5-2-26
Thursday

# Software Engineering Interview notes by Keshav Sharma

Q3, Q4, Q5, Q6, Q7, Q9, Q11, Q12, Q14, Q15, Q19, Q28

## (Q3) SDLC & its phases

→ Software development life cycle.
→ Phases

→ Planning & Analysis
→ understand project goals, cost & feasibility.

→ Requirment definition
→ write SRS = Software requirment specification.
→ Functional + Non-functional requirments.

→ Design
→ Create system architecture & blueprint

→ Development
→ Coding of th

→ Testing
→ Find bugs
properly

→ Deployme
→ Release
time.

## (Q4) Differen

→ Waterf
→ Line
→ Bes

→ V-Mo
→ Test
→ Ea
pl

→ Incre
→ S
→ E

→ Development
→ Coding of the software

→ Testing
→ Find bugs, ensure system works properly.

→ Deployment & Maintenance
→ Release software + fix/update over time.

---

(Q4) Different SDLC Models.

→ Waterfall Model
→ Linear, step by step process
→ Best when requirements are fixed.

→ V-model (Verification & Validation)
→ Testing done along with development
→ Each phase has matching testing phase.

→ Incremental model
→ Software built in small parts.
→ Early working version delivered.

→ RAD Model (Rapid application developme

→ focus on fast development + prototype
→ Contineous user feedback

→ Iterative Model
→ development in repeated cycles.
→ improve software after each version

→ Spiral Model
→ Combines iteration + risk analysis
→ best for large & high risk projects

→ Prototype model
→ Early prototype shown to users.
→ Feedback to refine system.

→ Agile Model
→ Development in small sprints
→ Highly flexibee & customer feedback driven.

Q5) Wa
→ A l
mod
bef
Phase
1) Req
2) Desi
3) imp
4) Int
5) De
6) Mo
Feat
→ N
→ I
→
→
Us
→)

## Q5) Waterfall method & use cases

→ A linear and sequential SDLC model where each phase must finish before the next starts.

**Phases**

1) Requirment analysis :- collect & finalize requirments.

2) Design :- system architecture & design.

3) implementation + Unit testing → coding modules.

4) Integration & testing (system) :- combines module & test

5) Deployment :- Release Software

6) Maintainance :- bug fix + update

**Features**

→ No going back in previous phase.
→ Documentation heavy.
→ Simple & Easy to manage.
→ changes are difficult after start.

**Use cases**

→ Government projects → Banking System
→ Healthcare System → Large enterprise

## (Q6) Black box testing

→ internal code hidden
( Tester check input → output)

→ No coding knowledge

→ Functionality & user requirement focus

→ done by tester/End user.

→ Tested
- → user interface
- → input & output
- → System behaviour
- → functional requirements

→ Eg! - login page → enter username & password
↓
check if login works

## (Q7) White box testing

→ Internal code & logic are known for testing.

→ tester knows source code

→ Focus: Code logic, paths, conditions

→ done by d

→ Tested
- → code struct
- → loops & con
- → internal l

→ Eg! - All i

## (Q9) Debugg

→ Process
in soft

→ After

→ Perf

→ impl

→ Bug fo

## (Q11) Use-

→ cli
intera

→ Comp

(1) Sys

→ done by developers.

→ Tested

→ Code structure

→ loops & condition

→ internal logic & paths.

→ Eg: - All if-else work correctly.

(Q9) Debugging

→ Process of finding & fixing bugs/error in software.

→ After testing find bugs.

→ Performed mainly by developers.

→ improves performance & correctness.

→ Bug found → Locate cause → fix code
↓
retest

(Q11) Use-case diagram (UML)

→ diagram showing how user interact with a system.

→ Components

(1) System: - Application being built.

(2) Actor: User or external system

(3) Use-case: Action by user.

→ Purpose
- → understand system functionality.
- → Show user requirment clearly.

→ Eg:- Actor: User
Usecase: Login, Register, make payment

## (Q12) Verification Vs Validation

| Verification | Validation |
|---|---|
| → check S/w built Correctly as per SRS. | → Check wether Software meets user needs. |
| → Static: No code exeution | → Dynamic: involve running code |
| → Done using reviews, inspection | → Done using testing |
| → "Are we buildig the product right? " | → "Are we building the right product? " |

(Q14) Cohesi

→ Cohesion

→ Mea
the task

→ High

→ Better

→ Eg:-

→ Couple

→ Mea

→ Low

→ Less

→ Eg:-

(Q15) A

→ A
Softwar
iteratio

## (Q14) Cohesion and Coupling

→ Cohesion

→ Measure of how strongly related the tasks are inside module are

→ High cohesion = module does one specific task

→ Better design → preffered.

→ Eg:- Login module only handles login.

→ Coupling

→ Measure of dependency b/w modules.

→ Low coupling = modules are indepen-dent.

→ Less dependency → better system.

→ Eg:- Login module works without affecting payment module.

---

## (Q15) Agile Software development model?

→ A flexible SDLC model where Software is developed in small iteration with Continuous feedback.

→ Features

→ Works in Short cycle (sprints)
→ Contireous customer feedback.
→ Allows changes anytime
→ Frequent working software delivery.

Team work

→ Developer + Testers + customer work together.

→ Advantages

→ Fast delivery.
→ High customer satisfaction.
→ Easily adapts to changing requirments.

## (Q19) Regression Testing

→ Testing done to ensure new code changes does not break existing features

→ Re-run old test cases.
→ Done after bug fix or update.
→ Ensures existing functionalities

still works.

→ Eg:- After adding payment feature
→ check if login still works.

## Q28) What is SRS?

→ Software requirment specification.

→ A document that describes all software requirments before development starts.

→ Contains functional + non-functional requirments.

→ Act as agreement b/w clients & developers

→ Used as base for design & development.

→ Purpose

→ Avoid misunderstanding.

→ Provide clear project scope.

END