Q-1 What is Agile Alliance? Explain each of the manifesto statements with suitable examples.

→ • A group of industry professionals met in early 2001, inspired by the observation that software teams in many organizations were stuck in a maze of ever-increasing process, to describe the values and principles that would allow software teams to operate fast and respond to change. They called themselves the Agile Alliance.

• The Manifesto of the Agile ~~Software Development~~ Alliance
Manifesto for Agile Software Development.
We are revealing better ways of developing software by doing it and helping others do it. Through this work we have come to value.

- Individuals and interactions over processes & tools.
- Working software over comphrehensive documentation
- Customer collaboration over contract negotiation
- Responding to change over following a plan.

- Individuals and interactions over processes & tools.
- people are most vital ingredient of success. A decent ~~team~~ process will not save the project from failure if the team doesn't have solid players, but a corrupt process can make even the solidest of players unproductive. Even a group of strong players can fail severely if they don't work as a team.

VISION

- Working software over comprehensive Documentation
  - Software without documentation is a tragedy. code is not the perfect medium for communicating the rationale and structure of a system. Somewhat, the team needs to produce human-readable documents that define the system and the rationale for their design decisions.
  - The two documents that are greatest at transferring information to new team members are the code and team. The code does not falsehood about what it does. It may be hard to extract rationale and intent from the code, but the code is the only unmistakable source of information.

- Customer collaboration over contract negotiation
  - Successful projects contain customer feedback on a consistent and frequent basis. Rather than contingent on a contract or a statement of work, the customer of the software works throughly with the development team, providing regular feedback on their efforts.

- Responding to change over following a plan
  - The ability to respond to change often determines the success or failure of a software project. When we build ~~parts~~ plans, we need to make sure that they are flexible and ready to adapt to changes in the business and technology.

Q-2 What is the user story and story point? Explain the concept of estimation in XP with suitable numeric examples.

→. In Extreme Programming (XP), a user story is a brief, informal description of a feature or functionality from the perspective of an end-user. It typically follows a template like "As a [user type], I want [an action] so that [benefit / value]."

- User stories are used to capture requirements in a more human-readable and user-focused manner.

- Story points are a unit of measure used in XP and Agile methodologies to estimate the effort required to complete a user story. They are a relative measure of complexity, uncertainty, and effort needed to implement a story, rather than a precise measure of time.

- Story points are assigned to each user story during the estimation process.

- In XP, estimation is often done using a technique called Planning Poker. Team members assign story points to user stories based on their collective understanding of the work involved.

VISION

- Each team member selects a card that they think best represents the effort required for the story. Then, they discuss their estimates and re-estimate if there's a significant difference in opinions until a consensus is reached.

- Example :-

- Imagine a software development team is estimating user stories for an e-commerce website. They have two user stories:

1 User story A : " As a customer, I want to be able to add items to my shopping cart".

2 User story B : " As a customer, I want to be able to checkout and complete my purchase".

- During the estimation meeting, team members use Planning Poker cards to assign story points:

- Team member 1 : Story A - 3 story points, story B - 5 story points.
- Team member 2 : Story A - 2 story points, Story B - 8 story points
- Team member 3 : Story A - 3 story points, story B - 5 story points

vision

- The team reaches a consensus:

- User story A is estimated at 3 story points, indicating it's relatively straight forward and requires less effort.

- User story B is estimated at 5 story points, suggesting it's more complex and might take more effort to complete.

Q-3 What is the single Responsibility Principle (SRP)? consider the scenario of the university system with roles like Students, Faculties and Exam co-ordinator. Give suitable examples to demonstrate the violation of OCP. Also provide solution to avoid violation of OCP. You can use UML diagram or OOP (Java code) for demonstration.

→ • The single Responsibility principle (SRP) is one of the SOLID principles of object-oriented design, and it states that a class should have only one reason to change. In other words, a class should have a single responsibility or encapsulate a single piece of functionality.

- Let's consider a university system scenario with roles like Students, Faculties and an Exam Co-ordinator. We'll look at examples of potential SRP violations and provide solutions to avoid them

VISION

using Java code.

* **SRP violation example:**

```
public class Student {
    private String name;
    private int id;

  public Student (String name, int id) {
      this.name = name;
      this.id = id;
  }
public void enrollInCourse (String course) {
  }
public void scheduleExam (String course, Date date)
  {
  }
```

* **Solution to Avoid SRP Violation:-**

```
public class Student {
    private String name;
    private int id;
public Student (String name, int id) {
    this.name = name;
    this.id = id;
  }
```

```
public void enrollIncourse(String course)
{
}

public class ExamCoordinator {
    public void scheduleExam(String course, Date date)
  {
  }
}
```

Q-4 What is the open Close Principle (OCP)? Consider the scenario of the university system with roles like Students, Faculties and Exam co-ordinator. Give suitable examples to demonstrate the violation of OCP. Also provide solution to avoid violation of OCP. You can use UML diagram or OOP (Java code) for demonstration.

→ • The open-Closed principle (OCP) is one of the SOLID principles of object oriented design. It states that software entities should be open for extension but closed for modification. In other words, you should be able to extend the behaviour of a module without changing its source code.

• Let's consider a scenario in a university system with roles like Students, faculties, and an Exam Co-ordinator and explore an example of a violation of the OCP principle, followed by a solution to avoid the violation.

* Violation of OCP example:

```
public class university System {
    private List <student> students;
    private List <faculty> faculties;

public void addstudent (student student) {
        students.add(student);
    }

public void addfaculty ( Faculty faculty) {
        faculties.add (faculty);
    }
}
```

* Solution of avoid OCP Violation:

```
public interface Role {
}

public class student implements Role {
}

public class Faculty implements Role {
}

public class Exam coordinator implements Role {
}

public class university system {
    private List<Role> roles;

    public void addRole (Role role) {
        roles.add (role);
    }
}
```

VISION

Q-5 Explain Agile Testing? What are the principles of Agile Testing?

→ • Agile Testing is a testing approach and mindset that aligns with the principles of agile software development. It emphasizes collaboration, flexibility and customer-centricity to ensure that software is continuously tested and delivered with high quality.

• Agile Testing is an integral part of Agile methodologies such as Scrum, Kanban and XP. Here are key principles of Agile Testing:

1 Testing Throughout the Software Lifecycle:
Agile Testing is not a phase that occurs after development; it starts from the beginning and continues throughout the software development cycle.

2 Customer Collaboration:
Agile Testing prioritizes customer feedback and collaboration. Testers work closely with customers, product owners and business analysts to understand user needs and expectations, ensuring that the delivered software meets those requirements.

3 continuous feedback.

VISION

Agile Testing emphasizes the rapid feedback loop. Testers provide feedback on the software's quality after each iteration, allowing for quick adjustments and improvements.

4 Test Automation:
Automated testing is a cornerstone of Agile Testing. Automated tests are used to validate & verify software changes quickly, especially in continuous integration and continuous delivery (CI/CD) pipelines.

5 Cross-functional Teams:
Agile teams are cross-functional, meaning they consist of developers, testers, designers and other roles necessary to deliver a complete product.

6. Test-Driven Development (TDD):
In TDD, tests are written before writing code. This approach encourages developers to think about the desired behaviour and requirements before implementation.

7 Exploratory Testing:
In addition to scripted tests, Agile Testing encourages exploratory testing. Testers use their domain knowledge and creativity to find defects and issues that scripted tests might not cover.

VISION