

# Software Engineering (4353202) - Winter 2024 Solution

Milav Dabgar

November 25, 2024

## Question 1(a) [3 marks]

Define software and explain its characteristics.

### Solution

**Software** is a collection of computer programs, procedures, and documentation that performs tasks on a computer system.

**Table 1.** Software Characteristics

Characteristic	Description
<b>Intangible</b>	Cannot be touched, only experienced
<b>Developed</b>	Engineered, not manufactured
<b>Maintainable</b>	Can be modified and updated
<b>Reliable</b>	Should work consistently
<b>Efficient</b>	Uses resources optimally

- **Key point:** Software = Programs + Documentation + Procedures

### Mnemonic

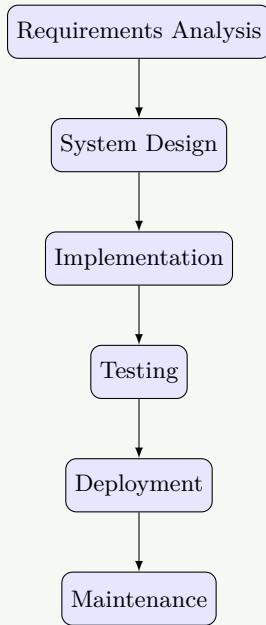
“I Don’t Make Reliable Electronics (Intangible, Developed, Maintainable, Reliable, Efficient)”

## Question 1(b) [4 marks]

Explain classical waterfall model.

### Solution

**Waterfall Model** is a linear sequential software development approach where each phase must be completed before the next begins.

**Figure 1.** Classical Waterfall Model**Key Features:**

- **Sequential phases:** No overlap between phases
- **Documentation-driven:** Heavy documentation at each phase
- **Simple structure:** Easy to understand and manage
- **Fixed requirements:** Changes are difficult once started

**Mnemonic**

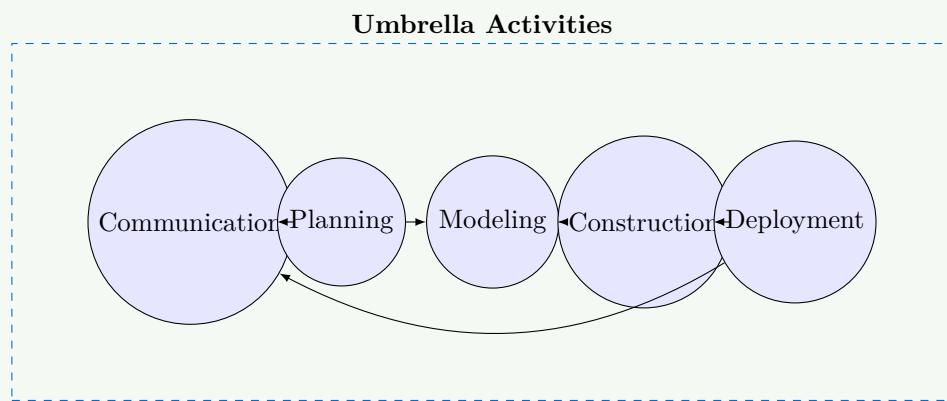
“Real Systems Include Testing, Deployment, Maintenance”

**Question 1(c) [7 marks]**

Explain software process framework and umbrella activities.

**Solution**

**Software Process Framework** provides the foundation for complete software engineering process by identifying key process areas.

**Figure 2.** Process Framework & Umbrella Activities

**Table 2.** Framework Activities vs Umbrella Activities

Framework Activities	Umbrella Activities
Communication	Software project tracking
Planning	Risk management
Modeling	Quality assurance
Construction	Technical reviews
Deployment	Configuration management

**Framework Activities:**

- **Communication:** Gather requirements from stakeholders
- **Planning:** Create project plan and schedule
- **Modeling:** Create design models
- **Construction:** Code generation and testing
- **Deployment:** Software delivery and feedback

**Umbrella Activities** run throughout the project:

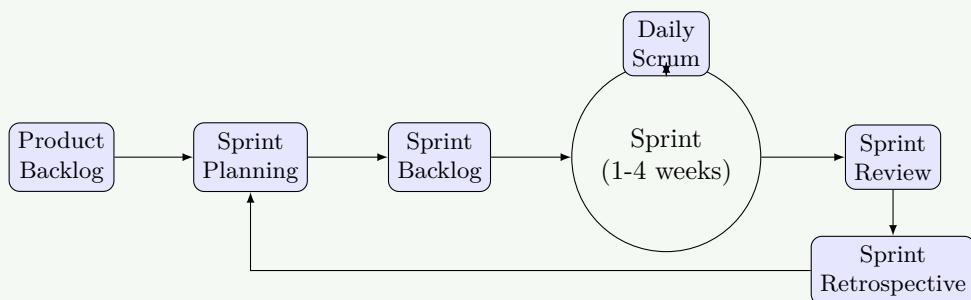
- **Project tracking:** Monitor progress
- **Risk management:** Identify and control risks
- **Quality assurance:** Ensure quality standards
- **Configuration management:** Control changes

**Mnemonic**

“Can People Make Construction Deploy (Communication, Planning, Modeling, Construction, Deployment)”

**Question 1(c) OR [7 marks]****Write a short note on SCRUM.****Solution**

SCRUM is an agile framework for managing software development projects using iterative and incremental practices.

**Figure 3.** SCRUM Process Flow**Table 3.** SCRUM Roles and Artifacts

Component	Description
<b>Product Owner</b>	Defines requirements and priorities
<b>Scrum Master</b>	Facilitates process and removes obstacles
<b>Development Team</b>	Self-organizing team that builds product
<b>Product Backlog</b>	Prioritized list of features
<b>Sprint Backlog</b>	Tasks selected for current sprint

**Key Events:**

- **Sprint Planning:** Select work for upcoming sprint
- **Daily Scrum:** 15-minute daily synchronization
- **Sprint Review:** Demonstrate completed work
- **Sprint Retrospective:** Reflect and improve process

**Benefits:** Fast delivery, flexibility, continuous improvement, customer collaboration

**Mnemonic**

“People Sprint Daily Reviewing Retrospectively”

**Question 2(a) [3 marks]**

Explain characteristic of good SRS.

**Solution**

**SRS (Software Requirements Specification)** document should have specific qualities to be effective.

**Table 4.** Good SRS Characteristics

Characteristic	Meaning
<b>Complete</b>	All requirements included
<b>Consistent</b>	No contradictory requirements
<b>Unambiguous</b>	Clear and single interpretation
<b>Verifiable</b>	Can be tested and validated
<b>Modifiable</b>	Easy to change when needed

- **Complete:** Contains all functional and non-functional requirements
- **Consistent:** No conflicts between different requirements
- **Unambiguous:** Each requirement has only one interpretation

**Mnemonic**

“Complete Computers Use Verified Modifications”

**Question 2(b) [4 marks]**

Describe advantage and disadvantages of prototype model.

**Solution**

**Prototype Model** creates a working model of software to understand requirements better.

**Table 5.** Prototype Model - Pros and Cons

Advantages	Disadvantages
<b>Better requirement understanding</b>	<b>Time consuming</b>
<b>User involvement</b>	<b>Cost increase</b>
<b>Early error detection</b>	<b>Incomplete analysis</b>
<b>User satisfaction</b>	<b>Prototype confusion</b>

**Advantages:**

- **Clear requirements:** Users see working model
- **Early feedback:** Reduces final product risks
- **User involvement:** Better user acceptance

**Disadvantages:**

- **Extra time:** Building prototype takes time
- **Additional cost:** Resources needed for prototype
- **Scope creep:** Users may expect prototype features

### Mnemonic

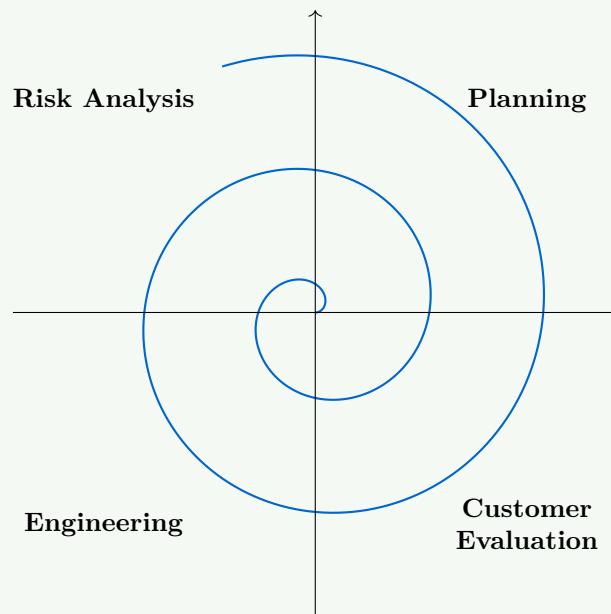
“Better Users Experience vs Time Costs Increase”

## Question 2(c) [7 marks]

Design and describe Spiral model and give advantages and disadvantages.

### Solution

**Spiral Model** combines iterative development with systematic risk management through repeated cycles.



**Figure 4.** Spiral Model

**Table 6.** Spiral Model Phases

Phase	Activities
Planning	Requirements gathering, resource planning
Risk Analysis	Identify and resolve risks
Engineering	Development and testing
Customer Evaluation	Customer reviews and feedback

**Advantages:**

- **Risk management:** Early risk identification
- **Flexibility:** Accommodates changes easily
- **Customer involvement:** Regular customer feedback
- **Quality focus:** Continuous testing and validation

**Disadvantages:**

- **Complex management:** Difficult to manage
- **High cost:** Expensive due to risk analysis
- **Time consuming:** Long development cycles
- **Risk expertise needed:** Requires risk assessment skills

**Best for:** Large, complex, high-risk projects

#### Mnemonic

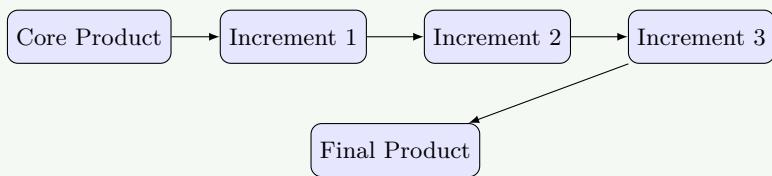
“Plan Risks Engineering Customer”

## Question 2(a) OR [3 marks]

Explain Incremental model.

#### Solution

**Incremental Model** delivers software in small, functional pieces called increments.



**Figure 5.** Incremental Delivery

#### Key Features:

- **Partial implementation:** Each increment adds functionality
- **Early delivery:** Core features delivered first
- **Parallel development:** Multiple increments can be developed simultaneously

**Table 7.** Incremental Model Characteristics

Aspect	Description
<b>Delivery</b>	Multiple releases
<b>Functionality</b>	Grows with each increment
<b>Risk</b>	Reduced through early delivery
<b>Feedback</b>	Continuous user feedback

#### Mnemonic

“Deliver Functionality Reducing Feedback”

## Question 2(b) OR [4 marks]

Write concept of Rapid Application Development model and explain it.

#### Solution

**RAD (Rapid Application Development)** emphasizes rapid prototyping and quick feedback over extensive planning.

**Table 8.** RAD Model Phases

Phase	Duration	Activities
<b>Business Modeling</b>	Short	Define business functions
<b>Data Modeling</b>	Short	Define data requirements
<b>Process Modeling</b>	Short	Convert data to business info
<b>Application Generation</b>	Short	Use tools to create software
<b>Testing &amp; Turnover</b>	Short	Test and deploy

**Key Concepts:**

- **Reusable components:** Pre-built components speed development
- **Powerful tools:** CASE tools and code generators
- **Small teams:** 2-6 people per team
- **Time-boxed:** Strict time limits (60-90 days)

**Requirements for RAD:**

- **Well-defined business requirements**
- **User involvement** throughout process
- **Skilled developers** familiar with RAD tools

**Mnemonic**

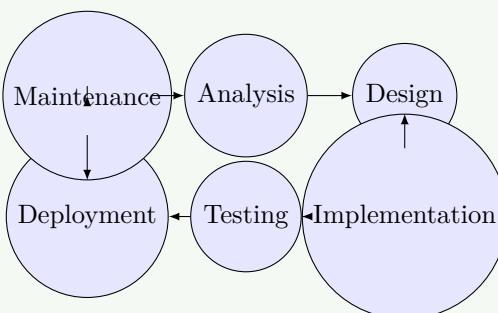
“Business Data Process Application Testing”

## Question 2(c) OR [7 marks]

Define SDLC and explain each phase.

**Solution**

**SDLC (Software Development Life Cycle)** is a systematic process for building software through well-defined phases.



**Figure 6.** SDLC Cycle

**Table 9.** SDLC Phases Detailed

Phase	Activities	Deliverables
<b>Planning</b>	Project planning, feasibility study	Project plan
<b>Analysis</b>	Requirement gathering	SRS document
<b>Design</b>	System architecture, UI design	Design document
<b>Implementation</b>	Coding, unit testing	Source code
<b>Testing</b>	System testing, integration	Test reports
<b>Deployment</b>	Installation, user training	Live system
<b>Maintenance</b>	Bug fixes, enhancements	Updated system

**Phase Descriptions:**

- **Planning:** Define project scope and resources
- **Analysis:** Understand what system should do
- **Design:** Plan how system will work
- **Implementation:** Build the actual system
- **Testing:** Verify system works correctly
- **Deployment:** Release system to users
- **Maintenance:** Ongoing support and updates

#### Mnemonic

“People Always Design Implementation, Test Deployment, Maintain”

### Question 3(a) [3 marks]

Describe skills to manage software projects.

#### Solution

**Software Project Management** requires combination of technical and soft skills.

**Table 10.** Essential Project Management Skills

Skill Category	Specific Skills
<b>Technical</b>	Understanding SDLC, tools, technologies
<b>Leadership</b>	Team motivation, decision making
<b>Communication</b>	Clear communication with team and clients
<b>Planning</b>	Resource allocation, scheduling
<b>Problem-solving</b>	Risk management, conflict resolution

#### Key Skills:

- **People management:** Lead and motivate team members
- **Technical knowledge:** Understand development process and tools
- **Communication:** Bridge between technical team and stakeholders

#### Mnemonic

“Technical Leaders Communicate Planning Problems”

### Question 3(b) [4 marks]

Briefly write responsibility of Software Project manager.

#### Solution

**Software Project Manager** oversees entire project from initiation to completion.

**Table 11.** Project Manager Responsibilities

Area	Responsibilities
Planning	Create project plans, schedules, budgets
Team Management	Hire, train, and manage team members
Communication	Regular updates to stakeholders
Quality Control	Ensure deliverables meet quality standards
Risk Management	Identify and mitigate project risks

**Primary Responsibilities:**

- **Project Planning:** Define scope, timeline, and resources
- **Team Leadership:** Guide and support development team
- **Stakeholder Communication:** Keep everyone informed of progress
- **Quality Assurance:** Ensure project meets requirements
- **Risk Management:** Handle project risks and issues

**Success Factors:** On-time delivery, within budget, meeting requirements

**Mnemonic**

“Plan Team Communication Quality Risk”

## Question 3(c) [7 marks]

Classify types of Requirements in SRS (1) Functional Requirements (2) Non-Functional Requirements.

**Solution**

**Requirements Classification** helps organize and understand different types of system needs.

**Table 12.** Functional vs Non-Functional Requirements

Aspect	Functional Requirements	Non-Functional Requirements
<b>Definition</b>	What system should do	How system should perform
<b>Focus</b>	System functionality	System quality attributes
<b>Examples</b>	Login, search, calculate	Performance, security, usability
<b>Testing</b>	Functional testing	Performance testing

**Functional Requirements:**

- **User interactions:** Login, registration, data entry
- **Business rules:** Validation rules, calculations
- **System features:** Reports, notifications, workflows
- **Data processing:** CRUD operations

**Examples:**

- User can login with username/password
- System calculates tax automatically
- Generate monthly sales report

**Non-Functional Requirements:**

**Table 13.** Non-Functional Requirement Types

Type	Description	Example
<b>Performance</b>	Speed and responsiveness	Response time < 2 seconds
<b>Security</b>	Data protection	Encrypted data transmission
<b>Usability</b>	User experience	Easy to learn interface
<b>Reliability</b>	System dependability	99.9% uptime
<b>Scalability</b>	Growth handling	Support 1000+ users

**Quality Attributes:**

- **Performance:** Response time, throughput
- **Security:** Authentication, authorization, encryption
- **Usability:** User-friendly interface, accessibility
- **Reliability:** Uptime, error handling
- **Maintainability:** Code quality, documentation

**Mnemonic**

“Performance Security Usability Reliability Maintainability”

**Question 3(a) OR [3 marks]**

Illustrate importance of SRS.

**Solution**

**SRS (Software Requirements Specification)** is crucial document that defines what software should do.

**Table 14.** SRS Importance

Aspect	Benefit
<b>Clear Communication</b>	All stakeholders understand requirements
<b>Project Planning</b>	Basis for estimation and scheduling
<b>Quality Assurance</b>	Foundation for testing
<b>Change Management</b>	Controlled requirement changes
<b>Legal Protection</b>	Contract reference document

**Key Importance:**

- **Communication tool:** Bridge between clients and developers
- **Planning foundation:** Helps estimate time, cost, and resources
- **Testing basis:** Test cases derived from SRS requirements

**Mnemonic**

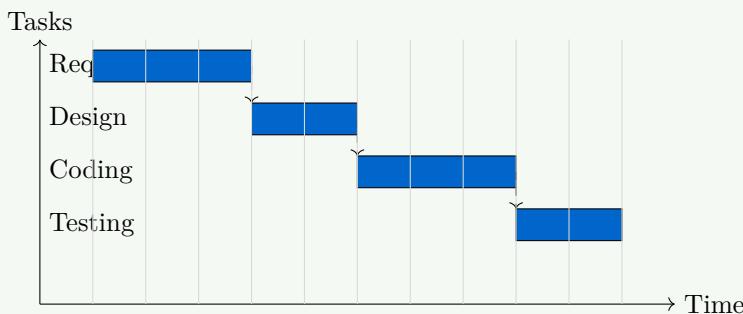
“Clear Planning Quality Change Legal”

**Question 3(b) OR [4 marks]**

Explain Gantt Chart.

### Solution

**Gantt Chart** is a visual project management tool showing tasks, timelines, and dependencies.



**Figure 7.** Simple Gantt Chart

**Table 15.** Gantt Chart Components

Component	Description
<b>Tasks</b>	Work items to be completed
<b>Timeline</b>	Horizontal time scale
<b>Bars</b>	Task duration and progress
<b>Dependencies</b>	Task relationships
<b>Milestones</b>	Important project events

#### Benefits:

- **Visual timeline:** Easy to see project schedule
- **Progress tracking:** Monitor task completion
- **Resource planning:** Allocate resources effectively
- **Dependency management:** Understand task relationships

### Mnemonic

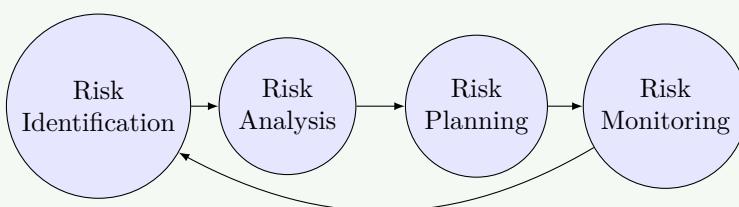
“Tasks Timeline Bars Dependencies Milestones”

## Question 3(c) OR [7 marks]

Write a short note on Risk Management.

### Solution

**Risk Management** is systematic process of identifying, analyzing, and controlling project risks.



**Figure 8.** Risk Management Cycle

**Table 16.** Risk Management Process

Phase	Activities	Output
<b>Identification</b>	Find potential risks	Risk list
<b>Analysis</b>	Assess probability and impact	Risk priority
<b>Planning</b>	Develop response strategies	Risk response plan
<b>Monitoring</b>	Track and control risks	Updated risk status

**Risk Categories:**

**Table 17.** Types of Software Risks

Category	Examples
<b>Technical</b>	Technology changes, complexity
<b>Project</b>	Schedule delays, resource shortage
<b>Business</b>	Market changes, funding issues
<b>External</b>	Vendor problems, regulatory changes

**Risk Response Strategies:**

- **Avoid:** Eliminate risk source
- **Mitigate:** Reduce probability or impact
- **Transfer:** Share risk with others
- **Accept:** Live with the risk

**Risk Assessment:** Probability × Impact = Risk Exposure

**Benefits:** Proactive problem solving, better project success rate, stakeholder confidence

### Mnemonic

“Identify Analyze Plan Monitor (Process), Avoid Mitigate Transfer Accept (Strategies)”

## Question 4(a) [3 marks]

What is metric for size estimation? Explain FP with example.

### Solution

**Size Estimation Metrics** help predict software project size and effort.

**Table 18.** Size Estimation Metrics

Metric	Description
<b>LOC</b>	Lines of Code
<b>Function Points</b>	Functionality-based measurement
<b>Object Points</b>	For object-oriented systems
<b>Feature Points</b>	Enhanced function points

**Function Points (FP)** measure software size based on user functionality.

**FP Components:**

- **External Inputs:** Data entry screens
- **External Outputs:** Reports, messages
- **External Queries:** Database queries
- **Internal Files:** Data stores
- **External Interfaces:** System connections

**FP Calculation Example:** For a Library Management System:

- External Inputs: 5 (Book entry, Member entry, etc.)
- External Outputs: 3 (Reports)

- External Queries: 4 (Search functions)
- Internal Files: 2 (Book DB, Member DB)
- External Interfaces: 1 (Online catalog)

**Simple FP = 5 + 3 + 4 + 2 + 1 = 15 Function Points**

#### Mnemonic

“Inputs Outputs Queries Files Interfaces”

## Question 4(b) [4 marks]

Explain project estimation techniques using basic COCOMO model.

#### Solution

COCOMO (COnstructive COst MOdel) estimates software development effort and schedule.

**Table 19.** COCOMO Model Types

Type	Description	Accuracy
<b>Basic</b>	Simple size-based estimation	±75%
<b>Intermediate</b>	Includes cost drivers	±25%
<b>Detailed</b>	Phase-level estimation	±10%

#### Basic COCOMO Formula:

- Effort =  $a \times (KLOC)^b$  person-months
- Time =  $c \times (Effort)^d$  months
- People = Effort / Time

**Table 20.** COCOMO Constants

Project Type	a	b	c	d
<b>Organic</b>	2.4	1.05	2.5	0.38
<b>Semi-detached</b>	3.0	1.12	2.5	0.35
<b>Embedded</b>	3.6	1.20	2.5	0.32

**Example:** For 10 KLOC organic project

- Effort =  $2.4 \times (10)^{1.05} = 25.47$  person-months
- Time =  $2.5 \times (25.47)^{0.38} = 8.64$  months
- People =  $25.47 / 8.64 = 3$  people

#### Mnemonic

“Organic Semi Embedded”

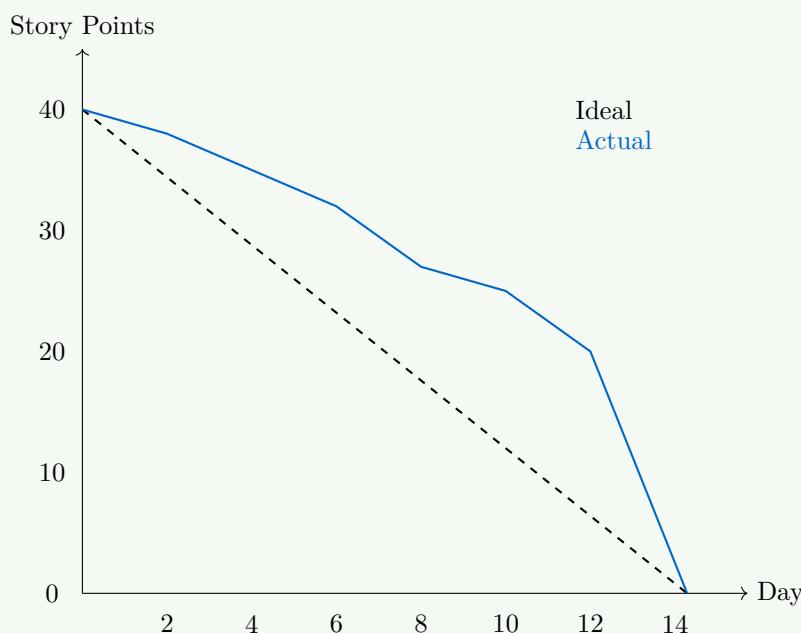
## Question 4(c) [7 marks]

Prepare Sprint burn down chart for system of your choice.

#### Solution

**Sprint Burn Down Chart** tracks remaining work during a sprint for **Online Shopping System**.

**Sprint Goal:** User Authentication Module (40 Story Points, 2 Weeks Estimate)

**Figure 9.** Sprint Burn Down Chart

Sprint Backlog:

**Table 21.** Sprint Tasks

Task	Story Points	Day Assigned
User Registration	8	Day 1-2
User Login	6	Day 3-4
Password Reset	5	Day 5-6
Profile Management	8	Day 7-8
Session Management	6	Day 9-10
Testing & Bug Fixes	7	Day 11-14

Burn Down Chart Data:

**Table 22.** Daily Progress

Day	Ideal	Actual	Status
Day 0	40	40	Sprint Start
Day 2	36	38	Registration delay
Day 4	32	32	Login completed
Day 6	28	27	Password reset done
Day 8	24	26	Profile issues
Day 10	20	20	Back on track
Day 12	16	15	Testing well
Day 14	0	0	Completed

Chart Analysis:

- Green line: Ideal burn down
- Red line: Actual progress
- Variations: Show challenges and recoveries
- Completion: Sprint finished on time

Benefits: Visual progress tracking, early problem identification, team motivation

**Mnemonic**

“Track Progress Daily, Identify Issues Early”

**Question 4(a) OR [3 marks]**

Explain the component of USE CASE diagram.

**Solution**

**Use Case Diagram** shows system functionality from user perspective.

**Table 23.** Use Case Diagram Components

Component	Symbol	Description
<b>Actor</b>	Stick figure	External entity interacting with system
<b>Use Case</b>	Oval	System functionality
<b>System Boundary</b>	Rectangle	System scope
<b>Association</b>	Line	Actor-Use Case relationship
<b>Generalization</b>	Arrow	Inheritance relationship

**Relationships:**

- **Include:** One use case includes another (mandatory)
- **Extend:** Optional use case extension
- **Generalization:** Parent-child relationship

**Example Components:**

- **Primary Actor:** Customer, Admin
- **Use Cases:** Login, Search Products, Place Order
- **System:** Online Shopping System

**Mnemonic**

“Actors Use Systems, Associate Generally”

**Question 4(b) OR [4 marks]**

Compare Cohesion and Coupling.

**Solution**

**Cohesion and Coupling** are important software design principles affecting maintainability.

**Table 24.** Cohesion vs Coupling Comparison

Aspect	Cohesion	Coupling
<b>Definition</b>	Unity within module	Dependency between modules
<b>Desirable Level</b>	High cohesion preferred	Low coupling preferred
<b>Focus</b>	Internal module unity	Inter-module relationships
<b>Impact</b>	Module reliability	System flexibility
<b>Measurement</b>	How related are module elements	How dependent modules are

**Cohesion Types** (Low to High):

- Coincidental, Logical, Temporal, Procedural, Communicational, Sequential, Functional
- Coupling Types** (High to Low):

- Content, Common, External, Control, Stamp, Data
- Goal: High Cohesion + Low Coupling = Good Design

### Mnemonic

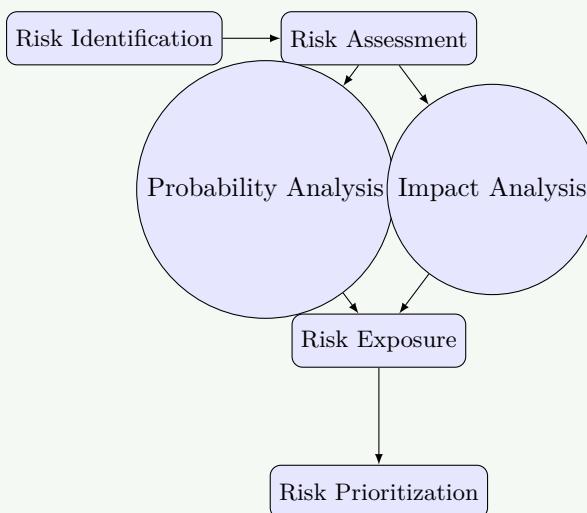
“High Cohesion, Low Coupling”

## Question 4(c) OR [7 marks]

Explain Risk Assessment in detail.

### Solution

**Risk Assessment** evaluates identified risks to prioritize management efforts.



**Figure 10.** Risk Assessment Process

**Table 25.** Risk Assessment Elements

Element	Description	Scale
<b>Probability</b>	Likelihood of risk occurring	0.1 to 1.0
<b>Impact</b>	Consequences if risk occurs	1 to 10
<b>Risk Exposure</b>	Probability × Impact	Calculated value
<b>Risk Level</b>	Priority classification	High/Medium/Low

**Assessment Process:**

**1. Probability Assessment:**

- Very Low (0.1) - Very High (0.9): Likelihood scale

**2. Impact Assessment:**

- Catastrophic (9-10) to Negligible (1-3): Consequence scale

**3. Risk Exposure Calculation:  $\text{Risk Exposure} = \text{Probability} \times \text{Impact}$**

**Example Risk Assessment:**

**Table 26.** Sample Risk Analysis

Risk	Prob	Imp	Exp	Priority
Key developer leaves	0.3	8	2.4	Medium
Requirements change	0.7	6	4.2	High
Technology failure	0.2	9	1.8	Low
Budget cuts	0.4	7	2.8	Medium

**Risk Matrix:**

- **High:** Exposure > 4.0 | **Medium:** 2.0-4.0 | **Low:** < 2.0

**Mnemonic**

“Probability Impact Exposure Priority”

**Question 5(a) [3 marks]**

Explain code inspection technique in code review.

**Solution**

**Code Inspection** is formal, systematic examination of code to find defects.

**Table 27.** Code Inspection Process

Phase	Participants	Activities
<b>Planning</b>	Moderator	Schedule, distribute code
<b>Overview</b>	Author, Team	Author explains code
<b>Preparation</b>	Individual	Reviewers study code
<b>Inspection</b>	All reviewers	Find defects
<b>Rework</b>	Author	Fix defects
<b>Follow-up</b>	Moderator	Verify fixes

**Key Features:**

- **Formal process:** Structured approach with defined roles
- **Systematic review:** Line-by-line examination
- **Defect focused:** Find errors, not solutions
- **No author criticism:** Focus on code, not coder

**Mnemonic**

“Plan Overview Prepare Inspect Rework Follow-up”

**Question 5(b) [4 marks]**

Prepare at least four test cases of ATM.

**Solution**

**ATM Test Cases** verify automated teller machine functionality.

**Table 28.** ATM Test Cases

ID	Scenario	Input	Output	Result
TC1	Valid PIN	Correct PIN	Access granted	Pass
TC2	Invalid PIN	Wrong PIN	Card blocked	Pass
TC3	Withdrawal	Amt $\leq$ Bal	Cash dispensed	Pass
TC4	Low Balance	Amt > Bal	Declined	Pass

**Detailed Test Cases:**

- **Valid Login:** Pre: Card inserted. Action: Enter PIN. Exp: Menu.
- **Cash Withdrawal:** Pre: Logged in. Action: Enter amount. Exp: Cash + Receipt.
- **Balance Inquiry:** Action: Check Balance. Exp: Show Balance.
- **PIN Change:** Action: Enter old/new PIN. Exp: Updated PIN.

**Mnemonic**

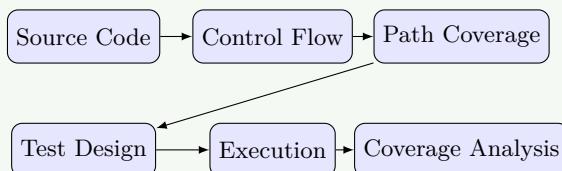
“Login Withdraw Inquiry Change”

**Question 5(c) [7 marks]**

Describe white box testing.

**Solution**

**White Box Testing** examines internal code structure and logic paths.



**Figure 11.** White Box Testing Process

**Table 29.** White Box Testing Characteristics

Aspect	Description
<b>Focus</b>	Internal code structure
<b>Knowledge</b>	Code implementation details
<b>Coverage</b>	Statements, branches, paths
<b>Techniques</b>	Basis path, loop testing

**Coverage Criteria:**

- **Statement Coverage:** Execute every statement
- **Branch Coverage:** Execute all if-else paths
- **Path Coverage:** Execute every possible path
- **Condition Coverage:** Test all conditions (true/false)

**Techniques:**

1. **Basis Path Testing:**
  - Cyclomatic Complexity  $V(G) = E - N + 2$
  - Generate independent paths equal to  $V(G)$
2. **Loop Testing:** Simple, Nested, Concatenated loops.
3. **Condition Testing:** Test logical conditions.

**Example Code:**

```

1 if (age >= 18 && income > 25000)
2     approve_loan();

```

```

3 | else
4 |     reject_loan();

```

**Test Cases:**

- age=20, income=30000 (Both True) → Approve
- age=16, income=30000 (First False) → Reject
- age=20, income=20000 (Second False) → Reject
- age=16, income=20000 (Both False) → Reject

**Advantages:** Thorough testing, Finds logic errors, Coverage measurement**Disadvantages:** Time consuming, Expensive, Code dependent**Mnemonic**

“Statement Branch Path Condition”

**Question 5(a) OR [3 marks]****Explain code walk through Technique in code review.****Solution****Code Walk Through** is informal code review technique where author presents code to team.**Table 30.** Walk Through Process

Phase	Description	Duration
<b>Preparation</b>	Author prepares presentation	30 min
<b>Presentation</b>	Author explains code logic	1-2 hours
<b>Discussion</b>	Team asks questions	30 min
<b>Documentation</b>	Record issues	15 min

**Key Characteristics:**

- **Author-led:** Code author drives the session
- **Informal process:** Less structured than inspection
- **Educational:** Team learns about code functionality
- **Collaborative:** Open discussion encouraged

**Participants:** Author (Presents), Reviewers (Ask questions), Moderator (Optional)**Mnemonic**

“Prepare Present Discuss Document”

**Question 5(b) OR [4 marks]****Explain software documentation.****Solution****Software Documentation** provides information about software system for various stakeholders.**Table 31.** Documentation Types

Type	Purpose	Audience
User Doc	How to use software	End users
System Doc	Technical details	Developers
Process Doc	Development process	Project team
Requirements	What system should do	Stakeholders

**Internal Documentation:** Code comments, Function headers, Variable names, README.

**External Documentation:** User manuals, Installation guides, API docs, Training materials.

**Benefits:** Maintainability, Knowledge transfer, User support, Quality assurance.

### Mnemonic

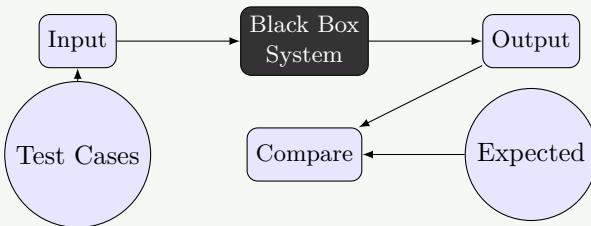
“User System Process Requirements”

## Question 5(c) OR [7 marks]

Write a short note on black box testing.

### Solution

**Black Box Testing** examines software functionality without knowledge of internal code structure.



**Figure 12.** Black Box Testing Concept

**Table 32.** Testing Techniques

Technique	Description	Example
<b>Equivalence Partitioning</b>	Valid/Invalid classes	Age: 0-17, 18-65, >65
<b>Boundary Value</b>	Test at boundaries	Age: 17, 18, 65, 66
<b>Decision Table</b>	Complex rules	Insurance premium
<b>State Transition</b>	State changes	ATM states

**Techniques Detail:**

1. **Equivalence Partitioning:** Test one value from each partition (Valid/Invalid).
2. **Boundary Value Analysis:** Test min, max, min-1, max+1.
3. **Decision Table:** Mapping conditions to actions.

**Comparison:**

- **User perspective** vs Code perspective
- **No code knowledge needed** vs Programming skills
- **Early testing** vs Testing after code

**Types:** Functional, Non-functional, Regression, UAT.

### Mnemonic

“Equivalence Boundary Decision State”