# Team Orbit

Rayyan Ibrahim B23110006137

Muhammad Mujtaba B23110006107

Muhammad Hasan Khan B23110006099

Muhammad Hamza Abbas B23110006096

Husain Ahmed Khan B23110006045

# Design Methodologies, Diagrams, and Testing Types in Software Project Management

In software project management, understanding design methodologies, diagrams, and testing types is essential for effective planning, communication, and quality assurance. Below is a well-structured overview presented in a professional yet approachable manner.

## 1. Design Methodologies

Design methodologies provide a framework for organizing and executing software development projects. Some commonly used methodologies include:

• Waterfall Model – A traditional, step-by-step process where each phase (requirements, design, implementation, testing, deployment) must be completed before moving to the next.  
• Agile Methodology – An iterative and flexible approach that emphasizes customer collaboration and rapid delivery. Frameworks such as Scrum and Kanban fall under this category.  
• Spiral Model – Combines iterative development with continuous risk assessment and refinement.  
• V-Model (Verification and Validation) – Aligns development phases with corresponding testing phases, ensuring quality at every step.  
• Rapid Application Development (RAD) – Focuses on quick prototyping and user feedback for faster delivery.  
• Incremental Model – Breaks the project into smaller, manageable increments that are developed and tested individually.

## 2. Types of Design Diagrams

Design diagrams are visual tools that help communicate system structure, workflow, and behavior. Common types include:

• Structural Diagrams:  
 - Class Diagram – Depicts classes, their attributes, and relationships.  
 - Component Diagram – Shows the software components and their interdependencies.  
 - Deployment Diagram – Maps software elements to hardware resources.  
 - Package Diagram – Groups related classes or components for better organization.  
  
• Behavioral Diagrams:  
 - Use Case Diagram – Illustrates system functionality and user interactions.  
 - Sequence Diagram – Displays how objects interact in a specific sequence over time.  
 - Activity Diagram – Represents workflows or processes within the system.  
 - State Diagram – Describes how an object transitions between states based on events.  
  
• Other Design Tools:  
 - Entity-Relationship Diagram (ERD) – Models database structures.  
 - Data Flow Diagram (DFD) – Shows the movement of data between processes.  
 - Flowchart – Represents process logic and decision-making.  
 - Wireframes/Mockups – Provide visual prototypes of the user interface.

## 3. Types of Software Testing

Software testing ensures that a system works as intended and meets its requirements. Major categories include:

• Based on Scope:  
 - Unit Testing – Tests individual components or functions.  
 - Integration Testing – Ensures combined modules work correctly together.  
 - System Testing – Validates the entire system against its specifications.  
 - Acceptance Testing – Confirms the software is ready for deployment, often involving User Acceptance Testing (UAT).  
• Based on Approach:  
 - White-Box Testing – Examines internal logic and code paths.  
 - Black-Box Testing – Focuses on inputs and outputs, without regard to internal code.  
 - Gray-Box Testing – Combines aspects of both white-box and black-box testing.  
• Based on Purpose:  
 - Functional Testing – Ensures individual features perform as expected.  
 - Non-Functional Testing – Evaluates performance, security, usability, scalability, and compatibility.  
 Examples include performance testing (load, stress), security testing, and usability testing.  
• Other Types:  
 - Regression Testing – Ensures new changes do not break existing functionality.  
 - Smoke Testing – Performs a quick check to verify core functionality.  
 - Sanity Testing – Focuses on validating small changes or fixes.  
 - Alpha and Beta Testing – Pre-release testing by developers (alpha) or real users (beta).  
 - Exploratory Testing – Involves unscripted, ad-hoc testing to discover unexpected issues.