

# 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”

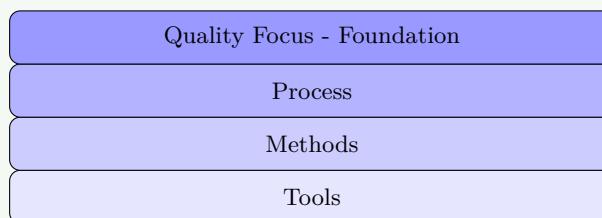
## Question 1(b) [4 marks]

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
<b>Quality Focus</b>	Foundation	Emphasis on delivering quality products
<b>Process</b>	Framework	Defines how software development is done
<b>Methods</b>	Techniques	Specific ways to perform activities
<b>Tools</b>	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.

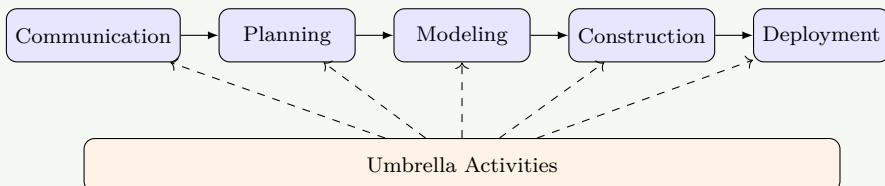
**Framework Activities:**

**Table 3.** Framework Activities

Activity	Purpose	Key Tasks
<b>Communication</b>	Understand requirements	Stakeholder interaction, requirement gathering
<b>Planning</b>	Create roadmap	Estimation, scheduling, risk assessment
<b>Modeling</b>	Create blueprints	Analysis and design models
<b>Construction</b>	Build software	Coding and testing
<b>Deployment</b>	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.

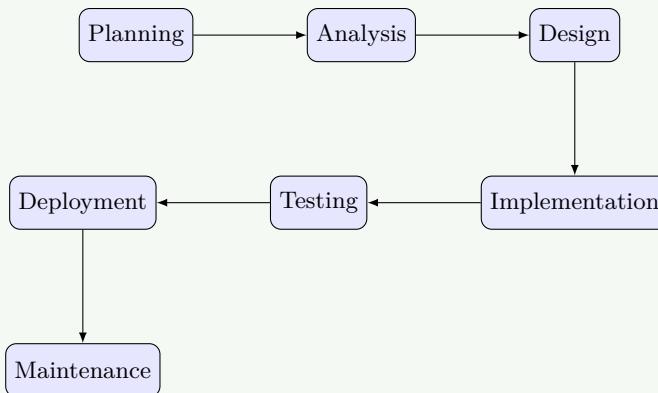
### Solution

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

#### SDLC Phases:

**Table 4.** SDLC Phases

Phase	Purpose	Key Activities	Deliverables
<b>Planning</b>	Define scope	Feasibility study, resource allocation	Project plan
<b>Analysis</b>	Gather requirements	Requirement collection, documentation	SRS document
<b>Design</b>	Create architecture	System design, database design	Design documents
<b>Implementation</b>	Write code	Programming, unit testing	Source code
<b>Testing</b>	Verify quality	System testing, bug fixing	Test reports
<b>Deployment</b>	Release software	Installation, user training	Live system
<b>Maintenance</b>	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
<b>Early feedback</b> from users	<b>Time consuming</b> development process
<b>Reduced risk of failure</b>	<b>Cost increase</b> due to iterations
<b>Better understanding</b> of requirements	<b>Scope creep</b> 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.

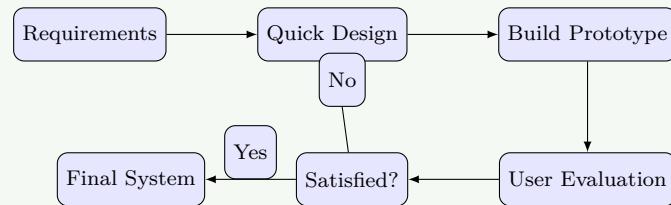
### Solution

Prototyping Model creates working model of software early in development process.

**When to Use:**

**Table 6.** Usage Scenarios

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



**Figure 4.** Prototyping Process

### Mnemonic

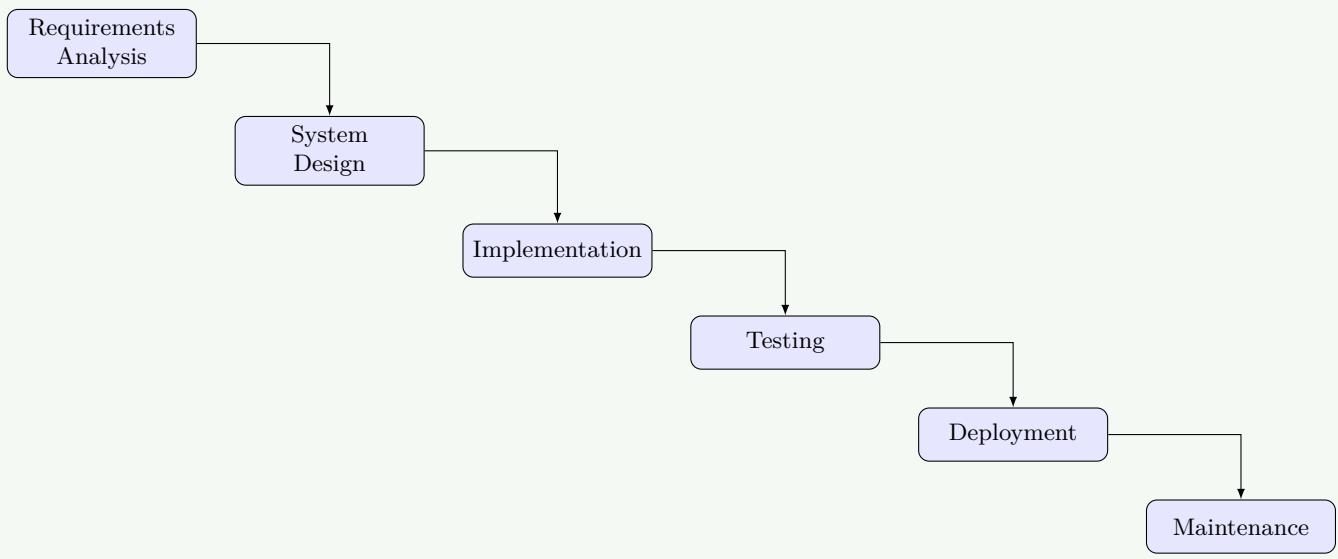
“Requirements Quick Build User Satisfied Final”

## Question 2(c) [7 marks]

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

### Solution

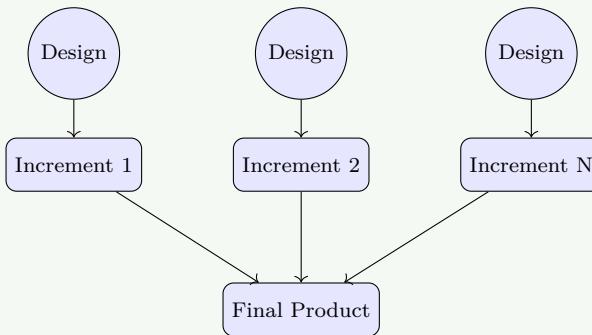
**(I) Waterfall Model:** Linear sequential approach where each phase must complete before next begins.



**Figure 5.** Waterfall Model**Table 7.** Waterfall Characteristics

Characteristics	Description
<b>Sequential</b>	One phase at a time
<b>Documentation driven</b>	Heavy documentation
<b>Suitable for</b>	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
<b>Flexibility</b>	Low	High
<b>Risk</b>	High	Low
<b>Delivery</b>	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
<b>Early delivery</b> of working software	<b>Total cost</b> may be higher
<b>Easier testing</b> of small increments	<b>System architecture</b> issues
<b>Reduced risk</b> through early feedback	<b>Management complexity</b> 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.

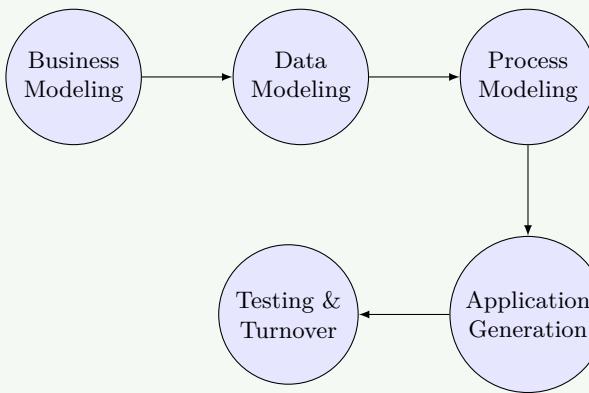
### Solution

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

#### RAD Components:

**Table 10.** RAD Process

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



**Figure 7.** RAD Flow

### Mnemonic

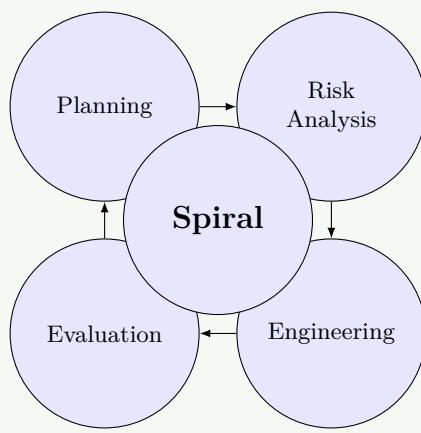
“Business Data Process Application Testing”

## 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

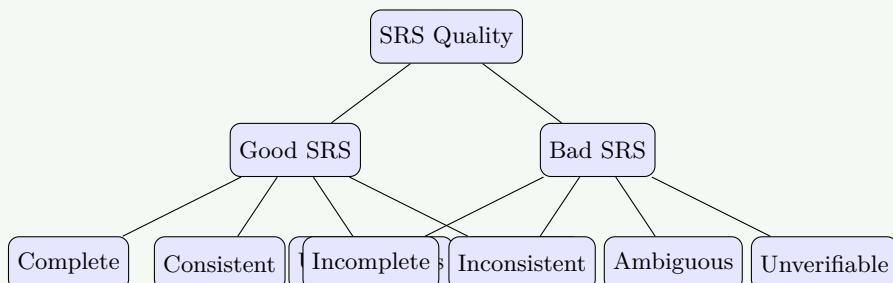
**Solution****SRS Quality Characteristics:**

**Table 14.** Good vs Bad SRS

Good SRS	Bad SRS
<b>Complete</b> - All requirements covered	<b>Incomplete</b> - Missing requirements
<b>Consistent</b> - No contradictions	<b>Inconsistent</b> - Conflicting statements
<b>Unambiguous</b> - Clear meaning	<b>Ambiguous</b> - Multiple interpretations
<b>Verifiable</b> - Can be tested	<b>Unverifiable</b> - 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.

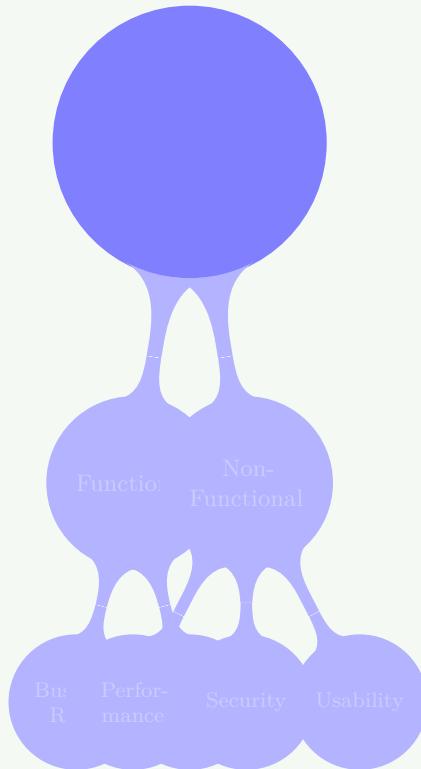
**Table 15.** Functional Requirements

Type	Description	Example
<b>Business Rules</b>	Core business logic	"Calculate tax based on income bracket"
<b>User Actions</b>	System responses	"Login with username/password"
<b>Data Processing</b>	Information handling	"Generate monthly sales report"
<b>External Interfaces</b>	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
<b>Performance</b>	Response time	"Page load < 3 seconds"	Time metrics
<b>Security</b>	Data protection	"Encrypt user passwords"	Security standards
<b>Reliability</b>	System uptime	"99.9% availability"	Failure rates
<b>Usability</b>	User experience	"Max 3 clicks to checkout"	User metrics
<b>Scalability</b>	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
<b>Technical</b>	Understanding technology	Architecture decisions
<b>Leadership</b>	Team motivation	Conflict resolution
<b>Communication</b>	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.

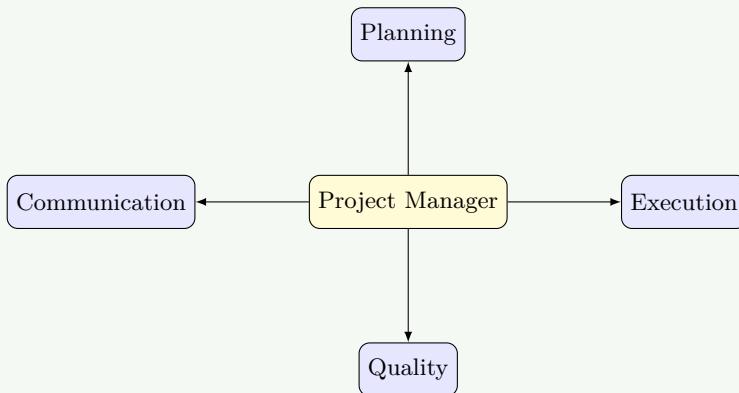
#### Key Responsibilities:

**Table 18.** PM Responsibilities

Area	Responsibility	Activities
<b>Planning</b>	Project roadmap	Schedule, budget, resource allocation
<b>Execution</b>	Team coordination	Task assignment, progress monitoring
<b>Quality</b>	Standard compliance	Code reviews, testing oversight
<b>Communication</b>	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

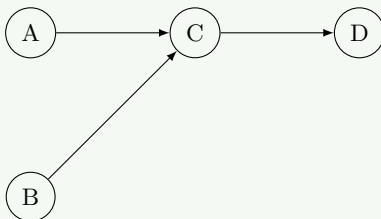
Both charts are project management tools but serve different purposes and have distinct characteristics.

#### Detailed Comparison:

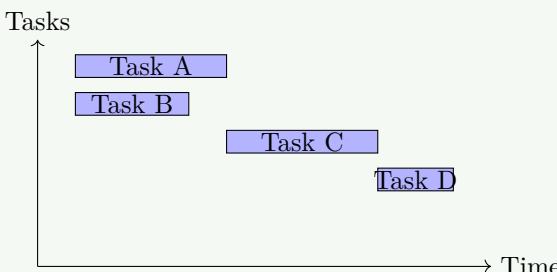
**Table 19.** PERT vs Gantt

Aspect	PERT Chart	Gantt Chart
<b>Purpose</b>	Show task dependencies	Show project timeline
<b>Structure</b>	Network diagram	Bar chart
<b>Focus</b>	Critical path analysis	Schedule visualization
<b>Time Display</b>	Estimated durations	Actual dates
<b>Dependencies</b>	Explicit arrows	Implicit connections
<b>Best For</b>	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
<b>Project Type</b>	Research & Development	Construction, Software
<b>Uncertainty</b>	High uncertainty	Well-defined tasks
<b>Audience</b>	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
<b>Track Progress</b>	Measure actual vs planned	Identify deviations
<b>Assess Quality</b>	Review deliverables	Ensure standards
<b>Take Action</b>	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
<b>Risk Identification</b>	Brainstorming, checklists	Risk list
<b>Risk Analysis</b>	Probability × Impact	Risk priority
<b>Risk Evaluation</b>	Risk matrix	Action priorities

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

**Table 23.** Mitigation Strategies

Strategy	Description	Example
<b>Avoidance</b>	Eliminate risk source	Change technology
<b>Reduction</b>	Minimize impact	Add testing
<b>Transfer</b>	Shift risk to others	Insurance, outsourcing
<b>Acceptance</b>	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

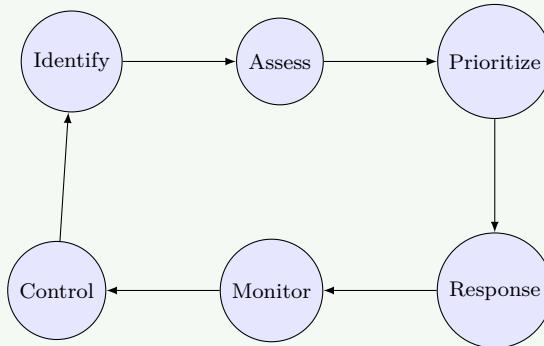
Project Risk is an uncertain event that, if occurs, has positive or negative effect on project objectives.

### Risk Characteristics:

**Table 24.** Characteristics

Characteristic	Description	Example
<b>Uncertainty</b>	May or may not occur	Technology failure
<b>Impact</b>	Affects project parameters	Cost, schedule, quality
<b>Probability</b>	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
<b>Identification</b>	Brainstorming	Checklists, SWOT	Risk register
<b>Assessment</b>	Prob/Impact analysis	Risk matrix	Risk ratings
<b>Response</b>	Develop strategies	Response templates	Action plans
<b>Monitoring</b>	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
<b>Definition</b>	Degree of relatedness within module	Degree of interdependence between modules
<b>Goal</b>	High cohesion desired	Low coupling desired
<b>Focus</b>	Internal module structure	Inter-module relationships
<b>Quality</b>	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**

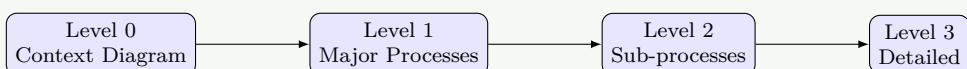
“Cohesion = Inside Strong, Coupling = Between Weak”

**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:**

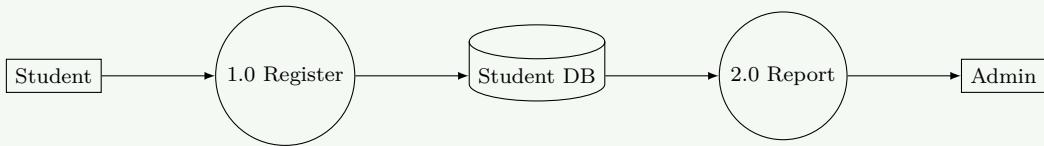
**Figure 15.** DFD Levels

**Level Descriptions:**

**Table 28.** Level Details

Level	Scope	Purpose	Detail
<b>Level 0</b>	Entire system	System boundary	Single process
<b>Level 1</b>	Major functions	High-level processes	5-7 processes
<b>Level 2</b>	Sub-functions	Process breakdown	Detailed view
<b>Level 3+</b>	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
<b>Simple</b>	Easy to understand	Reduced learning curve
<b>Consistent</b>	Uniform behavior	Predictable interaction
<b>Responsive</b>	Quick feedback	User satisfaction

#### Mnemonic

“Simple Consistent Responsive”

## Question 5(b) [4 marks]

Briefly explain Unit testing

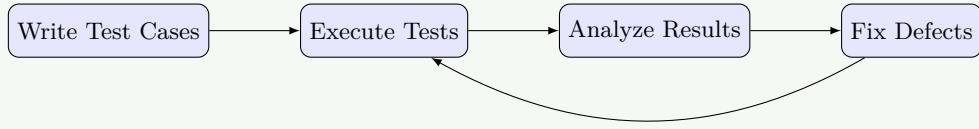
#### Solution

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

#### Unit Testing Overview:

**Table 30.** Testing Scope

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



**Figure 17.** Unit Testing Cycle

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

#### Mnemonic

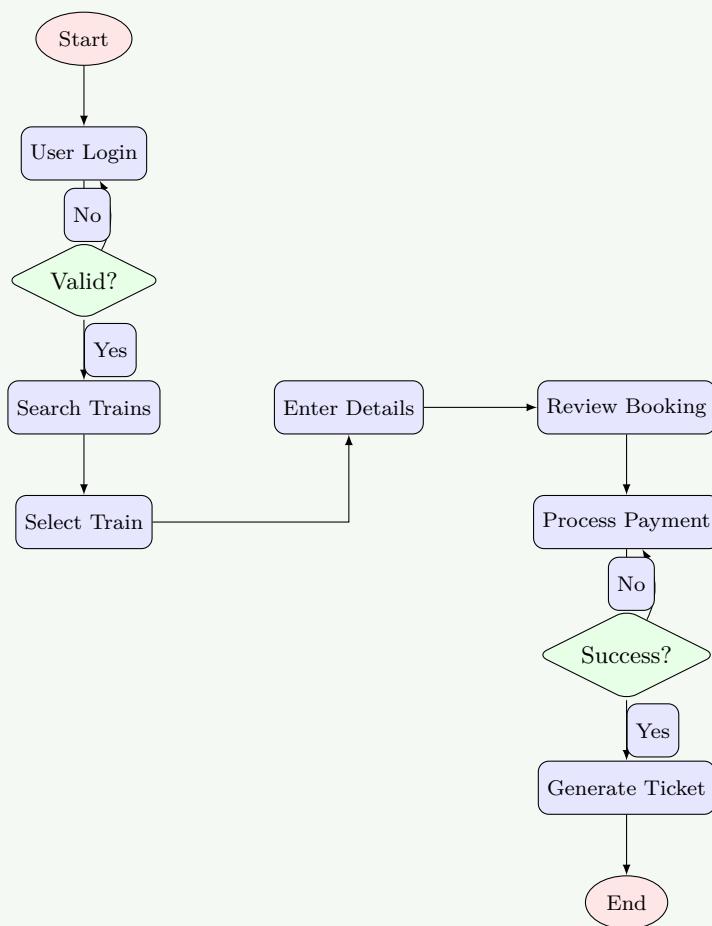
“Scope Isolation Automation Early”

### 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]**

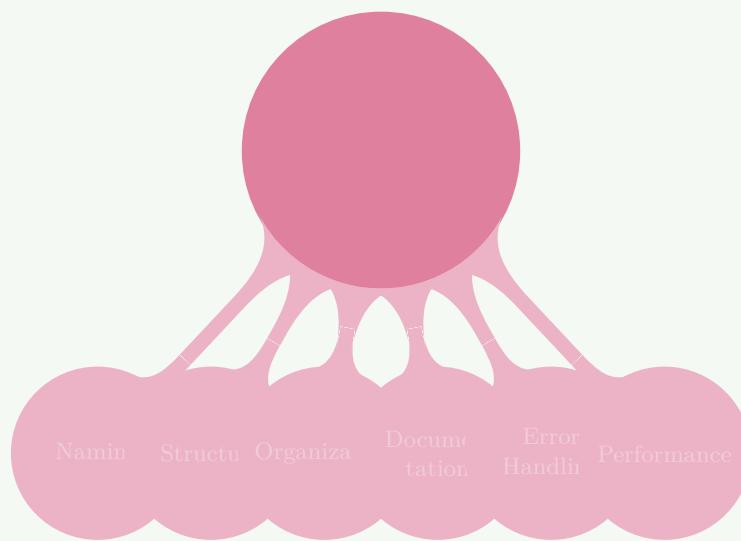
Describe each Coding standards and guidelines.

**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”