**PRACTICAL-5**

**AIM:**

List at least 10 software design principals & online/offline tools for the software development process and Draw the UML diagram.

**THEORY:**

**Software Design Principles**

The first five principles form the basics of the software development principles and are also known as “S.O.L.I.D principles”

**Single Responsibility Principle (SRP)**

* It's a software engineering theory that says a class should only change for one reason. In other words, it must only be responsible for one thing. We're talking about cohesion here. All elements in a given class hierarchy or module should have the same functional affinity. You can improve your class's cohesiveness by clearly identifying its responsibilities.
* It is denoted by “S”

**Open/Closed Principle (OCP)**

* The notion states that you should be able to change a class's behaviour without having to edit it. As a result, you can expand the behaviour of the class using composition, interface, and inheritance. You cannot, however, open it to make minor changes.
* It is denoted by “O”

**Liskov Substitution Principle (LSP)**

* The LSP principle is primarily concerned with the extent to which a software uses inheritance. While inheritance is advantageous, it is best to apply it in a context and in moderation. The principle aims to avoid situations in which classes are only extended through common things. Before conducting inheritance, you must evaluate the class's pre- and post-conditions.
* It is denoted by “L”

**Interface Segregation Principle (ISP)**

* Many particular interfaces are preferred by ISP over a broad interface. The goal is to create client-specific interfaces that are tightly grained. Interface cohesiveness must be improved, and modules with limited behaviours should be developed. It's difficult to maintain and evolve interfaces with multiple behaviours. As a result, they must be avoided entirely.
* It is denoted by “I”

**Dependency Inversion Principle (DIP)**

* The Dependency Inversion Principle, symbolised by the letter "D," is the fifth and final principle of SOLID. Programmers should rely on abstractions rather than real classes, according to this notion. This can be divided into two parts: We can divide it into two parts:
* High-level modules must be self-contained from low-level modules.
* Abstractions should be used in both cases.
* Abstractions should be devoid of specifics.
* Abstractions should determine the details.

We'll now look at some of the other fundamental principles besides SOLID.

**Keep It Simple**

* It's critical to make sure that the program's coding is simple and straightforward. The code should not be difficult to understand for a newcomer. Methods and functions should be as succinct as possible while yet being clear.
* Only one or two problems should be solved by each of them.
* There should also be few conditions in the project code (simple and nested conditions).
* The QA team would benefit from optimising the circumstances because it would be easier to understand and detect faults.

**You Aren't Gonna Need It (YAGNI)**

* Most programmers fall into the trap of attempting to implement all of the features at the same time, right from the start.
* In the end, some or all of these features will be rendered obsolete. Begin by creating a class with only a few methods.
* After that, as your product takes shape and new requirements emerge, you can add more features. You'll be able to create a lean development software in this manner.
* YAGNI saves you time, effort, and money that you would have spent trying to debug or comprehend the code.

**Measure Twice and Cut Once**

* If not done correctly, the requirement step of the development life cycle frequently causes more than 50% code errors.
* As a result, a methodical methodology should be created. It's critical to double-check all of the project requirements to ensure that no crucial sections (features) are overlooked or that nothing is added inadvertently.
* After that, create blueprints to guide the entire process and ensure high-quality code all the way through. Always test your project from the ground up to ensure everything is in working order.
* This method produces far more predictable results, particularly when the project's cost is already high. You'll avoid the hassles of having to delete or add code lines to meet requirements.

**Don’t Repeat Yourself (DRY)**

* Don't make the same mistake twice when coding your code. To put it another way, don't copy-paste your code into different places. Future maintenance will be tough if this is not done.
* The reason for this is that you'll need to update the coding in a few different places. Adjustments in the tests will be required as a result of these changes in order for the results to turn green.
* All of this will necessitate extra time, effort, and financial resources. You can avoid this issue by extracting common logic into functions. Also, if there are any manual tasks that you can automate to keep your code lean, do so.
* The aforementioned methods will aid in the reusability of code in software development without the need to duplicate it.

**Least Astonishment**

* According to the principle of least astonishment, it's best to avoid designing a feature with a high level of surprise. End-users should be able to assume certain behaviours from your system's components.
* As a result, only clear, predictable, and consistent project outputs will be lucrative. Users will be hesitant to employ features or structures that surprise, surprise, or confound them if they aren't familiar with them.
* You're a software developer who creates products that people can utilise. Thus, creating user-friendly features will pay off handsomely. Make an effort to match the mental models, experiences, and expectations of human beings. Keep in mind that you need to grab the user's attention as soon as feasible. The attention span of today's users has dwindled, as we all know.

**SOFTWARE DEVELOPMENT PROCESS TOOLS:**

1. **JIRA**

Jira is the most popular software development tool that is used by agile teams for planning, tracking and releasing the software.

**Features:**

* This tool is customizable and also has some prevailing features that are used in every development phase.
* Using Jira, we can accomplish the work in progress, generate reports, backlogs etc.
* Few other important features of Jira software are Scrum boards, Kanban boards, GitHub integration, Disaster recovery, Code Integration, Portfolio Management, Sprint Planning, Project Management etc.
* Jira works for Windows and Linux/Solaris operating systems.

1. **Atom**

Atom is a solid all-around text-editor. It is fully free and open source. It can be customized to do anything but without a need of modifying the config file.

**Features:**

* Atom works across many popular operating systems like OS X, Windows, or Linux.
* It helps developers to write code faster with a smart, flexible autocomplete.
* Easily browse and open whole project or multiple projects in one window.
* It is possible to split Atom interface into multiple panes to compare and edit code across files.
* Find, preview, and replace text type in a file or across the entire project.

1. **GitHub:**

GitHub is a powerful collaboration tool and development platform for code review and code management. With this GitHub, the users can build applications and software, manage the projects, host the code, review the code etc.

**Features:**

* With GitHub, developers can easily document their code and can host the same from the repositories.
* GitHub’s project management tools help its users to stay aligned, co-ordinate easily and get their task done accordingly.
* Few features of GitHub that make it a useful tool are its code security, access control among the team members, integration with other tools etc.
* Few developers use GitHub for experimenting new programming languages in their personal projects.
* GitHub can be hosted on servers and on a cloud platform. It runs on Windows and Mac OS.
* GitHub is free for open source projects and public use. For developers it is charged based on different criteria and services requested.

1. **Embold:**

Fixing bugs before deployment saves a lot of time and energy in the long run. Embold is a software analytics platform that analyses source code and uncovers issues that impact stability, robustness, security, and maintainability.

**Features:**

* With the Embold plugins, you can pick up code smells and vulnerabilities as you code, before making commits.
* Unique anti-pattern detection prevents the compounding of unmaintainable code.
* Integrate seamlessly with Github, Bitbucket, Azure, and Git and plugins available for Eclipse and IntelliJ IDEA.
* Get deeper and faster checks than standard code editors, for over 10 languages.

1. **Linx:**

Linx is a low code IDE and server. IT pros use Linx to quickly create custom automated business processes, integrate applications, expose web services and to efficiently handle high workloads.

**Features:**

* Easy-to-use, drag-and-drop interface
* Over 100 pre-built functions and services for rapid development
* One-click deployment to any local or remote Linx Server directly from the IDE
* Input and outputs include nearly any SQL & NoSQL databases, numerous file formats (text and binary) or REST and SOAP Web services
* Live debugging with step through logic
* Automate backend processes via timer, directory events or message queue or expose web services, and call APIs via HTTP requests

1. **Kite:**

Kite is IDE for Software Development that automatically completes multiple line codes. This editor supports more than 16 languages. It helps you to code faster with no hassle.

**Features:**

* It offers Software Development documentation.
* This editor provides a function signature as you type.
* You will get a tooltip on mouse hover.
* Provides support in email.
* Uses machine learning models for Software Development language.
* Also it is a free to use open source tool.

1. **Studio 3T:**

Studio 3T for MongoDB helps you to build queries fast, generate instant code, import/export in multiple formats, and much more.

**Features:**

* Query MongoDB faster with our Visual Query Builder, IntelliShell, or SQL Query tool.
* Our Data Masking tool enables data compliance and bolsters security with powerful field-level data obfuscation.
* Import to MongoDB from JSON, CSV, BSON/mongodump, and SQL, and get a preview of your output documents as you make changes.
* Migration from MongoDB to SQL (or vice versa) has never been easier with our Migration tools."

1. **Bitbucket:**

Bitbucket is a distributed, web-based version control system that is used for collaboration between software development teams (code and code review). It is used as a repository for source code and development projects.

**Features:**

* Useful features of Bitbucket that makes it a powerful tool are its flexible deployment models, unlimited private repositories, code collaboration on steroids etc.
* Bitbucket supports few services like code search, issue tracking, Git large file storage, bitbucket pipelines, integrations, smart mirroring etc.
* Using Bitbucket, one can organize the repositories into the projects with which they can focus easily on their goal, process or product.
* To rationalize the development process of any software it can integrate into the prevailing workflow.
* Bitbucket offers a free plan for 5 users with unlimited private repositories.

1. **Cloud9 IDE:**

Cloud9 IDE is an online integrated software development environment. It supports many programming languages like C, C++, PHP, Ruby, Perl, Python, JavaScript and Node.js.

**Features:**

* Allows to clone entire development environment.
* Built-In Terminal for command-line wizard.
* Code Completion suggestions helps software developers to code faster and avoid typos.
* The Debugger helps developers to set breakpoints, and inspect variables of any JS/Node.js app.
* Simply drag any file or Terminal to create multiple split views.
* Developers can select an extensive set of default Runners to execute app, such as Ruby, Python, PHP/Apache.

1. **NetBeans:**

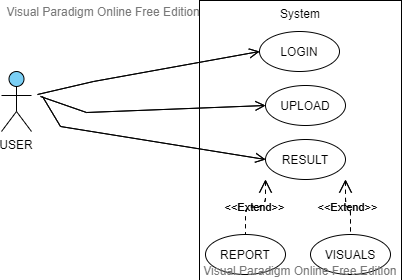
NetBeans is an open source and a free software development tool written in Java that develops world-class web, mobile, and desktop applications easily and quickly. It uses C / C++, PHP, JavaScript, Java etc.

**Features:**

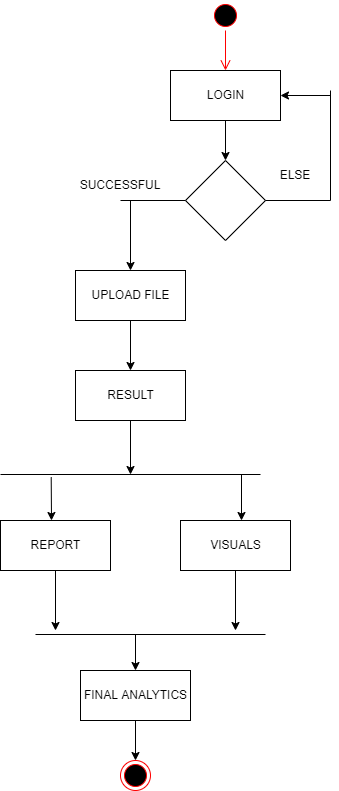
* Support for fast & smart code editing.
* Easy & Efficient Project Management process.
* Rapid User Interface Development.
* Helps to write bug-free code.
* NetBeans IDE offers superior support for C/C++ and PHP developers.

**UML DIAGRAM:**

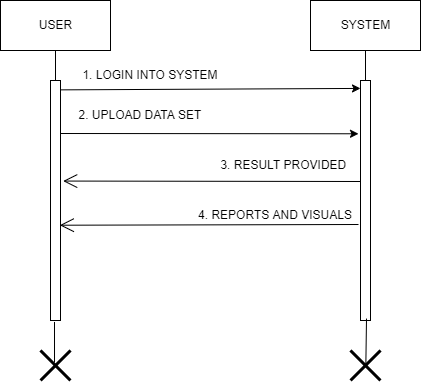
**USE-CASE DIAGRAM:**



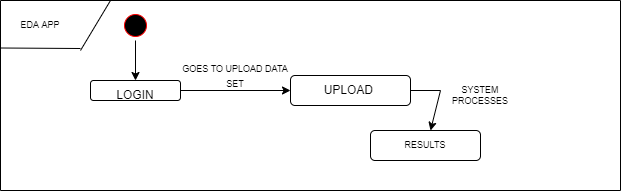
ACTIVITY DIAGRAM:



SEQUENCE DIAGRAM:



STATE DIAGRAM



LEVEL 0 DFD

DISPLAY RESULT

USERR

UPLOADS DATASET

EDA SYSTEM

DOWNLOAD RESULT

LEVEL 1 DFD

FORMATTED RESULT

USER

Data set Storage

EDA PROCESS

DOWNLOAD REPORT

**CONCLUSION:**

By performing the practical, I learnt the basics of software design principles and software development process and UML diagram.