**EXPERIMENT 2**

**Aim : Software Development LifeCycle and Case Study.**

**Theory :**

The software development lifecycle (SDLC) is the cost-effective and time-efficient process that development teams use to design and build high-quality software. The goal of SDLC is to minimize project risks through forward planning so that software meets customer expectations during production and beyond.

**Importance of SDLC**

Software development can be challenging to manage due to changing requirements, technology upgrades, and cross-functional collaboration. The software development lifecycle (SDLC) methodology provides a systematic management framework with specific deliverables at every stage of the software development process. As a result, all stakeholders agree on software development goals and requirements upfront and also have a plan to achieve those goals.

Here are some benefits of SDLC:

● Increased visibility of the development process for all stakeholders involved ● Efficient estimation, planning, and scheduling

● Improved risk management and cost estimation

● Systematic software delivery and better customer satisfaction. **Working of SDLC Process**

**Step 1: Plan**

The planning phase typically includes tasks like cost-benefit analysis, scheduling, resource estimation, and allocation. The development team collects requirements from several stakeholders such as customers, internal and external experts, and managers to create a software requirement specification document. The document sets expectations and defines common goals that aid in project planning. The team estimates costs, creates a schedule, and has a detailed plan to achieve their goals.

**Step 2: Design**

In the design phase, software engineers analyze requirements and identify the best solutions to create the software. For example, they may consider integrating pre-existing modules, make technology choices, and identify development tools.

They will look at how to best integrate the new software into any existing IT infrastructure the organization may have.

**Step 3: Implement**

In the implementation phase, the development team codes the product. They analyze the requirements to identify smaller coding tasks they can do daily to achieve the final result.

**Step 4: Test**

The development team combines automation and manual testing to check the software for bugs. Quality analysis includes testing the software for errors and checking if it meets customer requirements. Because many teams immediately test the code they write, the testing phase often runs parallel to the development phase.

**Step 5: Deploy**

When teams develop software, they code and test on a different copy of the software than the one that the users have access to. The software that customers use is called production, while other copies are said to be in the build environment, or testing environment.

Having separate build and production environments ensures that customers can continue to use the software even while it is being changed or upgraded. The deployment phase includes several tasks to move the latest build copy to the production environment, such as packaging, environment configuration, and installation.

**Step 6: Maintain**

In the maintenance phase, among other tasks, the team fixes bugs, resolves customer issues, and manages software changes. In addition, the team monitors overall system performance, security, and user experience to identify new ways to improve the existing software.

**Case Study of Amazon**

**1. Requirement Analysis**

● Customer Focus: Amazon’s development starts with the customer in mind. Product managers and business analysts gather detailed requirements by interacting with customers and stakeholders.

● PR/FAQ Document: Amazon uses an internal process called the "Press Release/Frequently Asked Questions (PR/FAQ)" document. Teams write a mock press release announcing the new feature/product as if it were complete. This helps clarify the customer problem being solved and the proposed solution.

**2. Planning**

● Roadmap Creation: A detailed project roadmap is created, outlining the timeline, resources, and key milestones.

● Resource Allocation: Teams are formed, and resources are allocated based on expertise and project requirements.

● Risk Management: Potential risks are identified, and mitigation strategies are developed.

**3. Design**

● High-Level Design (HLD): The system architecture is designed, including the database schema, application architecture, and integration points. ● Low-Level Design (LLD): Detailed designs for each component or module are created, specifying class diagrams, sequence diagrams, and detailed workflows.

● Design Reviews: Regular design reviews are conducted to ensure the architecture meets Amazon’s high standards for scalability and reliability.

**4. Implementation (Coding)**

● Agile Development: Amazon uses Agile methodologies, specifically Scrum or Kanban. The development process is broken into sprints, each lasting 2-3 weeks.

● Coding Standards: Developers follow strict coding standards and best practices to ensure code quality and maintainability.

● Code Reviews: Peer reviews and automated code analysis tools are used to catch bugs early and ensure adherence to coding standards.

**5. Testing**

● Automated Testing: Extensive use of automated testing, including unit tests, integration tests, and end-to-end tests.

● Continuous Integration (CI): Developers integrate code into a shared repository frequently. Each integration is verified by an automated build and test to detect issues early.

● Beta Testing: Features are often rolled out to a small group of users (internal or external) for beta testing to gather feedback and identify any issues in a real-world environment.

**6. Deployment**

● Staging Environment: Code is first deployed to a staging environment that mimics the production environment for final testing.

● Blue-Green Deployment: Amazon often uses blue-green deployments to minimize downtime and reduce the risk of deployment failures. This involves maintaining two identical production environments and switching traffic between them during deployment.

● Canary Releases: New features are gradually rolled out to a small subset of users to monitor for issues before a full-scale release.

**7. Maintenance**

● Monitoring: Continuous monitoring of the system using tools like Amazon CloudWatch to track performance, detect anomalies, and ensure uptime. ● Incident Management: A robust incident management process is in place to quickly address any issues that arise. This includes automated alerting and on-call rotations.

● Regular Updates: Regular software updates and patches are released to improve functionality, security, and performance.

**8. Feedback and Improvement**

● Customer Feedback: Constantly gather feedback from customers through surveys, support interactions, and direct feedback mechanisms. ● Data-Driven Decisions: Use data analytics and A/B testing to make informed decisions about product improvements and new features.

● Iterative Development: Continuous iteration and improvement of the software based on feedback and changing market conditions.

**Methodologies and Tools**

● Project Management: JIRA, Trello

● Version Control: Git, AWS CodeCommit

● CI/CD: Jenkins, AWS CodePipeline

● Testing: Selenium, JUnit, TestNG

● Monitoring: Amazon CloudWatch, New Relic

**Security and Compliance**

● Security Practices: Security is integrated into every stage of the SDLC. Regular security audits, code reviews, and the use of tools like Amazon Inspector for vulnerability scanning.

● Compliance: Compliance with regulations and standards such as GDPR, HIPAA, and PCI-DSS.

**Conclusion:**

The SDLC process is vital for developing effective, efficient, and user-centric software. By following a structured SDLC, organizations can deliver high-quality products, improve customer satisfaction, and maintain a competitive edge in the market.