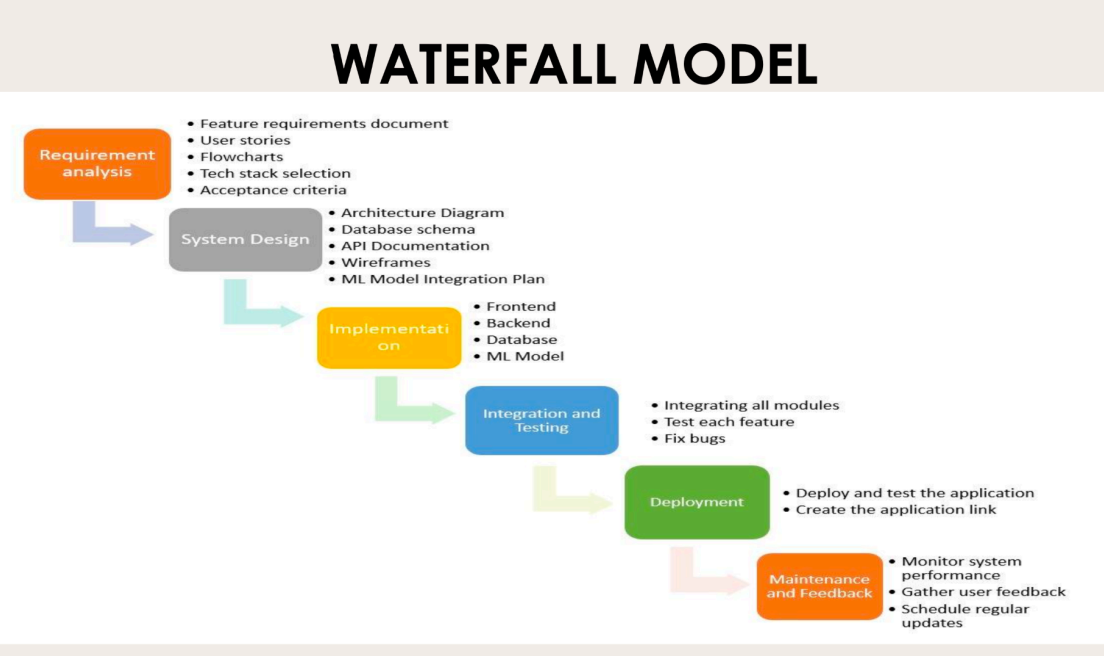
# LAB-1

**Software Engineering:**

Software engineering is the systematic approach to designing, developing, testing, deploying, and maintaining software applications. It applies engineering principles to software development to ensure the creation of high-quality, scalable, and maintainable software solutions.

**Traditional SDLC model:**

- **Waterfall Model:** The Waterfall model is a linear, sequential software development approach best suited for projects with well-defined requirements, where each phase follows a set path with minimal deviation, making it difficult and costly to revisit previous stages once completed.




- **V-model:** It is a structured SDLC approach extending the Waterfall model, where each development stage has a corresponding testing phase, ensuring continuous verification, validation, and full traceability between requirements and tests
- **Spiral model:** The Spiral Model, introduced by Barry Boehm in 1986, combines iterative and sequential development with a risk-driven approach, making it ideal for large, complex, and high-risk projects..

**Agile SDLC:**

- SCRUM: A lightweight Agile framework that organizes work into iterative sprints with defined roles, events, and artifacts for continuous improvement.
- SAFe (Scaled Agile Framework): A structured Agile framework designed to scale Lean and Agile practices across large enterprises.

- Extreme Programming (XP): An Agile methodology that emphasizes continuous development, frequent releases, and close collaboration with customers for high-quality software.
- Kanban: A visual workflow management method that optimizes efficiency by limiting work in progress and enabling continuous delivery.

**In Requirement Gathering and Analysis, we:**

1. **Identify Stakeholders** – Engage with clients, users, and key stakeholders to understand their needs.
2. **Collect Requirements** – Use interviews, surveys, workshops, and document analysis to gather functional and non-functional requirements.
3. **Analyze and Validate** – Ensure requirements are clear, feasible, and aligned with business objectives.
4. **Prioritize Requirements** – Categorize them based on importance, feasibility, and impact.
5. **Document Requirements** – Create Software Requirement Specifications (SRS) for reference.
6. **Review and Finalize** – Validate with stakeholders to confirm completeness and accuracy.

This phase ensures that the project starts with a clear and well-defined scope, reducing future risks.

## Project Architecture & Development Process

## 1) Requirements

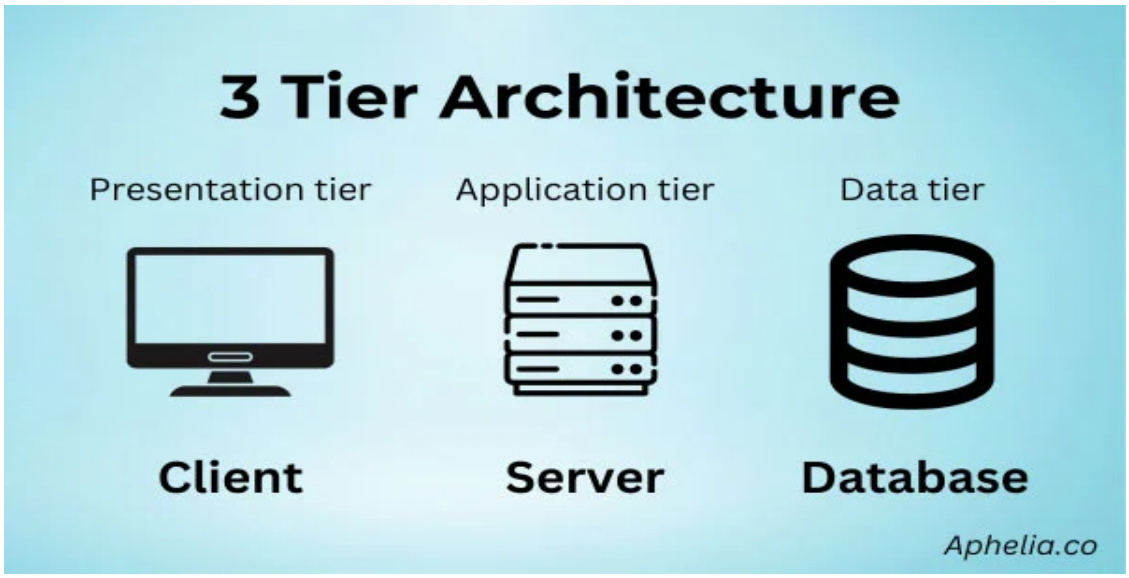
- **$R_1$ → GUI Development (Web-based)**
- **$R_2$ → Web Server Framework**
- **$R_3$ → Database**

## 2) Architecture

**The project follows a 3-tier architecture, consisting of:**

- **GUI (Frontend)** → Handles user interactions and displays results.
- **Web Server** → Manages requests, processes data, and communicates between frontend & backend.
- **Database (DB)** → Stores historical stock data, user preferences, and prediction results.

**Diagram Representation:**




## 3) Design

- Identify key functions and features.
- Structure the frontend, backend, and database accordingly.

## 4) Implementation

- Develop and integrate the frontend, backend, and database.
- Train & deploy ML/DL/QML/QNN models for stock market prediction.

## 5) Testing

- **UT** → Unit Testing (Verify individual components).
- **IT** → Integration Testing (Check interactions between components).
- **ST** → System Testing (Ensure overall system functionality).

## 6) Advantages & Disadvantages of the Waterfall Model

**Advantages:**

- Simple and easy to understand.
- Well-structured with clear phases.
- Suitable for projects with well-defined requirements.

**Disadvantages:**

- Difficult to accommodate changes once development starts.

- Late-stage testing may lead to costly modifications.
- Not ideal for dynamic and iterative software projects.

| SDLC Step | DevOps Tool |
| --- | --- |
| **Requirement Analysis (SRS)** | No specific DevOps tool, but tools like **JIRA**, **Confluence**, or **IBM DOORS** are used for documentation and tracking requirements. |
| **Design** | **PlantUML**, **Enterprise Architect**, or **Lucidchart** (for UML & OOAD diagrams) |
| **Coding** | **Pylint** (for Python static code analysis), **SonarQube** (for code quality and security) |
| **Unit Testing (UT)** | **PyUnit** (Python unit testing framework), **pytest** (advanced unit testing with fixtures), **unittest** (built-in Python testing framework) |
| **Integration Testing (IT)** | **Postman** (API testing), **Selenium** (UI automation), **PyTest with Selenium/ Webdriver** (for end-to-end integration) |
| **System Testing (ST)** | **Robot Framework** (automation testing), **Selenium** (UI testing), **Appium** (mobile testing) |
| **Deployment** | **Docker** (containerization), **Kubernetes** (orchestration), **Jenkins** (CI/CD), **Ansible** (configuration managen |