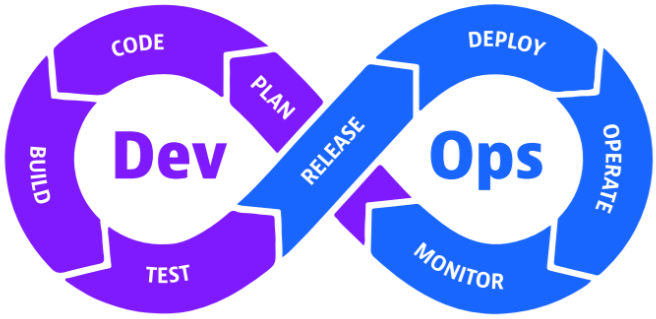
1. what is DevOps diagrams and activities?

DevOps is a flexible framework of software development practices organizations use to create and deliver software by aligning and coordinating software development efforts between two key teams: development (or “Dev") and IT operations (“Ops”). The easiest way to understand DevOps is to visualize it as a continuous loop. Instead of discrete processes, tasks in development and operations become part of an ongoing cycle that includes building, testing, and monitoring applications and services. The DevOps methodology embodies a cultural shift that requires vision, planning, executive buy-in, and tight collaboration to successfully establish an integrated way of developing and delivering applications.



1.DevOps infinitive loop

By embracing a few fundamental practices, teams can improve their efficiency and develop a deeper understanding of their workflows, toolsets, and processes to release better software faster. The following describes the basic tenets and practices that constitute a DevOps approach:

Continuous integration

Continuous integration (CI) is a software development practice in which developers regularly commit their code to a shared repository. With CI, the distributed nature of microservices architecture allows developers to own discrete, manageable chunks of code alongside individual features and work on them in parallel. Each chunk is a software artifact that teams can manage individually.

Continuous delivery

Complementing CI, continuous delivery (CD) takes artifacts that have been pushed to an artifact repository

and then deploys the artifacts to multiple environments with different testing and quality assurance criteria that determine the artifacts’ promotion from one stage to another.

Continuous testing and validation

Continuous testing evaluates software quality at each stage of the delivery lifecycle. Ideally using automation, continuous testing identifies poor-quality code and provides fast and continuous feedback to the development teams with the necessary information to address quality concerns.

2.What is platform engineering?

Platform engineering is a software development and operations discipline that helps software teams build self-service IT capabilities for cloud-native environments. A platform encompasses a set of tools, services, and infrastructure that enables developers to **build, test,** and **deploy** software applications.

3. what is TDD (test-driven development)?

Test-Driven Development (TDD) is a method in software development where the focus is on writing an Automation Tests before writing the actual code for any feature of an application or product. This approach uses short development cycles that repeat to verify the quality and correctness.

TDD simply means a method of coding in which you first write a test, and it fails, then write the code to pass the test of development, and clean up the code. This process is recycled for one new feature or change. In other methods in which you write either all the code or all the tests first, TDD will combine and write tests and code together into one.

Process of Test-Driven Development (TDD)

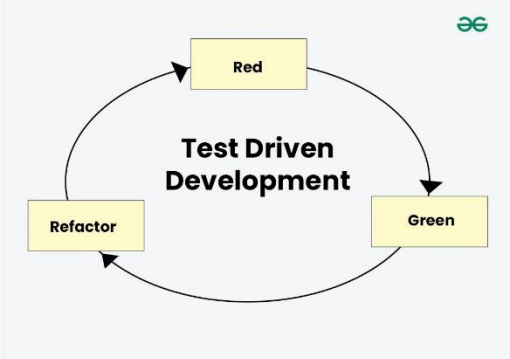
It is the process in which Test Cases are written before the code that validates those cases. It depends on the repetition of a concise development cycle. Test-driven Development is a technique in which automated Unit tests are used to drive the design and free decoupling of dependencies.

The process of Test-Driven Development (TDD) follows a repetitive cycle called Red-Green-Refactor.

Test Driven Development (TDD)

Run all the test cases and make sure that the new test case fails.

- Red – Create a test case and make it fail, Run the test cases
- Green – Make the test case pass by any means.
- Refactor – Change the code to remove duplicate/redundancy and Refactor code – This is done to remove duplication of code.



2.flow of TDD

Once you completed through the Red-Green-Refactor cycle, you continue repeating the process for the next piece of functionality or unit of code. Every time you write a new test, your code gets better and more reliable, making the overall software stronger.

4. what is UML (Unified Modeling Language)?

A UML diagram is a way to visualize systems and software using Unified Modeling Language (UML). Software engineers

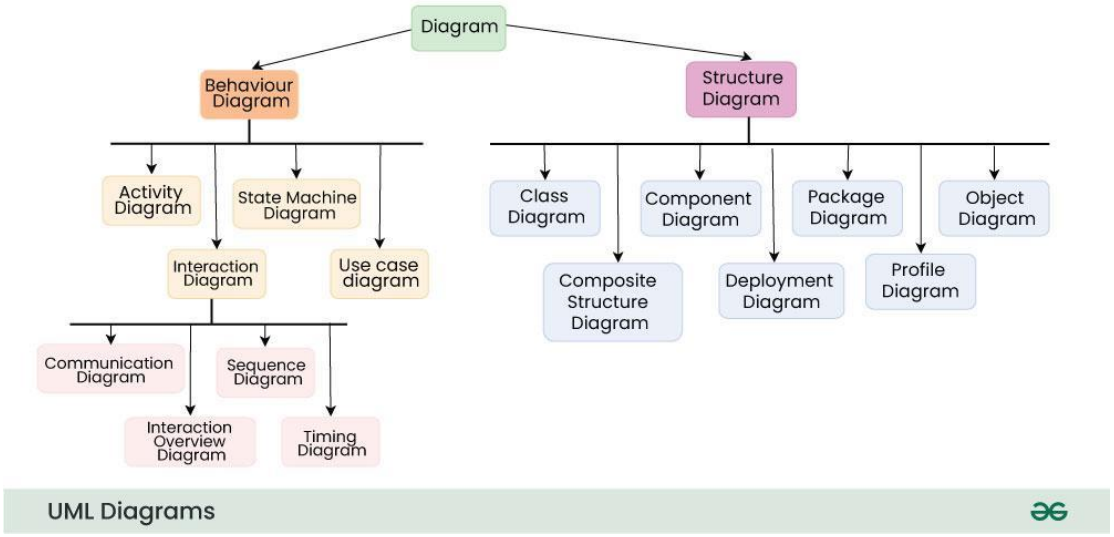
create UML diagrams

to understand the designs, code architecture, and proposed implementation of complex software systems. UML diagrams are also used to model workflows and business processes.

Coding can be a complicated process with many interrelated elements. There are often thousands of lines of programming language that can be difficult to understand at first glance. A UML diagram simplifies this information into a visual reference that's easier to digest. It uses a standardized method for writing a system model and capturing conceptual ideas.

Beyond code, UML diagrams are also great for visualizing relationships and hierarchies within software components, similar to decision trees or flowcharts — but specific to software.

By making complex systems easier to grasp, UML diagrams help engineers and non-technical team members alike track project progress and communicate effectively throughout the software development process. They break down software into essential parts and make it easier to understand how everything fits together.



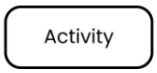
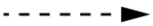
3.UML Diagrams

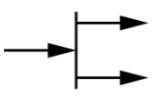
5.what is activity diagram?

An activity diagram is a type of Unified Modeling Language (UML) flowchart that shows the flow from one activity to another in a system or process. It's used to describe the different dynamic aspects of a system and is referred to as a 'behavior diagram' because it describes what should happen in the modeled system.

Even very complex systems can be visualized by activity diagrams. As a result, activity diagrams are often used in business process modeling or to describe the steps of a use case diagram within organizations. They show the individual steps in an activity and the order in which they are presented. They can also show the flow of data between activities.

Activity diagrams show the process from the start (the initial state) to the end (the final state). Each activity diagram includes an action, decision node, control flows, start node, and end node.

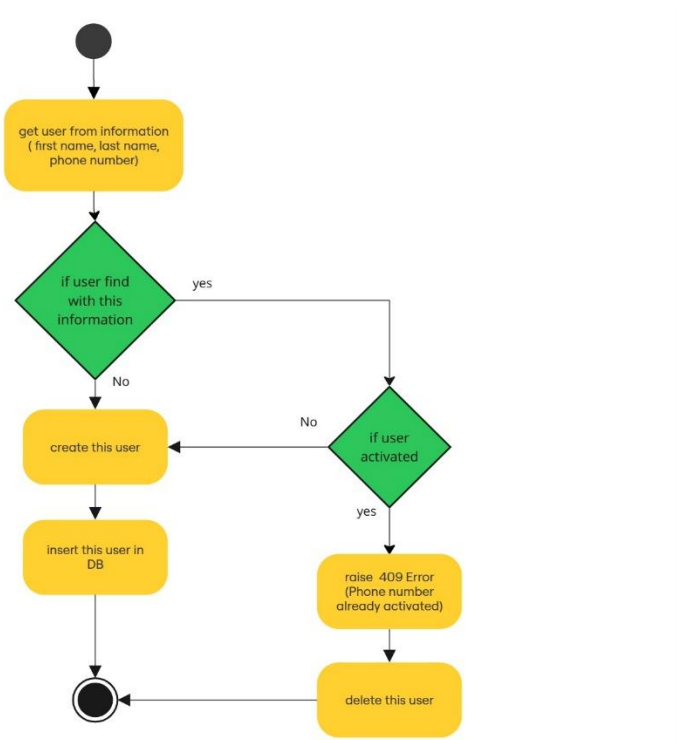
	Initial node	Represents the starting point of an activity.
	Activity state	Represents the activities within the process.
	Action	Represents the executable sub-areas of an activity.
	Control flow	Represents the flow of control from one action to another.
	Object flow	Represents the path of the objects moving through the activity.

	Activity final node	Represents the end of all control flows within the activity.
	Flow final node	Represents the end of a single control flow.
	Decision node	Represents a conditional branch point with a single input and multiple outputs.
	Merge node	Represents the merging of flows with several inputs and only one output.
	Fork	Represents a flow that can branch into two or more parallel flows.

6. Activity diagram in danayar:

- Activity Diagram Description (Endpoint: POST /signup)

    1. Start
    2. For /signup POST:
        - Receive phone number, first name, last name.
        - Check existing phone number:
            - If activated: Raise HTTP 409 Conflict.
            - If unactivated: Delete existing entries, create new user, insert into DB.
        - Return UserResponse for new user.
    3. End



- Activity Diagram Description (Endpoint: POST /auth/token)
    1. Input: Phone number (as username) and OTP code (as password).
       If not fetch: Raise HTTP 422 Error.

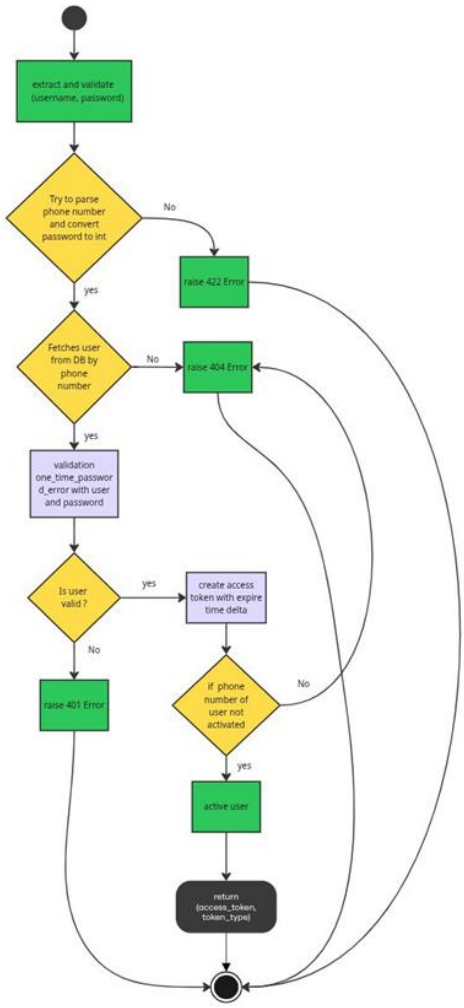
    2. Flow: Parses and validates the phone number and OTP format.
    3. Fetches user from DB by phone number:
       If not fetch: Raise HTTP 404 Error.
       Validates OTP: Checks expiration, correctness, and usage status.

       If valid:

        - Generates JWT access token using create_access_token().
        - Activates user phone number if not already activated.
        - Returns token with type "bearer".

       If OTP invalid or user doesn't exist → Raises 401 HTTP Error.

    4. End



- Activity Diagram Description (Endpoint: POST /Upload):

  The user uploads a file through the application:

  1. The file is first stored as an object in the **MinIO** database (object storage).
  2. Then, metadata or reference information about the file is saved in the **MongoDB** database.
  3. A **background task** is triggered to:
     - Convert the file into a **.md** (Markdown) format.
     - Store the converted **.md** content in the database (**MinIO)**.
     - Generate a short **summary** of the document.
     - Save the **summary** to the database (**MinIO)**.
  4. End