SNS COLLEGE OF ENGINEERING

Kurumbapalayam (po), Coimbatore – 641 107

Accredited by NAAC-UGC with 'A' Grade

Approved by AICTE & Affiliated to Anna University, Chennai

DEPARTMENT OF INFORMATION TECHNOLOGY

Unit 3

User Interface Design

User Interface Design (UI Design)

User Interface Design (UI Design) is the process of designing the layout, look, and feel of an application that users interact with. This includes buttons, icons, text, and how everything is arranged on the screen.

In Object-Oriented Software Engineering (OOSE), **UI design** is created by focusing on how users will interact with the application's **objects** (like a "Book" or "Order"), and how the interface will show the user data or actions related to these objects.

Concepts in UI Design

1. User-Centered Design (UCD)

User-Centered Design (UCD) means designing the application to meet the needs of the user. The main goal is to make the system easy for the user to understand and use.

- **Example**: Think of a **shopping app**. If users often want to filter products by category (e.g., clothes, electronics), UCD would focus on making that filter button large, easy to find, and use.

2. Usability

Usability is about how easy it is for a user to use the app. A good UI should be simple, easy to learn, and allow users to complete tasks without confusion or errors.

- **Example**: Imagine using a **calculator app**. If all the buttons are in the same order and easy to understand (numbers at the top, functions like "+" or "-" clearly separated), it makes the app easy to use and learn quickly.

3. Consistency

Consistency ensures that similar actions or elements behave in the same way, making it easier for users to predict how things work.

- **Example**: In a **social media app**, if the "Home" button is always at the bottom left of the screen, users will easily find it every time they open the app, even if the content changes.

4. Feedback

Feedback is how the app lets users know that their actions were successful or if there was an error. It's important to give clear, immediate feedback.

- **Example**: If you press the "Submit" button on a **contact form**, the system should show a message like "Thank you for submitting!" to let you know that your action was successful. If you missed a required field, the app should show an error message like "Please fill out all fields."

UI Design Patterns

UI design patterns are solutions to common design problems, and they help make the application easier to use. Let's look at a few patterns:

1. Model-View-Controller (MVC)

MVC is a design pattern that separates the app into three parts:

- **Model**: The data (e.g., a list of tasks or books).
- **View**: How the data is displayed (e.g., a list on the screen).

- **Controller**: The logic that handles user input (e.g., when you click a button, the controller decides what happens).
- **Example**: In a **task management app**:
    - **Model**: The list of tasks.
    - **View**: The screen showing tasks and buttons to add or remove tasks.
    - **Controller**: The code that adds, deletes, or updates tasks when the user interacts with the app (e.g., clicking a button).

2. Observer Pattern

The **Observer Pattern** helps keep the user interface up-to-date when the data changes. The app (model) sends updates to all parts of the UI that need to show the latest data.

- **Example**: In a **stock market app**, when the stock price changes, the model updates the price, and the UI (e.g., price charts) refreshes automatically.

3. Composite Pattern

The **Composite Pattern** is used when you need to treat both individual elements and groups of elements in the same way. It allows you to build complex interfaces from simple components.

- **Example**: In a **file explorer app**, a folder can contain both files and other folders. The app treats both files and folders similarly, allowing users to click on either and navigate through the system.

UI Design Guidelines

1. Visual Design Principles

These are basic rules to make the app look clean and easy to use:

- **Balance**: The screen should look evenly spaced, not too crowded on one side.
- **Alignment**: Items should be aligned properly (e.g., buttons in a row).

- **Contrast**: Important items (like buttons) should stand out by using colors that are different from the background.
- **Hierarchy**: Make the most important items more noticeable. For example, the **"Submit"** button might be bigger or in a different color than other buttons.
- **Example**: In a **news website**, the most important story might be placed in the center of the page with large, bold text, making it easy for users to notice first.

2. Interaction Design

This is about how users interact with the system:

- **Consistency**: Buttons that do similar things should look similar.
- **Simplicity**: Don't overload users with too many options or complex processes.
- **Affordance**: Elements should look like they can be interacted with. A button should look "clickable."
- **Example**: In a **photo gallery app**, a photo thumbnail that is clickable should have a border or shadow to show that it can be clicked on. A button to "Delete" a photo should look different from a button to "Like" a photo, so users know what they do.

UI Design in Object-Oriented Software

In Object-Oriented Software Engineering, the UI interacts with **objects** that represent real-world things. For example, a **Book** object might contain information like title, author, and price. The UI would allow users to view and modify this data.

Example: In a **bookstore app**:

- **Model**: The data representing a book (e.g., title, author).
- **View**: The UI showing a list of books, with buttons to add, edit, or remove a book.
- **Controller**: The part of the system that responds to user actions (e.g., when a user clicks on "Add Book", the controller adds the new book to the list).