Task 1:

What do you mean by GOOD Code and BAD CODE?

GOOD Code:

1. Easy to read and understand

2. Well-documented

3. Follows naming conventions

4. Properly organized and structured

5. Easy to maintain and modify

6. Efficient and performs well

7. Has proper error handling

8. Follows coding principles (SOLID, DRY, etc.)

BAD Code:

1. Confusing and hard to read

2. Poor or no documentation

3. Random/unclear naming

4. Messy and unorganized

5. Hard to maintain or change

6. Inefficient and slow

7. No error handling

8. Violates coding principles

Task 2:

What do you understand by databinding?

Data Binding is:

1. Connection between UI (user interface) and data source

2. Two types:

- One-way: Data flows in one direction

- Two-way: Data flows in both directions

3. Automatically updates data when UI changes and vice versa

4. Helps keep UI and data in sync

5. Common in modern frameworks (Angular, React, etc.)

Task 3:

What do you know about continuous development?

Continuous Development means:

1. Constantly improving and updating software

2. Regular code updates and deployments

3. Automated testing and integration

4. Quick feedback and fixes

5. Key parts:

- Continuous Integration (CI)

- Continuous Delivery (CD)

- Continuous Deployment

6. Helps deliver better software faster

Task 4:

What are the conditions for polymorphism?

For polymorphism to work, you need:

1. Inheritance: Parent and child classes

2. Method Overriding: Same method name in parent and child

3. Parent class reference: Using parent type to refer to child object

4. Runtime binding: Method call decided at runtime

Task 05:

What is, why is it used , where is it used..

TDD and BDD approach..

TDD (Test-Driven Development)

WHAT IS IT?

- Write tests before writing actual code

- Follow Red-Green-Refactor cycle

WHY USE IT?

1. Better code quality

2. Fewer bugs

3. Clear requirements

4. Easy maintenance

5. Built-in documentation

WHERE IS IT USED?

1. Unit testing

2. Method level testing

3. Small, focused functionality

BDD (Behavior-Driven Development)

WHAT IS IT?

- Focus on system behavior

- Written in plain English

- Given-When-Then format

WHY USE IT?

1. Better communication

2. Clear requirements

3. Non-technical stakeholders can understand

4. Business value focused

WHERE IS IT USED?

1. Feature testing

2. User story validation

3. End-to-end testing

Key Differences:

TDD:

- Code focused

- Technical approach

- Unit level testing

- Developer oriented

BDD:

- Behavior focused

- Business approach

- Feature level testing

- Stakeholder oriented

Common Tools:

TDD Tools:

- JUnit

- TestNG

- NUnit

BDD Tools:

- Cucumber

- SpecFlow

- JBehave

Task 06:

List down the Manual and automated testing tools

MANUAL TESTING TOOLS:

1. Test Case Management

- TestRail

- Zephyr

- qTest

- TestLink

2. Bug Tracking

- JIRA

- Bugzilla

- Mantis

- Trello

3. Test Planning

- Microsoft Excel

- Microsoft Word

- Google Docs

- Confluence

AUTOMATED TESTING TOOLS:

1. Web Testing

- Selenium

- Cypress

- Playwright

- TestCafe

2. API Testing

- Postman

- SoapUI

- REST Assured

- JMeter

3. Mobile Testing

- Appium

- XCUITest

- Espresso

- Detox

4. Performance Testing

- JMeter

- LoadRunner

- Gatling

- K6

5. Unit Testing

- JUnit

- TestNG

- NUnit

- PyTest

6. BDD/Functional

- Cucumber

- SpecFlow

- Serenity

- Robot Framework

7. CI/CD Tools

- Jenkins

- GitLab CI

- CircleCI

- TeamCity

8. Cross-browser Testing

- BrowserStack

- Sauce Labs

- LambdaTest

- CrossBrowserTesting