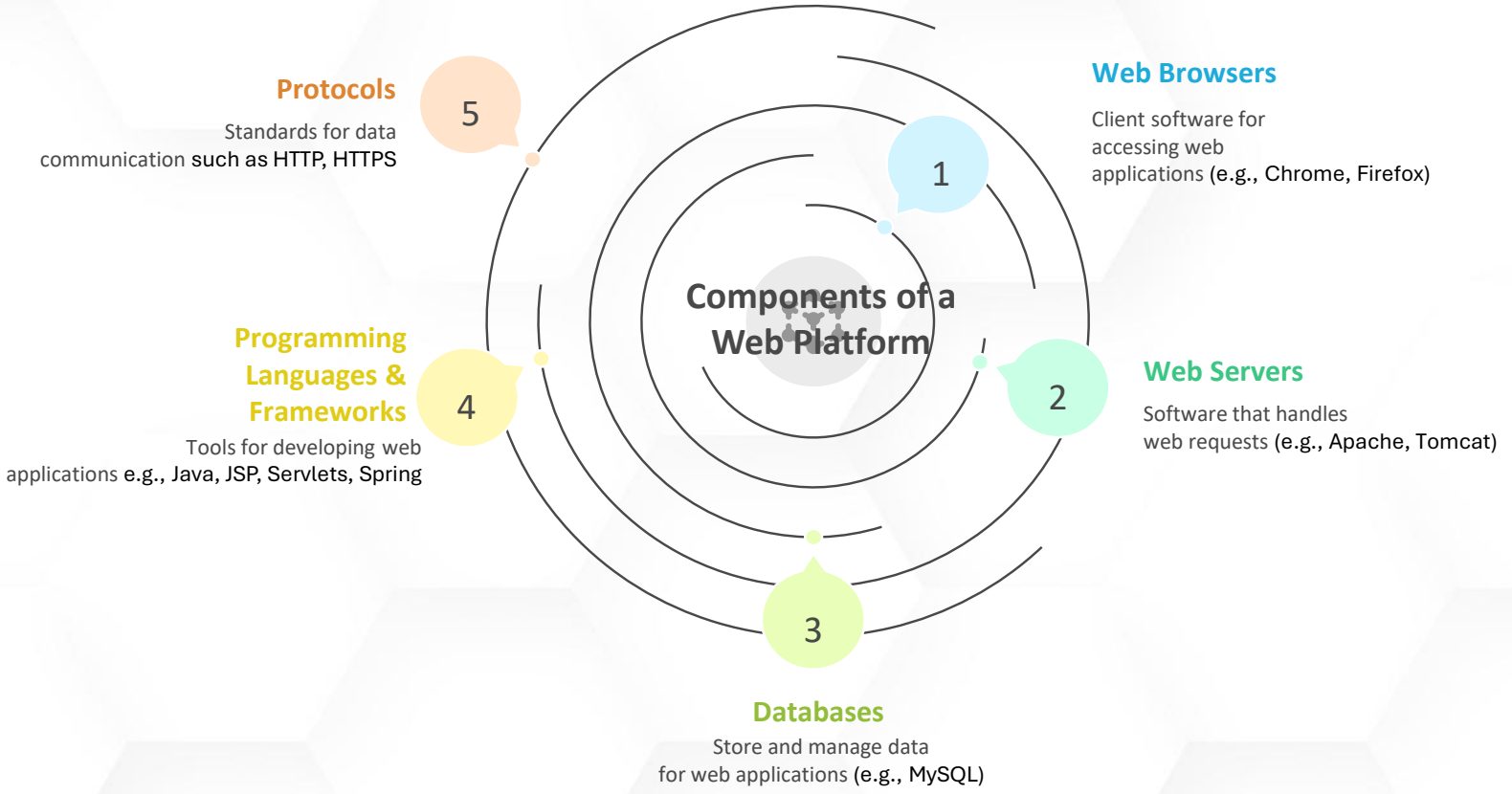
DFP50283
WEB DEVELOPMENT TECHNOLOGY

CHAPTER 1

INTRODUCTION TO JAVA WEB TECHNOLOGIES

1.1 What is a Web Platform?

A **web platform** is an environment that enables the development and deployment of web applications.

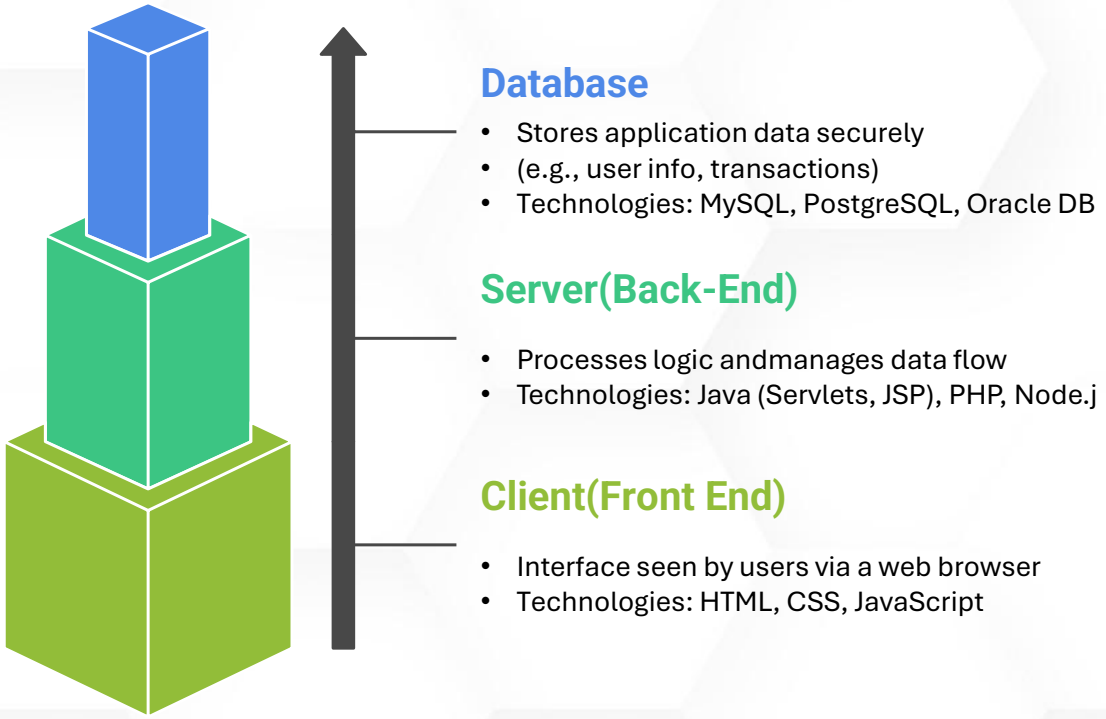


Popular Java Web Platforms:

Platform	Description
Java EE / Jakarta EE	Standard for building enterprise-level applications
Spring Framework	Lightweight and flexible framework for creating web apps with Java
Apache Tomcat	A widely-used servlet container for deploying Java web apps
GlassFish	Reference implementation for Java EE

1.1.1 Identify Components of a Web Application

A web application consists of **three main components** that work together:



Database

- Stores application data securely
- (e.g., user info, transactions)
- Technologies: MySQL, PostgreSQL, Oracle DB

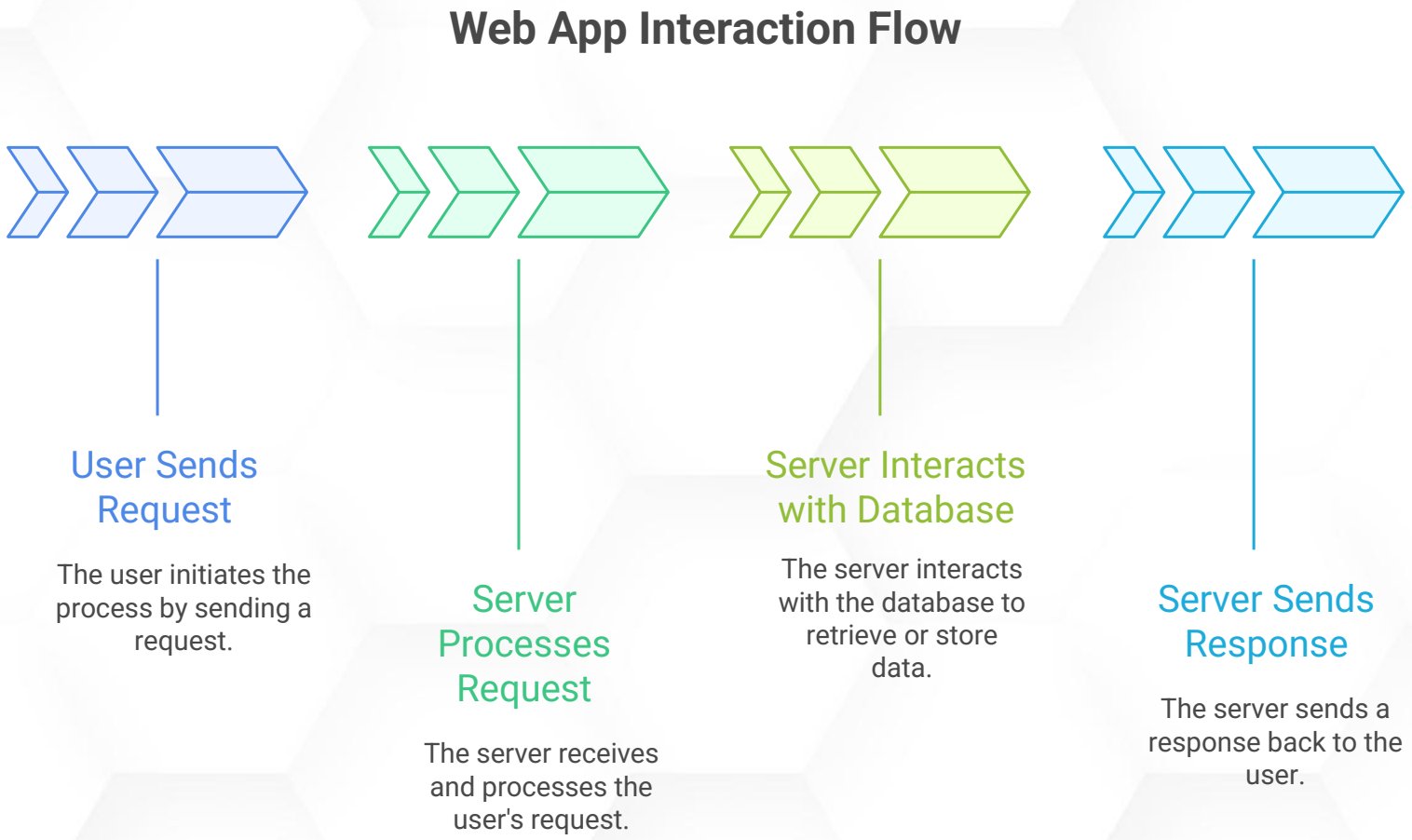
Server(Back-End)

- Processes logic andmanages data flow
- Technologies: Java (Servlets, JSP), PHP, Node.j

Client(Front End)

- Interface seen by users via a web browser
- Technologies: HTML, CSS, JavaScript

1.1.1 Identify Components of a Web Application

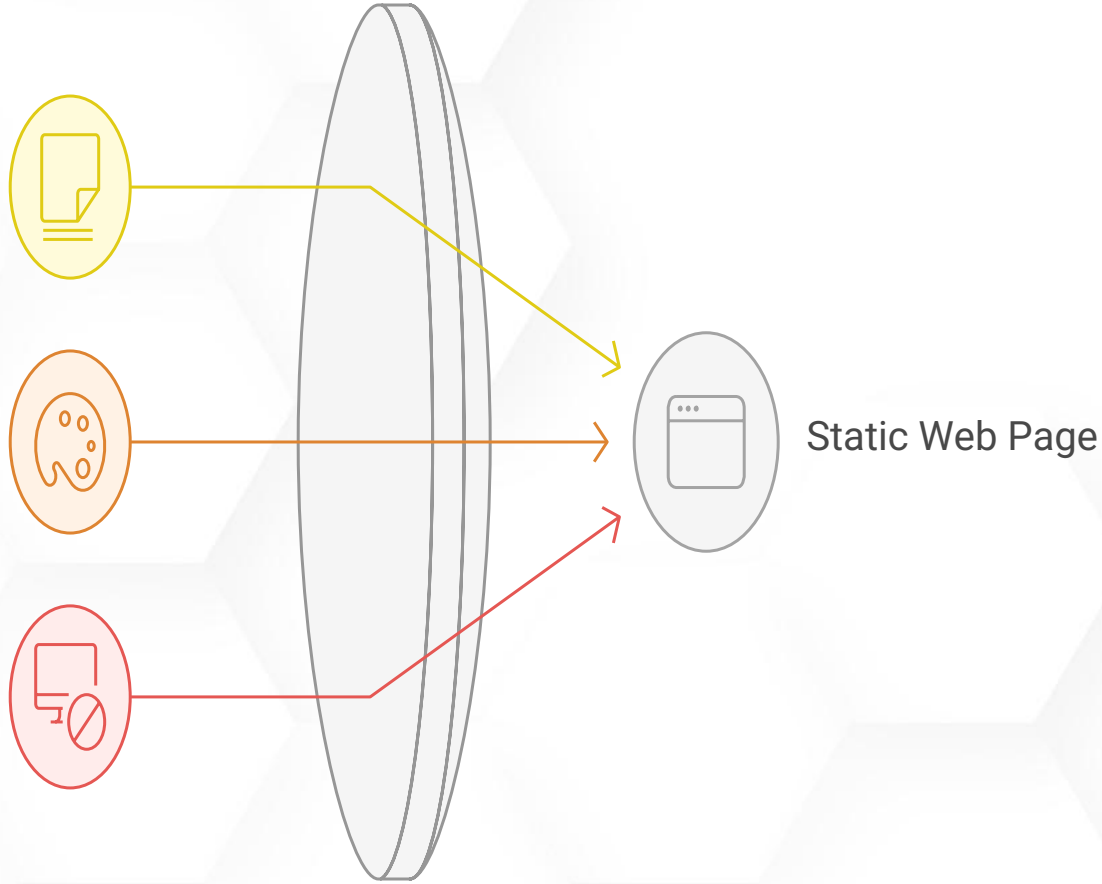


1.1.2 Understand Static Web Pages

Fixed content – **same for all users**

Content written directly in **HTML/CSS**

No interaction with databases or servers for data processing



1.1.2 Understand Static Web Pages

Pros

1. **Fast loading**
   Quick content delivery enhances user experience and site performance.
2. **No backend processing**
   Eliminates server load, reducing complexity and maintenance costs.
3. **Simple site ideal**
   Perfect for straightforward sites like portfolios and landing pages.

Characteristics of Static Web Page

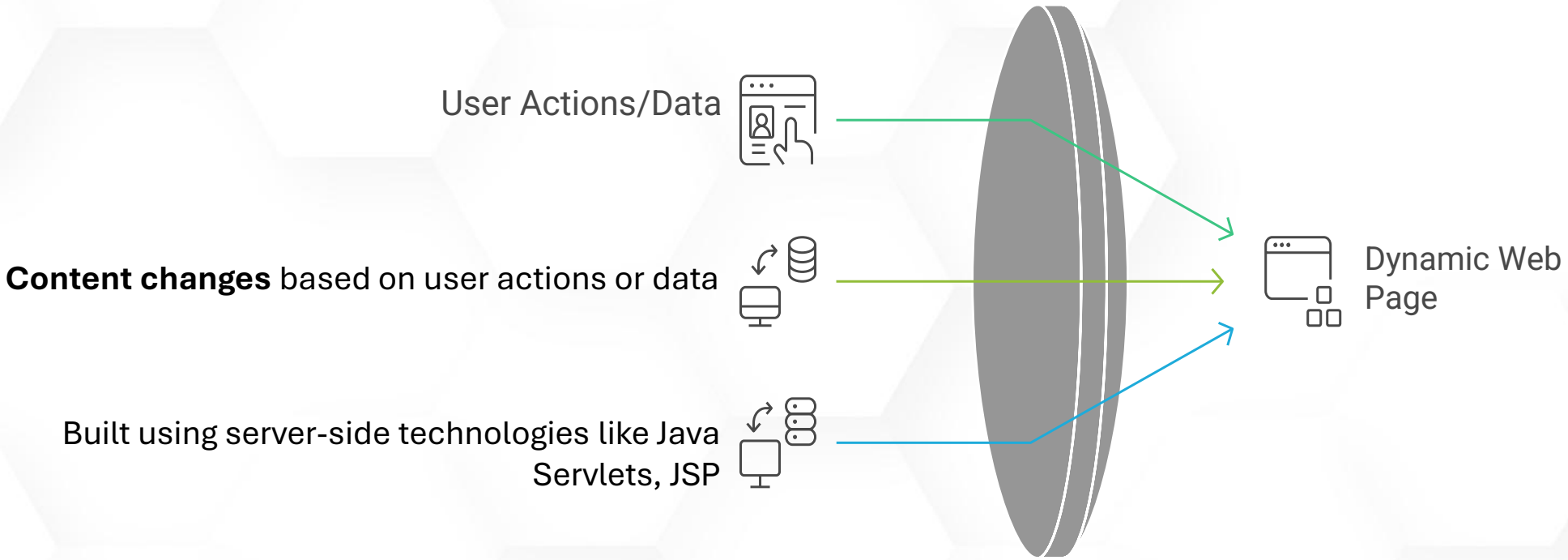
Cons

1. **No dynamic content**
   Lacks real-time updates, limiting user engagement.
2. **No user interaction**
   Prevents personalized experiences and interactive features.
3. **Not scalable**
   Difficult to expand with growing content or user base.

🛠️ Example:

```
<!DOCTYPE html>
<html>
 <head><title>Welcome</title></head>
 <body><h1>Hello, World!</h1></body>
</html>
```

1.1.3 Understand Dynamic Web Pages



1.1.3 Understand Dynamic Web Pages

Pros

Personalized content
1

Tailored content enhances user experience and engagement. (e.g., user profiles, shopping cart)

Interactive features
2

Interactive elements increase user involvement and satisfaction.

Database integration
3

Database connectivity enables real-time data updates and personalization.

Dynamic Web Pages Characteristics

Cons

Development complexity
1

Complex development requires skilled backend developers.

Server-side technologies
2

Server-side technologies demand advanced technical knowledge.

Backend logic
3

Backend logic necessitates robust server infrastructure and maintenance.

🛠️ Java Example (using JSP):

```
<%@ page language="java" %>
<html>
 <body>
  <h2>Hello, <%= request.getParameter("name") %>!</h2>
 </body>
</html>
```

If user inputs "Ali", page shows: **Hello, Ali!**

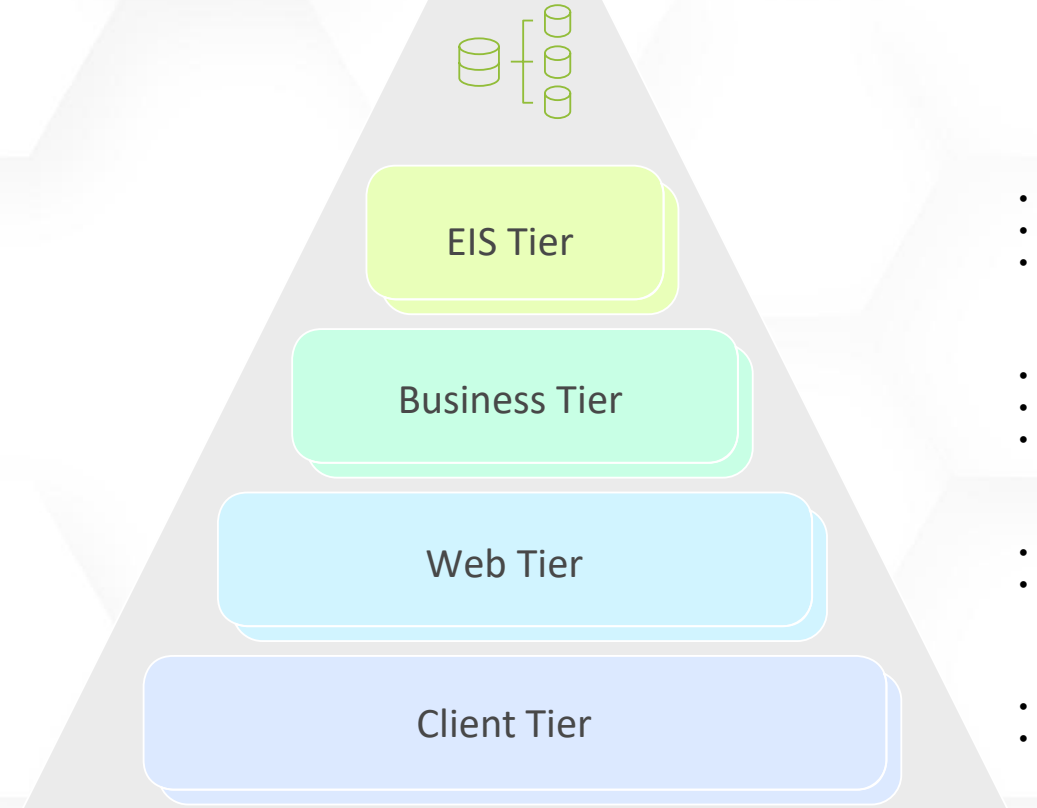
Activity

Discuss the differentiation between Static and Dynamic Web Pages

🌐 1.2 Explain the Java EE Platform in Web Technologies

- Java EE (now Jakarta EE) is a powerful **platform for building large-scale, secure, and distributed enterprise web applications**.
- It extends the Java SE (Standard Edition) by providing libraries and tools specifically for web-based and enterprise-level development.

1.2.1 Describe Java EE Architecture



Java EE Multi-tier Architecture

- **Used by end users**
- Devices like browsers, mobile apps, or Java-based GUI
- Technologies: HTML, Java Applet, JavaFX

- Handles user interface and web page rendering
- Technologies: Java Servlets, JavaServer Pages (JSP), JSF (JavaServer Faces)
- Acts as a bridge between client and business logic

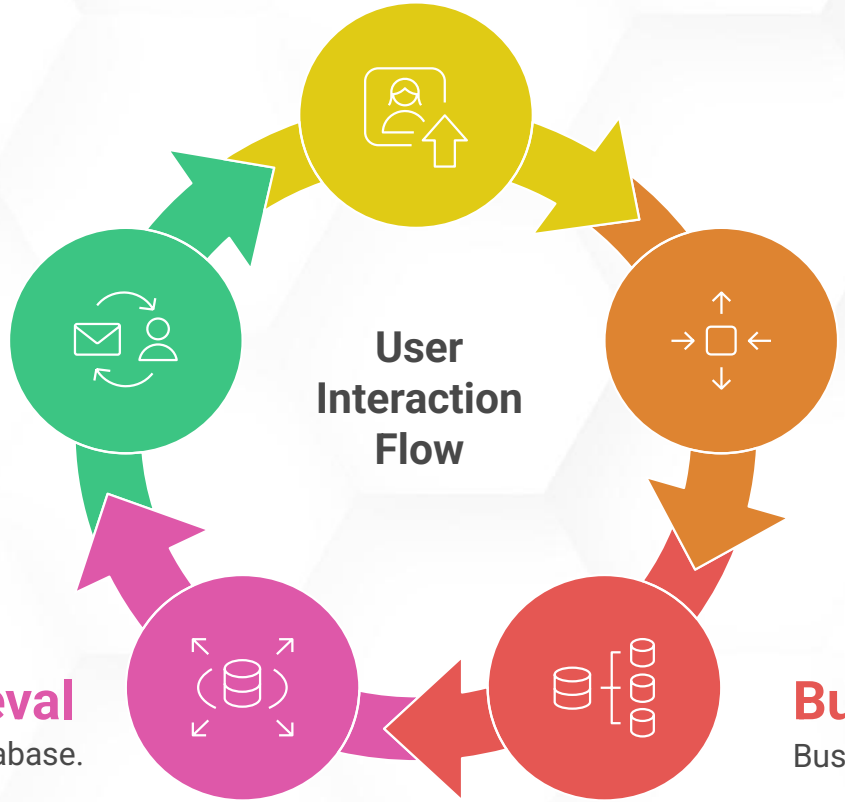
- Contains business logic, rules, and operations
- Technologies: Enterprise JavaBeans (EJB), CDI (Contexts and Dependency Injection)

- Handles data storage and integration with legacy systems
- Technologies: JDBC, JPA, JNDI, databases like MySQL, Oracle

1.2.1 Describe Java EE Architecture

User Request

User initiates a request to the system.



Web Layer Processing

Request is processed through the web layer.

Business Logic Application

Business logic is applied to handle the request.

Data Retrieval

Necessary data is retrieved from the database.

Response to User

Response is sent back to the user.

🛠️ 1.2.2 Identify Various Technologies in Java EE

✅ a. Client-Side Technologies

Technology	Description
Java Application	Standalone program written in Java, may interact with web server
Java Applet	Mini Java program embedded in HTML, runs in browser (deprecated in modern web)
HTML	Basic markup language used to create web pages and forms

🛠️ 1.2.2 Identify Various Technologies in Java EE

✅ b. Server-Side Technologies

Technology	Description
Servlets	Java classes that handle HTTP requests and generate dynamic responses
JSP (JavaServer Pages)	Allows embedding Java code inside HTML for dynamic content generation
JSF	Component-based UI framework for building user interfaces for web apps

🧠 Example Servlet Code:

```
@WebServlet("/hello")
public class HelloServlet extends HttpServlet {
    protected void doGet(HttpServletRequest request, HttpServletResponse response)
throws IOException {
        response.getWriter().println("Hello from Servlet!");
    }
}
```

🧠 Example JSP Code:

```
<%@ page language="java" %>
<html>
 <body>
  <h1>Welcome <%= request.getParameter("user") %></h1>
 </body>
</html>
```

🛠️ 1.2.2 Identify Various Technologies in Java EE

✅ c. Server-Side Business Technologies

Technology	Description
EJB (Enterprise JavaBeans)	Server-side components that encapsulate business logic and transaction management

Types of EJB:

- **Session Beans**	Handle business logic (Stateless/Stateful)
- **Message-Driven Beans**	Handle asynchronous messaging
- **Entity Beans** (deprecated)	Used for representing database records

🧠 Example:

```
@Stateless
public class OrderService {
    public void placeOrder(String productId, int quantity) {
        // business logic to place order
    }
}
```

Summary

Layer	Technology	Role
Client	HTML, Java Applet, Java App	User interface and input
Web Tier	Servlets, JSP	Handles requests/responses
Business Tier	EJB	Business rules & processing
Data Tier	JDBC, JPA	Connects to database systems

🌐 1.3 Organize Servlet to Build Web Application

Servlet is the backbone of many Java-based web applications. It enables **dynamic content generation**, **request-response processing**, and **interaction with users** over the web.

✅ 1.3.1 Define Servlet

- A **Servlet** is a Java class that runs on a web server and handles **client requests** (usually from browsers) and sends back **responses**.

- **Key points:**
  - Runs on server-side
  - Part of **Java EE / Jakarta EE**
  - Follows **request-response model** using **HTTP protocol**

💡 Example: When a user submits a login form, the servlet processes the data and returns the appropriate result.

✅ 1.3.2 Explain the Architecture of Servlet

- Servlet architecture follows a **client-server model**.
- 🔄 **Flow:**
  - **Client (Browser)** sends a request via HTTP
  - **Web Server** (e.g., Tomcat) receives it
  - Server forwards the request to the **Servlet**
  - **Servlet processes** the request (e.g., reading form data)
  - **Servlet sends response** back (HTML or data)
  - **Client** receives and renders the response

Components Involved:

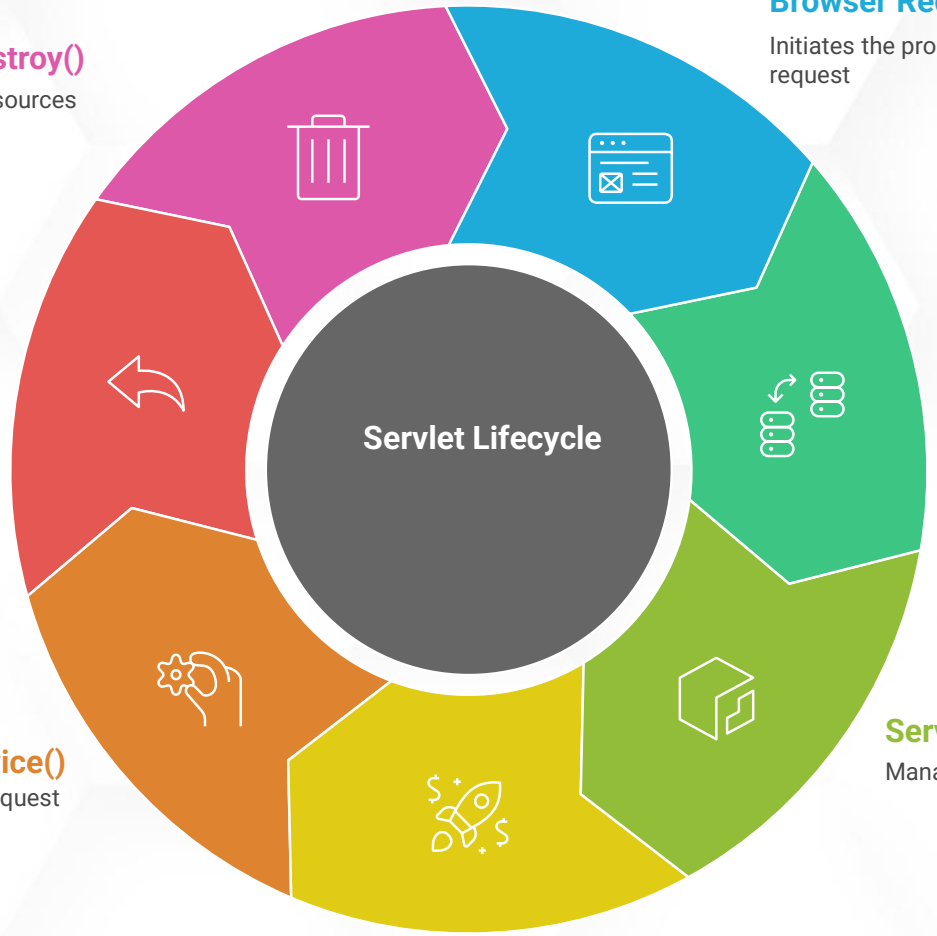
Component	Role
Servlet Class	Handles logic for requests/responses
Web Container	Manages lifecycle, maps URLs
Web.xml / Annotations	Maps servlet to URL pattern

✅ 1.3.3 Describe Servlet Lifecycle

- Servlet lifecycle is controlled by the **Servlet Container** and consists of **three key methods**:

Method	Description
init()	Called once when servlet is first loaded. Used to initialize resources.
service()	Called every time a request is made. Handles logic and sends a response.
destroy()	Called once when servlet is being removed. Used to release resources.

Lifecycle Diagram:



Browser Request
Initiates the process with an HTTP request

Web Server
Receives and forwards the request

Servlet Container
Manages servlet instances

init()
Initializes the servlet

service()
Handles the request

Response
Sends back the response

destroy()
Cleans up resources

✅ 1.3.4 Apply Core Servlet API: GenericServlet, ServletRequest, ServletResponse

❖ GenericServlet

- Abstract class implementing Servlet interface
- Used as a base class to build protocol-independent servlets
- Must override service() method

```
public class MyServlet extends GenericServlet {
    public void service(ServletRequest req, ServletResponse res) throws IOException {
        res.getWriter().println("Hello from GenericServlet");
    }
}
```

✅ 1.3.4 Apply Core Servlet API: GenericServlet, ServletRequest, ServletResponse

❖ ServletRequest

- Carries client request data to the servlet
- Useful methods:
  - getParameter(String name)
  - getInputStream()

✅ 1.3.4 Apply Core Servlet API: GenericServlet, ServletRequest, ServletResponse

❖ ServletResponse

- Used to send output to client
- Useful methods:
  - getWriter() – to send character data (HTML)
  - getOutputStream() – to send binary data

✅ 1.3.5 Explain HTTP Servlets: HttpServletRequest & HttpServletResponse

- Most modern web applications use **HTTP protocol**, so we usually extend **HttpServlet** instead of GenericServlet.

❖HttpServletRequest

- Special version of ServletRequest for HTTP
- Additional methods:
  - getParameter("username")
  - getHeader("User-Agent")
  - getCookies()

✅ 1.3.5 Explain HTTP Servlets: HttpServletRequest & HttpServletResponse

- Most modern web applications use **HTTP protocol**, so we usually extend **HttpServlet** instead of GenericServlet.

❖ HttpServletResponse
  - Special version of ServletResponse for HTTP
  - Additional methods:
    - sendRedirect("page.jsp")
    - setStatus(200)
    - addCookie(cookie)

Example: HttpServlet Code

```
@WebServlet("/hello")
public class HelloServlet extends HttpServlet {
    protected void doGet(HttpServletRequest req, HttpServletResponse
res) throws IOException {
        String name = req.getParameter("name");
        res.getWriter().println("Hello, " + name + "!");
    }
}
```

URL: http://localhost:8080/hello?name=Ali

Output: Hello, Ali!

Summary

Topic	Description
Servlet	Java class that handles HTTP requests/responses
Lifecycle Methods	init(), service(), destroy()
GenericServlet	Protocol-independent servlet class
ServletRequest/Response	Base interfaces for request and response handling
HttpServletRequest/Response	HTTP-specific request/response handling