1. Introduction to Software Project Management

1.What is project? Why is software project management important.

- A project is a temporary work done to achieve a specific goal.
- It has a start and end date, uses resources (people, money, time), and is limited by time, cost, and scope.
- Example: Developing an online shopping website.

Importance of Software Project Management:

1. Handles Complexity – Software projects are large and difficult, management makes them systematic.
2. Time Management – Ensures project is completed within the deadline.
3. Cost Control – Keeps the project within the planned budget.
4. Quality Assurance – Ensures software meets user requirements.
5. Risk Management – Identifies and reduces possible problems.
6. Team Coordination – Improves communication and work distribution.
7. Customer Satisfaction – Delivers software as per customer needs.

Characteristics of a Project:

1. **Temporary nature** – Every project has a start date and an end date.
2. **Unique output** – Produces a new product, service, or result.
3. **Goal oriented** – Done to achieve specific objectives.
4. **Progressive elaboration** – Planned and developed step by step.
5. **Resource based** – Requires people, money, time, and materials.
6. **Constraints bound** – Limited by time, cost, scope, and quality.
7. **Team effort** – Involves teamwork and coordination.

Project Management:-

Project Management is the process of **planning, organizing, leading, and controlling resources** to achieve specific goals and complete a project **successfully within a defined time, cost, and quality**.

2. Write the activities covered by software project management.

Main Activities in Software Project Management:

1. **Project Planning** – Defining scope, objectives, schedule, resources, and budget.
2. **Project Scheduling** – Creating timelines, milestones, and task dependencies (e.g., Gantt chart, PERT/CPM).
3. **Cost Estimation & Budgeting** – Estimating effort, resources, and controlling expenses.
4. **Resource Allocation** – Assigning people, tools, and technologies effectively.
5. **Risk Management** – Identifying possible risks and preparing solutions.
6. **Quality Management** – Ensuring software meets required standards.
7. **Monitoring & Controlling** – Tracking project progress and taking corrective actions.
8. **Communication & Team Management** – Coordinating among stakeholders and team members.
9. **Project Closure** – Delivering the final product, documentation, and evaluation.

3. Explain Plan, Method and Methodologies in Project Management

1) Plan

1. A plan is a **roadmap** that guides project execution.
2. It defines **what work will be done, when, by whom, and at what cost**.
3. Plans help in **tracking progress** and controlling deviations.
4. Examples: Project plan, schedule plan, budget plan.

2) Method

1. A method is a **specific procedure or technique** to complete a task.
2. It tells **how to do an activity** in the project.
3. Methods are smaller units that can be applied to solve problems.
4. Examples: PERT (Program Evaluation and Review Technique), CPM (Critical Path Method).

3) Methodology

1. A methodology is a **framework** that combines multiple methods, tools, and practices.
2. It covers the **entire project life cycle** (planning to closure).
3. Provides a **structured approach** for managing projects.
4. Examples: Waterfall Model, Agile, Scrum, PRINCE2.

4. Explain the Ways of Categorizing Software Projects in Project Management

1) Based on Size

- This categorization considers the **scale of the project** in terms of team size, duration, and cost.
- **Small Projects:** Few team members, short duration, low cost. Example: Personal website.
- **Medium Projects:** Moderate team size and complexity. Example: College management system.
- **Large Projects:** Large teams, long duration, high cost. Example: Banking software.

2) Based on Complexity

- Complexity depends on the **number of modules, integration needs, and difficulty of requirements**.
- **Simple Projects:** Easy requirements, less risk.
- **Complex Projects:** Multiple modules, integration required, higher risk. Example: ERP system.

3) Based on Application Area

- Projects are classified according to the **field or domain** where the software will be used.
- **Business Software:** Payroll, accounting systems.
- **System Software:** Operating systems, device drivers.
- **Scientific/Engineering Software:** Weather forecasting, CAD tools.
- **Embedded Software:** Software inside devices like mobiles, cars, and appliances.

4) Based on Technology Used

- Categorization by technology considers the **platform or tools** used to develop the software.
- **Web-based Projects:** E-commerce sites, social media apps.
- **Mobile-based Projects:** Android/iOS applications.
- **AI/ML Projects:** Chatbots, recommendation systems.
- **Cloud-based Projects:** SaaS products like Google Drive.

5) Based on Risk Level

- This is based on **uncertainty and potential problems** in a project.
- **Low Risk Projects:** Clear requirements, stable technology.
- **High Risk Projects:** New technology, unclear requirements, higher chance of failure.

5. Describe in Brief Stakeholders of a Project

- Stakeholders are people, groups, or organizations who are affected by or have an interest in a project.
- They are important because their needs, opinions, and decisions can influence the success of the project.

Types of Stakeholders:

1. **Internal Stakeholders:** Project team, managers, employees.
2. **External Stakeholders:** Clients, customers, suppliers, government, investors.

Main Types of Stakeholders:

1. Project Manager: Plans, executes, and monitors the project.
2. Project Team Members: Carry out tasks and contribute to deliverables.
3. Customers/Clients: Use the product or service; their requirements guide the project.
4. Sponsors/Investors: Provide funding and resources.
5. End Users: People who will actually use the software.
6. Regulatory Bodies: Ensure project follows legal and industry standards.
7. Suppliers/Vendors: Provide tools, technology, or materials needed for the project

6. Explain

1) Management

Management is the process of **planning, organizing, leading, and controlling resources** (people, money, materials, and time) to achieve specific objectives efficiently and effectively.

Key Points:

1. **Planning:** Deciding **what needs to be done**, how, and by whom.
2. **Organizing:** Arranging **resources and tasks** in a structured way to achieve goals.
3. **Leading/Directing:** Guiding and motivating people to **perform their tasks efficiently**.
4. **Controlling:** Monitoring performance and taking **corrective actions** if required.

Importance:

- Helps achieve **goals and objectives** on time.
- Reduces **wastage and errors**.
- Improves **coordination and communication**.

2) **Contract:-**

A contract is a formal agreement between two or more parties in which they agree to do (or not do) certain things, usually in exchange for something of value, such as money, goods, or services.

Key Points:

1. **Legally Binding:** Both parties are legally obligated to fulfill their promises.
2. **Mutual Agreement:** All parties must agree to the terms voluntarily.
3. **Terms & Conditions:** Specifies **what, when, and how** the work or service will performed.
4. **Purpose:** Ensures clarity, protects rights, and avoids disputes.

Example:

- A company signs a contract with a software developer to create a website within 2 months for a fixed payment.

3) **Contract Management:-**
Contract Management is the process of creating, monitoring, and managing contracts made with clients, suppliers, or partners to ensure that all parties meet their obligations and the agreed terms are properly followed.

Key Points:

1. **Planning:** Define the terms, deliverables, responsibilities, and timelines before signing the contract.
2. **Execution:** Implement the contract according to the agreed terms.
3. **Monitoring:** Track progress, compliance, and deadlines.
4. **Closure:** Ensure all obligations are completed, resolve disputes, and formally close the contract.

Importance:

- Ensures **clarity and legal protection** for all parties.
- Helps **avoid disputes and delays**.
- Ensures **deliverables meet quality and timelines**.
- Improves **relationships with clients, vendors, and stakeholders**.

Example:
In a software project, the contract with a client specifies project scope, deadlines, payment terms, and support. Contract management ensures the project follows these terms until completion.

7.Define Business Case and Explain the Concept of Business Case:

- A **Business Case** is a **document or justification that explains why a project or investment is worth undertaking**.
- It presents the **benefits, costs, risks, and alternatives** to help decision-makers approve or reject the project.

Concept of Business Case:

The **business case** helps organizations decide whether a project aligns with their **strategic goals** and is **worth the resources**. It ensures that the project:

1. **Provides value:** Clearly shows benefits for the organization, customers, or stakeholders.
2. **Justifies cost:** Explains the expected costs and return on investment (ROI).
3. **Assesses risks:** Identifies potential challenges and ways to mitigate them.
4. **Evaluates alternatives:** Compares different ways to achieve the objective.
5. **Guides decision-making:** Helps stakeholders approve, modify, or reject the project.

Key Elements of a Business Case:

1. **Project Objectives:** What the project aims to achieve.
2. **Benefits:** Tangible and intangible gains from the project.
3. **Costs:** Estimated budget, resources, and time required.
4. **Risks:** Potential problems and mitigation strategies.
5. **Alternatives:** Different options to achieve the same goal.
6. **Recommendation:** Suggested approach based on analysis.

Example:

A company wants to implement a new payroll system:

- **Objective:** Automate payroll to reduce errors.
- **Benefits:** Save time, reduce mistakes, improve employee satisfaction.
- **Costs:** Software purchase, training, and implementation.
- **Risks:** System downtime or resistance from staff.
- **Recommendation:** Proceed with implementation using a cloud-based solution.

8. How one can find out the success and failure of a project.

* Success of a Project

A project is considered **successful** when it **achieves its objectives** and meets the expectations of stakeholders.

Key Points:

1. **Meets Goals:** Delivers the planned product, service, or results.
2. **On Time:** Completed within the scheduled timeline.
3. **Within Budget:** Costs are managed efficiently.
4. **Good Quality:** Deliverables meet required standards.
5. **Stakeholder Satisfaction:** Client, users, and team are happy with results.
6. **Delivers Benefits:** Provides value, such as profit, efficiency, or customer satisfaction.

Example:
A software project delivered on time, with all required features working correctly, and the client is satisfied.

* Failure of a Project

A project is considered **failed** when it **does not achieve its objectives** or does not meet stakeholder expectations.

Key Points:

1. **Misses Goals:** Fails to deliver the expected product or results.
2. **Over Budget:** Costs more than planned.
3. **Delayed:** Exceeds the scheduled timeline.
4. **Poor Quality:** Deliverables are defective or do not meet requirements.
5. **Stakeholder Dissatisfaction:** Client, users, or team are unhappy.
6. **No Benefits:** Does not provide the expected value or ROI.

Example:
A software project that is late, over budget, buggy, and the client is unhappy.

9.Management Responsibilities of a Manager in Software Project Management

A **software project manager** is responsible for planning, executing, and closing software projects successfully. Their responsibilities cover **people, processes, and resources**.

1) Planning and Scheduling:

- Define **project objectives** and scope.
- Prepare **detailed project plans, schedules, and timelines**.
- Allocate **tasks and responsibilities** to team members.

2) Resource Management:

- Manage **human resources**, assign tasks based on skills.
- Allocate **budget, tools, and infrastructure** effectively.
- Ensure availability of **necessary software and hardware resources**.

3) Risk and Quality Management:

- Identify **potential risks** and create mitigation plans.
- Ensure **quality standards** are maintained in software development.
- Monitor progress to prevent **scope creep or delays**.

4) Communication and Coordination:

- Serve as a link between **team members, clients, and stakeholders**.
- Conduct **meetings, reviews, and status updates**.
- Resolve **conflicts and misunderstandings** within the team.

5) Monitoring and Control:

- Track **project progress against the plan**.
- Take **corrective actions** in case of deviations.
- Ensure the project is completed **on time, within budget, and as per requirements**.

6) Documentation and Reporting:

- Maintain **project documentation** including plans, reports, and contracts.
- Provide **regular progress reports** to clients and management.

10.Explain traditional project management and modern project management

1) Traditional Project Management (TPM)

Traditional project management follows a **structured, linear, and sequential approach** where each phase is completed before moving to the next.

Key Features:

1. **Linear Process:** Follows phases like **requirement → design → development → testing → deployment** (Waterfall model).
2. **Predictive Planning:** Detailed plans are made at the start.
3. **Fixed Scope:** Changes are minimized once the project begins.
4. **Control-Oriented:** Emphasis on strict schedules, budgets, and documentation.

Advantages:

- Clear structure and well-defined deliverables.
- Easy to track progress.
- Suitable for projects with **stable requirements**.

Disadvantages:

- Inflexible to changes.
- Late detection of errors or problems.
- Not ideal for **dynamic or complex projects** like modern software development.

2) Modern Project Management (MPM)

Modern project management is **flexible, adaptive, and iterative**, focusing on collaboration, continuous improvement, and customer feedback.

Key Features:

1. **Iterative Approach:** Work is done in **sprints or phases**, with continuous review (Agile, Scrum).
2. **Adaptive Planning:** Plans can change based on **feedback and evolving requirements**.
3. **Collaborative:** Encourages **teamwork and stakeholder involvement** throughout the project.
4. **Value-Oriented:** Focus on delivering **quality products and customer satisfaction** rather than just following a plan.

Advantages:

- Adapts to changing requirements easily.
- Early detection of problems and errors.
- Higher stakeholder satisfaction.

Disadvantages:

- Less predictable schedule and budget.
- Requires experienced and collaborative teams.

2.Project Design and Evaluation

1.What is Project Evaluation? Explain Its Benefits

Definition:

Project Evaluation is the process of **assessing a project's performance and outcomes** to determine whether it **achieved its objectives, delivered expected benefits, and used resources efficiently**.
It helps in understanding **what went well, what didn't, and how future projects can improve**.

Purpose of Project Evaluation:

1. Measure **project success** against objectives, time, cost, and quality.
2. Identify **strengths and weaknesses** of the project.
3. Ensure **accountability** of resources and team performance.
4. Learn **lessons for future projects** to improve planning and execution.

Benefits of Project Evaluation:

1. **Improves Decision-Making:** Helps management decide whether to continue, modify, or stop a project.
2. **Enhances Efficiency:** Identifies resource wastage and suggests ways to optimize.
3. **Ensures Accountability:** Tracks use of funds, time, and manpower.
4. **Increases Stakeholder Confidence:** Provides transparency to clients, management, and team.
5. **Supports Future Planning:** Lessons learned help in better planning and execution of new projects.
6. **Measures Project Impact:** Evaluates if the project delivered expected benefits and value.

Example:

After completing a software project, evaluation can check:

- Was it delivered on time and budget?
- Did it meet quality standards and client requirements?
- Were resources used efficiently?
- What lessons can improve the next project?

2.Cost-Benefit Evaluation Techniques :-

Cost-Benefit Evaluation is a method used to **analyze whether the expected benefits of a project exceed its costs**. It is essential for **project approval, prioritization, and efficient resource allocation**.

1) Cost-Benefit Analysis (CBA)

Concept:

- Compares **total expected costs** and **total expected benefits**.
- Helps determine **financial viability**.

Steps:

1. List all **costs**: development, hardware, manpower, maintenance.
2. List all **benefits**: revenue, efficiency, productivity, quality improvements.
3. Subtract costs from benefits → **Net Benefit**.

Decision Rule:

- Net Benefit > 0 → Project is viable.

Example:

- Cost = ₹5 lakh; Expected revenue = ₹8 lakh → Net Benefit = ₹3 lakh → Acceptable.

Advantages:

- Simple, easy to understand.
- Provides clear financial justification.

Limitations:

- Difficult to quantify intangible benefits.
- May ignore future risks or uncertainties.

2) Payback Period Method

Concept:

- Measures **time required to recover the initial investment** from project benefits.
- Focuses on **liquidity and risk**.

Formula:

$$Payback\ Period = \frac{Initial\ Investment}{Annual\ CashInflows}$$

Decision Rule:

- Shorter payback period → more attractive project.

Example:

- Project cost = ₹6 lakh, Annual benefit = ₹2 lakh → Payback Period = 6 ÷ 2 = 3 years.

Advantages:

- Simple and quick to calculate.
- Helps in **risk assessment**.

Limitations:

- Ignores benefits beyond payback period.
- Does not consider **time value of money**.

3) Return on Investment (ROI)

Concept:

- Measures **profitability of a project** relative to its cost.

Formula:

$$ROI(\%) = \frac{Net\ Benefits}{Total\ Costs} \times 100$$

Decision Rule:

- Higher ROI → More profitable project.

Example:

- Cost = ₹5 lakh, Net Benefits = ₹2 lakh → ROI = (2/5) × 100 = 40% → Profitable.

Advantages:

- Shows **profitability in percentage**, easy to compare projects.
- Supports decision-making and prioritization.

Limitations:

- ROI does not consider **time value of money**.
- May ignore intangible benefits.

4) Net Present Value (NPV) & Discounted Cash Flow (DCF)

Concept:

- Calculates **present value of future benefits minus costs**, considering **time value of money**.
- Useful for **long-term projects**.

Decision Rule:

- NPV > 0 → Project is acceptable.
- NPV < 0 → Project should be rejected.

Example:

- Project generates ₹1 lakh per year for 5 years, Cost = ₹4 lakh.
- Discounted cash flow = ₹4.5 lakh → NPV = ₹0.5 lakh → Accept project.

Advantages:

- Accounts for **time value of money**.
- Measures **profitability over lifecycle**.

Limitations:

- Complex calculation.
- Requires accurate estimation of cash flows and discount rate.

5) Benefit-Cost Ratio (BCR)

Concept:

- Ratio of **present value of benefits to present value of costs**.

Formula:

$$BCR = \frac{Total\ Benefits}{Total\ Costs}$$

Decision Rule:

- BCR > 1 ⟶ Project is beneficial.
- BCR < 1 ⟶ Project is not beneficial.

Example:

- Total benefits = ₹10 lakh, Total costs = ₹8 lakh ⟶ BCR = 10 ÷ 8 = 1.25 ⟶ Acceptable.

Advantages:

- Simple, provides **quick comparison** between projects.
- Useful for **prioritization**.

Limitations:

- Ignores qualitative benefits.
- May not consider **time factor unless discounted**.

3.Techniques of Process Analysis

Definition:
Process analysis is the systematic study of a process to **understand how it works, identify inefficiencies, and improve performance**. In software project management, it helps in **analyzing workflows, optimizing resources, and improving quality**.

---

1) Flowcharting

Definition:

- Flowcharting represents a process as a **sequence of steps using symbols and arrows**.

Purpose:

- Helps **visualize the workflow**.
- Makes it easy to **identify bottlenecks, redundancies, and errors**.

Example:

- Software development process:
  - Requirement → Design → Coding → Testing → Deployment

Diagram:

css Copy Edit

```css
[Start] → [Requirement] → [Design] → [Coding] → [Testing] → [Deployment] → [End]
```

---

2) Data Flow Diagram (DFD)

Definition:

- Shows **how data moves between processes, storage, and users**.

Purpose:

- Helps **identify inputs, outputs, processes, and storage points**.
- Useful for **software systems analysis**.

Example:

- Library Management System: Users input book requests, system stores data, outputs report to users.

Diagram:

scss Copy Edit

```scss
[User] → (Input Data) → [Process] → (Store Data) → [Database] → (Output Report) → [User]
```

3) Process Mapping

Definition:

- Detailed representation of **activities, responsibilities, and timelines.**

Purpose:

- Identifies **delays, redundancies, and inefficiencies.**
- Helps in **resource allocation.**

Example:

- Mapping a software project lifecycle with responsible roles:
    - Requirement → Design → Coding → Testing → Deployment

Diagram:

less Copy Edit

```
[Requirement] → [Design] → [Coding] → [Testing] → [Deployment]
   | Analyst | Designer | Developer | QA | Admin
```

---

4) Root Cause Analysis (RCA)

Definition:

- Technique to **identify the underlying cause of problems** rather than just symptoms.

Purpose:

- Prevents **recurring issues.**
- Improves **long-term process efficiency.**

Example:

- Software release delays caused by inefficient coding practices.

Diagram (Fishbone/Ishikawa):

arduino Copy Edit

```
            Problem
          ←───────
        /    |    \
     People  Process  Tools  Environment
```

4.The GQM Paradigm

- **GQM** stands for **Goal, Question, Metric**.
- It is a **structured approach to define and measure software project goals**.
- Helps organizations **understand, control, and improve software processes and products**.

1) Goal (G)

- Define **what you want to achieve**.
- Goals are **specific, measurable, and aligned with project objectives**.
- Example: "Improve the quality of software by reducing defects in the coding phase."

2) Question (Q)

- Ask **questions that help evaluate whether the goal is being met**.
- Questions should focus on **specific aspects of the goal**.
- Example: "How many defects are found per module during coding?"

3) Metric (M)

- Identify **quantitative measures** to answer the questions.
- Metrics provide **objective evidence of progress**.
- Example: "Number of defects per 1,000 lines of code (KLOC)."

Process Flow:

Goal → Questions → Metrics

Explanation:

1. Start with a **Goal** for the project.
2. Break it down into **Questions** to understand how the goal can be achieved.
3. Define **Metrics** to provide data for answering the questions.

Benefits of GQM Paradigm:

- Provides a **clear measurement framework**.
- Helps in **tracking progress toward project goals**.
- Ensures **data-driven decisions**.
- Supports **continuous improvement** in software processes.

5. Improvement Process

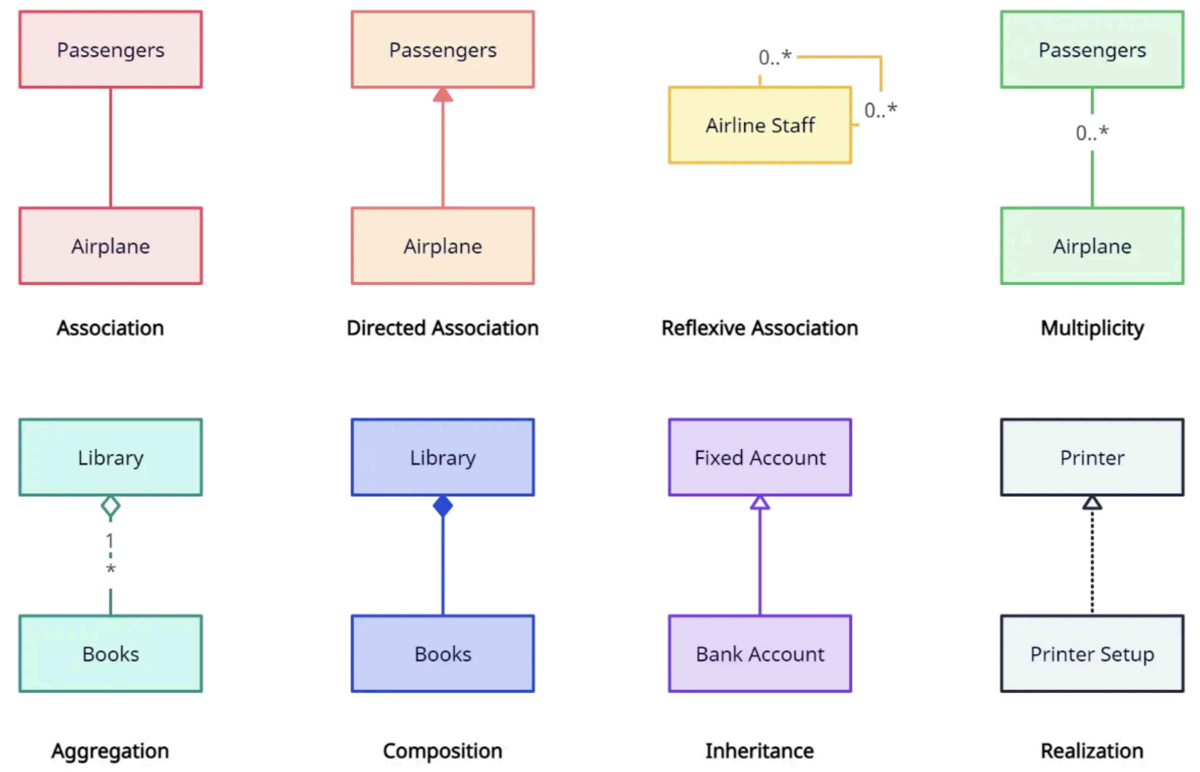
- A structured approach to **identify, analyze, and enhance existing processes** in software development.
- Helps organizations **adapt to changing requirements and improve productivity**.

Steps in the Process Improvement Cycle

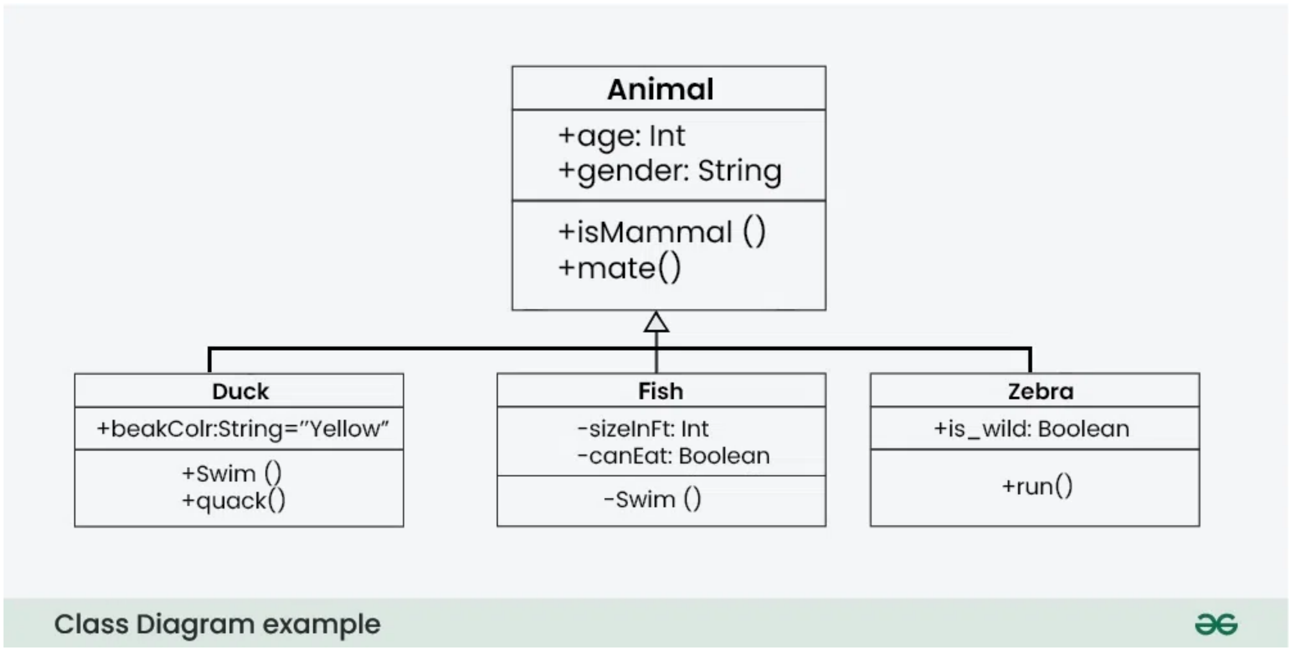
1. **Identify Process Areas for Improvement**
    - Evaluate current processes.
    - Find **bottlenecks, inefficiencies, and defects**.
    - Example: Long testing phase causing project delays.
2. **Analyze and Understand Processes**
    - Use techniques like **flowcharts, process mapping, and root cause analysis**.
    - Understand **why problems occur**.
3. **Define Improvement Goals**
    - Set **clear objectives** for what the improved process should achieve.
    - Example: Reduce defect rate by 20% in coding phase.
4. **Develop and Implement Changes**
    - Introduce **new methods, tools, or practices**.
    - Example: Adopt automated testing tools to reduce manual errors.
5. **Measure and Evaluate Results**
    - Collect **data on the updated process**.
    - Check whether improvement goals are achieved.
    - Example: Track defect rate after implementing automated testing.
6. **Review and Standardize**
    - Standardize **successful improvements**.
    - Make them part of **organizational best practices**.

6.Explain relationships in class diagram with the help of suitable example.

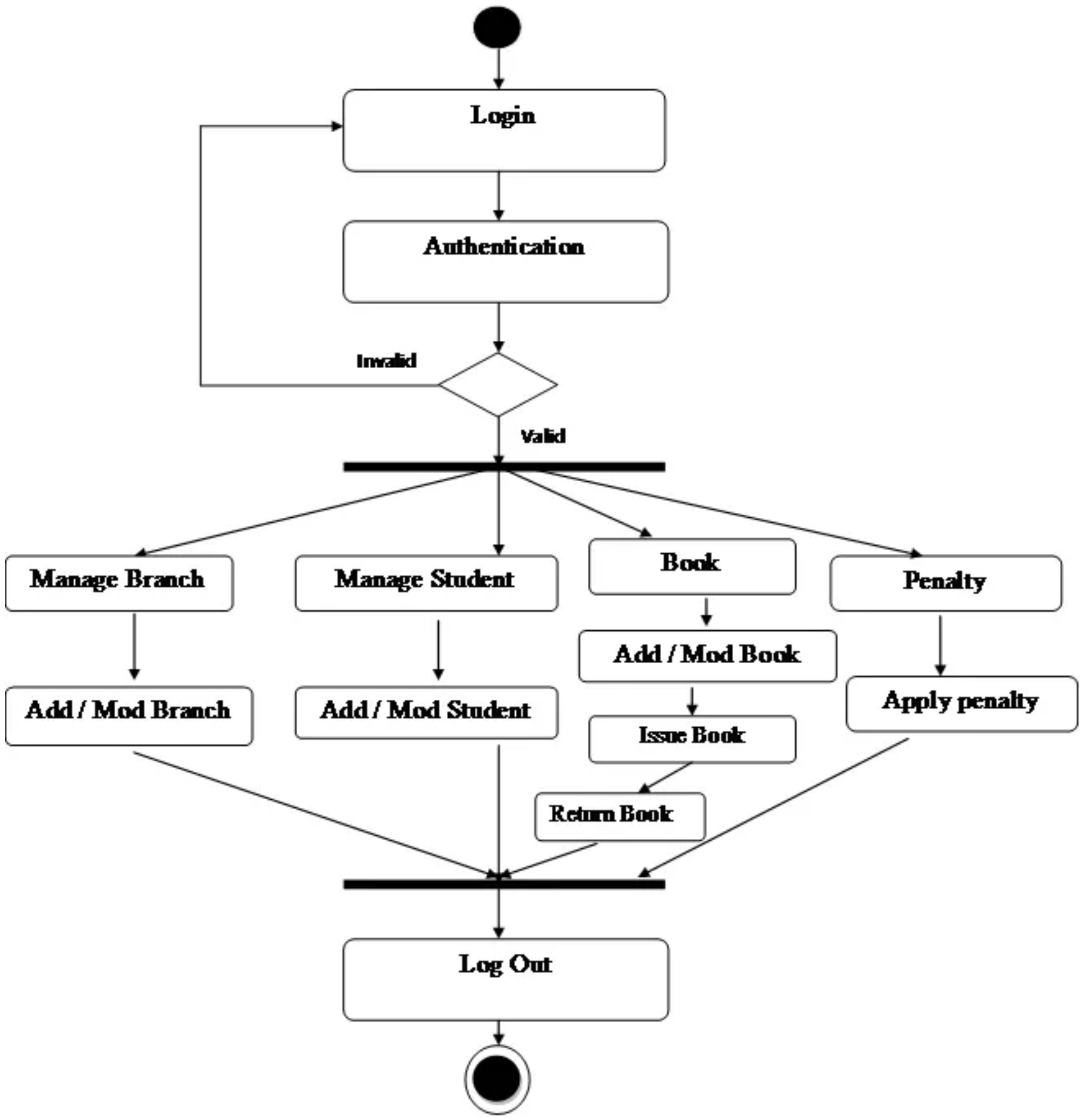
Types of Relationships:



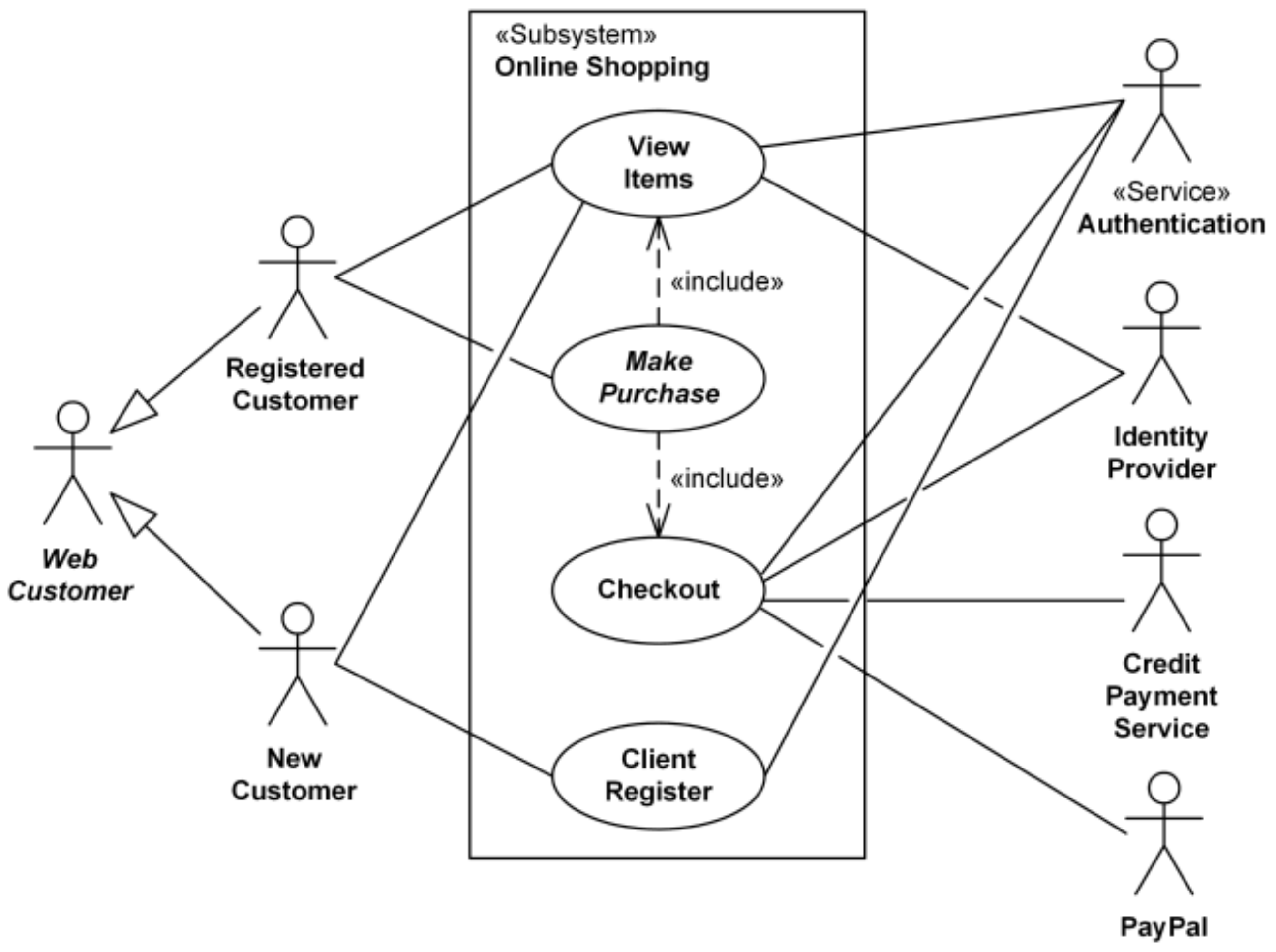
1. **Association**
   - It is the **basic relationship** between two classes.
   - Example: A Teacher teaches a Student.
2. **Aggregation (Has–a relationship)**
   - A **whole–part relationship** where the part can **exist independently** of the whole.
   - Example: A Department has Professors. Even if the department closes, professors still exist.
3. **Composition (Strong Has–a relationship)**
   - A **whole–part relationship** where the part **cannot exist without the whole**.
   - Example: A House has Rooms. If the house is destroyed, rooms do not exist.
4. **Inheritance (Generalization)**
   - Represents an **“is-a” relationship** where one class is a specialized version of another.
   - Example: A Car is a type of Vehicle.
5. **Dependency (Uses relationship)**
   - One class **depends on or uses** another class.
   - Example: A Customer uses a Payment service



7.Draw an activity diagram for a book issue process in library management system.



8. Draw a use case diagram for placing an online order in a sales company.



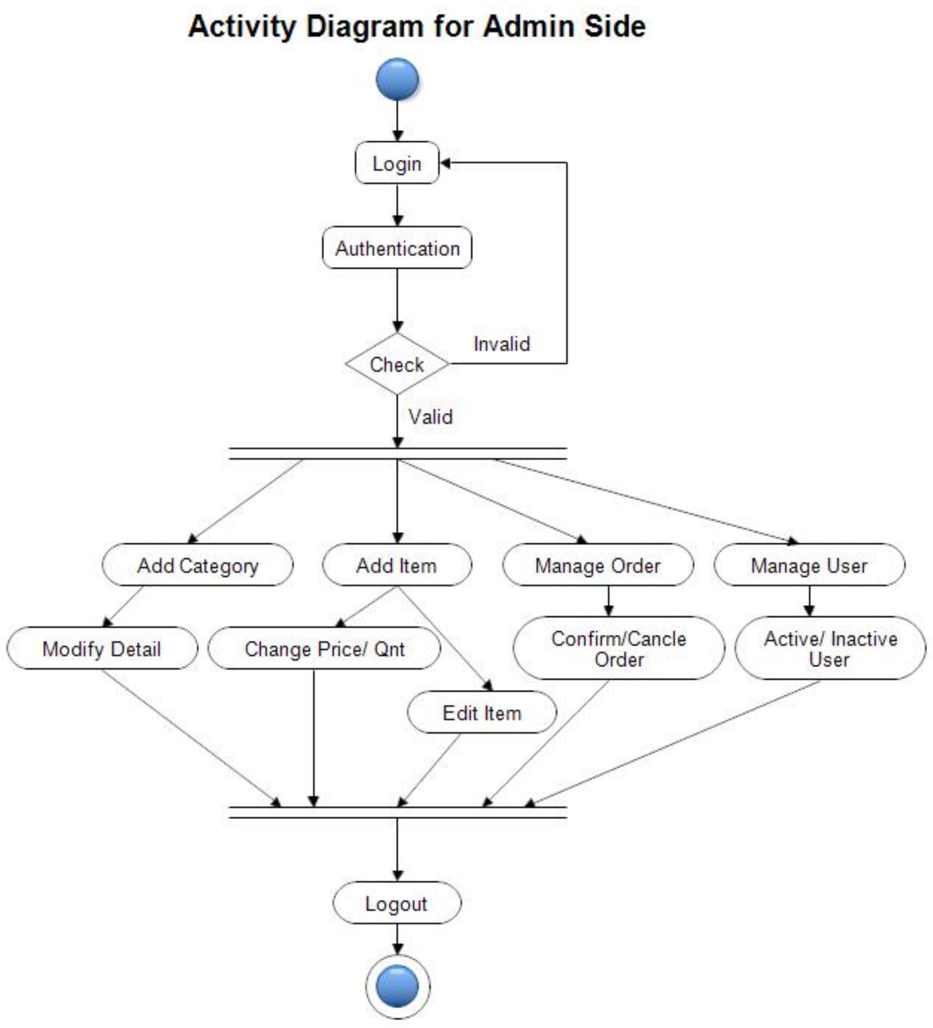
9. Describe sequence diagram and its notations with the help of suitable example.

Notations Used in Sequence Diagram:-

- Actors
- Lifelines
- Messages
- Create message
- Delete Message
- Self Message
- Reply Message
- Found Message
- Lost Message
- Guards

1. **Actor (Stick Figure)**
   - Represents a **user or external system** interacting with the system.
2. **Object / Class (Rectangle with name)**
   - Represents an object or class that participates in the interaction.
3. **Lifeline (Vertical dashed line)**
   - Shows the **existence of an object** over time.
4. **Activation Box (Thin vertical rectangle on lifeline)**
   - Represents the time when an object is performing an action.
5. **Message (Horizontal arrow)**
   - Represents **communication** between objects.
   - Types:
     - **Solid arrow** → synchronous call (function call).
     - **Dashed arrow** → return message.
6. **Start/End**
   - Interaction starts with an actor and ends with completion of the process.

10.Activity Diagram of online shopping website


Activity Diagram for Admin Side
Login
Authentication
Check
Invalid
Valid
Add Category
Add Item
Manage Order
Manage User
Modify Detail
Change Price/ Qnt
Confirm/Cancle Order
Active/ Inactive User
Edit Item
Logout