SDLC🡪**SDLC** stands for **Software Development Life Cycle**, and it refers to the structured process used by software developers to design, develop, test, and deploy software applications. The goal of SDLC is to produce high-quality software that meets or exceeds customer expectations, is delivered on time, and is within budget. It provides a systematic approach to software development and helps in managing and controlling the software project.

The SDLC process is typically divided into several stages, each with its own specific tasks and objectives. Here's an overview of the key phases in the SDLC:

**1. Requirement Gathering and Analysis**

* **Objective**: Understand and document the customer’s needs and the requirements of the software.
* **Activities**:
  + Collecting requirements from stakeholders, customers, and end-users.
  + Analyzing the requirements and preparing a **Requirements Specification Document** (SRS).
  + Identifying system requirements and constraints.
* **Outcome**: Clear and detailed understanding of what the software should do, which sets the foundation for design and development.

**2. System Design**

* **Objective**: Plan the architecture and design of the software system.
* **Activities**:
  + High-level design: Creating an architecture that outlines the software’s structure, components, and interactions.
  + Low-level design: Detailed design, including database schemas, data structures, and algorithms.
  + Preparing the **Design Specification Document**.
* **Outcome**: A blueprint for building the system, including how it will function and how different parts of the system will interact.

**3. Implementation (Coding/Development)**

* **Objective**: Actual development of the software according to the design specifications.
* **Activities**:
  + Writing the code based on the design document and requirements.
  + Developers follow coding standards and use appropriate technologies.
  + Version control is used to track and manage code changes.
* **Outcome**: The actual software product is created in the form of code.

**4. Testing**

* **Objective**: Identify and fix defects to ensure the software works as expected.
* **Activities**:
  + Conducting various types of testing, such as unit testing, integration testing, system testing, and acceptance testing.
  + Identifying bugs and issues in the software.
  + Ensuring that the software meets the requirements and functions as intended.
* **Outcome**: A verified and validated software product that is free of major bugs and issues.

**5. Deployment**

* **Objective**: Deploy the software to the production environment and make it available to end-users.
* **Activities**:
  + Releasing the software to the production environment or distributing it to users.
  + Installation and configuration on customer machines or cloud environments.
  + Ensuring smooth deployment without affecting existing systems (if applicable).
* **Outcome**: The software is live and available for use by end-users.

**6. Maintenance**

* **Objective**: Ongoing support, bug fixes, updates, and improvements to the software.
* **Activities**:
  + Monitoring the software for any issues or performance bottlenecks.
  + Releasing patches and updates as necessary.
  + Adding new features or enhancing existing functionality based on user feedback.
* **Outcome**: A continuously improved system that adapts to new requirements and remains functional and secure over time.

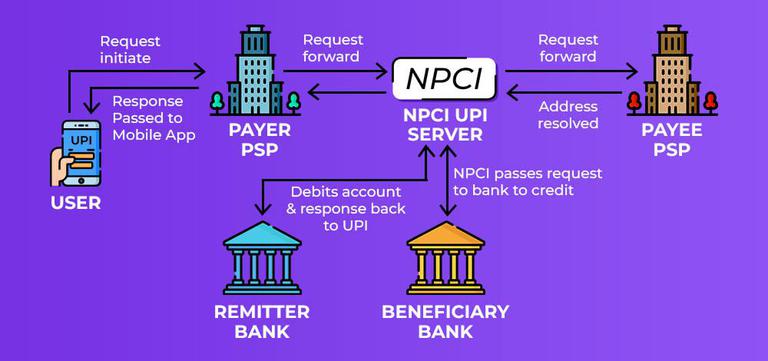
**SDLC Models:**

There are several SDLC models, and the choice of model depends on the project’s requirements, complexity, timeline, and resources. Some common SDLC models include:

1. **Waterfall Model**:
   * A linear and sequential approach where each phase is completed before moving to the next.
   * **Pros**: Simple to understand and implement; clear structure.
   * **Cons**: Inflexible to changes; no room for iteration.
2. **Agile Model**:
   * An iterative and incremental approach that focuses on flexibility, continuous feedback, and collaboration with stakeholders.
   * **Pros**
   * : High adaptability; quick delivery of working software.
   * **Cons**: Can be hard to manage for larger projects; requires frequent customer interaction.
3. **V-Model (Verification and Validation)**:
   * An extension of the Waterfall model, where each development phase is associated with a testing phase.
   * **Pros**: Clear and straightforward, with strong emphasis on validation.
   * **Cons**: Like Waterfall, it's rigid and doesn't accommodate changes easily.
4. **Spiral Model**:
   * Combines elements of both Waterfall and Agile, focusing on iterative development with regular risk analysis.
   * **Pros**: Suitable for large and complex projects with many risks.
   * **Cons**: Can be expensive and time-consuming due to frequent iteration.
5. **Iterative Model**:
   * Software is developed in iterations or small chunks, allowing portions of the system to be delivered early, tested, and improved in subsequent iterations.
   * **Pros**: Allows for early feedback and early delivery of parts of the system.
   * **Cons**: May require more time and resources due to multiple iterations.
6. **DevOps Model**:
   * Focuses on the continuous integration and continuous delivery (CI/CD) pipeline. It emphasizes collaboration between development and operations teams.
   * **Pros**: Fast deployment and quick delivery of features and bug fixes.
   * **Cons**: Requires a significant culture shift and proper tools for automation.

**Benefits of SDLC:**

* **Clear structure**: Provides a systematic approach for developing software, with defined phases and steps.
* **Improved project management**: Helps in managing time, cost, and resources effectively.
* **Quality control**: Ensures that software meets the desired quality and functional standards.
* **Risk management**: Helps in identifying risks early in the process and managing them effectively.
* **Customer satisfaction**: Through continuous involvement and feedback, the SDLC ensures that the final product meets customer expectations.



**Conclusion:**

The SDLC is an essential framework that guides software development from conception to deployment and maintenance. By following the SDLC phases and choosing the right model for the project, development teams can ensure a structured, efficient, and quality-driven approach to software creation.

Application

|

UI(html,css,angular,javascript,django)--dev

|

BE (application)—process(java,c,c++,dotnet)-dev—logs

|

DBMS (sql,mysql,mssql,mangodb)-dev (ddl,dml,dcl)+ support(dql)

Linux—server

SQL –DBA

ITRS—monitoring

Autosys – scheduler

Service now – ticketing (ITIL)

Jira (ITSM)

Data dog –

Kibana –

Splunk –

Git –

Jenkins – Udeploy

MQ – NDM -IHS –NAS –SAN—load balancers

L1(SOP, monitoring) 🡪 L2 (SOP, alerts, deployments, user related issues) 🡪DEV

Teams:

Poc—dev

Cab

Ps L1 and L2

Dev ops

Unix core SA

DBA SA

MQ

Tipco

WAS SA

Testing

DL

Cyberarc

APP teams