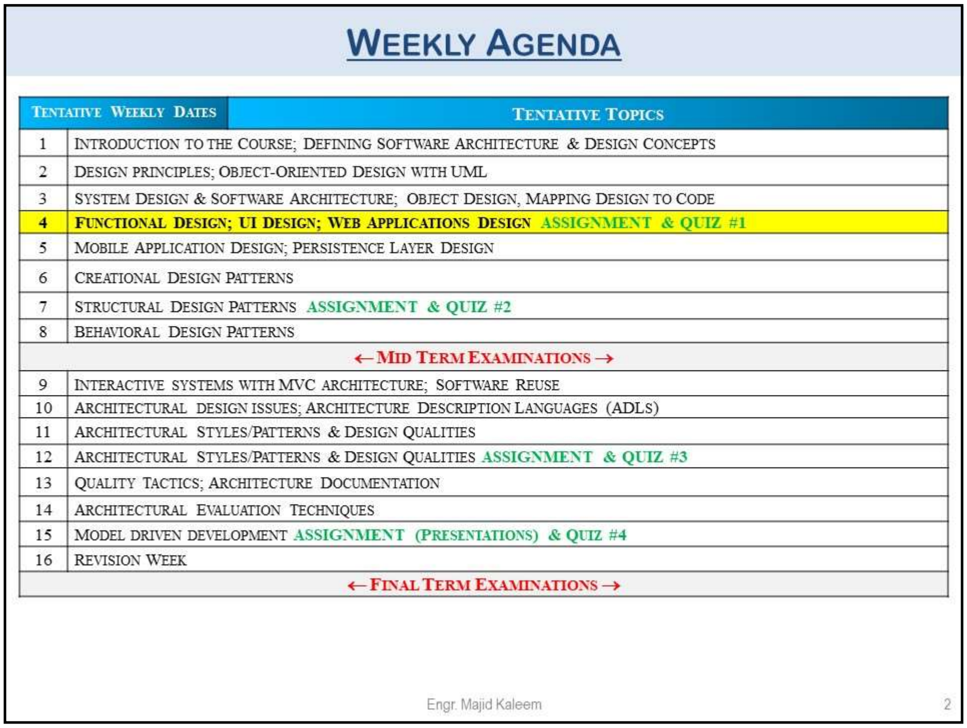
Bahria University-Karachi Campus

Software Design & Architecture

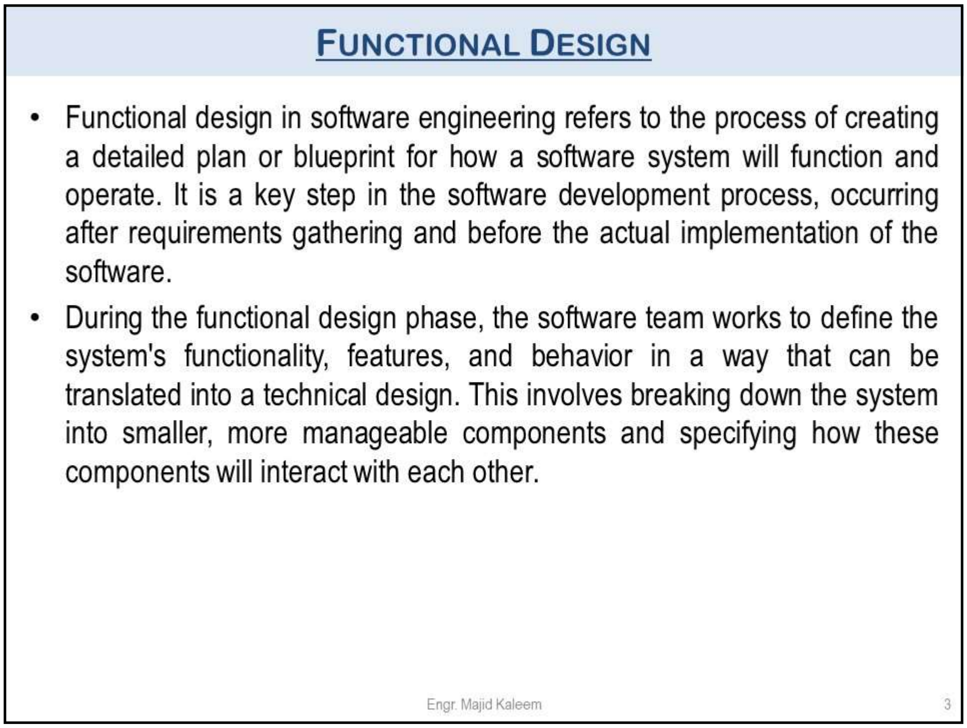
Lecture 4 of 16

Engr. Majid Kaleem



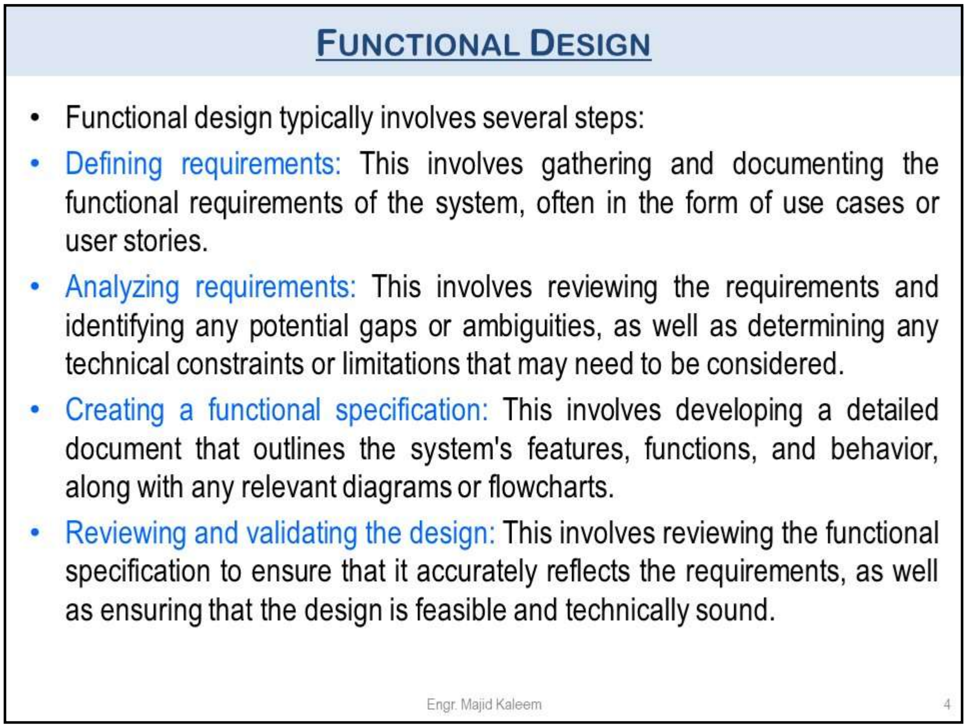
Weekly Agenda

Tentative Weekly Dates	Tentative Topics
1	Introduction to the Course; Defining Software Architecture & Design Concepts
2	Design principles; Object-Oriented Design with UML
3	System Design & Software Architecture; Object Design, Mapping Design to Code
4	**Functional Design; UI Design; Web applications Design** Assignment & Quiz #1
5	Mobile Application Design; Persistence Layer Design
6	Creational Design Patterns
7	Structural Design Patterns Assignment & Quiz #2
8	Behavioral Design Patterns
	← Mid Term Examinations →
9	Interactive systems with MVC architecture; Software Reuse
10	Architectural design issues; Architecture Description Languages (ADLs)
11	Architectural Styles/Patterns & Design Qualities
12	Architectural Styles/Patterns & Design Qualities Assignment & Quiz #3
13	Quality Tactics; Architecture Documentation
14	Architectural Evaluation Techniques
15	Model driven development Assignment (Presentations) & Quiz #4
16	Revision Week
	← Final Term Examinations →



FUNCTIONAL DESIGN

- Functional design in software engineering refers to the process of creating a detailed plan or blueprint for how a software system will function and operate. It is a key step in the software development process, occurring after requirements gathering and before the actual implementation of the software.
- During the functional design phase, the software team works to define the system's functionality, features, and behavior in a way that can be translated into a technical design. This involves breaking down the system into smaller, more manageable components and specifying how these components will interact with each other.



FUNCTIONAL DESIGN

- Functional design typically involves several steps:
- Defining requirements: This involves gathering and documenting the functional requirements of the system, often in the form of use cases or user stories.
- Analyzing requirements: This involves reviewing the requirements and identifying any potential gaps or ambiguities, as well as determining any technical constraints or limitations that may need to be considered.
- Creating a functional specification: This involves developing a detailed document that outlines the system's features, functions, and behavior, along with any relevant diagrams or flowcharts.
- Reviewing and validating the design: This involves reviewing the functional specification to ensure that it accurately reflects the requirements, as well as ensuring that the design is feasible and technically sound.

FUNCTIONAL DESIGN

- The Functional Specification Document (FSD) is written by the project's Business Analyst (BA) and provides detailed information on ***how*** the system solution will function based on what the requested behavior is.

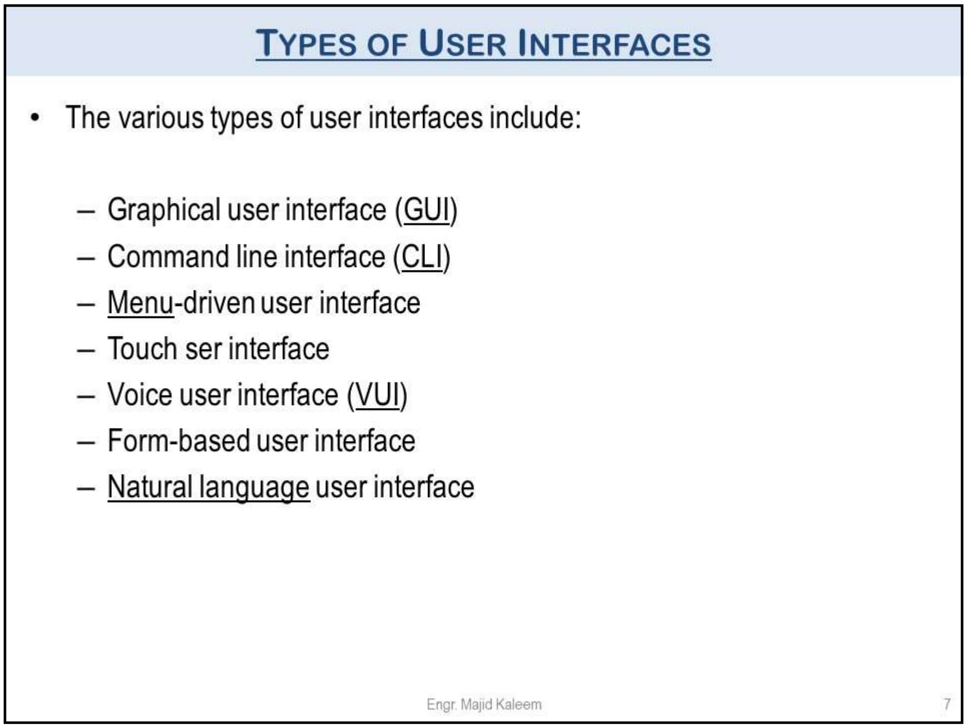
- This document is created based on the high-level requirements identified in the Business Requirements Document (SRS) and provides traceability from the functional specification back to the business requirements.

- Source: https://uit.stanford.edu/pmo/functional-design

USER INTERFACE DESIGN

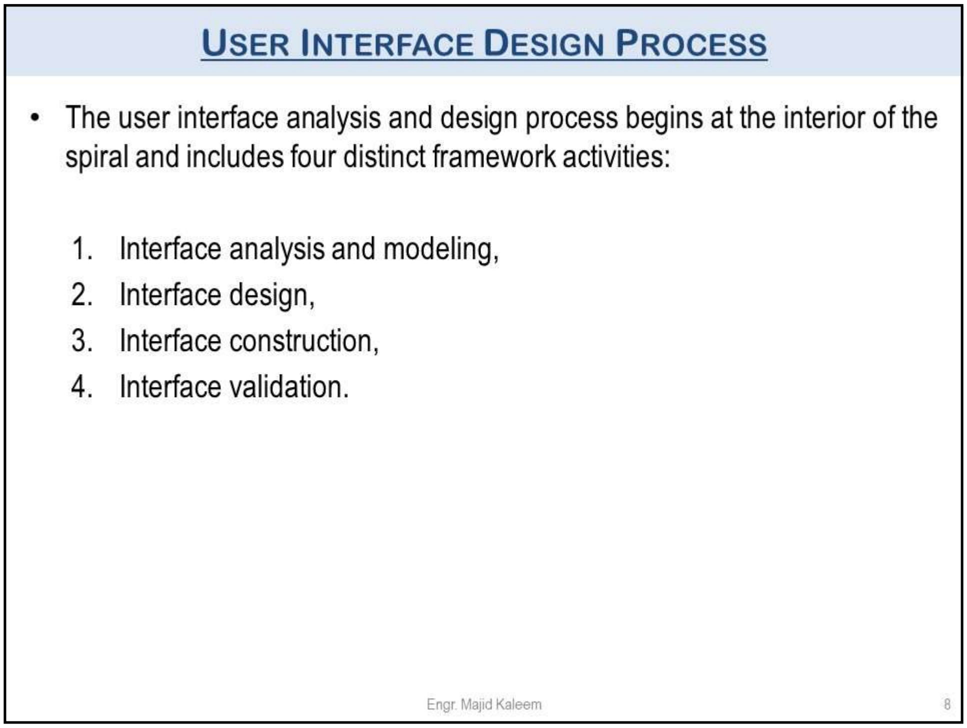
- The means by which the user and a computer system interact, in particular the use of input devices and software.
- The user interface (UI) is the point of human-computer interaction and communication in a device.
- This can include display screens, keyboards, a mouse and the appearance of a desktop.
- It is also the way through which a user interacts with an application or a website.
- The growing dependence of many businesses on web applications and mobile applications has led many companies to place increased priority on UI in an effort to improve the user's overall experience.

Courses in BSE:
Human-Computer Interaction
Usability Engineering



TYPES OF USER INTERFACES

- The various types of user interfaces include:
  - Graphical user interface (GUI)
  - Command line interface (CLI)
  - Menu-driven user interface
  - Touch ser interface
  - Voice user interface (VUI)
  - Form-based user interface
  - Natural language user interface



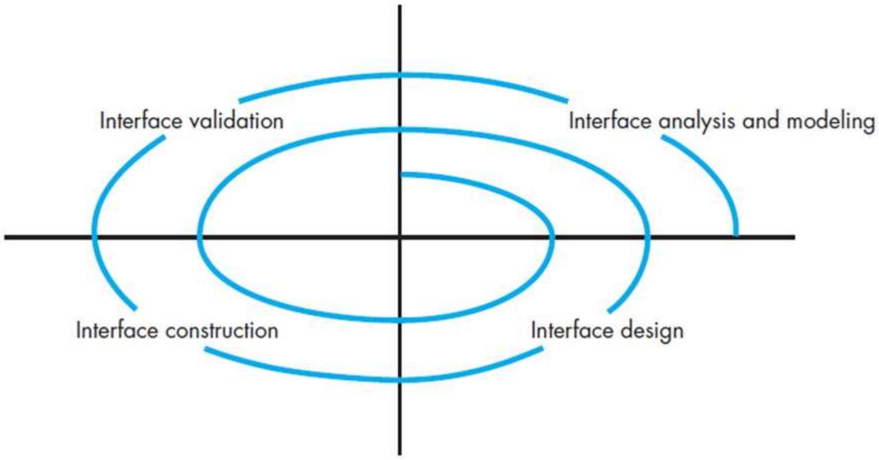
USER INTERFACE DESIGN PROCESS

- The user interface analysis and design process begins at the interior of the spiral and includes four distinct framework activities:

1. Interface analysis and modeling,
2. Interface design,
3. Interface construction,
4. Interface validation.

USER INTERFACE DESIGN PROCESS

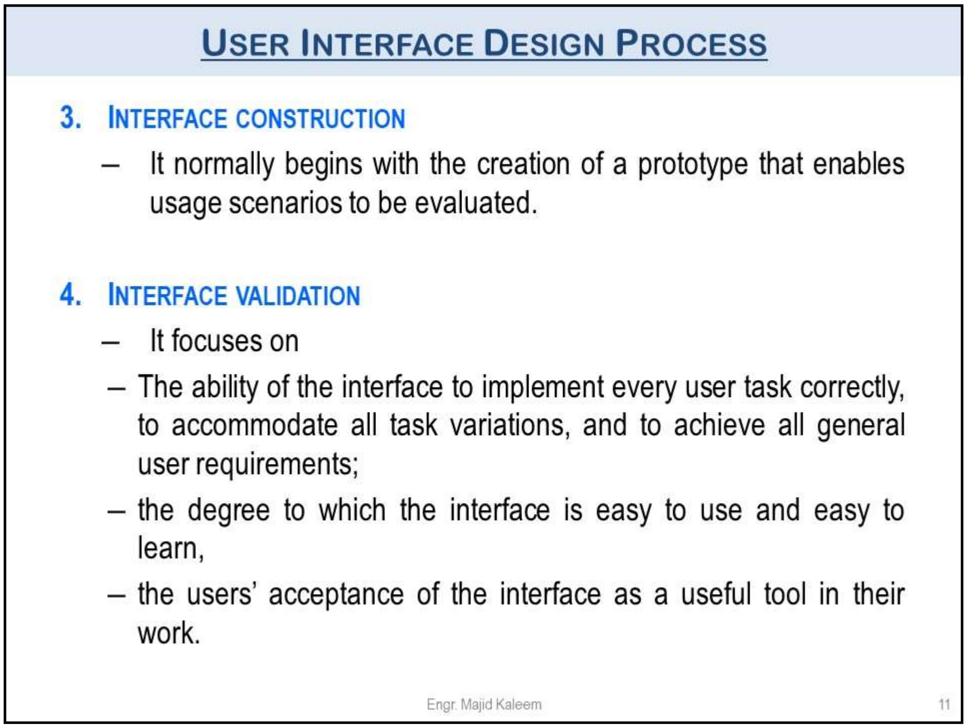
- The spiral shown in Figure implies that each of these tasks will occur more than one time



USER INTERFACE DESIGN PROCESS

1. **INTERFACE ANALYSIS :**
   - It focuses on the profile of the users who will interact with the system.
   - Skill level, business understanding, and general , different user categories are defined. For each user category, requirements are elicited.

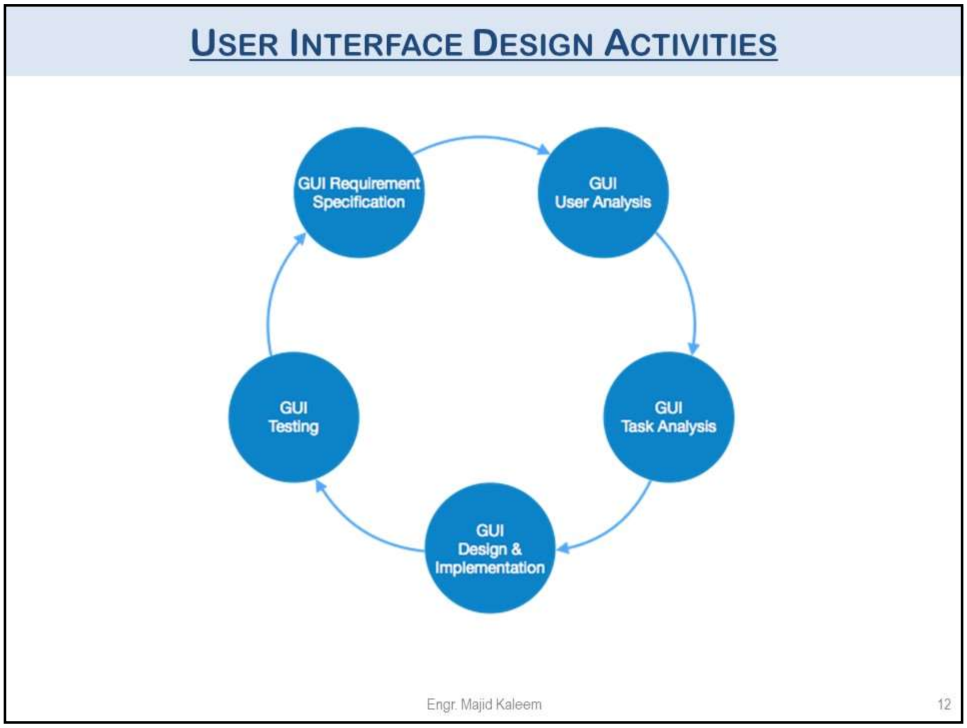
2. **THE GOAL OF INTERFACE DESIGN**
   - It is to define a set of interface objects and actions (and their screen representations) that enable a user to perform all defined tasks in a manner that meets every usability goal defined for the system.



USER INTERFACE DESIGN PROCESS

3. **INTERFACE CONSTRUCTION**
   - It normally begins with the creation of a prototype that enables usage scenarios to be evaluated.

4. **INTERFACE VALIDATION**
   - It focuses on
   - The ability of the interface to implement every user task correctly, to accommodate all task variations, and to achieve all general user requirements;
   - the degree to which the interface is easy to use and easy to learn,
   - the users' acceptance of the interface as a useful tool in their work.



USER INTERFACE DESIGN ACTIVITIES

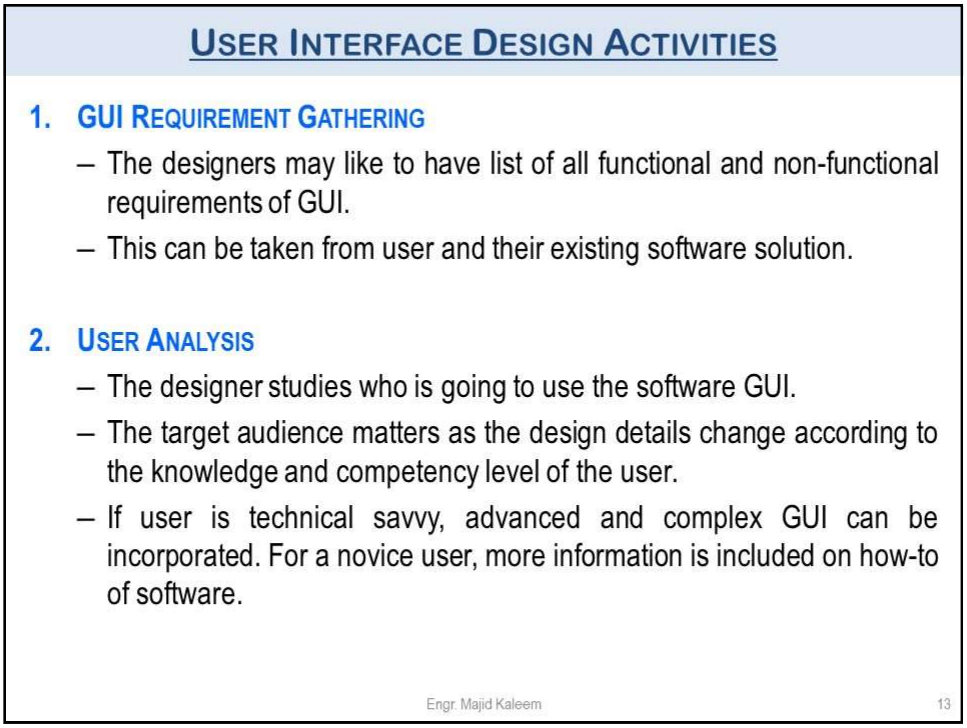
GUI Requirement Specification

GUI User Analysis

GUI Task Analysis

GUI Design & Implementation

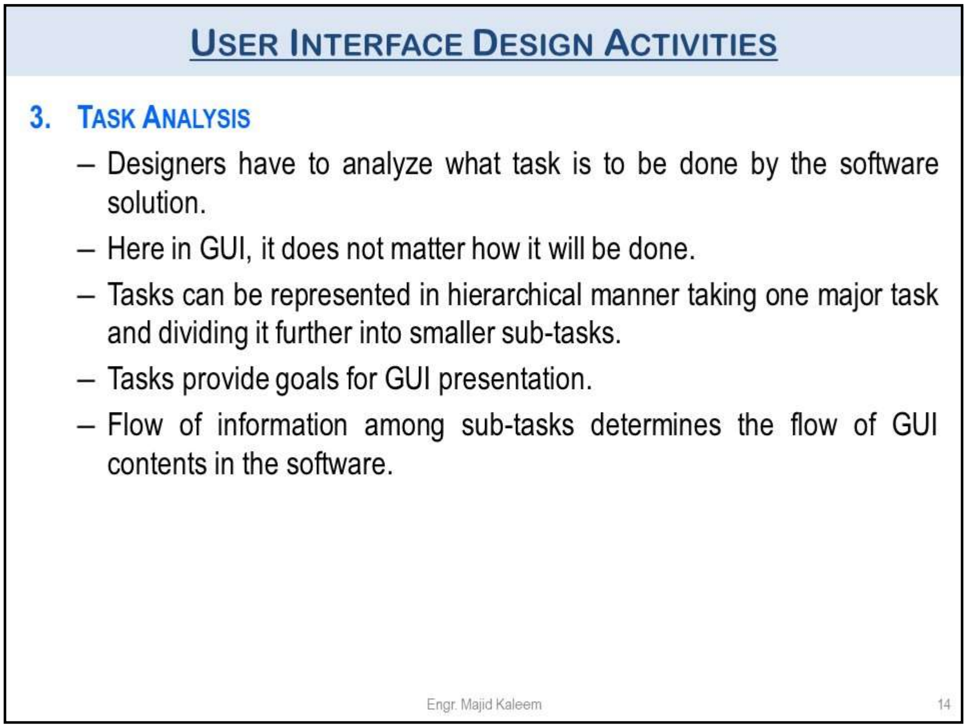
GUI Testing



USER INTERFACE DESIGN ACTIVITIES

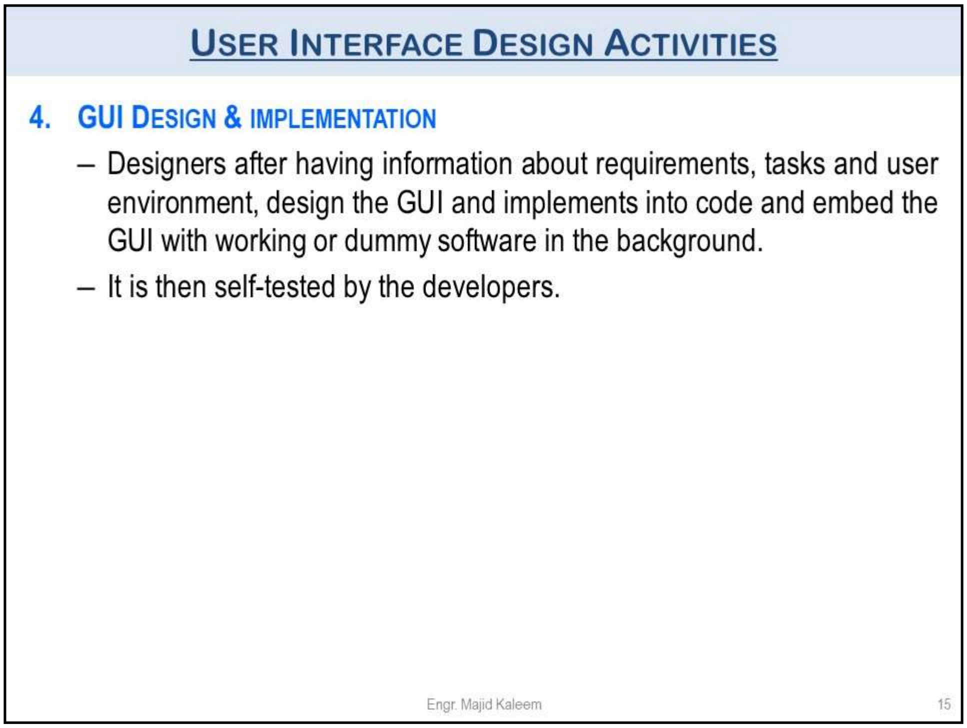
1. **GUI REQUIREMENT GATHERING**
   - The designers may like to have list of all functional and non-functional requirements of GUI.
   - This can be taken from user and their existing software solution.

2. **USER ANALYSIS**
   - The designer studies who is going to use the software GUI.
   - The target audience matters as the design details change according to the knowledge and competency level of the user.
   - If user is technical savvy, advanced and complex GUI can be incorporated. For a novice user, more information is included on how-to of software.



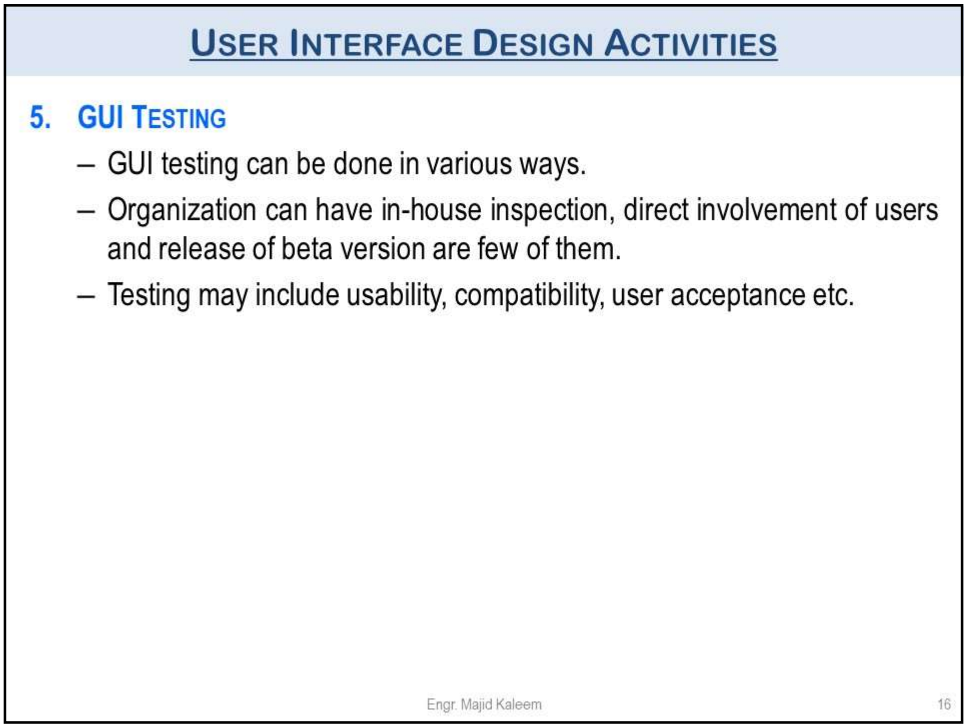
USER INTERFACE DESIGN ACTIVITIES

3. **TASK ANALYSIS**
   - Designers have to analyze what task is to be done by the software solution.
   - Here in GUI, it does not matter how it will be done.
   - Tasks can be represented in hierarchical manner taking one major task and dividing it further into smaller sub-tasks.
   - Tasks provide goals for GUI presentation.
   - Flow of information among sub-tasks determines the flow of GUI contents in the software.



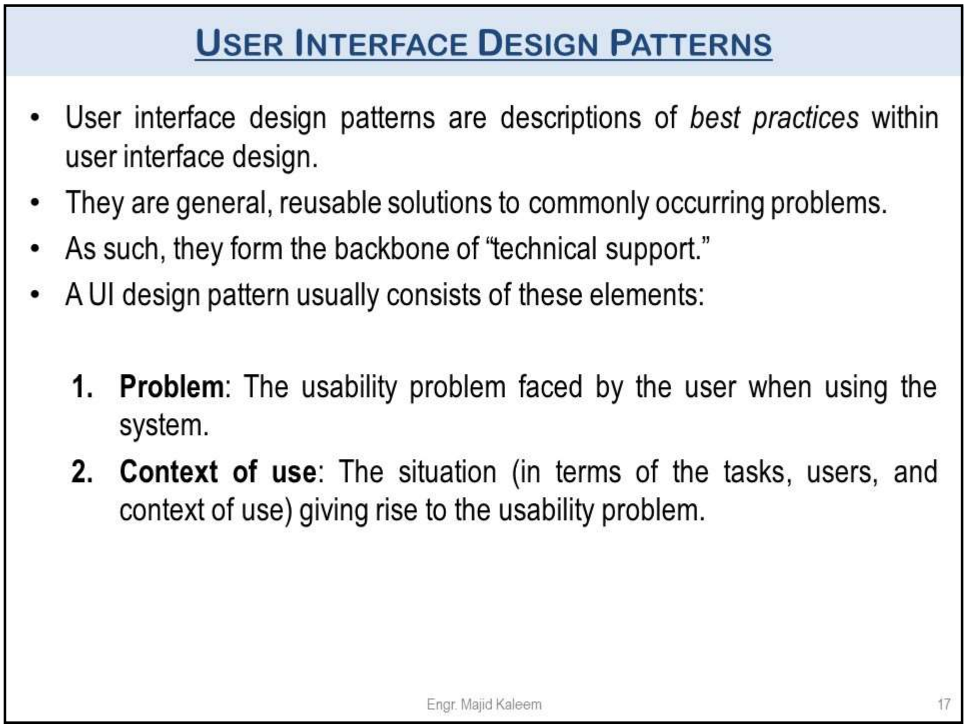
USER INTERFACE DESIGN ACTIVITIES

4. **GUI DESIGN & IMPLEMENTATION**
   - Designers after having information about requirements, tasks and user environment, design the GUI and implements into code and embed the GUI with working or dummy software in the background.
   - It is then self-tested by the developers.



USER INTERFACE DESIGN ACTIVITIES

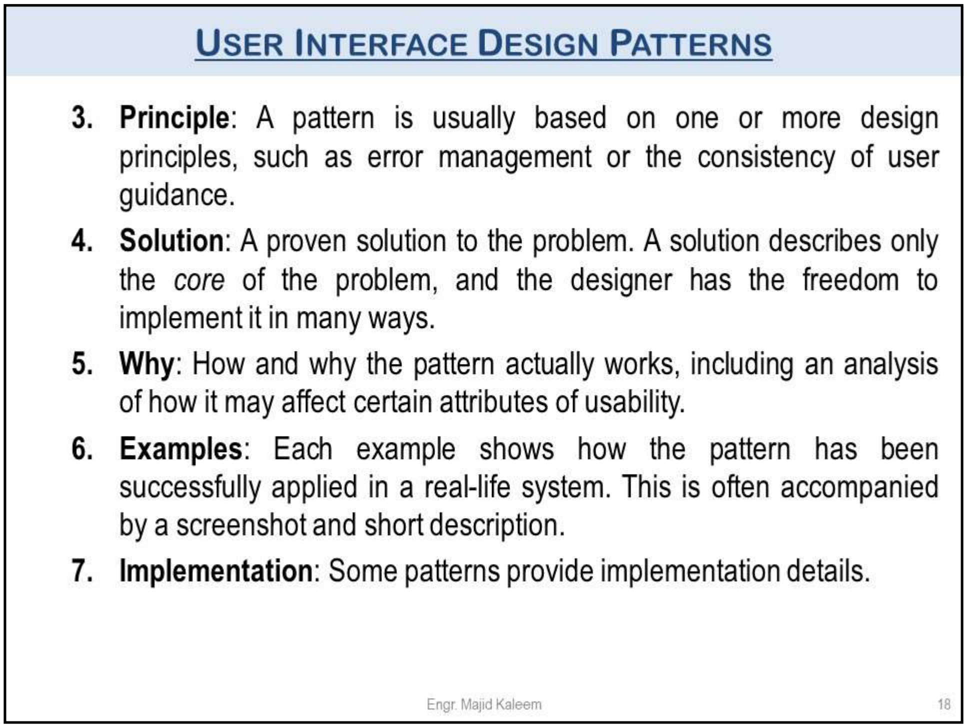
5. **GUI TESTING**
   - GUI testing can be done in various ways.
   - Organization can have in-house inspection, direct involvement of users and release of beta version are few of them.
   - Testing may include usability, compatibility, user acceptance etc.



User Interface Design Patterns

- User interface design patterns are descriptions of *best practices* within user interface design.
- They are general, reusable solutions to commonly occurring problems.
- As such, they form the backbone of "technical support."
- A UI design pattern usually consists of these elements:

1. **Problem**: The usability problem faced by the user when using the system.
2. **Context of use**: The situation (in terms of the tasks, users, and context of use) giving rise to the usability problem.



User Interface Design Patterns

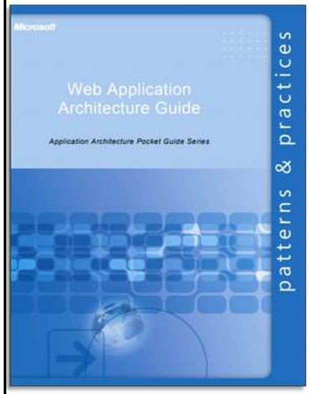
3. **Principle**: A pattern is usually based on one or more design principles, such as error management or the consistency of user guidance.
4. **Solution**: A proven solution to the problem. A solution describes only the *core* of the problem, and the designer has the freedom to implement it in many ways.
5. **Why**: How and why the pattern actually works, including an analysis of how it may affect certain attributes of usability.
6. **Examples**: Each example shows how the pattern has been successfully applied in a real-life system. This is often accompanied by a screenshot and short description.
7. **Implementation**: Some patterns provide implementation details.

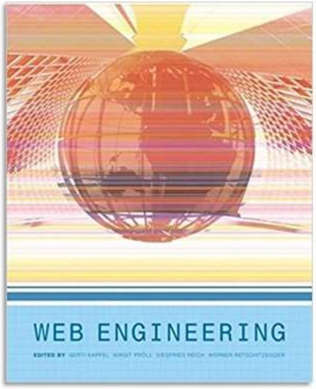
User Interface Design Patterns

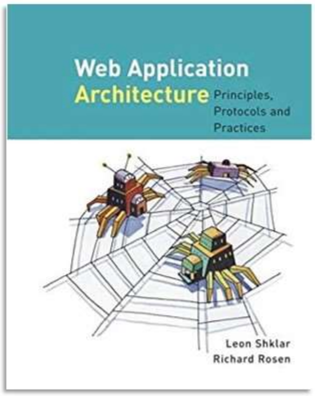
- Visit the following site to study complete list of patterns:
- http://ui-patterns.com/patterns

Web Application

- You may download free pdf books for further study:







Course in BSE:
Web Engineering

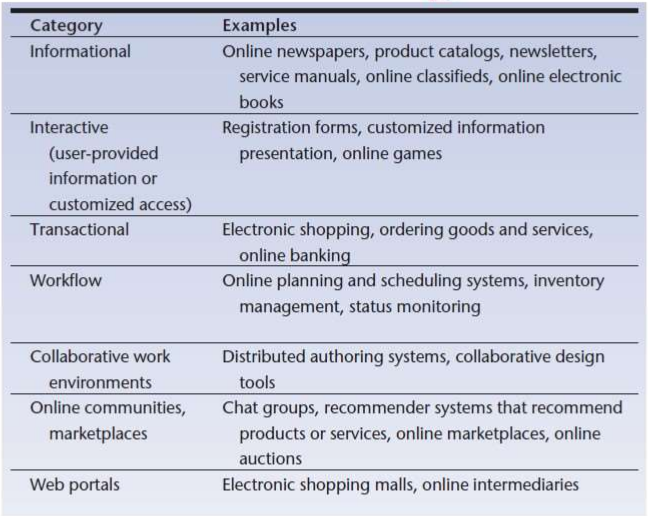
WEB APPLICATION DESIGN

- Unlike traditional software, the Web serves as both development & user platform
- A Web application is a system that utilizes W3C standards & technologies to deliver Web-specific resources to clients.
- What Is Web Application Design?
- *Web application design is an important stage when building a web application.*
- It focuses on the *look* and *feel* of the web application.
- The design stage encompasses several different aspects, including user interface design (UI), usability (UX), content production, and graphic design.

CATEGORIES OF WEB APPLICATION

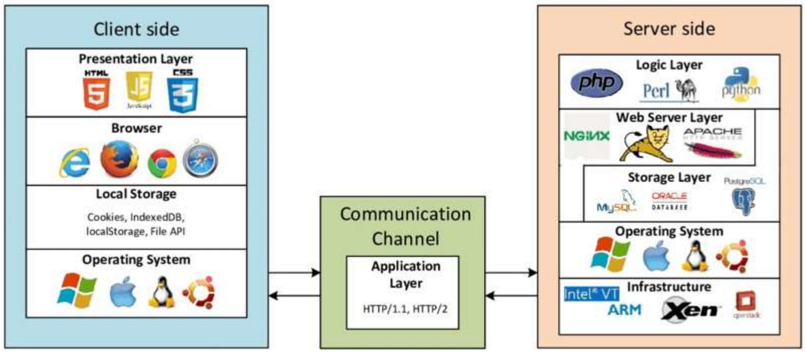
- Most frequent types include:

VERY IMPORTANT



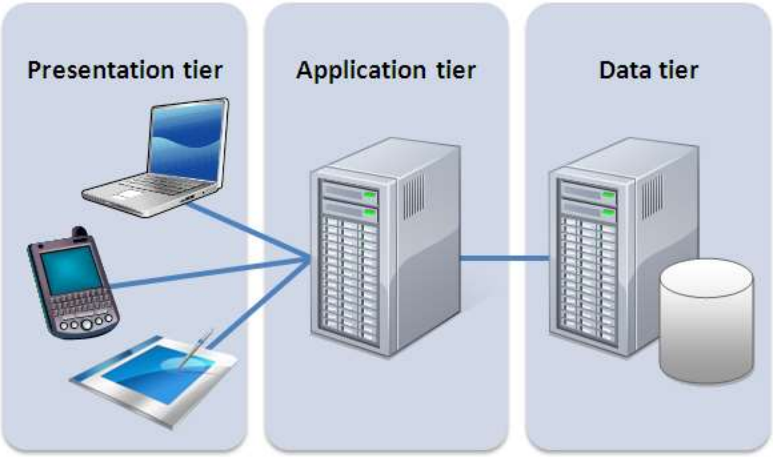
Category	Examples
Informational	Online newspapers, product catalogs, newsletters, service manuals, online classifieds, online electronic books
Interactive (user-provided information or customized access)	Registration forms, customized information presentation, online games
Transactional	Electronic shopping, ordering goods and services, online banking
Workflow	Online planning and scheduling systems, inventory management, status monitoring
Collaborative work environments	Distributed authoring systems, collaborative design tools
Online communities, marketplaces	Chat groups, recommender systems that recommend products or services, online marketplaces, online auctions
Web portals	Electronic shopping malls, online intermediaries

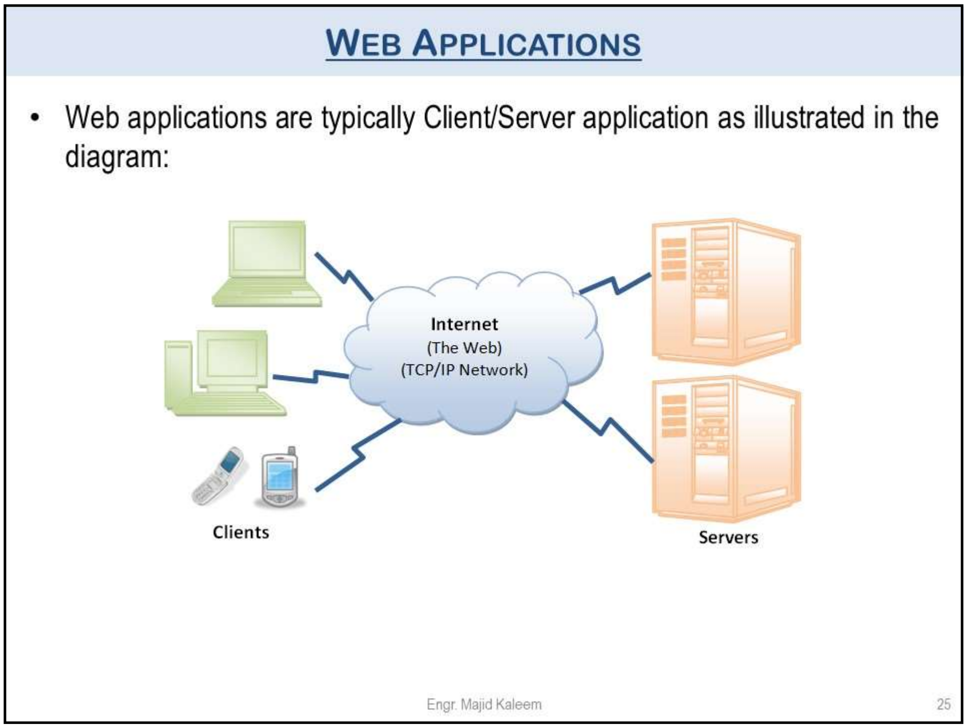
WEB APPLICATIONS



TIER VS. LAYER

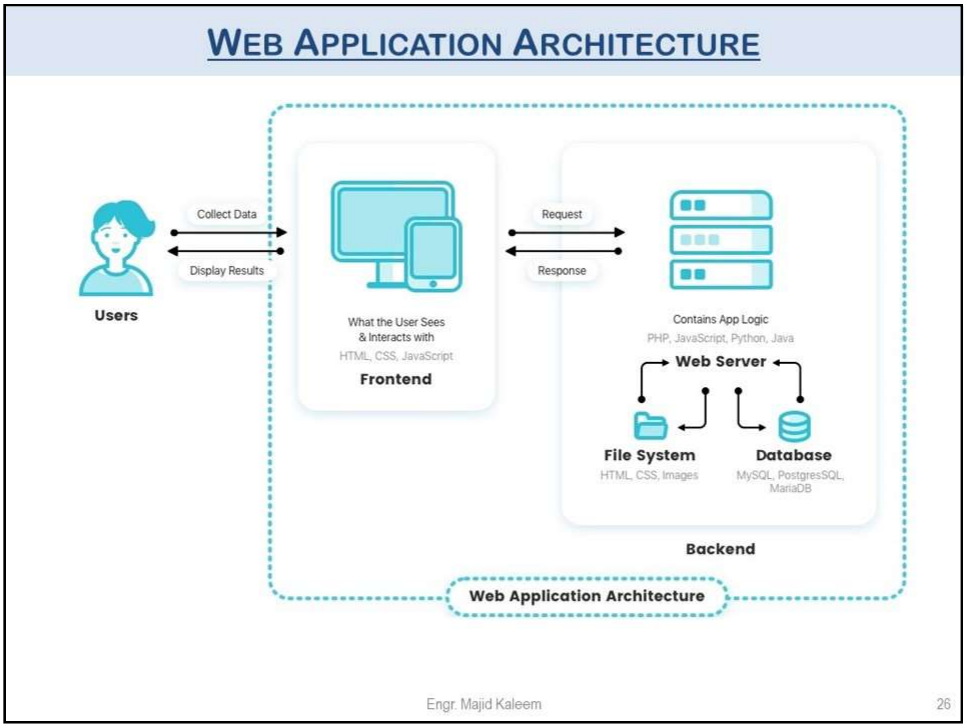
- https://dotnettutorials.net/what-is-the-difference-between-layers-and-tiers/



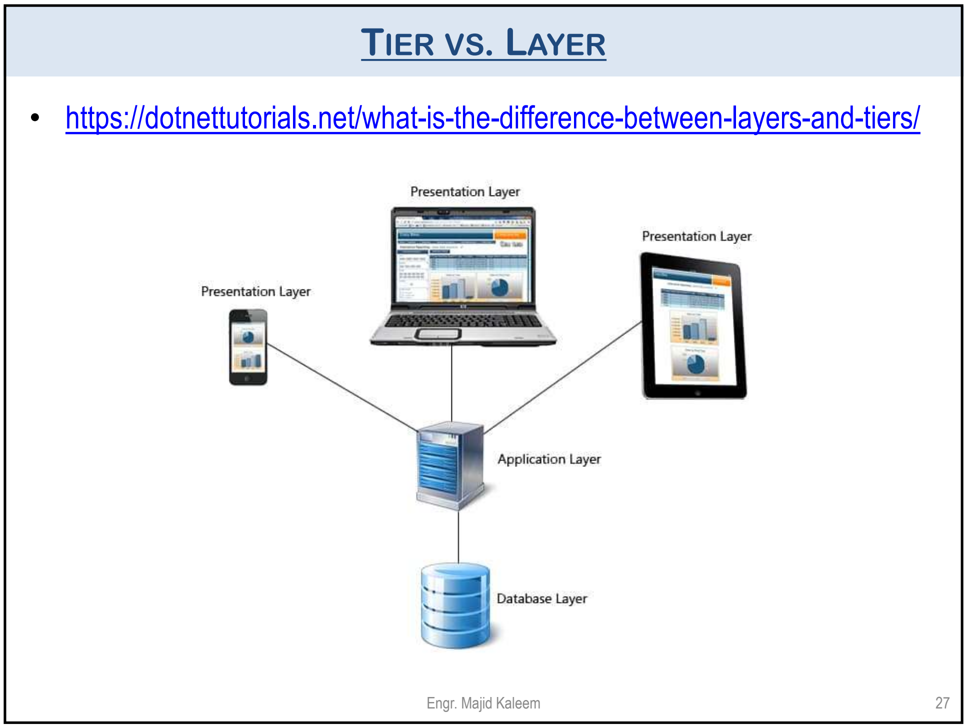


WEB APPLICATIONS

- Web applications are typically Client/Server application as illustrated in the diagram:

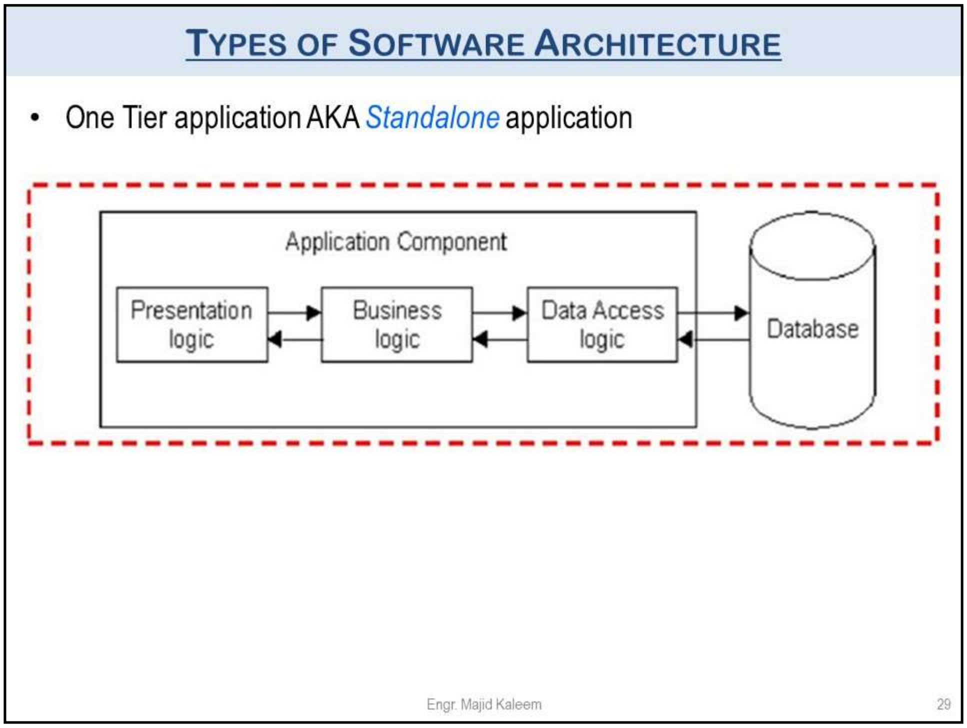


WEB APPLICATION ARCHITECTURE



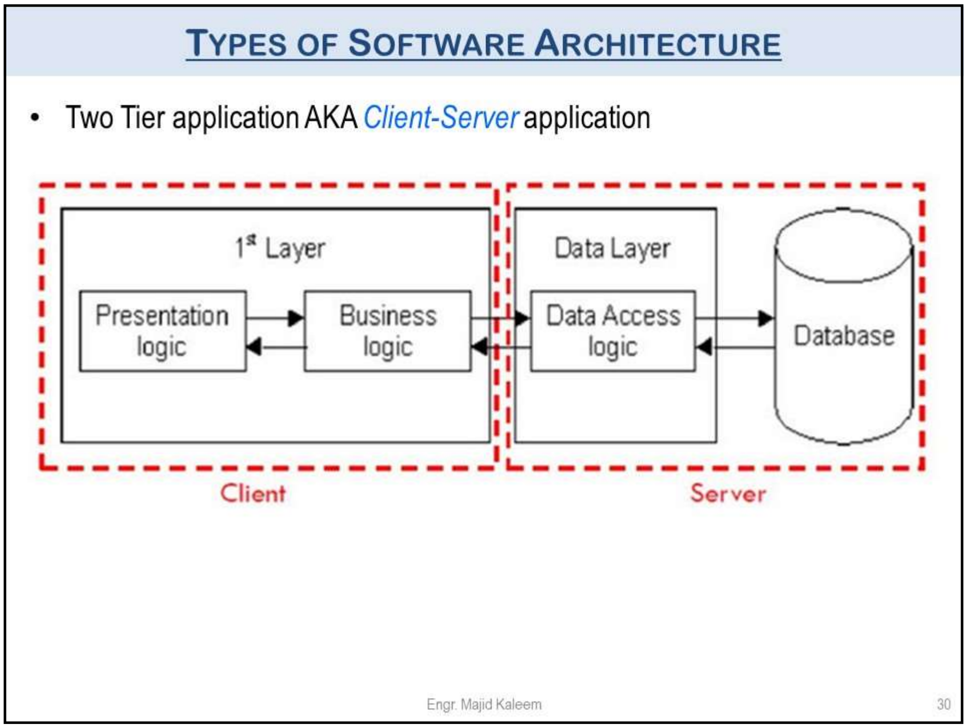
Tier vs. Layer

- https://dotnettutorials.net/what-is-the-difference-between-layers-and-tiers/



Types of Software Architecture

- One Tier application AKA *Standalone* application



TYPES OF SOFTWARE ARCHITECTURE

- Two Tier application AKA *Client-Server* application

1st Layer

Presentation logic

Business logic

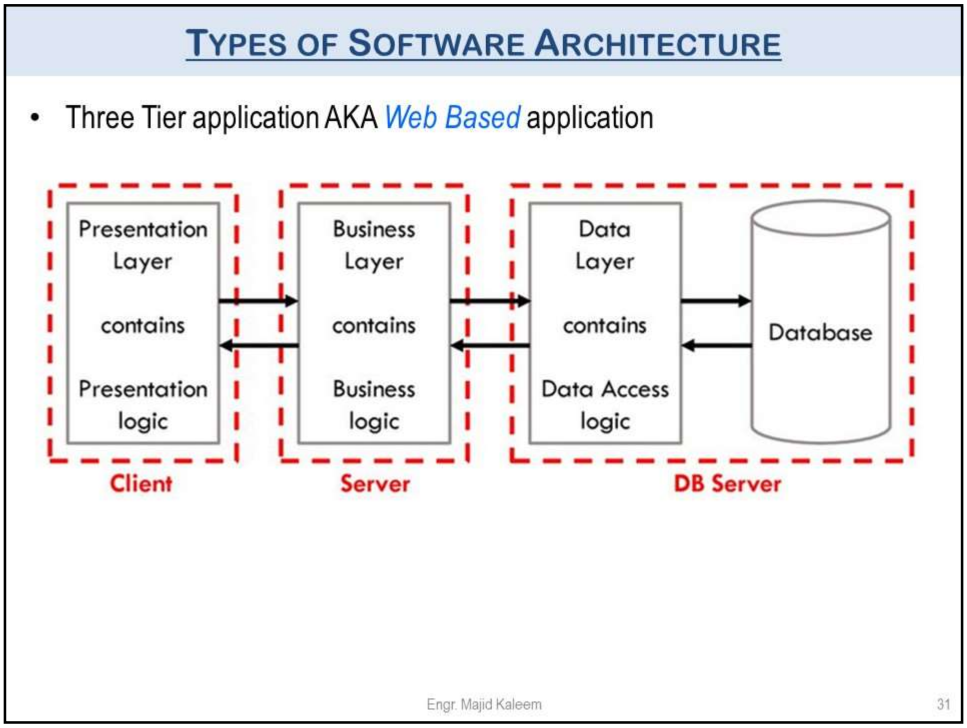
Data Layer

Data Access logic

Database

Client

Server



TYPES OF SOFTWARE ARCHITECTURE

- Three Tier application AKA *Web Based* application

Presentation Layer contains Presentation logic

Business Layer contains Business logic

Data Layer contains Data Access logic

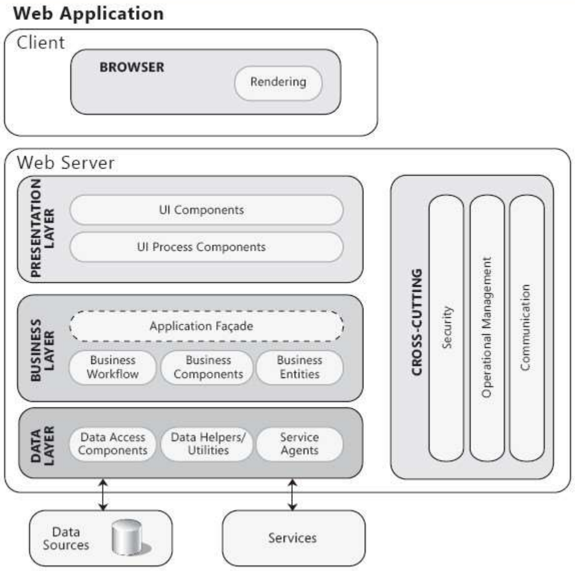
Database

Client

Server

DB Server

TYPICAL WEB APPLICATION



Web Application

Client

BROWSER

Rendering

Web Server

PRESENTATION LAYER

UI Components

UI Process Components

BUSINESS LAYER

Application Façade

Business Workflow

Business Components

Business Entities

DATA LAYER

Data Access Components

Data Helpers/ Utilities

Service Agents

CROSS-CUTTING

Security

Operational Management

Communication

Data Sources

Services

CROSS-CUTTING CONCERNS

- *Cross-cutting concerns in programming refer to aspects of a software system that affect multiple parts of the system and cannot be cleanly encapsulated within a single module or component.*

- These concerns often involve aspects of the system that are not related to its core functionality but are important for its *operation and management*.
- Here are some examples of cross-cutting concerns:
  - Logging: Logging is a cross-cutting concern because it affects many different parts of a system.
  - Logging information about errors, warnings, and other events can help diagnose problems and debug issues.
  - But logging code can't be confined to a single component because it needs to be integrated into all components of the system.

Cross-Cutting Concerns

- Security: Security is another cross-cutting concern because it is important throughout the system.
- Security measures such as authentication, access control, and encryption must be implemented across all system components to ensure that data is *protected from unauthorized access*.

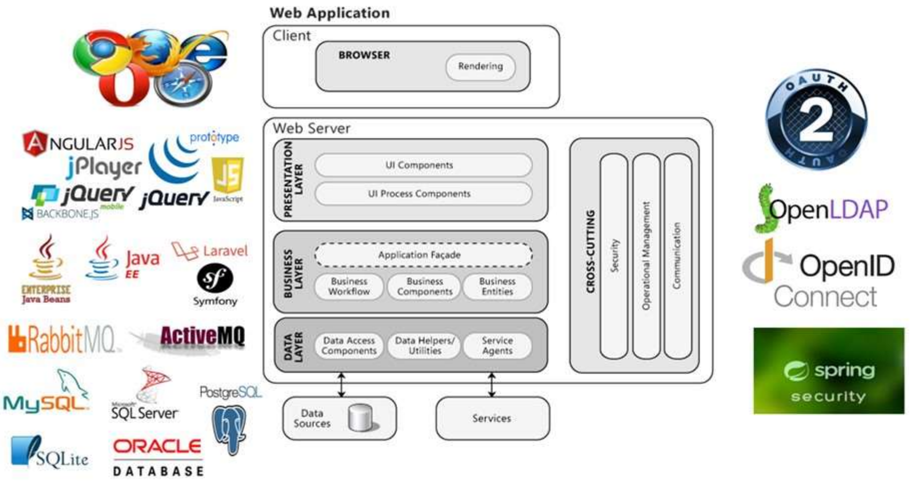
- Caching: Caching is a cross-cutting concern because it affects performance across the entire system.
- Caching is used to *speed up access to frequently accessed data*, but it must be implemented consistently across all components of the system to be effective.

Cross-Cutting Concerns

- Transaction management: Transaction management is a cross-cutting concern because it involves coordination between multiple components of the system.
- Ensuring that transactions are properly managed and committed is important for maintaining data consistency.

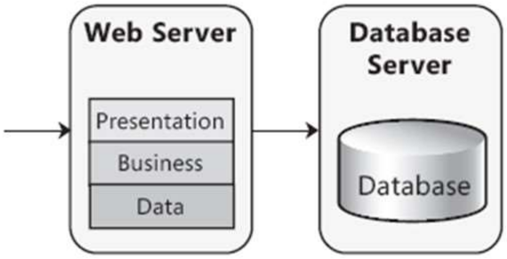
- Configuration management: Configuration management is a cross-cutting concern because it affects how the system operates.
- Managing configuration data such as system settings, database connections, and environment variables must be done consistently across all components of the system to ensure that the system runs smoothly.

Typical Web Application



Nondistributed Deployment

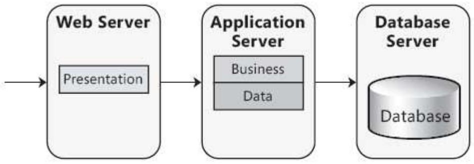
- In a nondistributed deployment scenario, all the logically separate Web application layers are physically located on the *same* Web server, except for the database.



Nondistributed deployment of a Web application

DISTRIBUTED DEPLOYMENT

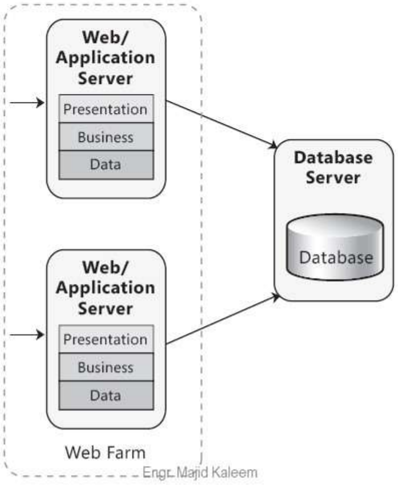
- In a distributed deployment scenario, the presentation and business layers of the Web application reside on separate physical tiers, and communicate remotely. You will typically locate your business and data access layers on the same sever.

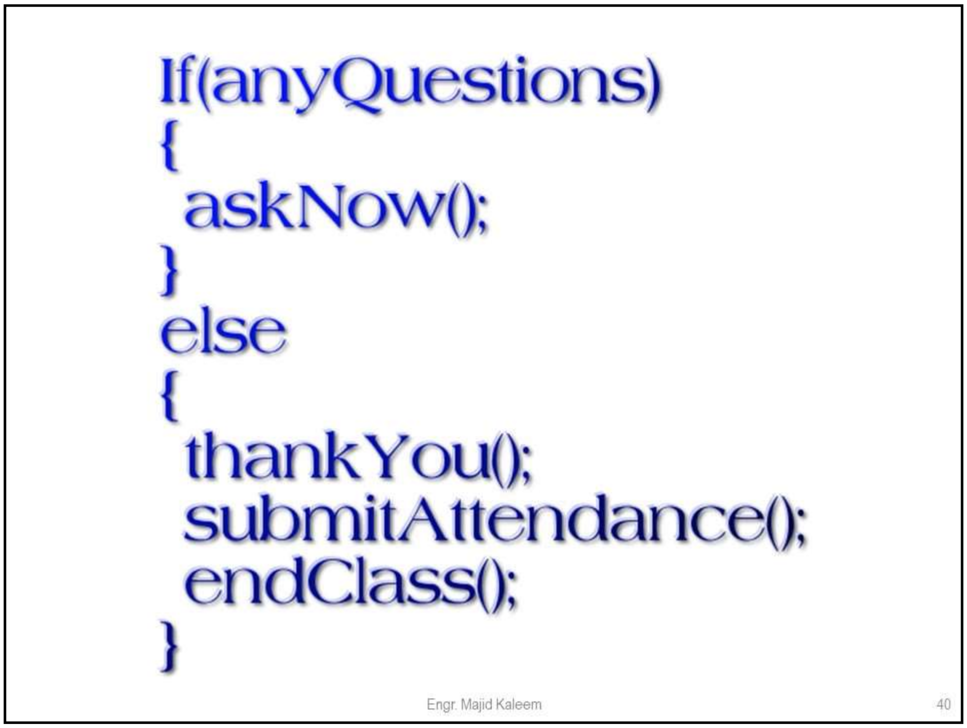


Distributed deployment of a Web application

LOAD BALANCING

- When you deploy your Web application on multiple servers, you can use load balancing to distribute requests so that they are handled by different Web servers. This helps to maximize response times, resource utilization, and throughput.





```
If(anyQuestions)
{
 askNow();
}
else
{
 thankYou();
 submitAttendance();
 endClass();
}
```