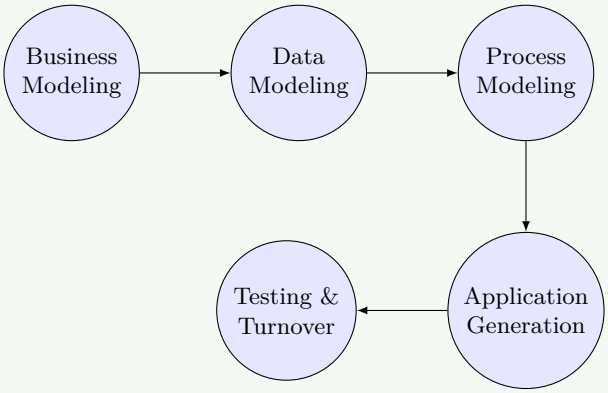

Figure 7. RAD Flow

Mnemonic

"Business Data Process Application Testing"

Question 2(c OR) [7 marks]

Design and describe Spiral Model and give advantage and disadvantage.

Solution

Spiral Model combines iterative development with systematic risk analysis.

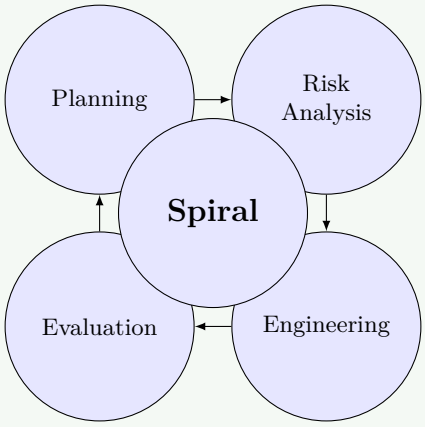

Figure 8. Spiral Model Quadrants

Spiral Quadrants:

Table 11. Quadrant Details

Quadrant	Activity	Purpose
Planning	Objective setting	Define requirements and constraints
Risk Analysis	Risk assessment	Identify and resolve risks
Engineering	Development	Build and test the product
Evaluation	Customer assessment	Evaluate results and plan next iteration

Advantages vs Disadvantages:

Table 12. Spiral Pros and Cons

Advantages	Disadvantages
High risk projects handled well	**Complex management** required
Good for large applications	**Expensive** for small projects
Customer involved throughout	**Risk analysis expertise** needed

Mnemonic

"Plan Risk Engineer Evaluate: Quadrants" "High Good Customer vs Complex Expensive Risk"

Question 3(a) [3 marks]

Illustrate importance of SRS

Solution

SRS (Software Requirements Specification) is crucial foundation document for software development.
Importance Table:

Table 13. SRS Benefits

Aspect	Importance	Benefit
Communication	Stakeholder understanding	Clear expectations
Contract	Legal agreement	Dispute resolution
Testing basis	Validation criteria	Quality assurance

Mnemonic

"Communication Contract Testing"

Question 3(b) [4 marks]

Specify characteristics of good & bad SRS

Solution

SRS Quality Characteristics:

Table 14. Good vs Bad SRS

Good SRS	Bad SRS
Complete - All requirements covered	**Incomplete** - Missing requirements
Consistent - No contradictions	**Inconsistent** - Conflicting statements
Unambiguous - Clear meaning	**Ambiguous** - Multiple interpretations
Verifiable - Can be tested	**Unverifiable** - Cannot be validated

Additional Good Characteristics:

- **Modifiable**: Easy to change and maintain
- **Traceable**: Links to source and design

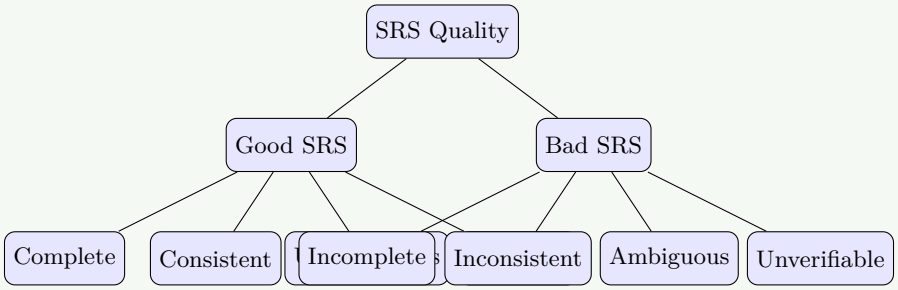

Figure 9. SRS Characteristics

Mnemonic

"Complete Consistent Unambiguous Verifiable vs Incomplete Inconsistent Ambiguous Unverifiable"

Question 3(c) [7 marks]

Classify Types of Requirements in SRS

Solution

Software requirements are classified into two main categories.
(i) Functional Requirements: Define what the system should do - specific behaviors and functions.

Table 15. Functional Requirements

Type	Description	Example
Business Rules	Core business logic	”Calculate tax based on income bracket”
User Actions	System responses	”Login with username/password”
Data Processing	Information handling	”Generate monthly sales report”
External Interfaces	System interactions	”Connect to payment gateway”

(ii) Non-functional Requirements: Define how the system should perform - quality attributes and constraints.

Table 16. Non-functional Requirements

Category	Requirement	Example	Measurement
Performance	Response time	”Page load < 3 seconds”	Time metrics
Security	Data protection	”Encrypt user passwords”	Security standards
Reliability	System uptime	”99.9% availability”	Failure rates
Usability	User experience	”Max 3 clicks to checkout”	User metrics
Scalability	Growth capacity	”Support 10,000 users”	Load capacity

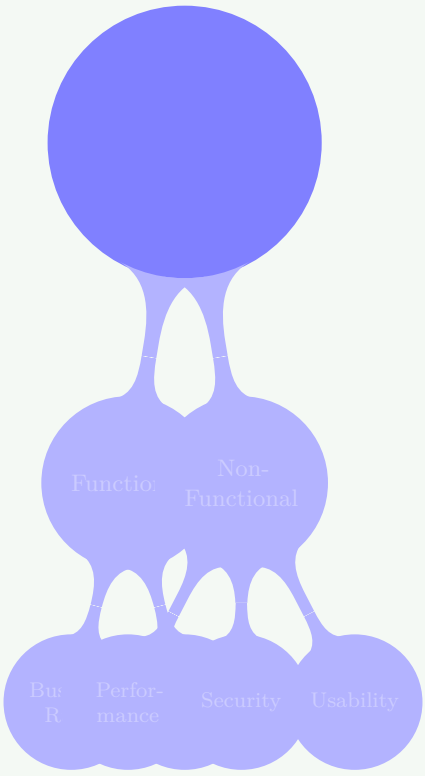

Figure 10. Requirement Types

Mnemonic

“Functional = What, Non-Functional = How”

Question 3(a OR) [3 marks]

Describe skill to manage software projects

Solution

Project management requires diverse skill set for successful software delivery.
Essential Skills:

Table 17. PM Skills

Skill Category	Description	Application
Technical	Understanding technology	Architecture decisions
Leadership	Team motivation	Conflict resolution
Communication	Stakeholder interaction	Status reporting

Mnemonic

"Technical Leadership Communication"

Question 3(b OR) [4 marks]

Briefly give the Responsibility of software project Manager.

Solution

Software Project Manager oversees entire project lifecycle and ensures successful delivery.
Key Responsibilities:

Table 18. PM Responsibilities

Area	Responsibility	Activities
Planning	Project roadmap	Schedule, budget, resource allocation
Execution	Team coordination	Task assignment, progress monitoring
Quality	Standard compliance	Code reviews, testing oversight
Communication	Stakeholder updates	Status reports, risk communication

Additional Duties:

- **Risk Management**: Identify and mitigate project risks
- **Team Development**: Mentor team members and resolve conflicts

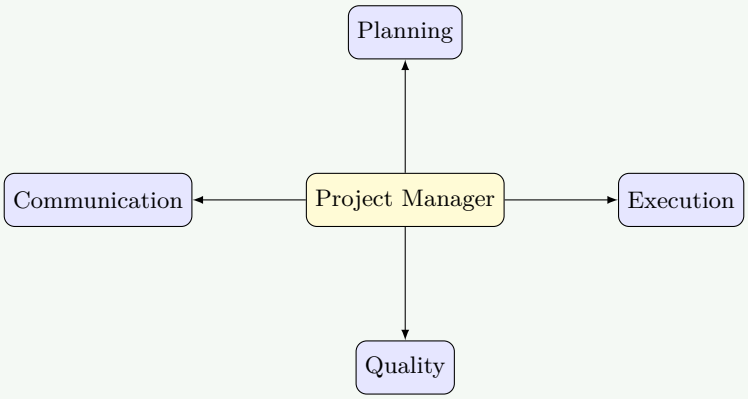

Figure 11. PM Functions

Mnemonic

"Plan Execute Quality Communicate Risk Team"

Question 3(c OR) [7 marks]

Compare PERT chart – Gantt chart side by side.

Solution

Both charts are project management tools but serve different purposes and have distinct characteristics.
Detailed Comparison:

Table 19. PERT vs Gantt

Aspect	PERT Chart	Gantt Chart
Purpose	Show task dependencies	Show project timeline
Structure	Network diagram	Bar chart
Focus	Critical path analysis	Schedule visualization
Time Display	Estimated durations	Actual dates
Dependencies	Explicit arrows	Implicit connections
Best For	Complex projects	Simple scheduling

Visual Representation:

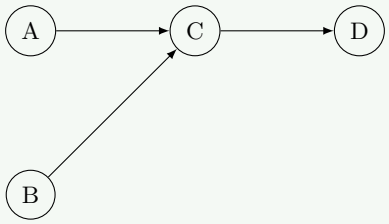

Figure 12. PERT Chart Concept

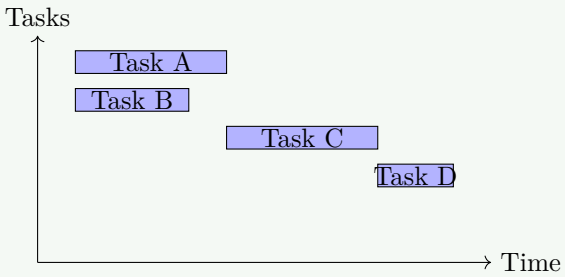

Figure 13. Gantt Chart Concept

When to Use:

Table 20. Usage Guide

Scenario	PERT	Gantt
Project Type	Research & Development	Construction, Software
Uncertainty	High uncertainty	Well-defined tasks
Audience	Technical team	Management, Clients

Advantages Comparison:
- **PERT**: Critical path, Flexible timing, Risk analysis
- **Gantt**: Easy to understand, Progress tracking, Resource allocation

Mnemonic

"PERT = Path, Gantt = Bars"

Question 4(a) [3 marks]

Give steps of Project Monitoring and control process

Solution

Project monitoring ensures project stays on track through systematic observation and corrective actions.
Monitoring Steps:

Table 21. Process Steps

Step	Activity	Purpose
Track Progress	Measure actual vs planned	Identify deviations
Assess Quality	Review deliverables	Ensure standards
Take Action	Implement corrections	Maintain alignment

Mnemonic

"Track Assess Take"

Question 4(b) [4 marks]

Discuss i)Risk Assessment ii)Risk Mitigation

Solution

(i) Risk Assessment: Process of identifying and evaluating potential project risks.

Table 22. Assessment Components

Assessment Type	Method	Output
Risk Identification	Brainstorming, checklists	Risk list
Risk Analysis	Probability × Impact	Risk priority
Risk Evaluation	Risk matrix	Action priorities

(ii) Risk Mitigation: Strategies to reduce risk impact and probability.

Table 23. Mitigation Strategies

Strategy	Description	Example
Avoidance	Eliminate risk source	Change technology
Reduction	Minimize impact	Add testing
Transfer	Shift risk to others	Insurance, outsourcing
Acceptance	Live with risk	Contingency planning

Mnemonic

"Avoid Reduce Transfer Accept"

Question 4(c) [7 marks]

Define project risk and how Manage Risk Management it.

Solution

Project Risk is an uncertain event that, if occurs, has positive or negative effect on project objectives.
Risk Characteristics:

Table 24. Characteristics

Characteristic	Description	Example
Uncertainty	May or may not occur	Technology failure
Impact	Affects project parameters	Cost, schedule, quality
Probability	Likelihood of occurrence	30% chance of delay

Risk Management Process:

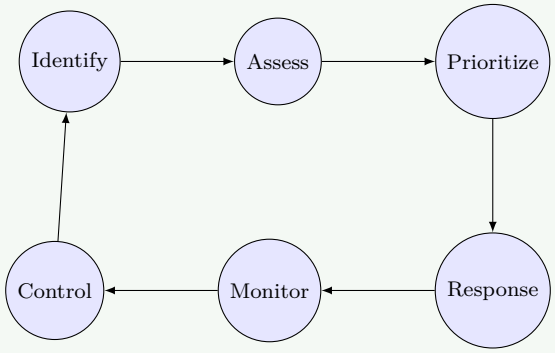

Figure 14. Management Loop

Risk Management Steps:

Table 25. Process Details

Step	Activities	Tools	Output
Identification	Brainstorming	Checklists, SWOT	Risk register
Assessment	Prob/Impact analysis	Risk matrix	Risk ratings
Response	Develop strategies	Response templates	Action plans
Monitoring	Track indicators	Dashboards	Status reports

Risk Response Strategies:
- **Negative Risks**: Avoid, Transfer, Mitigate, Accept
- **Positive Risks**: Exploit, Share, Enhance, Accept

Mnemonic

"Identify Assess Respond Monitor + Avoid Transfer Mitigate Accept"

Question 4(a OR) [3 marks]

Describe Software design process and explain Design methodologies.

Solution

Software design transforms requirements into blueprint for implementation through systematic approach.
Design Process:

Table 26. Process Structure

Phase	Activity	Output
Analysis	Understand requirements	Problem definition
Architecture	High-level structure	System architecture
Detailed Design	Component specification	Design documents

Mnemonic

"Analysis Architecture Detail"

Question 4(b OR) [4 marks]

Compare Cohesion and Coupling side by side.

Solution

Both concepts measure module design quality but focus on different aspects.
Comprehensive Comparison:

Table 27. Cohesion vs Coupling

Aspect	Cohesion	Coupling
Definition	Degree of relatedness within module	Degree of interdependence between modules
Goal	High cohesion desired	Low coupling desired
Focus	Internal module structure	Inter-module relationships
Quality	Stronger = Better	Weaker = Better

Types Comparison (Best to Worst):
- **Cohesion**: Functional, Sequential, Communicational, Procedural, Temporal, Logical, Coincidental
- **Coupling**: Data, Stamp, Control, External, Common, Content

Impact on Design: High cohesion and low coupling leads to better maintainability, reusability, and testing.

Mnemonic

"Cohesion = Inside Strong, Coupling = Between Weak"

Question 4(c OR) [7 marks]

Sketch Data Flow Diagram with levels and explain.

Solution

Data Flow Diagram (DFD) shows how data moves through system using graphical notation with multiple levels of detail.
DFD Levels:

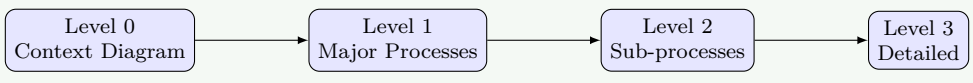

Figure 15. DFD Levels

Level Descriptions:

Table 28. Level Details

Level	Scope	Purpose	Detail
Level 0	Entire system	System boundary	Single process
Level 1	Major functions	High-level processes	5-7 processes
Level 2	Sub-functions	Process breakdown	Detailed view
Level 3+	Fine details	Implementation level	Very specific

Example - Level 1 DFD:

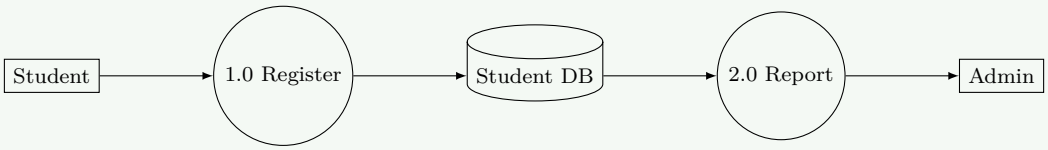

Figure 16. Level 1 Example

Benefits: Abstraction, Decomposition, Verification.

Mnemonic

"Context Major Sub Fine + Process Entity Store Flow"

Question 5(a) [3 marks]

Give Characteristics of good UI.

Solution

Good User Interface design ensures effective user interaction with software system.
UI Characteristics:

Table 29. Key Features

Characteristic	Description	Benefit
Simple	Easy to understand	Reduced learning curve
Consistent	Uniform behavior	Predictable interaction
Responsive	Quick feedback	User satisfaction

Mnemonic

"Simple Consistent Responsive"

Question 5(b) [4 marks]

Briefly explain Unit testing

Solution

Unit Testing verifies individual software components in isolation to ensure correct functionality.
Unit Testing Overview:

Table 30. Testing Scope

Aspect	Description	Purpose
Scope	Individual modules	Component verification
Isolation	Test in isolation	Independent validation
Automation	Automated execution	Efficient testing
Early Detection	Find bugs early	Cost-effective

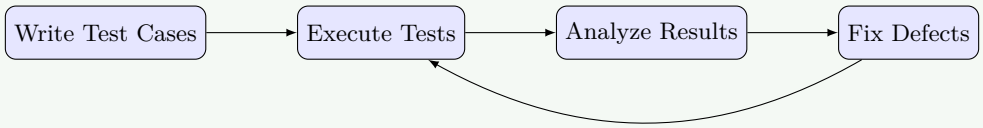

Figure 17. Unit Testing Cycle

Benefits: Early bug detection, Code quality improvement, Regression testing.

Mnemonic

"Scope Isolation Automation Early"

Question 5(c) [7 marks]

Draw activity diagrams of the train reservation system, explain each step.

Solution

Activity Diagram shows workflow of train reservation system from user request to ticket confirmation.

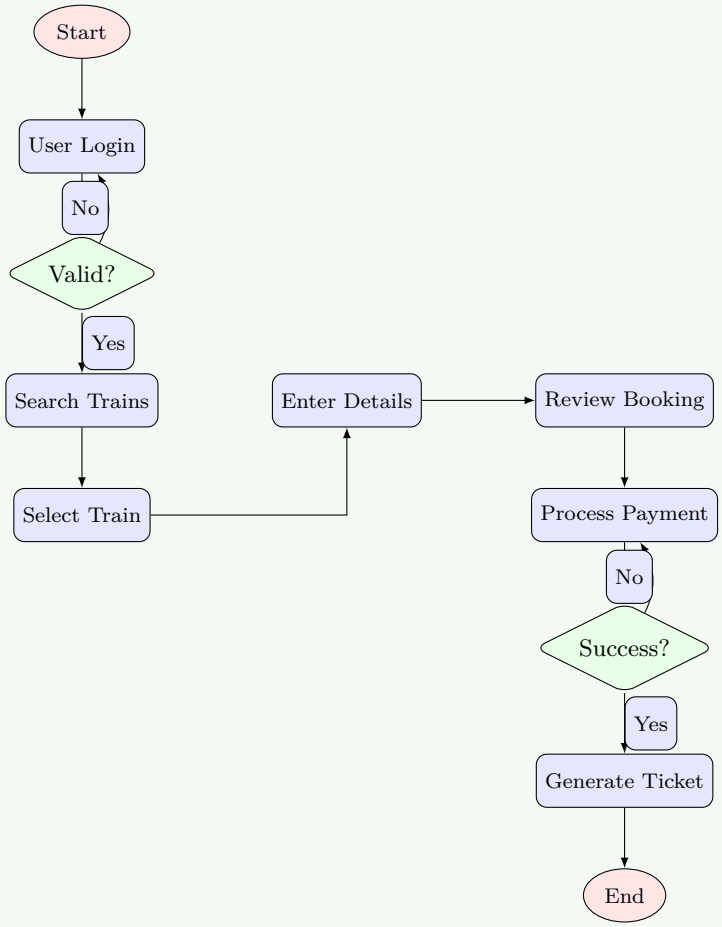

Figure 18. Train Reservation Activity

Step Explanation:

- **Login**: User authentication.
- **Search**: Find trains for route/date.
- **Selection**: Choose train and seats.
- **Details**: Enter passenger info.
- **Payment**: Process transaction.
- **Ticket**: Generate and send confirmation.

Mnemonic

"Login Search Select Choose Enter Review Pay Generate Send"

Question 5(a OR) [3 marks]

Compare Verification, Validation side by side.

Solution

Both are quality assurance activities but focus on different aspects of correctness.
Verification vs Validation:

Table 31. Comparison

Aspect	Verification	Validation
Question	”Are we building right?”	”Are we building right thing?”
Focus	Process correctness	Product correctness
Method	Reviews, inspections	Testing, user feedback

Mnemonic

“Verification = Right Process, Validation = Right Product”

Question 5(b OR) [4 marks]

Define Testing describe any two testing type.

Solution

Testing is process of evaluating software to detect errors and ensure it meets requirements.
Two Testing Types:

Table 32. Black Box vs White Box

Aspect	Black Box	White Box
Approach	Unknown internal structure	Known code structure
Focus	Functional requirements	Internal logic
Tester	User acceptance	Developer unit testing

Mnemonic

“Black = External, White = Internal”

Question 5(c OR) [7 marks]

Describe each Coding standards and guidelines.

Solution

Coding Standards are rules for writing consistent, maintainable code.
Major Categories:
1. **Naming Conventions**: camelCase for variables, PascalCase for classes.
2. **Code Structure**: Consistent indentation, line length limits.
3. **Organization**: Single responsibility, small functions.
4. **Documentation**: Header comments, meaningful inline comments.
5. **Error Handling**: Graceful exception handling, logging.
6. **Performance**: Avoid memory leaks, efficient algorithms.

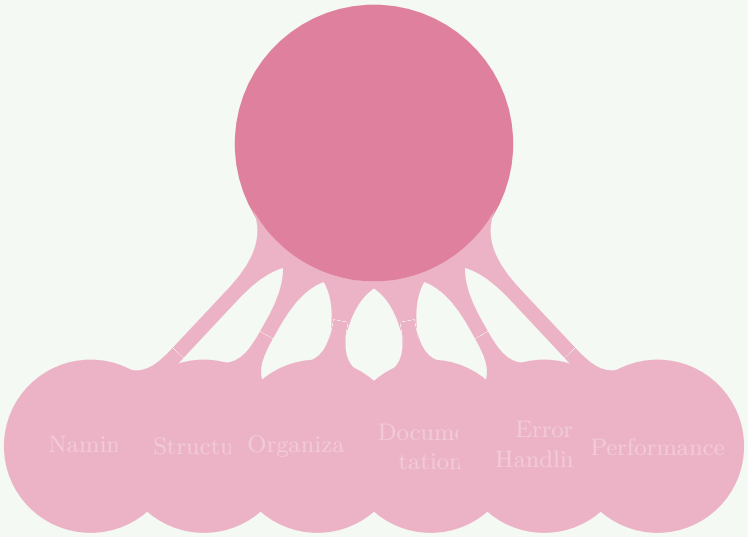

Figure 19. Standard Categories

Benefits: Improved readability, consistency, maintainability, and quality.

Mnemonic

"Name Structure Organize Document Handle Perform Review"