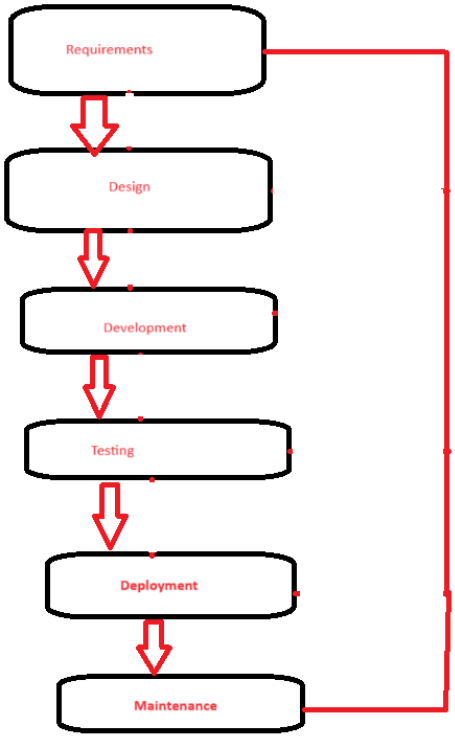
Name: Subha Ranjan Sahoo

Batch: CPPE_Java Full Stack

Assignment 1

Q. SDLC Overview - Create a one-page infographic that outlines the SDLC phases (Requirements, Design, Implementation, Testing, Deployment), highlighting the importance of each phase and how they interconnect.

Answer :



Software Development Life Cycle (SDLC).

1. Requirements

Importance: Gathering and understanding client needs.

Key Activities: Requirement Getting, analysis, and documentation.

2. Design

Importance: Translating requirements into a blueprint for development.

Key Activities: System architecture, database design, user interface design.

3. Development

Importance: Transforming design into actual code.

Key Activities: Writing code, unit testing.

4. Testing

Importance: Ensuring the software meets quality standards and functions correctly.

Key Activities: Unit testing, integration testing, system testing, acceptance testing.

5. Deployment

Importance: Releasing the software to users or clients.

Key Activities: Installation, configuration, user training, maintenance planning**.**

6. Maintenance

Importance: Sustaining and enhancing the software after deployment.

Key Activities: Bug fixing, performance optimization, feature enhancements, updates.

Q. Develop a case study analyzing the implementation of SDLC phases in a real-world engineering project. Evaluate how Requirement Gathering, Design, Implementation, Testing, Deployment, and Maintenance contribute to project outcomes.

Answer:

Case Study: Building a Mobile Banking App

1. Requirement Gathering:

A bank wants a mobile app for customers. They ask people what they want in the app, like checking balances or paying bills.

2. Design:

People make drawings of how the app will look and work. They make sure it's easy to use and keeps people's banking info safe.

3. Implementation:

Programmers start writing the code for the app. They make sure it works on both iPhones and Android phones. They also make it connect well with the bank's computers.

4. Testing:

Testers try out the app to make sure it works right. They check if people can use it easily and if it keeps their information safe.

5. Deployment:

When the app is good to go, it gets put in the app stores for people to download. The bank also keeps an eye on how the app is doing and fixes any problems.

6. Maintenance:

After the app is out, the team keeps updating it. They fix any bugs and add new features. They also help people if they have questions or problems with the app.

Evaluation:

Each step in making the app is important. Talking to people first helps make sure the app has what they need. Drawing how it will look and work makes it easy for people to use. Writing the code brings the app to life. Testing makes sure it works right. Putting it in the app stores lets people use it. And keeping it updated and helping people after it's out makes sure they keep liking and using it.

Q. Research and compare SDLC models suitable for engineering projects. Present findings on Waterfall, Agile, Spiral, and V-Model approaches, emphasizing their advantages, disadvantages, and applicability in different engineering contexts.

Answer:

Waterfall Model:

Advantages:

Easy to Understand: Steps are clear and easy to follow.

Good for Stable Projects: Works well when requirements are fixed.

Milestones: Clear checkpoints help track progress.

Disadvantages:

No Flexibility: Hard to make changes once a phase is done.

Late Feedback: Issues might not be found until late in the process.

High Risk: If requirements are misunderstood, it can lead to problems later.

Applicability: Best for projects with clear requirements, like building infrastructure.

Agile Model:

Advantages:

Flexible: Can adapt to changes easily.

Frequent Deliveries: Gives something usable to stakeholders regularly.

Team Collaboration: Teams work closely together for better results.

Disadvantages:

Complex for Large Projects: Might be tricky to use on big projects.

Needs Active Stakeholder Involvement: Requires a lot of input from stakeholders.

Minimal Documentation: Less documentation might make it harder to understand the project later.

Applicability: Great for projects with changing requirements or where early delivery is crucial.

Spiral Model:

Advantages:

Risk Management: Helps identify and address risks early.

Flexible: Combines elements of Waterfall and Agile for better adaptation.

Stakeholder Involvement: Keeps stakeholders involved throughout the process.

Disadvantages:

Complex: More complicated than some other models.

Resource-Heavy: Takes more time and money due to iterative cycles**.**

Not for Simple Projects: Might be too much for straightforward projects.

Applicability: Good for large and complex projects where risk management is vital.

V-Model:

Advantages:

Testing Emphasis: Testing is integrated from the start, catching issues early.

Clear Requirements: Makes sure requirements are understood before moving forward.

Traceability: Keeps track of requirements and tests, ensuring thoroughness.

Disadvantages:

Rigid Structure: Like Waterfall, can't easily adapt to changes.

Late Feedback: Testing happens late, which can delay finding problems.

Complex for Big Projects: Might be tough to scale up for large projects.

Applicability: Suitable for projects with strict requirements and high-quality standards, like healthcare or finance projects.