# MCS-14 : SYSTEMS ANALYSIS AND DESIGN

## 1. (a) What is Prototyping Approach for system design? Explain the steps involved in Prototyping process with the help of a diagram.

Prototyping is an iterative approach to system design that involves creating a preliminary version of a system in order to test and refine its functionality, design, and user experience. The prototyping process typically involves the following steps:

Identify Requirements: Gather requirements from stakeholders to understand the goals and functionality of the system.

Design Prototype: Create a preliminary design of the system based on the gathered requirements. This design should outline the basic structure, layout, and functionality of the system.

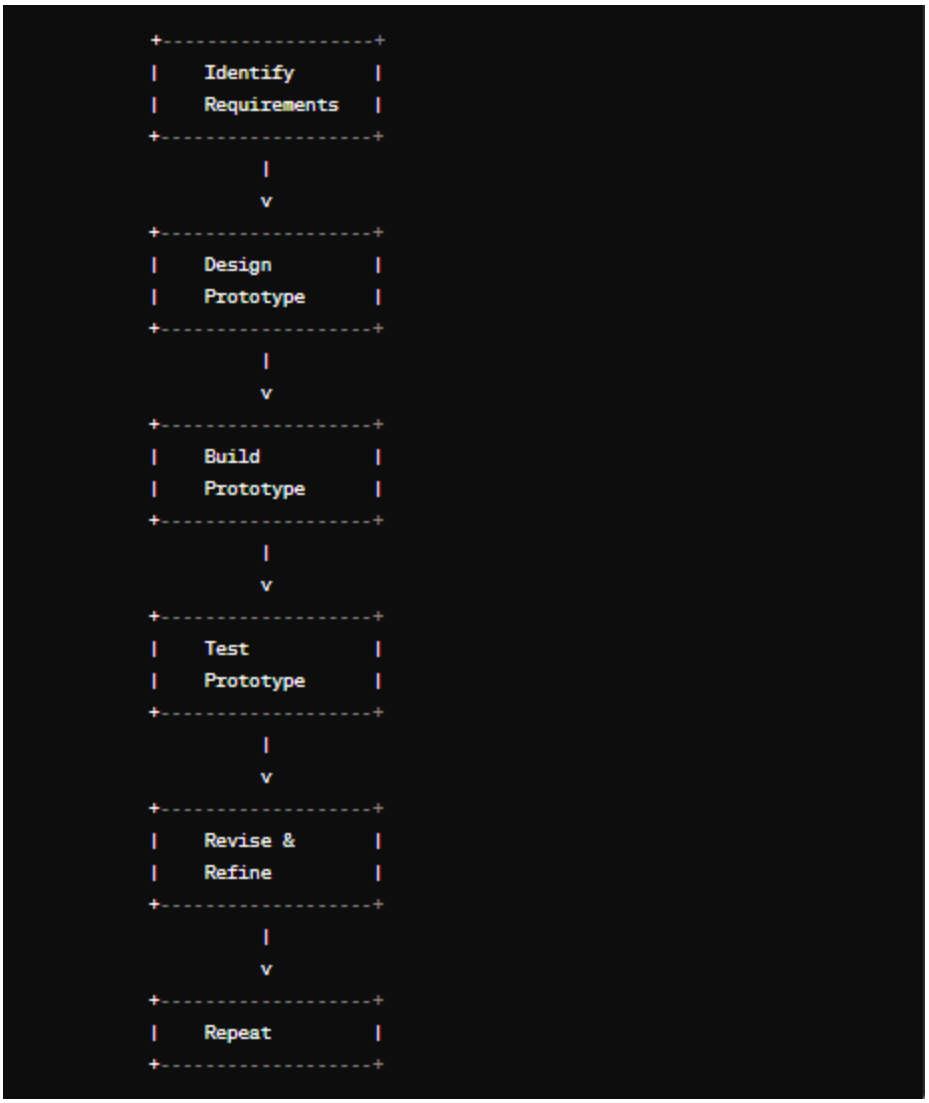
Build Prototype: Develop a working prototype of the system using rapid prototyping tools or by building a simplified version of the system using programming languages or development platforms.

Test Prototype: Test the prototype with users and stakeholders to gather feedback on its functionality, usability, and design. This feedback is used to identify areas for improvement and refinement.

Revise and Refine: Based on the feedback received during testing, make revisions and refinements to the prototype to address any issues or concerns raised by users and stakeholders.

Repeat: Iterate on the prototype by going through the design, build, test, and refine steps multiple times until the system meets the desired requirements and expectations.

Here's a diagram illustrating the prototyping process:




This cyclic process allows for continuous improvement and refinement of the system until it meets the desired requirements and expectations of the users and stakeholders.

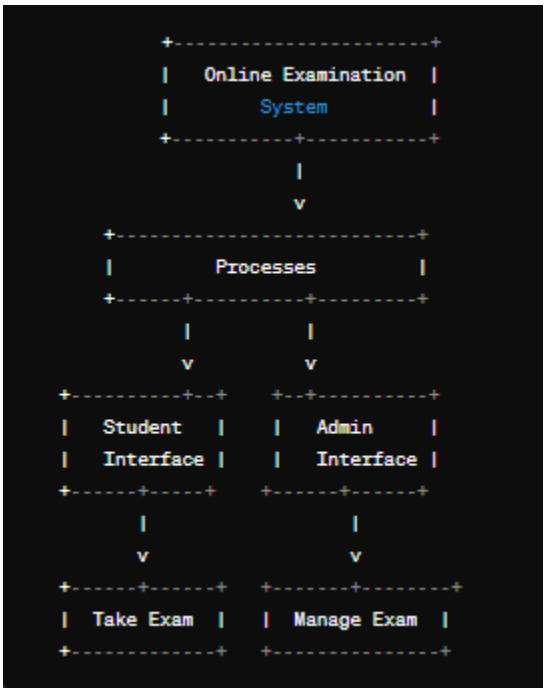
## (b) What is a DFD? Draw DFD's upto 2nd level for "Online Examination System".

## Note : Make necessary assumptions.

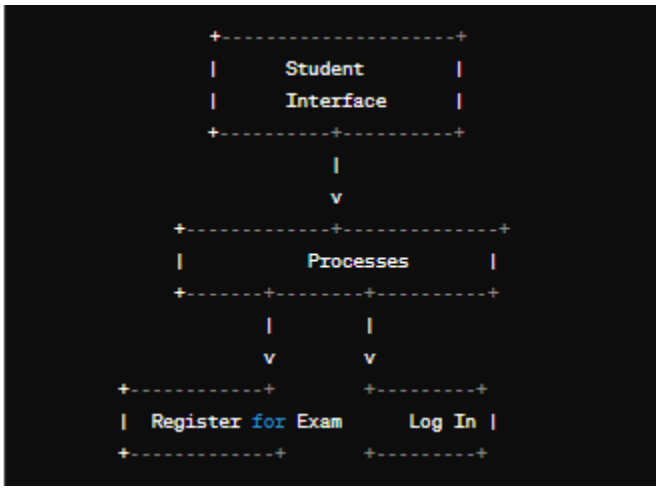
A Data Flow Diagram (DFD) is a graphical representation of the flow of data through a system. It consists of processes, data stores, data flows, and external entities. Each process represents a function performed within the system, while data flows represent the movement of data between processes, data stores, and external entities.

For an "Online Examination System," here's a simple DFD up to the 2nd level:

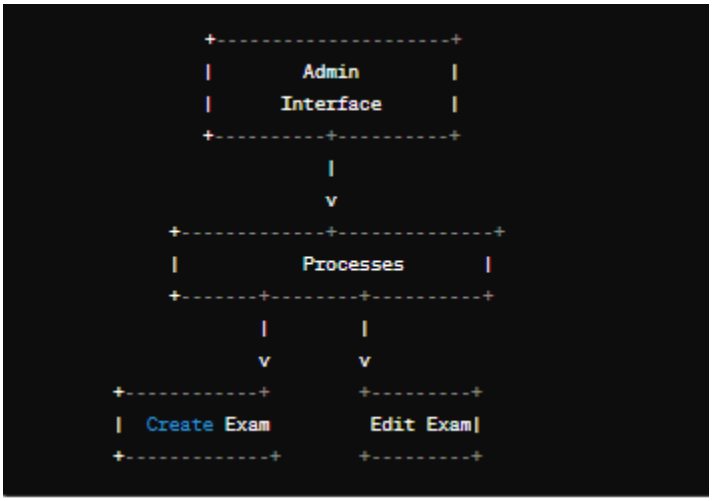
**Level 0 DFD:**



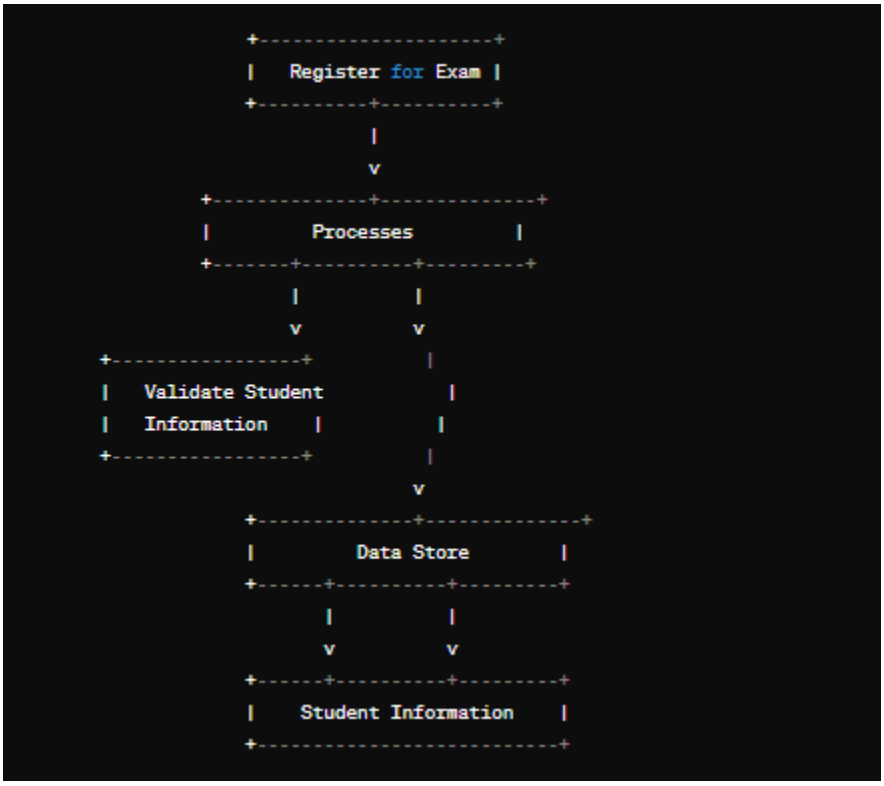

**Level 1 DFD (for Student Interface):**



**Level 1 DFD (for Admin Interface):**




**Level 2 DFD (for "Register for Exam" Process):**




This DFD depicts the basic flow of data and processes involved in an Online Examination System, including interfaces for students and administrators, processes such as registration, login, exam creation, and exam management, and data stores for student information and exam data.

# (c) What is MIS? How is it useful for management control and strategic planning in an organization?

MIS stands for Management Information System. It's a computerized system that provides managers and other decision-makers with the tools and information they need to efficiently and effectively manage an organization.

MIS collects, processes, stores, and disseminates information to support decision-making at various levels within an organization. Here's how MIS is useful for management control and strategic planning:

Data Consolidation: MIS gathers data from various sources within the organization, including transactional systems, databases, and external sources. By consolidating data into a single system, MIS provides managers with a comprehensive view of the organization's operations.

Real-time Information: MIS provides real-time or near-real-time information about the organization's performance, operations, and financials. This enables managers to make timely decisions based on the most current data available.

Performance Monitoring: MIS helps managers monitor key performance indicators (KPIs) and other metrics to track the organization's performance against goals and objectives. By identifying deviations from targets, managers can take corrective actions to address issues and improve performance.

Decision Support: MIS provides decision support tools, such as reporting and analytics, that help managers analyze data, identify trends, and evaluate alternative courses of action. This supports informed decision-making and reduces the risk of making decisions based on incomplete or inaccurate information.

Resource Allocation: MIS helps managers allocate resources, such as human resources, finances, and equipment, more effectively by providing insights into resource utilization, availability, and demand. This enables managers to optimize resource allocation and improve operational efficiency.

Strategic Planning: MIS supports strategic planning by providing managers with information about market trends, competitor activities, and internal capabilities. This enables managers to

identify opportunities and threats, formulate strategic goals and objectives, and develop plans to achieve them.

Forecasting and Predictive Analytics: MIS uses forecasting models and predictive analytics to anticipate future trends, market conditions, and customer behavior. This enables managers to proactively respond to changes in the business environment and stay ahead of the competition.

Overall, MIS plays a critical role in management control and strategic planning by providing managers with the information and tools they need to make informed decisions, monitor performance, allocate resources effectively, and plan for the future success of the organization.

# (d) What are CASE Tools? Explain the role of CASE Tools in technical and managerial aspects of software development process. List advantages and disadvantages of CASE Tools.

CASE (Computer-Aided Software Engineering) Tools are software applications that provide automated support for various stages of the software development lifecycle. These tools help developers and managers to automate tasks, streamline processes, and improve the overall efficiency and quality of software development. The primary goal of CASE tools is to enhance productivity and reduce the time and effort required to develop software systems.

Role of CASE Tools in Software Development:

**Technical Aspects:**

- Modeling and Design: CASE tools facilitate the creation of diagrams, models, and prototypes to visualize and design software systems. They provide support for various modeling techniques such as UML (Unified Modeling Language) diagrams, data flow diagrams, and entity-relationship diagrams.
- Code Generation: Some CASE tools offer code generation capabilities, allowing developers to automatically generate code from design models. This can significantly reduce the time and effort required for coding and minimize the risk of errors.
- Testing and Debugging: CASE tools provide features for automated testing, debugging, and validation of software systems. They help identify and fix defects early in the development process, leading to improved software quality.
- Version Control and Configuration Management: CASE tools often include version control and configuration management features to track changes to software artifacts, manage dependencies, and ensure consistency across development environments.

**Managerial Aspects:**

- Project Planning and Management: CASE tools support project planning, scheduling, and resource allocation by providing features for creating and tracking project tasks, milestones, and dependencies. They help managers monitor progress, identify bottlenecks, and allocate resources effectively.
- Collaboration and Communication: CASE tools facilitate collaboration and communication among team members by providing features for sharing documents, artifacts, and feedback in a centralized repository. They enable teams to work together more efficiently, even in distributed environments.
- Documentation and Reporting: CASE tools assist in documenting software requirements, design specifications, and test plans. They generate reports and documentation automatically, reducing the time and effort required for documentation tasks.

**Advantages of CASE Tools:**

- Increased Productivity: CASE tools automate repetitive tasks and provide tools for modeling, design, code generation, testing, and documentation, leading to increased productivity and efficiency.
- Improved Quality: By automating testing, debugging, and validation tasks, CASE tools help identify and fix defects early in the development process, leading to higher software quality.
- Consistency and Standardization: CASE tools enforce consistency and standardization across the development process by providing templates, guidelines, and best practices.
- Enhanced Collaboration: CASE tools facilitate collaboration and communication among team members by providing features for sharing documents, artifacts, and feedback in a centralized repository.
- Cost and Time Savings: CASE tools reduce the time and effort required for software development tasks, leading to cost savings and faster time-to-market.

**Disadvantages of CASE Tools:**

Complexity: Some CASE tools can be complex to learn and use, requiring training and expertise to fully utilize their features.

Cost: High-quality CASE tools can be expensive to acquire and maintain, especially for small organizations or individual developers.

Over-reliance: Over-reliance on CASE tools can lead to a lack of critical thinking and creativity among developers, resulting in less innovative solutions.

Compatibility Issues: CASE tools may have compatibility issues with other tools or environments, leading to integration challenges and workflow disruptions.

Maintenance: CASE tools require regular maintenance and updates to ensure compatibility with evolving technologies and standards.

Despite these disadvantages, the benefits of CASE tools generally outweigh the drawbacks, making them valuable assets in modern software development environments.

# 2. (a) Prepare an outline of SRS for an “Online Examination System”. Make necessary assumptions.

Sure, here's an outline for a Software Requirements Specification (SRS) document for an "Online Examination System":

**1. Introduction**

- Purpose of the Document
- Scope of the System
- Definitions, Acronyms, and Abbreviations
- References

**2. General Description**

- Product Perspective
- Product Functions
- User Characteristics
- Constraints
- Assumptions and Dependencies

**3. Specific Requirements**

3.1 External Interface Requirements

- User Interfaces
- Hardware Interfaces
- Software Interfaces
- Communication Interfaces

3.2 Functional Requirements

- User Registration
- User Authentication
- Exam Creation
- Exam Taking
- Exam Grading
- Reporting and Analytics
- Administration Functions

3.3 Non-Functional Requirements

- Performance Requirements
- Security Requirements
- Reliability Requirements
- Scalability Requirements
- Usability Requirements

3.4 System Quality Attributes

- Maintainability
- Portability
- Interoperability

**4. System Models**

- Use Case Diagrams
- Activity Diagrams
- Class Diagrams

**5. Other Requirements**

- Legal and Regulatory Requirements
- Documentation Requirements
- Training and Support Requirements

**6. Appendices**

- Glossary
- Index

This outline provides a structured framework for documenting the requirements of the Online Examination System. Each section addresses specific aspects of the system, including its functionality, interfaces, constraints, and quality attributes. Additional details and specific requirements would be added under each section during the development of the SRS document.

## (b) What are the duties of a systems analyst? Why technical and managerial skills are required for the job of systems analyst? Explain.

The duties of a systems analyst typically involve analyzing, designing, and implementing information systems to meet the needs of an organization. Here's a breakdown of their main responsibilities:

Analyzing Systems: Systems analysts study existing systems and processes within an organization to identify areas for improvement or automation. They gather requirements from users and stakeholders to understand the goals and functionality needed from the new or improved system.

Designing Systems: Based on the requirements gathered, systems analysts design new systems or enhancements to existing systems. This involves creating specifications, data models, process diagrams, and user interface designs to guide the development process.

Implementing Systems: Systems analysts work with developers and other IT professionals to implement and deploy the designed systems. They may oversee the development process, provide guidance to developers, and ensure that the system meets the specified requirements.

Testing and Quality Assurance: Systems analysts are responsible for testing the developed systems to ensure they function correctly and meet user needs. They may conduct various types of testing, including unit testing, integration testing, and user acceptance testing.

Documentation and Training: Systems analysts document the design, functionality, and usage of the systems they develop. They create user manuals, technical documentation, and training materials to help users understand and use the systems effectively.

Support and Maintenance: After the system is deployed, systems analysts provide ongoing support and maintenance to ensure its continued operation. They troubleshoot issues, implement updates and enhancements, and provide support to users as needed.

Now, regarding the need for both technical and managerial skills for the job of systems analyst:

Technical Skills: Systems analysts need strong technical skills to understand and work with various technologies, programming languages, databases, and development tools. They must be able to analyze complex systems, design efficient solutions, and communicate effectively with developers and other technical stakeholders.

Managerial Skills: Systems analysts also need managerial skills to effectively lead and coordinate the development process. They must be able to communicate with stakeholders, manage project timelines and resources, and prioritize tasks effectively. Additionally, they may need to negotiate with vendors, facilitate meetings, and resolve conflicts within the project team.

Overall, the job of a systems analyst requires a combination of technical expertise and managerial acumen to successfully analyze, design, and implement information systems that meet the needs of the organization.

## 3. (a) What is the need of feasibility study? Explain any four issues involved in feasibility study.

A feasibility study is conducted at the beginning of a project to assess the viability and potential success of a proposed system or project. It helps stakeholders determine whether the proposed project is feasible and worth pursuing. The need for a feasibility study arises from the following reasons:

Risk Management: A feasibility study helps identify potential risks and challenges associated with the proposed project. By evaluating feasibility factors upfront, stakeholders can make informed decisions and mitigate risks before investing significant time and resources.

Resource Allocation: Conducting a feasibility study helps stakeholders assess the resources required for the project, including financial resources, human resources, technology, and infrastructure. This enables stakeholders to allocate resources effectively and ensure that the project is feasible within the available constraints.

Decision Making: A feasibility study provides stakeholders with objective information and analysis to support decision-making. It helps stakeholders evaluate the potential benefits, costs,

and risks of the proposed project and determine whether it aligns with organizational goals and priorities.

Project Planning: The findings of a feasibility study inform the project planning process by providing insights into the project scope, objectives, timelines, and deliverables. This enables stakeholders to develop realistic project plans and set achievable goals.

**Four issues involved in a feasibility study include:**

Technical Feasibility: This issue assesses whether the proposed project can be implemented from a technical standpoint. It involves evaluating the availability of technology, infrastructure, and expertise required to develop and deploy the system.

Economic Feasibility: Economic feasibility examines whether the proposed project is financially viable and cost-effective. It involves estimating the project costs, including development, implementation, and ongoing maintenance costs, and comparing them to the expected benefits and returns on investment.

Operational Feasibility: Operational feasibility evaluates whether the proposed project will be accepted and adopted by users and stakeholders. It involves assessing the impact of the project on existing processes, workflows, and organizational culture, as well as the ability of users to adapt to the new system.

Legal and Regulatory Feasibility: This issue examines whether the proposed project complies with relevant laws, regulations, and industry standards. It involves identifying legal and regulatory requirements that may impact the project and assessing the feasibility of meeting these requirements.

Addressing these issues in a feasibility study helps stakeholders make informed decisions about whether to proceed with the proposed project and how to mitigate potential risks and challenges.

## (b) What is modularity in a system? Explain the concepts of coupling and cohesion. Is there any relationship between coupling and cohesion? Explain.

Modularity in a system refers to the degree to which a system's components (modules) are organized and structured independently yet functionally related to perform specific tasks or functions. A modular system is composed of discrete, self-contained modules that can be

developed, modified, and maintained independently without affecting the entire system. Modularity enhances flexibility, scalability, and maintainability by allowing changes to be made to individual modules without impacting other parts of the system.

Now, let's discuss the concepts of coupling and cohesion, which are closely related to modularity:

Coupling: Coupling refers to the degree of interdependence between modules in a system. It measures how closely connected or dependent one module is on another. Low coupling indicates that modules are relatively independent and can be modified or replaced without affecting other modules. High coupling, on the other hand, suggests tight interdependencies between modules, making the system more rigid and difficult to maintain. Types of coupling include:

Low Coupling: Modules are loosely connected, and changes in one module have minimal impact on other modules.

High Coupling: Modules are tightly connected, and changes in one module require modifications in multiple other modules.

Data Coupling: Modules communicate with each other through parameters or data structures.

Control Coupling: Modules share control information, such as flags or status variables.

Content Coupling: Modules share data directly, violating encapsulation principles.

Cohesion: Cohesion refers to the degree to which the elements within a module are logically related and focused on performing a single task or function. It measures the strength of the relationship between the elements within a module. High cohesion means that the elements within a module are closely related and work together to achieve a specific goal, while low cohesion indicates that the elements are loosely related and perform multiple unrelated tasks. Types of cohesion include:

Functional Cohesion: Elements within a module perform a single, well-defined task or function.

Sequential Cohesion: Elements within a module are related by sequence, where the output of one element becomes the input of the next.

Communicational Cohesion: Elements within a module operate on the same data or share related data.

Procedural Cohesion: Elements within a module are grouped based on the sequence in which they are executed.

Temporal Cohesion: Elements within a module are executed at the same time.

There is an inverse relationship between coupling and cohesion. In general, as coupling decreases (i.e., modules become more independent), cohesion tends to increase (i.e., modules become more focused and logically related). Conversely, as coupling increases (i.e., modules become more interdependent), cohesion tends to decrease (i.e., modules become less focused and more fragmented). Therefore, achieving a balance between low coupling and high cohesion is essential for designing modular, maintainable, and scalable systems.

# 4. (a) Explain the following with an example for each:

# (i) Class diagram

# (ii) E-R diagram

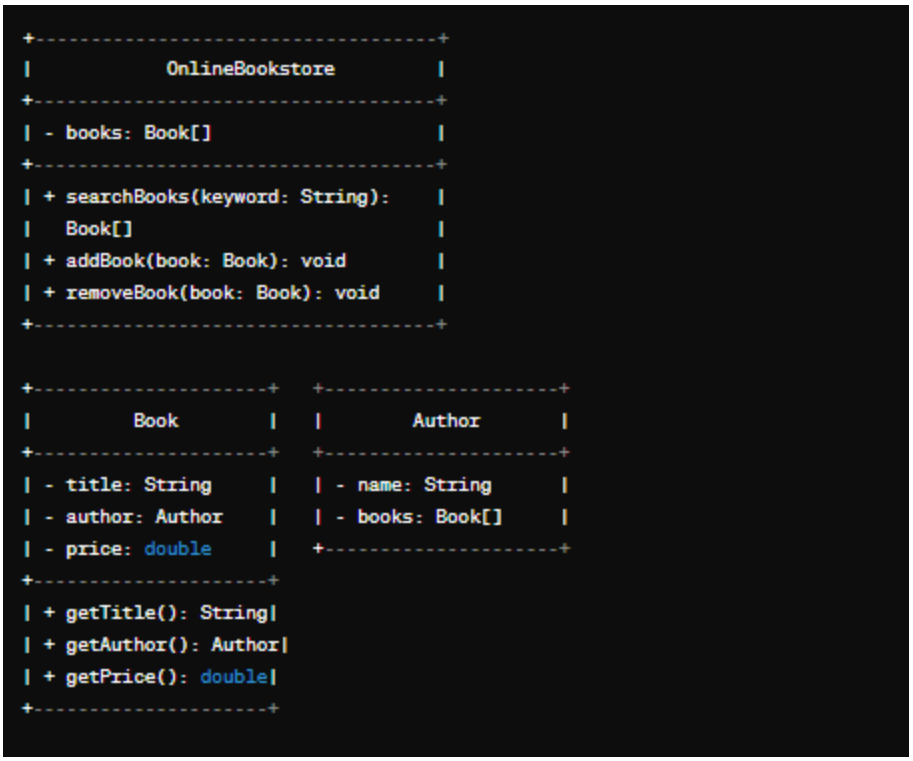
Sure, let's explain each concept with an example:

**(i) Class Diagram:**

A class diagram is a type of static structure diagram in UML (Unified Modeling Language) that represents the structure of a system by showing the classes, attributes, methods, and relationships between them. It provides a visual representation of the classes in a system and their associations.

Example:

Consider a simple online bookstore system. We can represent the classes involved in this system using a class diagram. Here's an example of a class diagram for the online bookstore system:




In this example:

There are three classes: OnlineBookstore, Book, and Author.

The OnlineBookstore class has attributes such as books and methods like searchBooks, addBook, and removeBook.

The Book class has attributes like title, author, and price, and methods like getTitle, getAuthor, and getPrice.

The Author class has attributes such as name and books.

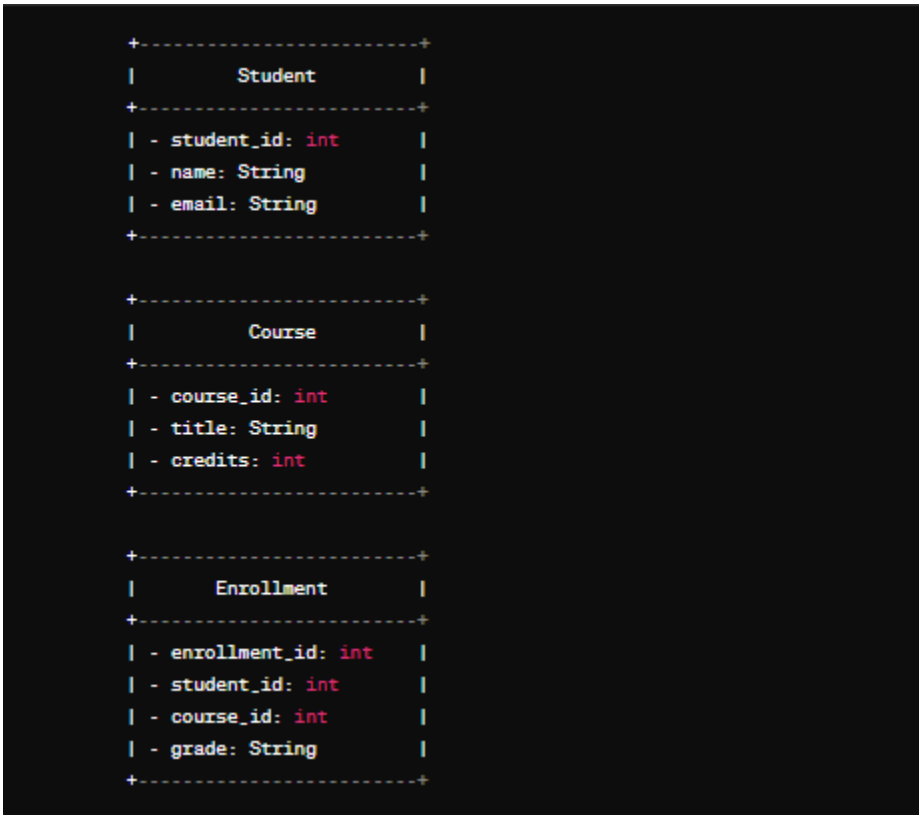
Relationships are represented using associations between classes, such as the association between Book and Author.

**(ii) E-R Diagram:**

An Entity-Relationship (E-R) diagram is a type of diagram used to visualize the relationships between entities in a database. It represents the entities (such as objects, people, or concepts), attributes (properties of entities), and relationships (associations between entities).

Example:

Consider a university database system. We can represent the entities and their relationships using an E-R diagram. Here's an example of an E-R diagram for a university database:




In this example:

There are three entities: Student, Course, and Enrollment.

Each entity has attributes such as student_id, name, email for Student, course_id, title, credits for Course, and enrollment_id, student_id, course_id, grade for Enrollment.

Relationships between entities are represented using lines connecting the entities. For example, there is a many-to-many relationship between Student and Course, which is represented by the Enrollment entity.

# (b) What is SDLC? Briefly explain all the seven phases of SDLC.

SDLC stands for Software Development Life Cycle. It is a structured process used by software development teams to design, develop, test, and deploy software systems. The SDLC framework provides a systematic approach to software development, ensuring that projects are completed efficiently, on time, and within budget.

**The seven phases of the SDLC are as follows:**

Planning: In this phase, project stakeholders define the scope, objectives, and requirements of the software project. This involves gathering requirements from stakeholders, conducting feasibility studies, and creating a project plan.

Analysis: During the analysis phase, software requirements are analyzed and documented in detail. This includes identifying user needs, defining system functionalities, and creating use cases or user stories.

Design: The design phase involves creating the architectural and technical design of the software system based on the requirements gathered in the previous phases. This includes defining system architecture, data models, user interfaces, and software components.

Implementation: In the implementation phase, the software is developed based on the design specifications. This involves writing code, building software components, integrating systems, and conducting unit testing.

Testing: During the testing phase, the software is tested to ensure that it meets the specified requirements and functions correctly. This includes various types of testing such as unit testing, integration testing, system testing, and user acceptance testing.

Deployment: The deployment phase involves deploying the software system to the production environment and making it available to users. This includes installing the software, configuring systems, and training users.

Maintenance: The maintenance phase involves maintaining and supporting the software system after it has been deployed. This includes fixing bugs, implementing updates and enhancements, and providing ongoing support to users.

These phases are often iterative, meaning that they may be revisited and repeated as needed throughout the software development process. Additionally, different SDLC models (such as waterfall, agile, or iterative) may have variations in the order or emphasis of these phases.

However, the core objectives of each phase remain consistent: to ensure that the software is developed efficiently, meets user needs, and achieves its intended goals.

# 5. Write short notes on the following:

# (a) Distributed Systems

# (b) Structure Charts

# (c) Internal and External Information

# (d) Risk Analysis

**(a) Distributed Systems:**

Distributed systems are computer systems composed of multiple interconnected computers that communicate and coordinate with each other to achieve a common goal or perform a task. These systems are designed to distribute processing, storage, and communication tasks across multiple nodes or computers, enabling scalability, fault tolerance, and resource sharing. Examples of distributed systems include cloud computing platforms, peer-to-peer networks, and distributed databases. Distributed systems pose unique challenges such as ensuring consistency, reliability, and security across distributed components, as well as managing communication and coordination between distributed nodes.

**(b) Structure Charts:**

Structure charts are graphical representations used in software engineering to depict the hierarchical structure of a software system. They illustrate the modular organization of a program by showing the relationships between modules or components. In a structure chart, each module is represented as a box, and the connections between modules indicate the flow of control or data between them. Structure charts are commonly used in structured programming methodologies such as top-down design and structured analysis to aid in the design, documentation, and understanding of complex software systems.

**(c) Internal and External Information:**

Internal information refers to data and information that is generated, stored, and used within an organization or system. It includes information about the organization's operations, processes, employees, and resources. Examples of internal information include sales records, inventory data, employee records, and financial statements. External information, on the other hand, refers to data and information that originates from sources outside the organization or system. This includes information about customers, competitors, market trends, regulatory requirements, and industry benchmarks. Effective management of internal and external information is essential for decision-making, strategic planning, and organizational success.

**(d) Risk Analysis:**

Risk analysis is the process of identifying, assessing, and mitigating risks and uncertainties associated with a project, system, or decision. It involves analyzing potential threats, vulnerabilities, and consequences to determine the likelihood and impact of adverse events occurring. Risk analysis helps organizations make informed decisions, prioritize resources, and develop risk management strategies to minimize negative outcomes and maximize opportunities. Techniques used in risk analysis include risk identification, risk assessment, risk evaluation, and risk treatment. Risk analysis is an integral part of project management, business continuity planning, and information security management.