Section A: Attempt all questions (40 Marks)

i. Briefly discuss what is meant by the term structured software design. **(2 marks)**

ii. Explain two problems with structured software design. **(2 Marks)**

iii. How does object Oriented design address the problems identified in (ii) above. **(2 marks)**

iv. Briefly explain the term iterative in relation to modern software analysis and design and explain how iterative processes contrasts to traditional systems such as the Waterfall Model. **(4 Marks)**

v. When designing data entry forms, reducing the number of input errors improves data quality. One way to reduce input errors is to eliminate unnecessary data entry. Describe four types of validation rules that you can use to ensure data quality. **(4 Marks)**

vi. When selecting architecture, explain any four items that a Systems Analyst should consider as part of the overall design checklist? **(4 Marks)**

vii. Define unit testing, integration testing, and system testing. **(3 Marks)**

viii. Describe the phases of the systems development life cycle. **(4 Marks)**

ix. What are three common reasons for systems projects? **(3 Marks)**

x. Define the term black box, and explain why it is an important concept in object-oriented analysis. **(3 Marks)**

xi. Explain the polymorphism as used in object –oriented analysis. **(2 marks)**

xii. What is enterprise resource planning (ERP) and why is it important? **(3 marks)**

xiii. Describe the Unified Modelling Language (UML) and how it can be used during systems development. **(4 marks)**

i. Structured Software Design

Structured software design is a methodology that decomposes a system into hierarchical, manageable modules using a top-down approach. It emphasizes clear organization, stepwise refinement, and functional decomposition to simplify development and maintenance.

ii. Problems with Structured Software Design

1. **Rigidity in Modification**: Changes in one module can propagate errors across dependent modules due to tight coupling.
2. **Scalability Issues**: Managing large systems becomes complex as hierarchical structures grow, leading to maintenance challenges.

iii. How Object-Oriented Design Addresses These Problems

1. **Encapsulation**: Bundles data and methods within objects, reducing unintended side effects during modifications.
2. **Modularity and Reusability**: Objects and classes promote reusable components, easing scalability and reducing redundancy.

iv. Iterative Processes in Modern Software Design

Iterative design involves incremental development cycles (e.g., Agile), where each iteration delivers a functional subset of the system, incorporates feedback, and refines the product. **Contrast with Waterfall**: Waterfall is linear (requirements → design → implementation) with no revisiting phases, while iterative allows flexibility and continuous improvement.

v. Validation Rules for Data Quality

1. **Format Check**: Ensures data matches a specified pattern (e.g., DD/MM/YYYY).
2. **Range Check**: Validates numerical input within defined limits (e.g., age 0-120).
3. **Mandatory Field Check**: Prevents submission if required fields are empty.
4. **Consistency Check**: Verifies logical relationships (e.g., end date ≥ start date).

vi. Architecture Design Checklist

A Systems Analyst should consider:

1. **Scalability**: Ability to handle future growth.
2. **Security**: Data protection mechanisms.
3. **Integration**: Compatibility with existing systems.
4. **Performance**: Speed, reliability, and resource efficiency.

vii. Testing Definitions

- **Unit Testing**: Testing individual components (e.g., functions) in isolation.
- **Integration Testing**: Verifying interactions between combined modules.
- **System Testing**: Evaluating the entire system against functional requirements.

viii. SDLC Phases

1. **Planning**: Define scope, objectives, and feasibility.
2. **Analysis**: Gather and document requirements.
3. **Design**: Create system architecture and specifications.
4. **Implementation**: Develop and code the system.
5. **Testing**: Validate functionality and fix defects.
6. **Deployment**: Release the system to users.
7. **Maintenance**: Ongoing updates and support.

ix. Common Reasons for Systems Projects

1. Outdated technology replacement.
2. Improving operational efficiency.
3. Compliance with regulatory standards.

x. Black Box Concept

A **black box** refers to analyzing a system based on inputs and outputs without knowledge of internal workings. In OO analysis, it emphasizes encapsulation, allowing objects to interact via interfaces while hiding implementation details.

xi. Polymorphism

Polymorphism enables objects of different classes to respond to the same

method call in unique ways. For example, a calculateArea() method behaves differently for Circle and Square classes.

xii. ERP and Its Importance

Enterprise Resource Planning (ERP) integrates core business processes (e.g., finance, HR, supply chain) into a unified system. It enhances data consistency, operational efficiency, and decision-making across departments.

xiii. UML in Systems Development

Unified Modeling Language (UML) is a visual modeling standard for documenting system design. Diagrams like use cases (requirements), class diagrams (structure), and sequence diagrams (interactions) aid in planning, communication, and validation during development.

TEST: 1

Instructions: Answer ALL Questions

Read the case study and use it to answer the questions below:

Mr Joseph Mukasa visited the Uganda National Theatre to pick up tickets for a play that he had made a booking for. However, due to an oversight which turns out to be rather frequent, his tickets were sold to another customer. Fortunately for Mr Mukasa, the manager was able to find 2 unclaimed tickets and handed them to Mr Mukasa. Mr Mukasa happens to own a software development company and is now interested in supporting the Theater to avoid such ticketing problems in the future and rather become more efficient and improve customer satisfaction. As a Systems Analyst who is a member of the team assigned to work on the potential Ticket/booking management system, use your knowledge of Systems development to answer questions below;

1. As a Systems Analyst on this project, what would be your key role(s)? **[4 Marks]**

b) Assuming that the Project leader has decided to use the 'Prototyping' approach to deliver the system, evaluate this chosen development approach and specify its potential benefits and weaknesses. **[6 Marks]**

c) From your knowledge of components of an Information system, identify any five (5) components of the ticketing system and give 1 example for each component identified **[10 Marks]**

2. In making a business case for the Ticket management system, what would be the purpose of conducting a feasibility study? **[2 Marks]**

b) Describe in detail what would constitute your feasibility study report. **[10 Marks]**

[END]

1a) Key Roles of a Systems Analyst

1. **Requirements Gathering**: Conduct interviews with theater staff and customers to identify pain points (e.g., ticket overselling).
2. **Process Analysis**: Evaluate the current manual ticketing workflow and propose improvements.
3. **System Design**: Create specifications for the new system (e.g., user interfaces, database structure).
4. **Stakeholder Communication**: Act as a liaison between the theater management and the development team to ensure alignment.

1b) Evaluation of Prototyping Approach

Benefits:

1. **Early User Feedback**: Theater staff can interact with a prototype to refine requirements (e.g., seat selection interface).
2. **Reduced Misunderstandings**: Visual prototypes clarify expectations better than textual specifications.
3. **Flexibility**: Allows iterative adjustments based on stakeholder input (e.g., adding a payment gateway).

Weaknesses:

1. **Scope Creep**: Users may demand additional features beyond the original scope, increasing costs.
2. **Time-Consuming**: Multiple iterations can delay final delivery.
3. **Misinterpretation**: Stakeholders might mistake the prototype for the final product, leading to dissatisfaction.

1c) Components of the Ticketing System

1. **Hardware**: Ticket kiosks for self-service bookings.
2. **Software**: Online reservation platform (e.g., web/mobile app).
3. **Data**: Customer details (names, contact information, booking history).
4. **Processes**: Automated ticket allocation to prevent overselling.
5. **People**: Customer support agents assisting with booking issues.

2a) Purpose of a Feasibility Study

To assess whether the proposed ticketing system is **technically achievable**, **economically viable**, and **operationally beneficial** for the theater. It ensures resources are invested wisely.

2b) Feasibility Study Report Contents

1. **Introduction**: Overview of the current problem (e.g., ticket mismanagement).
2. **Technical Feasibility**: Evaluate if existing IT infrastructure supports the system (e.g., server capacity).
3. **Economic Feasibility**:
    - **Costs**: Development, hardware, training.
    - **Benefits**: Reduced errors, increased ticket sales, improved customer satisfaction.
    - **ROI Analysis**: Compare costs to projected revenue gains.
4. **Legal Feasibility**: Compliance with data protection laws (e.g., GDPR for customer data).
5. **Operational Feasibility**: Assess staff readiness to adopt the new system.

6. **Schedule Feasibility**: Timeline for development, testing, and deployment.
7. **Recommendations**: Proceed, modify, or abandon the project based on findings.

Question 1

a) Briefly explain each of the following (use an illustration where applicable)
   i. Structured Systems Analysis and Design (5 marks)
   ii. Object Oriented Analysis and Design (5 marks)
   iii. Business process re-engineering (5 marks)
   iv. The systems development life cycle (5 marks)
   v. Prototyping (5 marks)
   vi. A decision tree (5 marks)

b) Using an illustration in each case, explain the following basic concepts of Object Orientation
   i. Generalization (5 marks)
   ii. The sequence diagram (5 marks)

i. Structured Systems Analysis and Design (SSAD)
SSAD is a linear, process-driven methodology that breaks down a system into functional components using tools like **data flow diagrams (DFDs)** and **flowcharts**. It focuses on modularity, stepwise refinement, and hierarchical structure.
Illustration: A DFD showing how a "Ticket Booking" process flows from user input → validation → database update → confirmation.

ii. Object-Oriented Analysis and Design (OOAD)
OOAD models systems as interacting objects with attributes and methods. Key concepts include **encapsulation**, **inheritance**, and **polymorphism**. UML diagrams (e.g., class diagrams) are used for visualization.
Example: A "Ticket" class with attributes (seat number, price) and methods (reserve(), cancel()).

iii. Business Process Re-engineering (BPR)
BPR involves radically redesigning core business processes to achieve

dramatic improvements in efficiency and customer satisfaction. It often leverages technology to eliminate redundant steps.
Example: Replacing manual ticket allocation with an automated system to prevent overselling.

iv. Systems Development Life Cycle (SDLC)
SDLC is a framework for developing systems through phases:

1. **Planning** (feasibility study),
2. **Analysis** (requirements gathering),
3. **Design** (system architecture),
4. **Implementation** (coding),
5. **Testing** (validation),
6. **Deployment** (launch),
7. **Maintenance** (updates).
   Illustration: Waterfall model diagram showing sequential phases.

v. Prototyping
Prototyping involves creating a preliminary model (prototype) of the system to validate requirements and gather feedback. Types include **throwaway** (quick mockups) and **evolutionary** (iterative refinement).
Example: A clickable UI prototype for the theater's online booking system.

vi. Decision Tree
A decision tree is a graphical tool for mapping decisions and their possible outcomes. Nodes represent decisions or events, and branches represent outcomes.

2i. Generalization
Generalization creates a hierarchical relationship where a **superclass** (parent) shares common attributes/methods with **subclasses** (children). It promotes code reusability.

2ii. Sequence Diagram

A sequence diagram visualizes interactions between objects over time, showing the order of messages exchanged.

Example: User login process:

Nile House is a carpentry shop that makes all kinds of furniture. You have been contacted to create an information system for them. When customers place orders at the company website, the system should check to see if the items are in stock, issues a status message to the customer and generates a delivery order to the warehouse, where the order is filled. The customer is required to make payment before the order is delivered. The system also produces various reports.

a) Draw a context diagram for the Order system **[4 Marks]**
b) Draw a DFD 0 diagram for the order system **[6 Marks]**

a) Context Diagram for the Order System (4 Marks)

A **context diagram** is a high-level, simplified view of the system showing:

- The **system as a single process** (a circle).
- The **external entities** interacting with it.
- The **data flows** between them.

Context Diagram Components

External Entities:

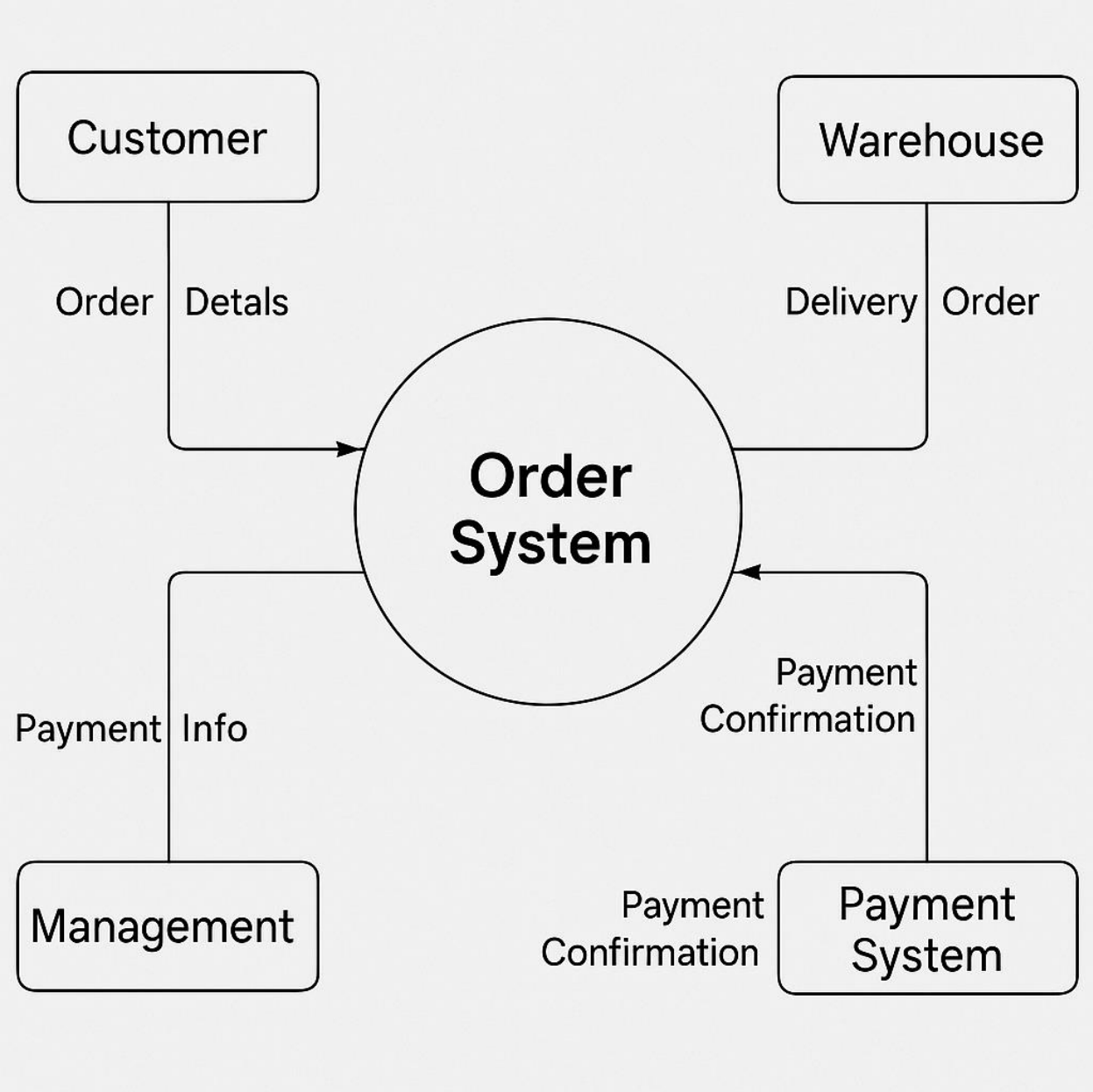
- Customer
- Warehouse
- Payment System (or Bank)
- Management (for reports)

System:

- Order System (the central process)

Data Flows:

- Customer → Order Details → Order System
- Order System → Order Status → Customer
- Order System → Delivery Order → Warehouse
- Order System → Reports → Management
- Customer → Payment Info → Payment System
- Payment Confirmation → Order System



b) DFD Level 0 (DFD 0) Diagram (6 Marks)

A **Level 0 Data Flow Diagram** breaks down the main system into **sub-processes**, shows **data stores**, and defines how data flows between them.

DFD 0 Components

Processes:

1. Take Order

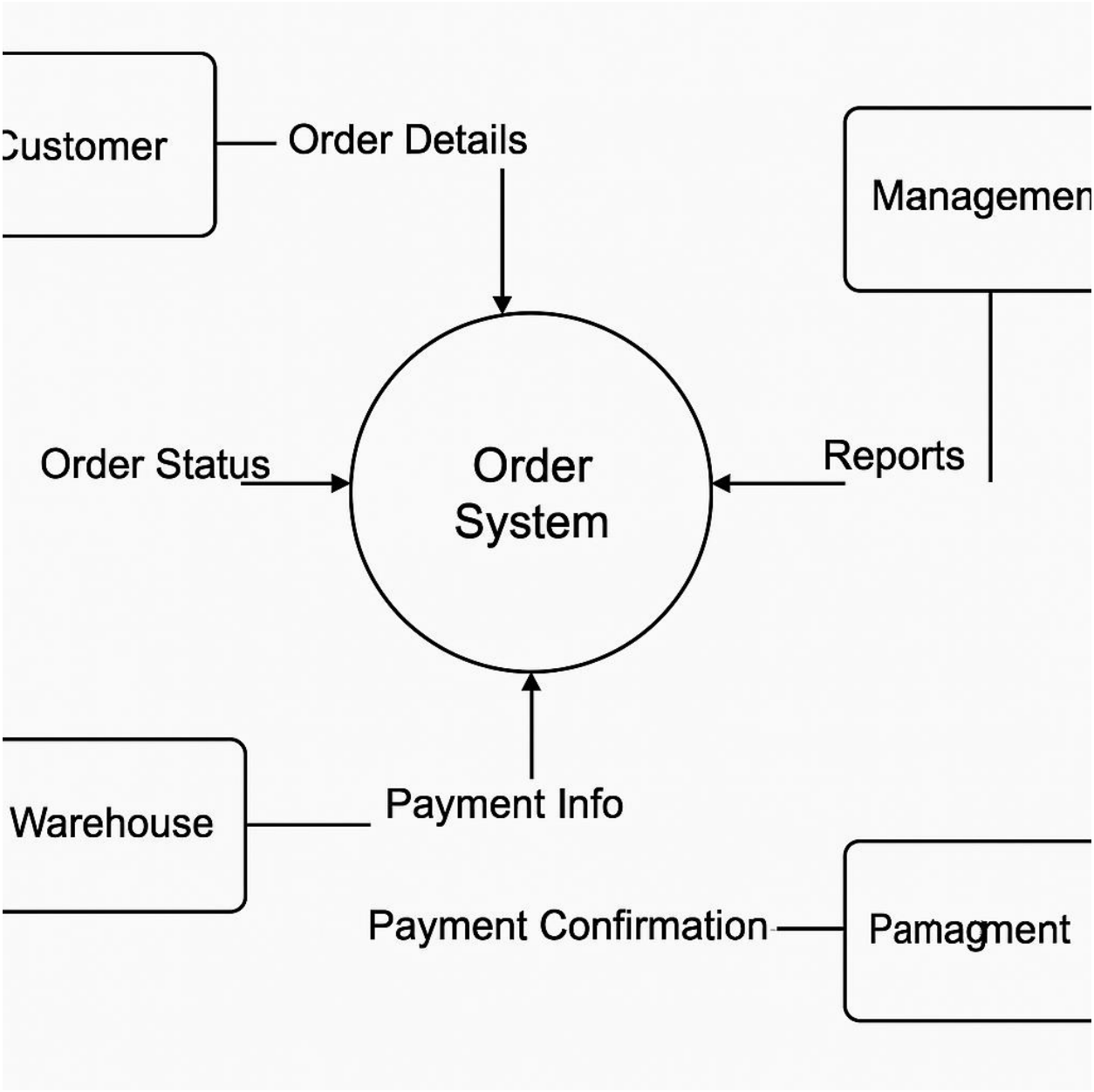
2. Check Inventory
3. Process Payment
4. Generate Delivery Order
5. Generate Reports

Data Stores:

- Inventory
- Orders
- Payments
- Reports

Entities (same as context):

- Customer
- Warehouse
- Payment System
- Management


Customer
Order Details
Managemen
Order Status
Order System
Reports
Warehouse
Payment Info
Payment Confirmation
Pamagment

Question 5

Uganda Registration Services Bureau (URSB) is planning to acquire a new Payroll system. The IT director opened the department staff meeting today by saying "I've got some good news and some bad news. The good news is that management approved the payroll system project this morning. The new system will reduce clerical time and errors, improve morale in the payroll department, and avoid possible fines and penalties for noncompliance. The bad news is that the system must be installed by the end of December in order to meet new URA reporting rules on Pay as you Earn (PAYE), costs must be within the budgeted amount, the new system must interact with existing systems, and the Finance Manager insists on approving the final design."

a) Name the constraints and indicate whether each is present, future, internal, external, mandatory, or desirable. **(8 Marks)**
b) Explain why it is important to define the payroll project's scope. Explain how to define project scope. **(6 Marks)**
c) Identify tangible and intangible benefits of the new payroll system. **(4 Marks)**
d) What topics should be included in a report to management at the end of the preliminary investigation? **(4 Marks)**

a) Constraints — Indicate if Present/Future, Internal/External, Mandatory/Desirable (8 Marks)

Constraint	Present/Future	Internal/External	Mandatory/Desirable
Must be installed by end of December	Future	Internal	Mandatory
Must meet new URA PAYE reporting rules	Future	External	Mandatory
Costs must be within the budgeted amount	Present	Internal	Mandatory
Must interact with existing systems	Future	Internal	Mandatory
Finance Manager must approve final design	Present	Internal	Mandatory
Reduce clerical time and errors	Future	Internal	Desirable
Improve morale in the payroll department	Future	Internal	Desirable
Avoid fines and penalties for noncompliance	Future	External	Desirable

b) Why It's Important to Define Project Scope & How to Define It (6 Marks)

Importance of Defining Scope:

- Clarifies project boundaries, deliverables, and expectations.
- Prevents scope creep and resource wastage.
- Helps in planning time, cost, and resource allocation.
- Ensures stakeholder alignment and satisfaction.

How to Define Project Scope:

1. **Identify Objectives** – What the payroll system should achieve (e.g. compliance, automation).
2. **List Deliverables** – Functional system, documentation, training.
3. **Determine Inclusions/Exclusions** – What is and isn't part of the project.
4. **Define Constraints** – Budget, time, tech stack, integration with existing systems.
5. **Consult Stakeholders** – Finance manager, IT team, payroll department.
6. **Document and Approve** – Prepare a scope statement and get sign-off.

---

c) Tangible and Intangible Benefits of the New Payroll System (4 Marks)

Tangible Benefits:

- Reduced processing time and costs.
- Lower risk of fines due to better compliance.
- Accurate tax and payroll reporting.

Intangible Benefits:

- Improved employee morale and satisfaction.
- Better decision-making with accurate data.
- Enhanced company reputation with timely tax compliance.

---

d) Topics for Preliminary Investigation Report to Management (4 Marks)

- **Project objectives and justification.**
- **Constraints and risks** (e.g. deadlines, compliance).
- **Requirements analysis** (functional and technical).
- **Stakeholder analysis and approvals needed.**
- **Feasibility study** (technical, financial, operational).
- **Cost estimates and budget alignment.**
- **Implementation timeline and milestones.**
- **Recommendations and next steps.**