

**Priyadarshini D.**

## **FULLSTACK DEVELOPMENT**

### **WEEK-02**

#### **Software development in agile methodology**

##### **Define goal of product**

Product goals help achieve the product vision and business objectives. Goals are a key piece of your product strategy.

##### **Define Epic**

Agile epics are large pieces of work that can be broken down into smaller and more manageable work items, tasks, or user stories. These series of work items share the same goal described by the epic.

##### **Road Map**

Roadmaps in Jira Software are team-level roadmaps useful for planning large pieces of work several months in advance at the Epic level within a single project

##### **Cost estimation**

Cost estimation in software engineering is the process of predicting the resources (money, time, and people) necessary to finish a project within the defined scope.

##### **Risk Management**

There are three main classifications of risks which can affect a software project:

1. Project risks
2. Technical risks
3. Business risks

1. Project risks: Project risks concern different forms of budgetary, schedule, personnel, resource, and customer-related problems.

2. Technical risks: Technical risks concern potential method, implementation, interfacing, testing, and maintenance issues. Most technical risks appear due to the development team's insufficient knowledge about the project.

3. Business risks: This type of risks contain risks of building an excellent product that no one need, losing budgetary or personnel commitments, etc.

### **Creating user stories for the epic using Jira**

#### Step 1: Create a new epic in Jira Software

There are three ways to create epics in Jira Software the Roadmap, Backlog, and Global Create issue button.

When you create an epic, you'll need to enter the following details:

**Epic name** – A Label to identify for your epic.

**Epic summary** - You'll see this whenever Jira displays the epic.

#### Create an epic on the Roadmap

The roadmap is useful for visualizing and planning large pieces of work that may be in progress right now or you may prioritize in the future.

First, enable the roadmap.

1. From your board select **More (• • • )** in the top right corner > **Board settings**
2. In the **Roadmap** tab, enable the feature for your project.
3. Then, select **Roadmap** from the menu on the far left.
4. Hit + **Create epic** on the **Roadmap**. If your **Roadmap** is empty, simply start typing to create your first epic.

#### Create an epic from a new issue

You can create epics and any other issue types using the global navigation menu.

1. Click the **create** button located in the global navigation bar at the top of the screen.
2. Select **Epic** for the issue type.

#### Create an epic from the Epic Panel in the backlog

1. Navigate to the **Backlog**.
2. Click on the **Epics Panel**.

### 3. Hit **Create Epic**.

#### Step 2: Add stories or child-issues

There are two ways to add a story to an epic:

##### **From the issue create screen**

1. Click **Issue** in the top-left corner. Select an issue type other than epic. Find the **Epic Link** field, and select your epic. Fill in any other details and click **Create**.

##### **From the Epics Panel**

1. Navigate to the **Backlog**.
2. Open the **Epics Panel**.
3. Hit **Create issue in epic**.

##### **On the roadmap**

1. Hover over an epic and click **+ Add a child issue**.
2. Choose an issue type, then hit enter.

##### **To remove an issue from an epic**

Navigate to either **Backlog** or **Active sprints**:

In the **Backlog**, drag the issue onto the **Issues without epics** section at the bottom of the **Epics Panel**; or

In either the **Backlog** or the **Active sprints**, click the relevant issue to display it on the right-hand side of the screen, then click the **x** in the epic name (e.g. "Apples" in *Screenshot 1* below).

#### Step 3: Viewing your epics


You can see information relating to all your epics in the Backlog.

1. **Epics Panel:** Go to the **Backlog** and open the **Epics Panel** to view and manage your epics.
2. **List of epics:** The **Epics Panel** displays a list of all epics in your project.
3. **View issues in epic:** Click an epic's name to view all the issues belonging to that epic, across all sprints.

Step 4: Set up swimlanes for your epics on your board

During a sprint, you might find it useful to divide your board into swimlanes for each epic, to make the board visually clearer.

**Here's how you can set this up in Jira Software:**

1. Navigate to the **Backlog** (or **active sprint**).
2. Select **more** (  ) > **Board settings**.
3. Click on **Swimlanes**.
4. Under **Base swimlanes on**, select **Epics**.

Step 5: Monitor the progress of your epic

It is important to keep track of all incomplete issues attached to an epic.

Step 6: Complete your epic

To complete an epic navigate to the backlog or roadmap

## **Backlog**

1. Navigate to the **Backlog**.
2. Open the **Epics Panel**.
3. Click the drop-down for your epic, and select **Mark as Done**.

## **Roadmap**

1. Navigate to the **Roadmap**
2. Click on the **epic**
3. Click the status drop-down in the detail view and select **Mark as Done**.

### Creating Acceptance Criteria:

1. Go to your Project and select Project Settings.
2. Select Issue Types from the left-hand menu and choose the User Story issue type.
3. Create a new "Paragraph" custom field from the right-hand "Create a Field" menu by clicking on it.
4. Give the field a name - Acceptance Criteria.
5. Click Save Changes.

### Sprint Planning:

#### **Create a sprint**

You can create a sprint for your current iteration, or multiple future sprints if you want to plan several iterations ahead.

1. If not already there, navigate to your company-managed Jira Software project.
2. From your project's sidebar, select your **Backlog**.
3. Click **Create sprint** at the top of the backlog section.

Once you've created a sprint, you can add issues to it.

#### **Plan a future sprint**

To plan a future sprint:

1. If not already there, navigate to your company-managed Jira Software project.
2. From your project's sidebar, select your **Backlog**.
3. Click **Create sprint** at the top of the backlog section. The new sprint will appear below the current sprint.
4. Select **Add dates** (located under the sprint's header) to plan the start and end date of your future sprint.

### Backlog Refinement:

The refinement stage. The purpose of the refinement stage is **to set clear expectations with stakeholders and other teams.**

- 1: Backlog Is Treated Like a Wish List
- 2: Sprint Objectives Are Not Aligned with Business Goals
- 3: Backlog Grooming Is Not Regulated
- 4: Issues Are Not Refined with Relevant Context

### **Sprint Demo**

It is the opportunity for the Scrum Team to showcase to Stakeholders and the Product Owner, things that have been done in a Sprint.

### **Burndown chart**

Burndown chart is a major parameter used in agile software development and scrum to detect how much work remains to be completed.

Burndown Chart is a graphical representation of work done, work to be done, the time required for pending work done.

#### **To view the sprint burndown chart:**

1. Navigate to your scrum project.
2. Select the **Backlog** or **Active sprint**.
3. Click **Reports**, then select **Burndown Chart**.

### **Sprint retrospective**

The sprint retrospective is a recurring meeting held at the end of a sprint used to discuss what went well during the previous sprint cycle and what can be improved for the next sprint.

#### **During the Sprint Retrospective, the team discusses:**

What went well in the Sprint

What could be improved

What will we commit to improve in the next Sprint

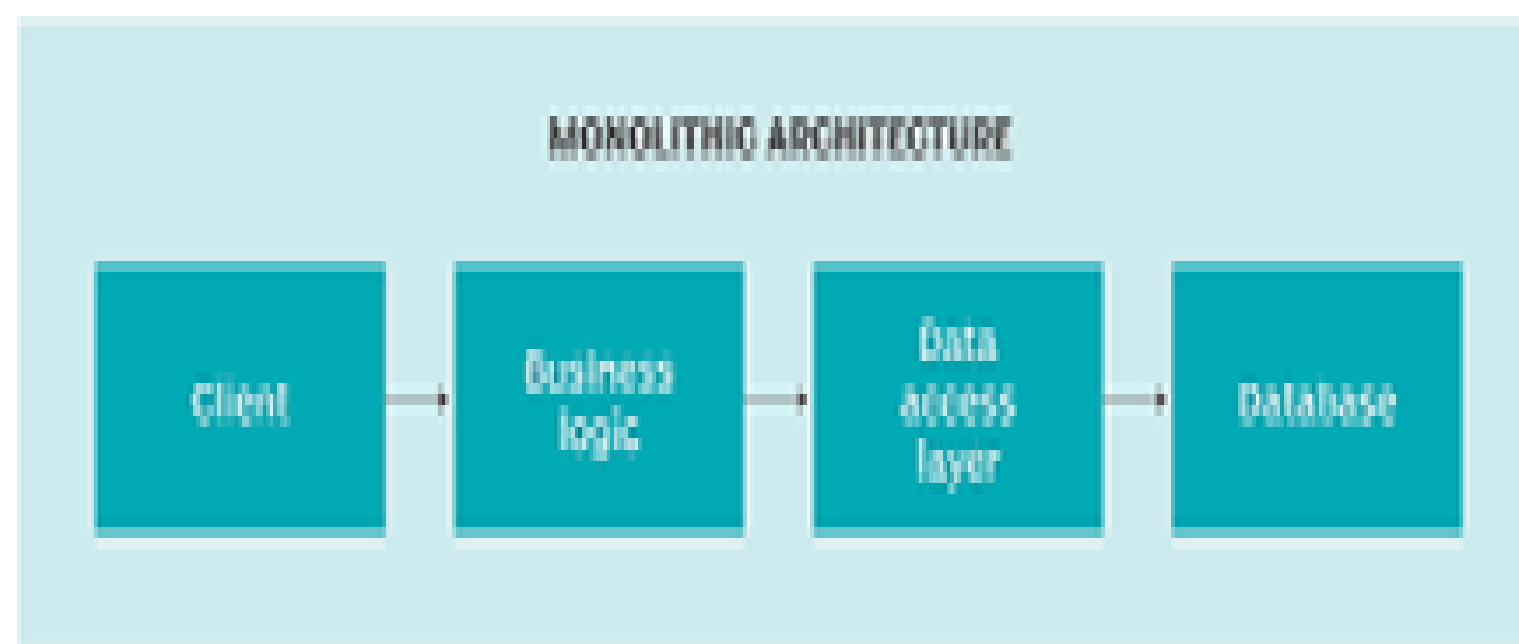
## Design Principles

1. **Availability** refers to the percentage of time that the infrastructure, system, or solution remains operational under normal circumstances in order to serve its intended purpose
2. **Performance** is an indicator of how well a software system or component meets its requirements for timeliness. Timeliness is measured in terms of response time or throughput. The response time is the time required to respond to a request
3. **Consistency** defined as the requirement that a series of measurements of the same project carried out by different raters using the same method should produce similar results, is one of the most important aspects to be taken into account in the measurement methods of the software.
4. **Scalability** is the measure of a system's ability to increase or decrease in performance and cost in response to changes in application and system processing demands.
5. **Manageability** How efficiently and easily a software system can be monitored and maintained to keep the system performing, secure, and running smoothly.
6. **Software cost** estimation is the process of predicting the effort required to develop a software system.

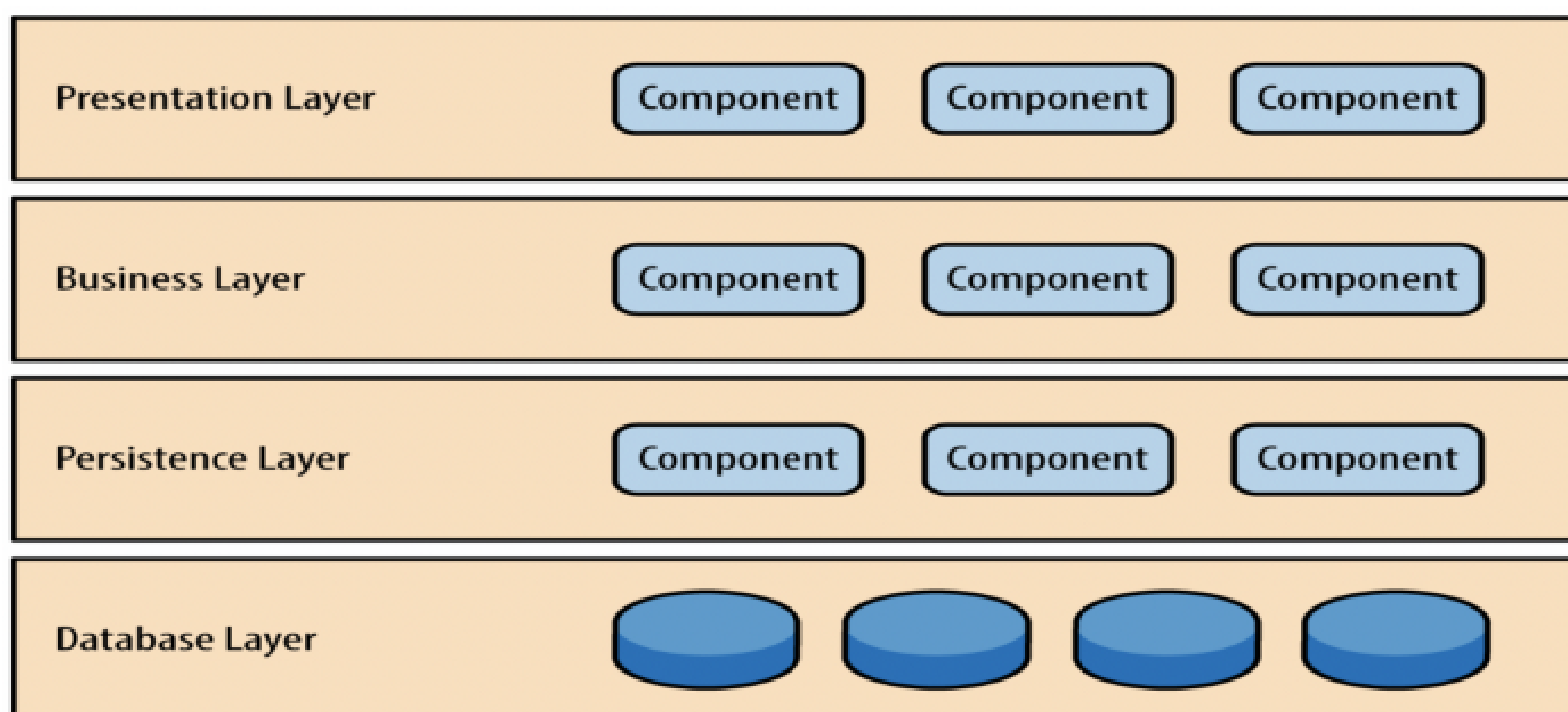
## Architecture Pattern:

1. **Monolithic Architecture** is like a big container, wherein all the software components of an app are assembled and tightly coupled, i.e., each component fully depends on each other.

As you can see in the example all the services provided by the application (Customer Services, Cost Services, Product Services) are directly connected. So if we want to change in code or something we have to change in all the services as well.



## 2. Layered architecture:



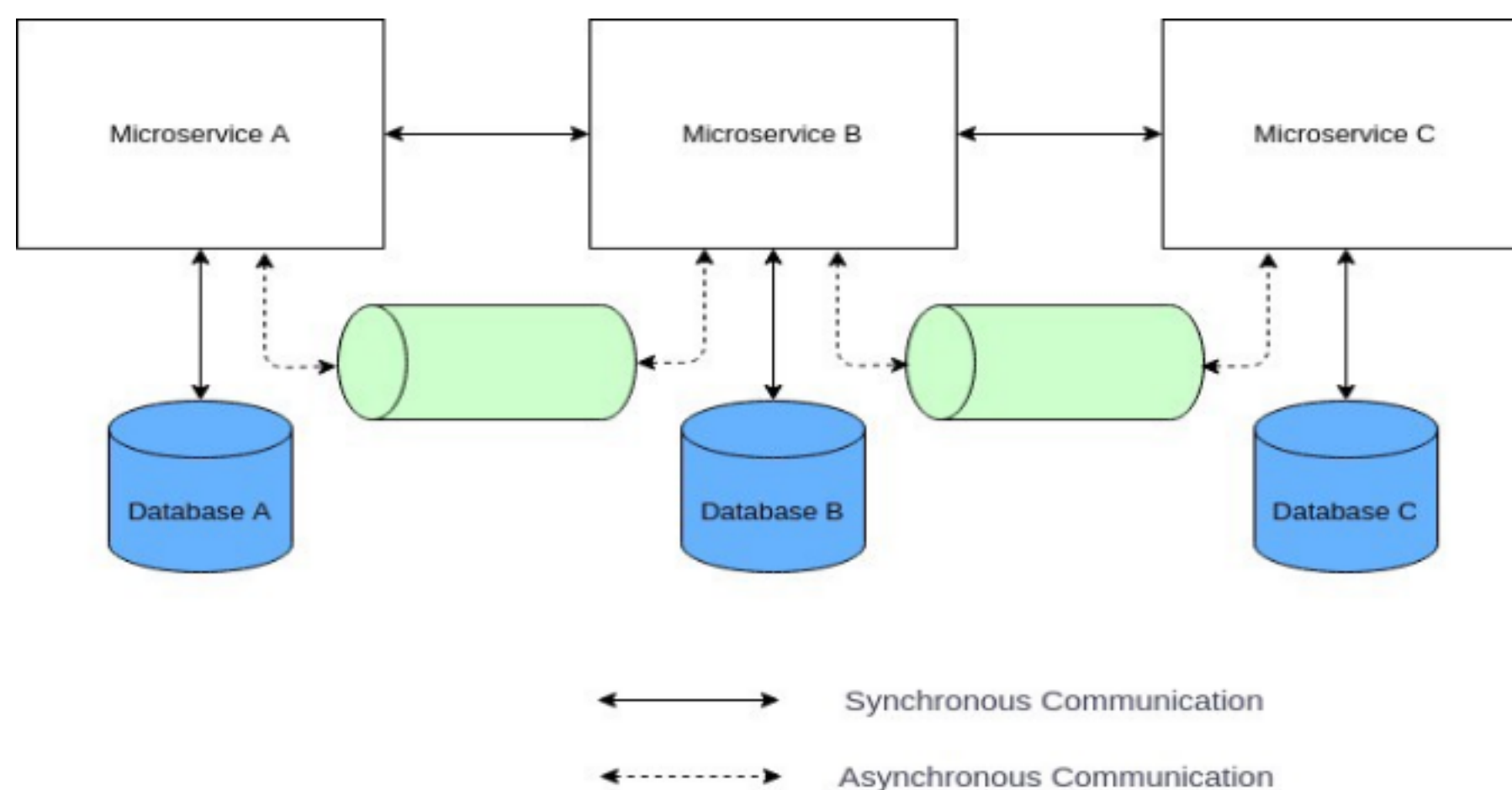
A number of different layers are defined with each layer performing a well-defined set of operations. Each layer will do some operations that becomes closer to machine instruction set progressively.

At the outer layer, components will receive the user interface operations and at the inner layers, components will perform the operating system interfacing (communication and coordination with OS)

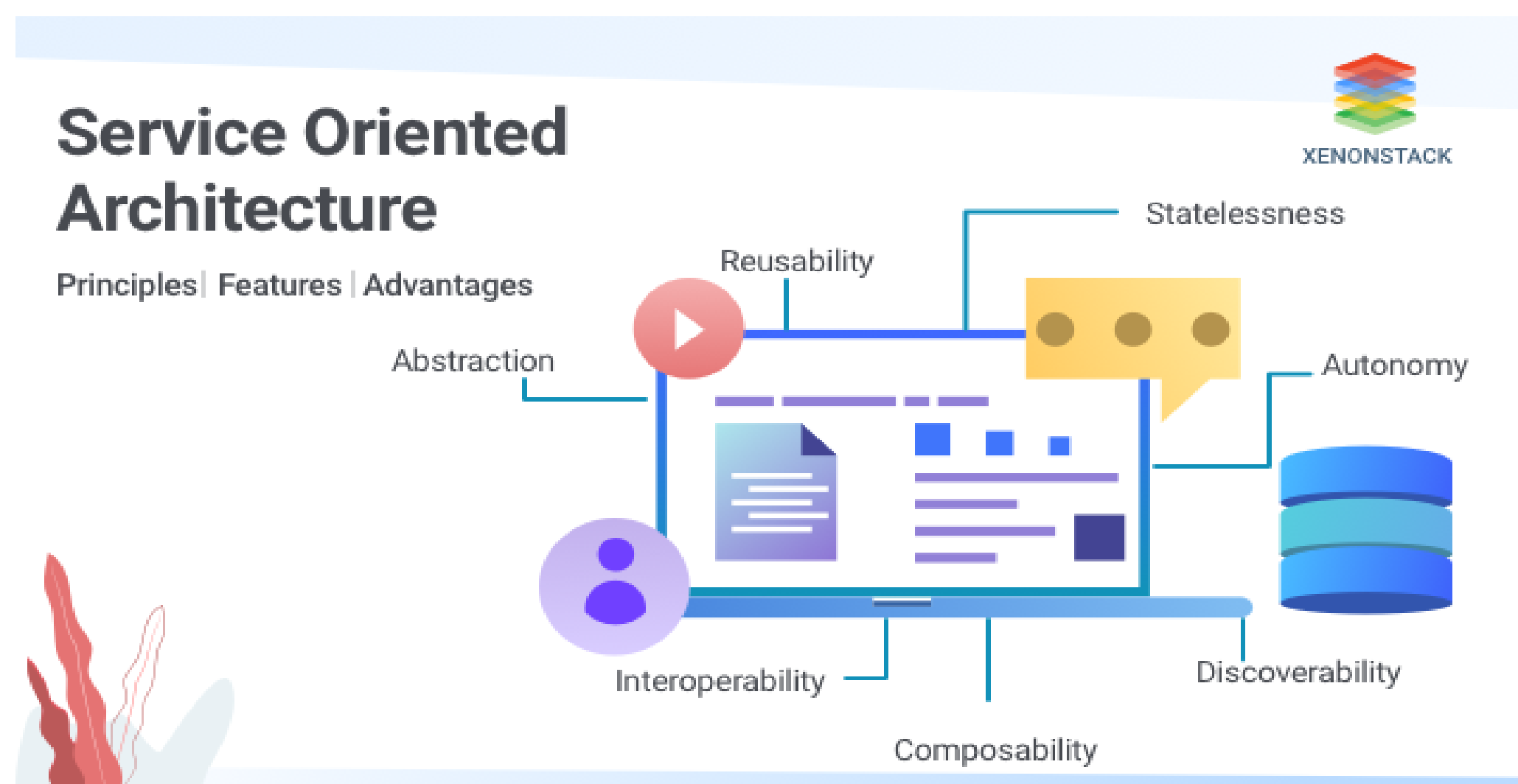
Intermediate layers to utility services and application software functions.



3. The **Microservices pattern** combines design patterns to create multiple services that work interdependently to create a larger application. Because each application is small, it's easier to update them when needed.



4. **Service-Oriented Architecture (SOA)** is a stage in the evolution of application development and/or integration. It defines a way to make software components.



## Design methods for security

### 1. Application security

#### Definition

Application security is the process of developing, adding, and testing security features within applications to prevent security vulnerabilities against threats such as unauthorized access and modification.

### **Types of application security**

1. **Authentication:** Authentication procedures ensure that a user is who they are. This can be accomplished by providing a user name and password when logging in to an application.
2. **Authorization:** The system can validate that a user has permission to access the application by comparing the user's identity with a list of authorized users.
3. **Encryption:** Encryption is the method by which information is converted into secret code that hides the information's true meaning.

### **Token Based security:**

Token-based authentication is a protocol that generates encrypted security tokens.

It enables users to verify their identity to websites, which then generates a unique encrypted authentication token.

That token provides users with access to protected pages and resources for a limited period of time without having to re-enter their username and password.

### **Cookie based Security:**

Cookie authentication uses HTTP cookies to authenticate client requests and maintain session information

### **Open ID:**

**OpenID** allows you to use an existing account to sign in to multiple websites, without needing to create new passwords.

### **Third party access:**

**Third-party access** refers to the process of an organization granting external vendors and service providers secure access to corporate IT assets for maintenance, administration and management purposes.

virtual private network (VPN) solutions to authenticate and authorize employees accessing corporate applications and IT services from outside the enterprise network.

### **SAML:**

**SAML** is an acronym used to describe the **Security Assertion** Markup Language (**SAML**).

Security Assertion Markup Language (SAML) is an open standard that allows identity providers (IdP) to pass authorization credentials to service providers (SP).

### **Multi factor Authentication:**

Multi-factor Authentication (MFA) is an authentication method that requires the user to provide two or more verification factors to gain access to a resource such as an application, online account, or a VPN.

### **Data Source:**

A datastore is a repository for storing, managing and distributing data sets on an enterprise level.

A Data Store is a **connection to a store of data, whether the data is stored in a database or in one or more files.**

### **Structured data:**

Data is stored in a spreadsheet

Excel spreadsheets, Comma-separated value file (.csv) and Relational database tables.

Data entries have the same format and a predefined length and follow the same order.

### **Semi Structured data:**

Data is stored in a text file.

With some structure like headers, paragraphs, etc

HTML files, JavaScript Object Notation (JSON) files and XML files.

Data entries have the hierarchical format and a but the size and order of elements can vary.

### **Unstructured data:**

Data has no specific format.

Images such as .jpeg or .png files, Videos such as .mp4 or m4a files, Sound files such as .mp3 or .wav files, Plain text files. Word files and PDF files.

Data that can take any form and thus be stored as any kind of file

### **Design Principles for UI/UX:**

**UX** design refers to the term “ **user experience** design” , while **UI** stands for “ **user interface** design” .

### **Technology for Application development**

JavaScript

Java

HTML

C

Git

C++

Python

CSS

Blockchain

SQL

### **Tools and framework for Application development**

GitHub

Atom

IntelliJ IDEA

Google Docs

Bootstrap

Node.js

Django

Angular JS

React JS

Microsoft Azure

NetBeans

Jenkins