⇒ Software development models and devops

1) Waterfall Model:-

Description:- A linear and sequential approach where each phase must be completed before the next one begins.

phases:

- Requirements
- Design
- Implementation
- Testing
- Deployment
- Maintenance

use case: suitable for projects with well-defined requirements and low risk of changes

2) Iterative model:-

Description:- Develops the system through repeated cycles (iterations), allowing for incremental improvements

phases:-

- planning
- Design
- implementation
- Testing
- Evaluation

use case:- suitable for large projects with clear requirements but expected changes

3. Spiral model:-

Description:- combines iterative development with systematic risk analysis, focusing on risk mitigation

phases:-

- planning
- Risk Analysis
- Engineering
- Evaluation

use case: Best for complex and high-risk projects

4. Agile Model:-

Description:- An iterative and incremental approach emphasizing flexibility, customer collaboration and small frequent releases

phases:-

- Requirements
- Development
- Testing
- Review

use case: Ideal for projects with frequently changing requirements.

5. Lean model :-

Description :- Focuses on maximizing value by eliminating waste and improving efficiency. It emphasizes delivering only what is necessary and continuosly improving the process

phases :-

- Identify value
- Map value stream
- create flow
- Establish pull
- seek perfection

use case: Best for projects aiming to reduce waste, improve process efficiency, and deliver value quickly

6. Devops Model :-

Description :- Integrates development and operations teams to improve collaboration and deploy software faster

phases :-

- continuous Development
- continuous Integration
- continuous Testing
- continuous Deployment
- continuous Monitoring

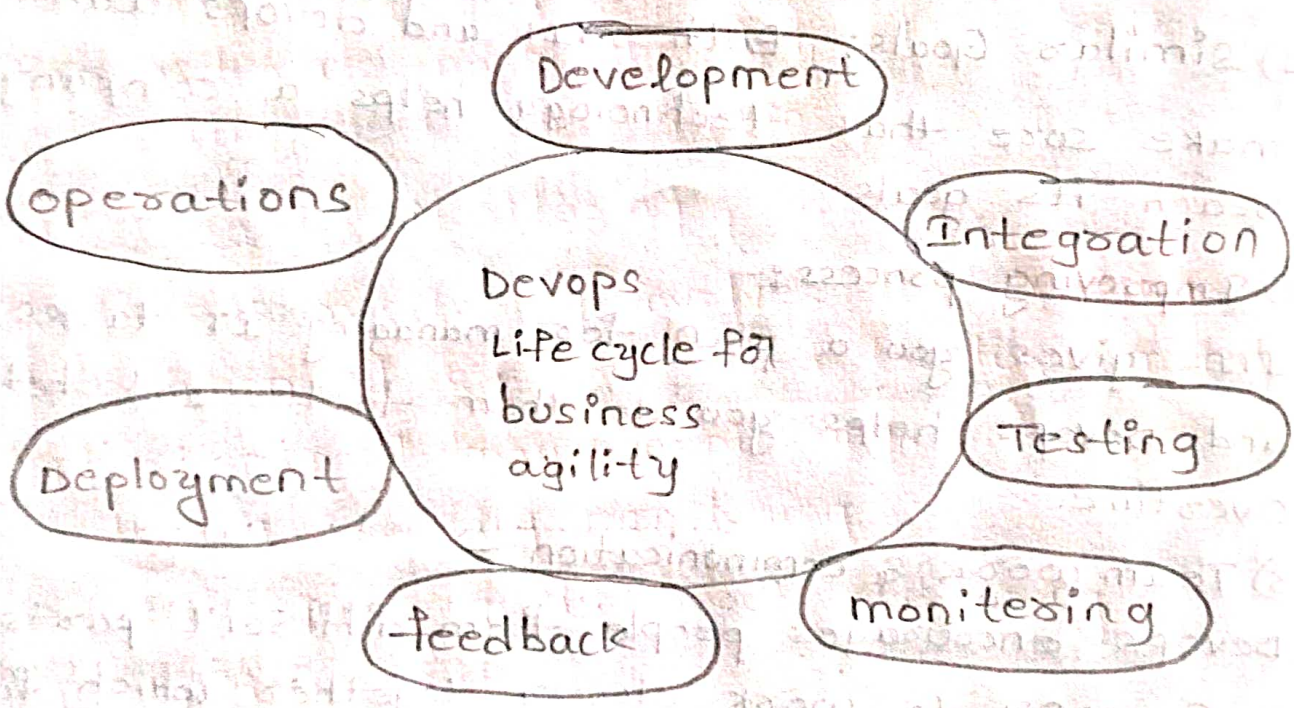
use case :- Best for organizations focusing on rapid delivery, operational efficiency and automation.

→ Devops life cycle for business agility

- Devops defines an agile relationship between operations and development
- It is a practice that is practiced by the development and operational engineers, together from begining to the final stage of the product

Business Agility:-

Business agility means a company can easily adjust to market changes by being flexible and quick to adapt



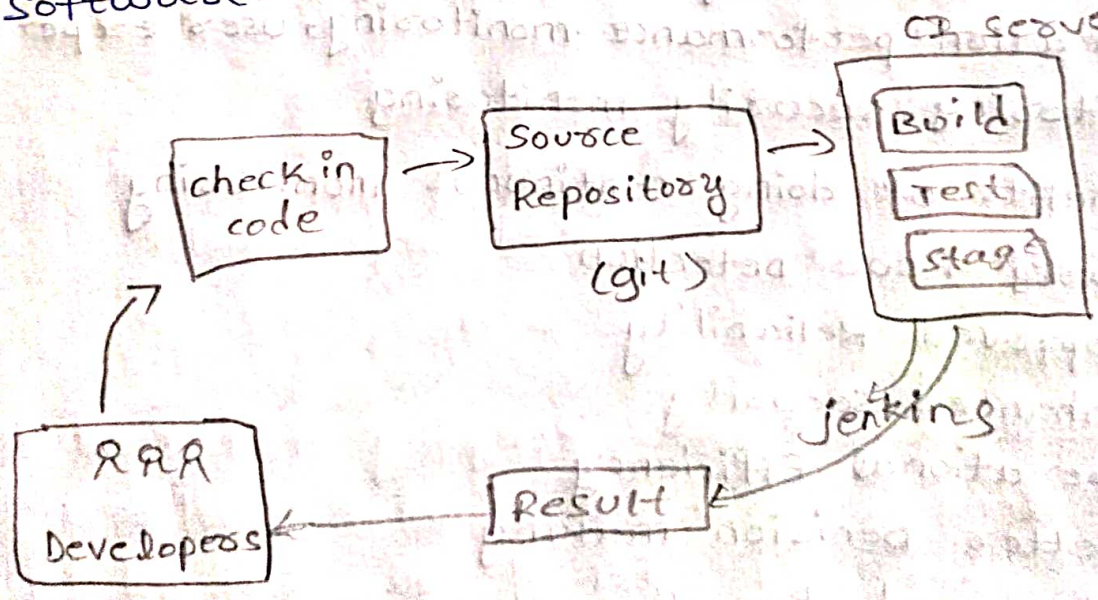
1) Continuous Development:-

- This plan involves the planning and coding of the software
- The vision of the project is declared during the planning phase
- The developers began developing the code for the application

- There are no devops tools required for planning, but there are several tools for maintaining the code

2) Continuous Integration:-

- It is a development practice in which the developers require to commit changes to the source code more frequently
- This may be on a weakly or daily basis
- when every commit is built, and this allows early detection of problems if they are present
- Building code not only involves compilation, but it also includes unit testing, integration testing, code review and packaging
- The code supporting new functionality is continuously integrated with the existing code.
- Therefore, there is continuous development of software.



- jenkins is a popular tool used in this phase

3) Continuous testing:-

- It is a phase, where the developed software is continuously testing for bugs
- For Constant testing, automation testing tools such as TestNG, Junit, selenium etc are used
- These tools allows quality Analysis to test multiple code-bases thoroughly in parallel to ensure that there is no flaw in the functionality
- In this phase, Docker containers can be used for simulating the test environment
- Selenium does the automation testing, and TestNG generates the reports. This entire testing phase can automate with the help of a continuous integration tool called jenkins

4) Continuous Monitoring:-

continuous monitoring involves the Ongoing observation of systems, infrastructure monitoring, Application performance monitoring, user experience monitoring, security monitoring

- Benefits of doing continuous monitoring
  - Early issue Detection
  - improved Reliability
  - Enhanced security
  - operational Efficiency
  - Better Decision making
- Tools used for this phase are:-
  - prometheus
  - Grafana

5) Continuous Feedback

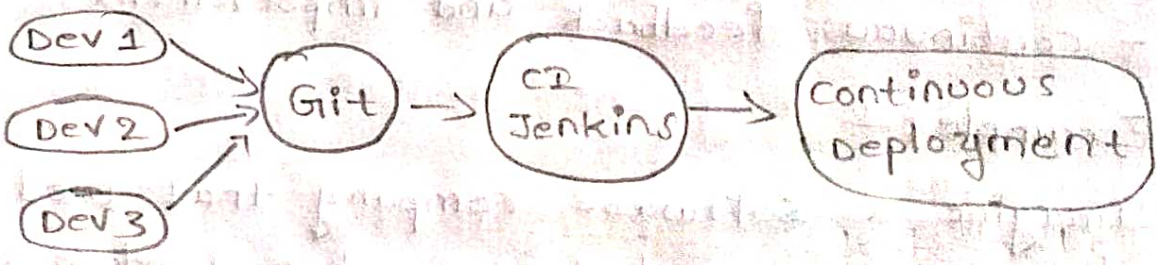
Continuous feedback in devops is critical for achieving business agility. It involves the constant collection, analysis, and integration of feedback from various stages of the development and operations to improve services.

- continuous feedback involves
  - customer feedback
  - Development feedback
  - operational feedback

6) Continuous Deployment

continuous deployment in devops is the practice of automatically deploying every change that passes automated tests to production. This approach enhances business agility by enabling rapid and reliable delivery of features and updates.

- key aspects of continuous deployment includes
  - Automated testing
  - Deployment pipelines
  - monitoring and logging
  - Rollback Mechanisms



- Docker tool is used for this phase

7) Continuous operations:-

Continuous operations in DevOps aim to ensure that systems and applications run smoothly without interruption, even during updates or changes. This approach supports business agility by maintaining high availability, reliability, and performance.

⇒ Impacts of DevOps on Achieving Business Agility

Business Agility refers to the ability of a business to adapt rapidly and efficiently to market changes, customer demands, and other external factors. DevOps, a set of practices that combine software development (Dev) and IT operations (Ops), plays a crucial role in enhancing business agility through several mechanisms

- Faster time-to market
- improved collaboration
- Enhanced Quality and Reliability
- Scalability and flexibility
- Continuous feedback and improvement

Example:-

Imagine a software company that used to have long waits between when developers finished their work and when it was deployed by the operations team. With DevOps they implemented automated systems that allows developers to send their code straight to deployment. This means new features and updates can reach

customers much faster and able to adopt quickly to customer feedback

Examples of companies that have adopted DevOps to achieve faster software delivery and increased customer responsiveness :-

Amazon :-

Amazon's DevOps practices enable them to rapidly deploy updates and new features to their e-commerce platform

For example, during peak shopping seasons like prime Day, they smoothly handle increased traffic and transactions, ensuring a smooth customer experience.

Netflix :-

Netflix relies on devops to continuously enhance their streaming service

Google :-

Google's use of containerization and kubernetes for managing its infrastructure has allowed it to maintain high levels of scalability & reliability

Facebook :-

Facebook's adoption of devops, including extensive use of automation and continuous monitoring, has facilitated rapid deployment cycles and robust performance. This results to delivery of high-quality user experience.

⇒ Continuous Testing in DevOps

In DevOps, continuous testing means testing your software non-stop, from the moment you start writing it until its ready to place in a server. its about automating tests so they run automatically whenever there's a change, helping to catch bugs early and ensure the software works well.

Definition and example for each type of continuous testing:-

1) unit testing:-

Tests individual components or modules for correctness

example: Testing a function that calculates the sum of two numbers

```
def test_sum():
    verify sum(1,2) == 3
```

Expected output: 3

2) Integration testing:-

Tests interactions between different modules or services

Example:- Testing user authentication and database retrieval.

```
def test_user_authentication():
    User = authenticate('username', 'password')
    verify user.is_authenticated is True
```

Expected output:- True

3) Functional testing:-

Validates that the software performs and functions correctly according to the requirements.

Example:- Testing a search feature to ensure it returns relevant results.

```
def test_search_function():
    response = client.get('/search?q=test')
    verify 'search result for "test"' in response.data
```

Expected output:- 'search results for "test"'

4) performance testing:-

Assesses speed, responsiveness, and stability under load

Example:- Testing response time with 1000 concurrent users.

```
def test_response_time_under_load():
    response_times = simulate_load(1000)
    verify all(rt < 2 for rt in response_times)
```

Expected output:- All response times < 2 seconds

5) security testing:-

Identifies vulnerabilities that could be exploited.

Example:- Testing for SQL injection vulnerabilities

```
def test_sql_injection():
    malicious_input = "' OR '1'='1"
    response = client.post('/login', data={'username':
    'admin', 'password': malicious_input})
    verify 'Invalid credentials' in response.data
```

Expected output: 'Invalid credentials'

challenges:-

1) Automation complexcity:-
Automating tests across various environments and platforms can be complex & time consuming

2) Tool integration:-
Integrating testing tools with other devops tools can be challenging

3) Test Data management:-
Ensuring the availability and manageability of test data for various scenarios and environments can be difficult

Benefits:-

- Early Bug detection
- faster feedback loop
- improved Release confidence
- cost and time saving
- collaboration between development, testing and operations teams

Tools commonly used for continuous testing in Devops includes:-

1) Junit & TestNG:- Unit testing
2) selenium & Cypress:- Functional testing
3) Jenkins & Travis CI:- continuous testing
4) JMeter & Gatling:- performance testing
5) OWASP ZAP & Nessus:- Security testing

⇒ Devops influence on Architecture:-

Introducing software Architecture:

Introducing software architecture in Devops is crucial for creating resilient, scalable, and maintainable systems. It involves the high-level design of a system, including its structure, components, and interactions. In Devops, architecture is developed collaboratively by developers, operations, and security teams to meet all stakeholder's needs and align with organizational goals.

Key principles in Devops software architecture include:

- Scalability:- Design to accommodate growth, increased usage, new features, and changing requirements.
- Resilience:- Ensure the system can withstand hardware failures, software bugs, & security breaches.
- Maintainability:- Make it easy to maintain and update, including adding new features & fixing bugs.
- Security:- Incorporate authentication, access control, and data encryption.
- Flexibility:- Ensure adaptability and the ability to integrate with third-party systems and technologies.

Benefits of Devops Architecture:-

1. Cost Reduction:- Enhanced efficiency in software production leads to lower operational costs

2. Increased productivity and faster Release Time:- shorter development cycles and streamlined processes boost team productivity and speed up software deployment.

3. Improved customer Satisfaction:- Emphasis on user feedback ensures client needs are met, leading to higher customer satisfaction.

4. Enhanced Efficiency over Time:- simplified development lifecycle and improved requirements gathering foster a culture of accountability, collaboration, and transparency.

⇒ The Monolithic scenario:-

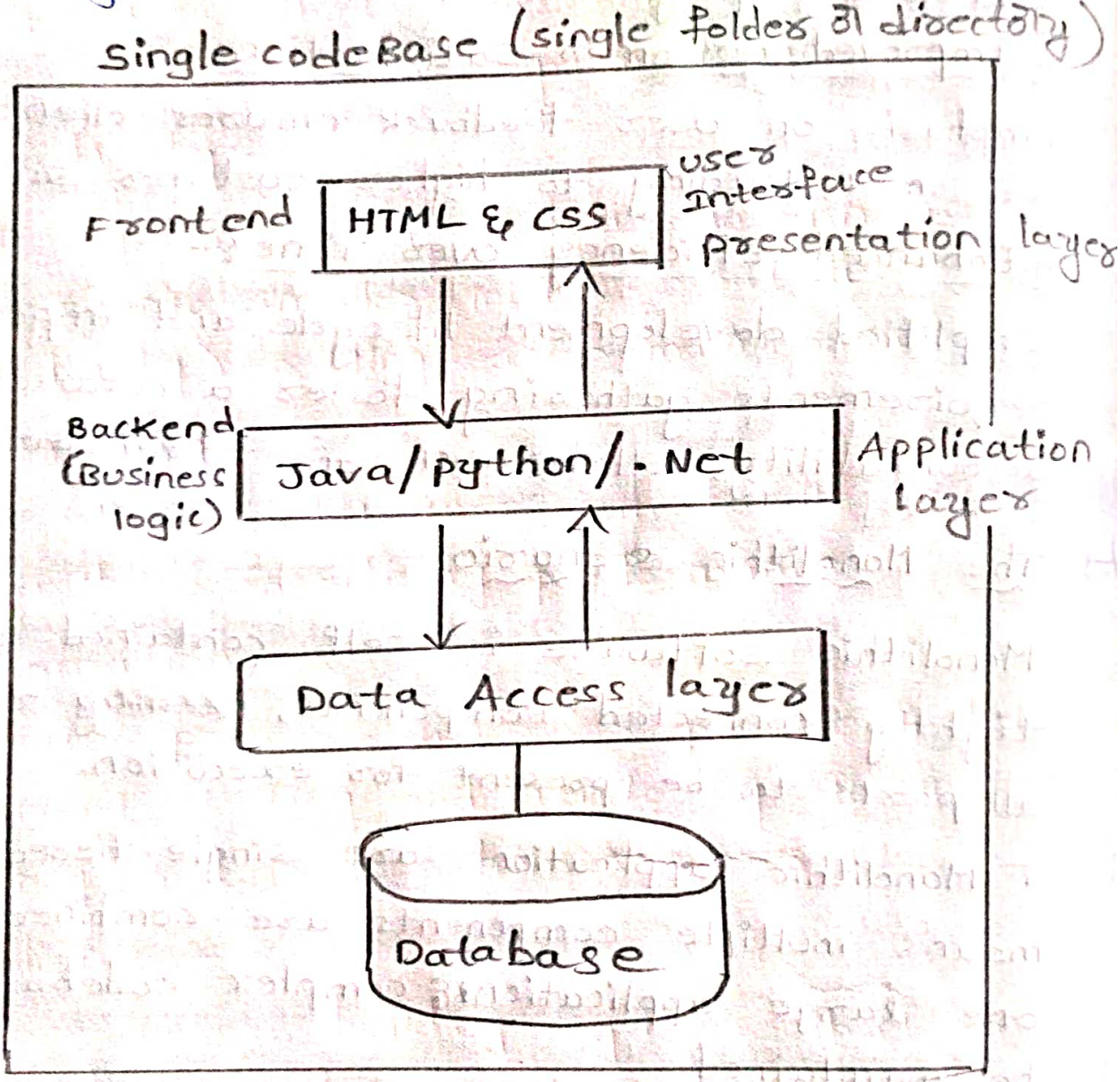
Monolithic software is self-contained with tightly connected components, requiring all parts to be present for execution.

. Monolithic applications are single-tiered, which means multiple components are combined into one large application & complex codebases can be maintained

updating one part often requires changes to others and a full recompilation is done

Monolithic Architecture in DevOps

Monolithic architecture in devops refers to a traditional software design where all components of an application like Front end, Backend and Database all these are tightly integrated into a single code base, which means everything is kept together with in a single folder



Monolithic Architecture

.In monolitic architecture, user interface, Business logic and data Access layer works as a single unit. so it's easier to manage all operations because all components of the software located in one place

. In monolithic architecture, changes made in one layer can affect other layers because they are tightly integrated

Example:-

If I created a college website and want to add a new feature such as new course, I would need to modify the presentation layer, the business logic layer and the data Access layer accordingly.

Benefits of monolithic Architecture:-

- simple to design
- single codebase
- Integration is simple
- Easier deployment
- Best for simple Applications

Drawbacks of monolithic Architecture

- when one component fails entire system fails
- error identification is difficult
- limited Agility
- challenging updates

⇒ Architecture Rules of thumb in devops :-

- "Rules of thumb" are general principles or guidelines widely used in a specific area. They are not strict rules, more like practical suggestions to achieve positive results in system design and implementation
- In Devops architecture, they represent general best practices that teams can follow to achieve effective and efficient system design.

1) Modularity :-

Design systems in a modular fashion (i.e, break software into smaller modules that work independently) enabling easier maintenance, updates

2) Scalability :-

Ensure that architectures can handle increased load without sacrificing performance

3) Automation :-

Automate processes wherever possible, including deployment, testing and monitoring, to increase efficiency and reduce human error

4) Resilience :-

Resilience means building systems that can handle failures by including backup plans, automatic switches

5) Infrastructure as code (IOC) :-

Manage infrastructure like settings to track changes, recreate setups and setup automatically

6) continuous integration / continuous deployment (CI/CD):-
Automate CI/CD process by creating pipelines

7) Monitoring and logging:-
creating monitoring and logging systems to check system performance and delete issues easly

8) Security:-
Integrate security practices into every stage of the development and deployment process, including encryption & access control.

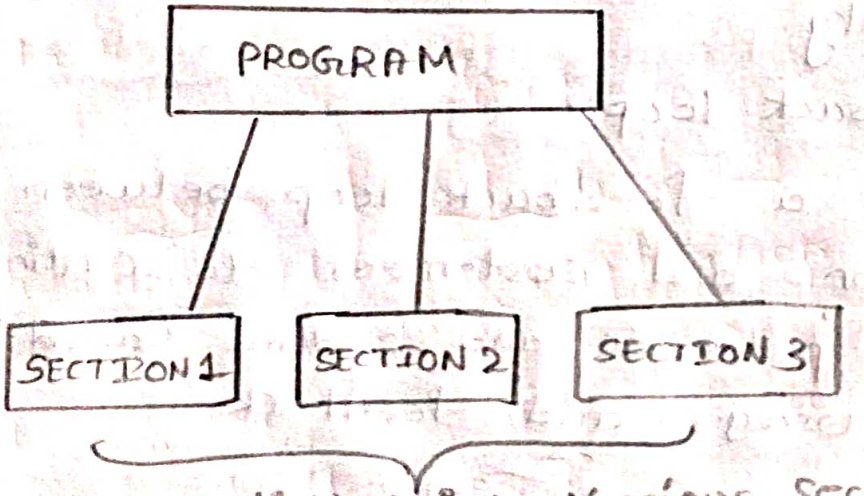
9) containerization:-
utilize containerization technologies like docker tool to package and manage applications efficiently

10) Feedback loop:-
Establish a feedback loop between development, operations and customess to continuously improve processes and systems based on real world usage and feedback.

⇒ Separation of Concerns in Software Architecture:-

Separation of concerns (SOC) means dividing entire program into various sections based on their functionalities, so that each section runs independently without affecting other services. changes made in one section will not affect other sections

"seperation" highlights the act of distinguishing aspects, while "concerns" refer to specific functionalities or responsibilities, such as user interface, data storage, business logic, error handling, and security.



program is divided into various sections based on their functionalities.

Separation of concerns (SoC) will improve

Maintainability:- Easier to update and fix parts of the code without affecting the whole system.

Scalability:- simple to expand the system to handle more load and simple to add new features

Readability:- code is easy to understand, making it simpler for developers to work with that code.

Debugging:- simple to identify and fix bugs

Example:-

I am creating a college application that consist of four pages, so I will divide it into four modules,

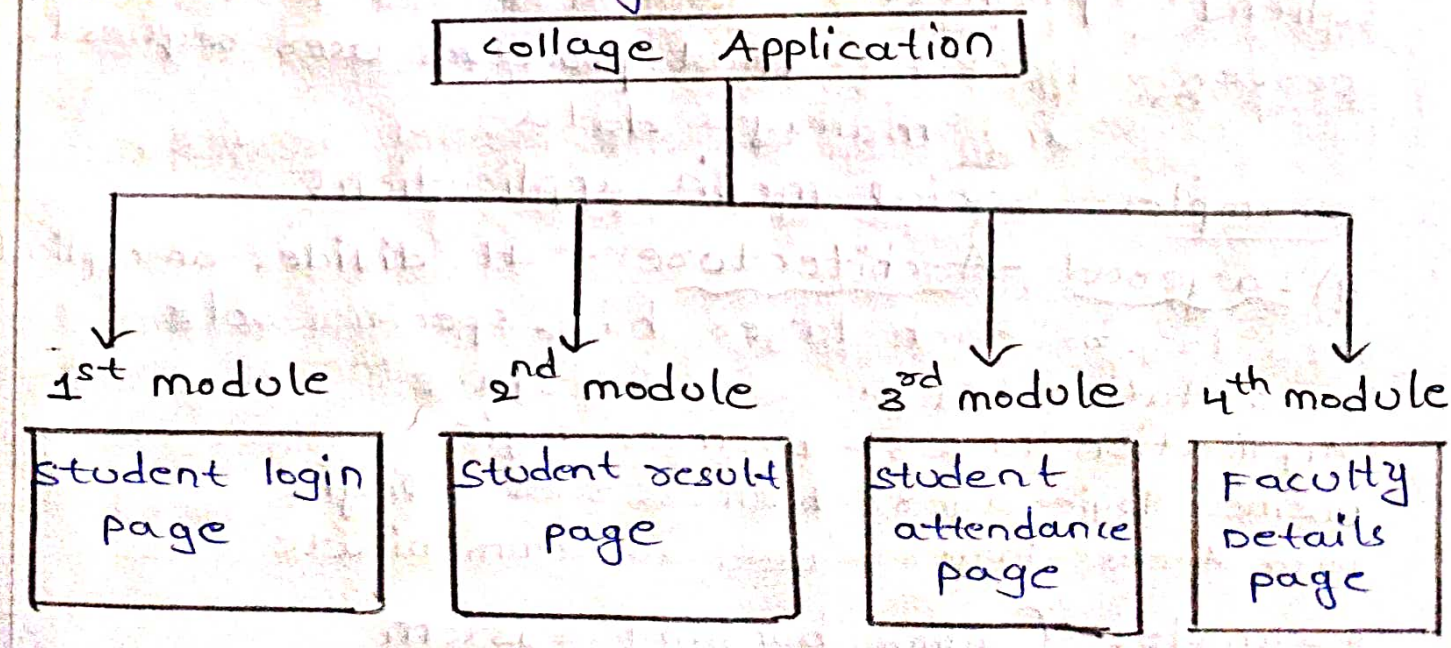
First module contains code related to student login page

Second module contains code related to student results page

Third module contains code related to student attendance page

Fourth module contains code related to faculty details page

changes made in one module will not affect other modules because they all are seperated.



Real world Examples of SoC implementation:-

1) Model - View - Controller (MVC) Architecture:-

MVC is a classic example of SoC because it divides an application into three components

Model:- Manages data and business logic

View:- Handles the presentation layer

Controller:- interface between model & view process user inputs

example:- web applications

2) Micro Services Architecture:-

In a micro service Architecture, an application is built as a collection of loosely coupled services. where each service will perform certain task

example:- E-commerce platforms

3) client - Server Architecture:-

This application splits an application into two main components client & server

client:- Frontend that interacts with users

server:- Backend that processes user requests and manages data

example:- Social media applications

4) Layered Architecture:- It divides an application into layers, each layer has specific role

presentation layer:- user interface

Application layer:- business logic

Business logic layer:- business rules & logic

Data Access layer:- used to communicate

example:- Online banking system

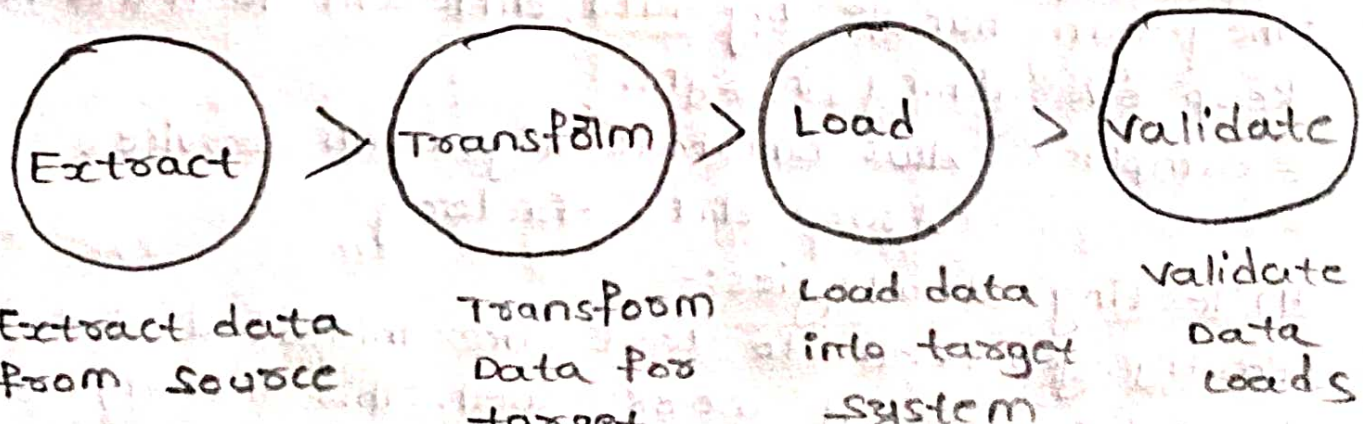
→ Handling Database Migrations in DevOps:-

Database schemas define how data is structured in a relational database. While a well designed schema is crucial, changing requirements often necessitate schema modifications. These changes must be carefully managed to avoid data loss and ensure applications continue to function correctly.

Database migrations are controlled changes that modify a database's structure. These changes can include adding or removing tables and columns, altering data types, or changing constraints. Migrations help transition a database from its current state to a desired state in a way that is repeatable, shareable, and testable. while ensuring it's done smoothly and without disturbances

This is important because changes to the database structure or data need to be matched with application updates to avoid errors.

Database Migration



Techniques used for database migration in devops:—

1) Version control:- (Git)

One of the essential techniques for database migration in devops is to use version control for both the application code and the database schema.

Tools like Liquibase and flyway can be integrated into CI/CD pipelines to automatically apply database schema changes when corresponding changes are made to the code.

2) Schema migration tools:—

These tools automate making and running SQL scripts that change your database structure they prevent mistakes and reduce downtime

examples:- SQL server data tools
SQL work bench

3) Data migration tools:—

Sometimes you need to move data between databases Data migration tools make this easy and safe. They can handle big data, different formats and keep everything in sync.

examples:- Aws Database migration Services
Azure data factory

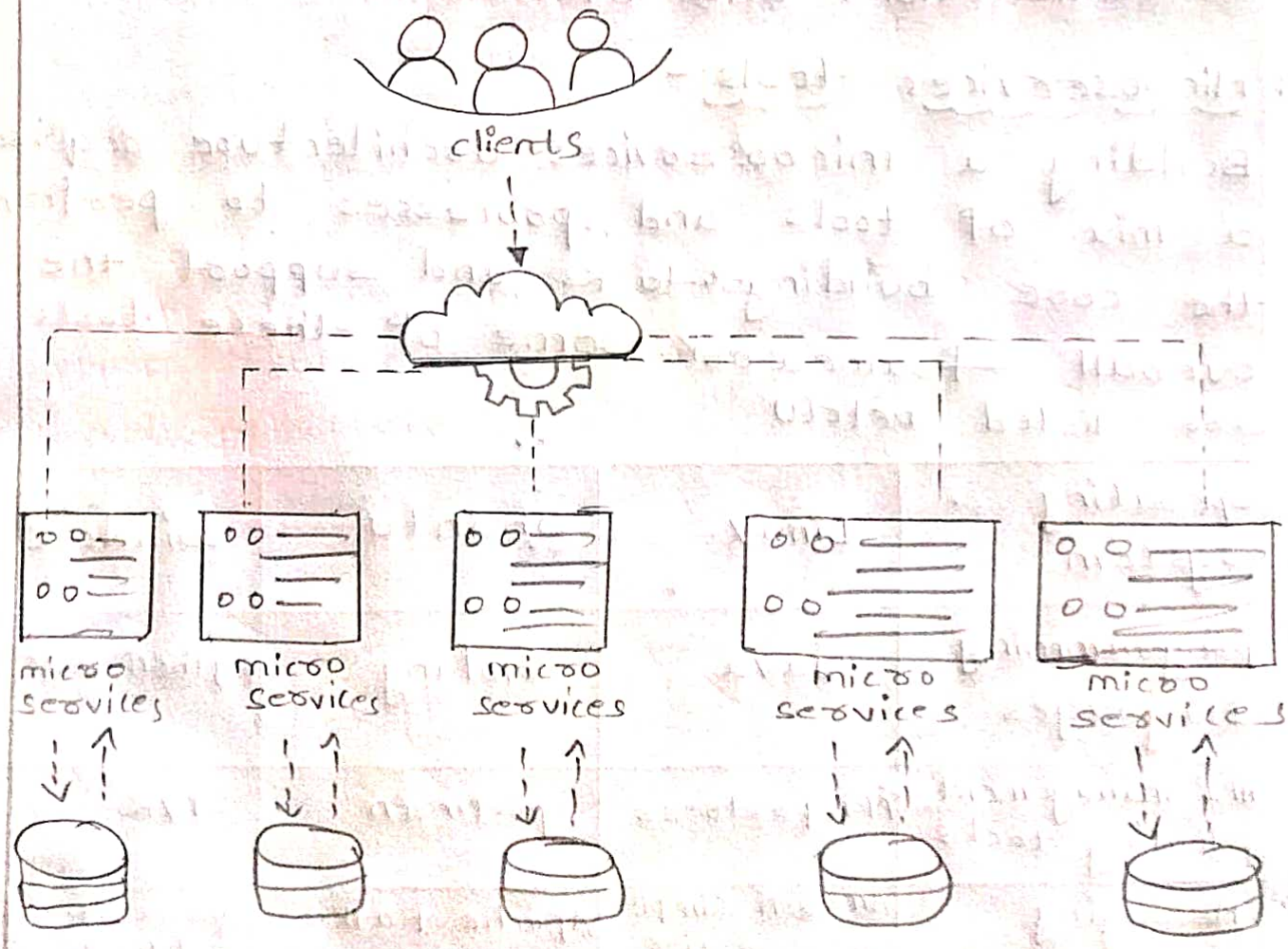
4) Testing strategies:-

Lastly, you need to test the migrated database to make sure it works well. This includs various tests like checking performance and security. It ensures the databases meets all requirements and fixes any issues.

examples:- SQL Test and Jmeter

Microservice Architecture:

Microservices architecture is a software development approach where a large application is divided into smaller, loosely coupled, and independently deployable services. Each service focuses on a specific business function and communicates with other services through APIs. This architecture contrasts with monolithic architectures, where all functionalities are intertwined in a single, large application.



MICROSERVICE ARCHITECTURE

Key benefits of microservices architecture

1) Less development effort
2) Improved scalability
3) Independent deployment
4) Error isolation
5) Integration with various tech stacks

microservices architecture used for :-

-> Data processing
-> media content
-> website migration
-> Transactions and invoices

⇒ Microservices tools:-

Building a microservices architecture requires a mix of tools and processes to perform the core building tasks and support the overall framework. some of these tools are listed below

operating System	Linux	ubuntu	window
programming Danguages	Java	Golang	Python, NodeJs
API management & testing tools	API fortress	postman	TYK
Messaging tools	Amazon Simple Queue service (SQS)	Apache Kafka	google cloud Pub/sub
Toolkits	Fabric 8	Google cloud Functions	Seneca
Architectural Frameworks	Helidon	Quarkus	Molecular

orchestration Tools	kubernetes	Azure Kubernetes Service (AKS)	conductore
Monitoring tool	Logstash	middleware	Elastic Stack
serverless tools	claudia	Apache openwhisk	kubeless

⇒ Data tier :-

The data tier in devops refers to the layer of the application architecture that is responsible for storing, retrieving, and processing data. The data tier is typically composed of databases, data warehouses, and data processing systems that manage large amounts of structured and unstructured data.

In Devops, the data tier is considered an important aspect of the overall application architecture and is typically managed as part of the Devops process. This includes:

1. Data management and migration:- Ensuring the data is properly managed and migrated as part of the software delivery pipeline
2. Data backup and recovery:- Implementing data backup and recovery strategies to ensure that data can be recovered in case of failures or disruptions
3. Data security:- Implementing data security measures to protect sensitive information and comply with regulations.

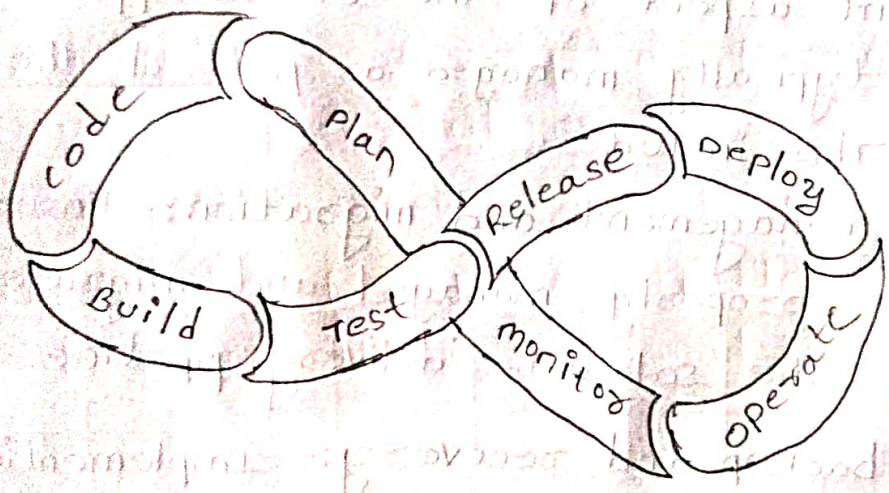
4. Data performance optimization:- optimizing data performance to ensure that applications and services perform well, even with large amounts of data.

5. Data integration:- Integrating data from multiple sources to provide a unified view of data and support business decisions.

By integrating data management into the Devops process, teams can ensure that data is properly managed and protected, and that data-driven applications and services perform well and deliver value to customers.

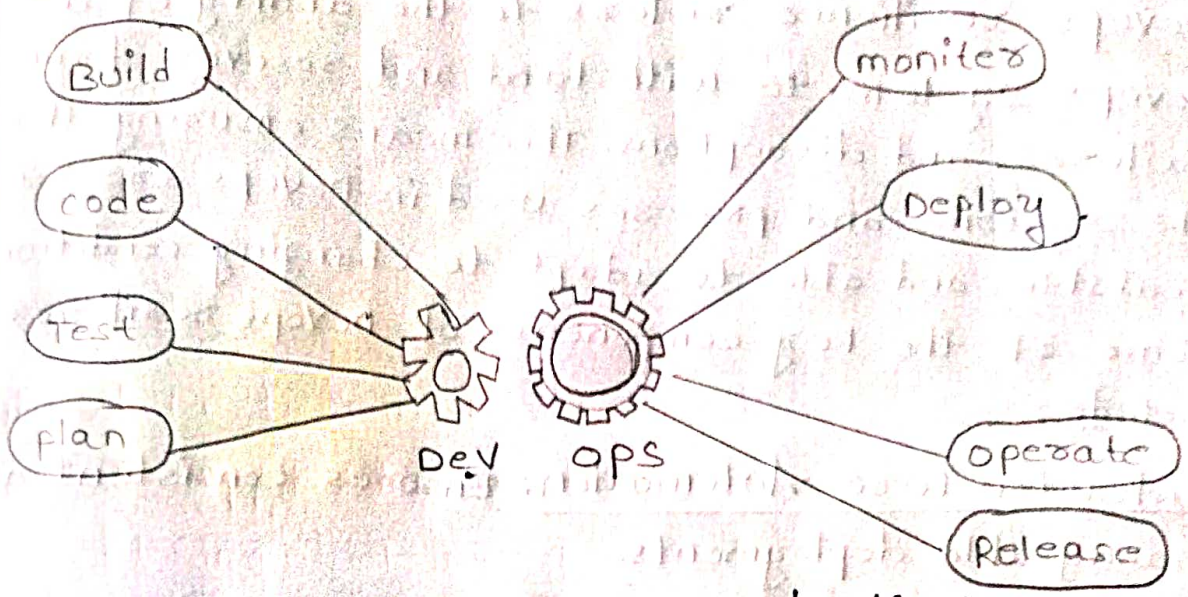
⇒ Devops architecture and resilience

⇒ Devops Architecture



Development and operations are crucial for delivering applications. Devops bridges the gap between these areas by combining them, resulting in faster delivery. It is especially effective for cloud-based and large distributed applications. Using Agile methods within Devops allows for continuous integration and delivery. By aligning development and operations, Devops minimizes delays & boosts productivity

Devops Components



Build: utilizes cloud resources and adjusts to user needs, optimizing resource consumption

code: Employs tools like Git for version control and tracking changes, ensuring code quality and manageability

Test: Automates testing to reduce time and effort, speeding up deployment to production

plan: Uses Agile methodology for effective planning and increased productivity.

Monitor: Continuously tracks system health and performance with tools like splunk to detect and address issues early.

Deploy: Automates deployments with scheduling and cloud management tools, ensuring accurate and efficient deployment.

Operate: Encourages collaboration between development and operations throughout the service lifecycle, improving overall efficiency.

Release: Manages production deployments with manual triggers to minimize customer impact.

→ Devops Resilience

Devops resilience refers to the ability of a Devops system to withstand and recover from failures and disruptions. This means ensuring that the systems and processes used in Devops are robust, scalable, and able to adapt to changing conditions. Some of the key components of Devops resilience include:

Infrastructure Automation: Ensures consistent and manageable deployments.

Monitoring and Logging: Facilitates quick issue detection and diagnosis

Disaster Recovery: provides a plan and tests for rapid recovery from disruptions

Continuous Testing: Identifies and resolves issues early.

High Availability: Designs systems to remain operational despite failures.

These practices ensure a robust, scalable, and adaptive Devops system capable of delivering high-quality applications even amid disruptions.