Task001:  
1.what do you understand by good coding and bad coding?  
Good Coding:  
Readable & Clear → Easy to understand, even for someone new to the code.  
Maintainable → Can be updated or fixed without breaking other parts.  
Efficient → Uses logic and resources   
Well-structured → Follows proper design patterns, indentation, and naming conventions.  
Reusable & Modular → Code is broken into small, reusable functions or classes.  
  
Bad Coding:  
Messy & Hard to Read → Poor indentation, confusing variable names.  
Spaghetti Code → Everything tangled together without modularity.  
Inefficient → repeated loops, unnecessary computations.  
Hard to Maintain → Any small change breaks the system.  
No Comments → Either no guidance or unnecessary explanations.  
  
Task002:  
What do you understand by databinding?  
Data Binding is a technique that connects the User Interface and the data source so that changes in one are automatically reflected in the other without writing extra code to synchronize them.  
It creates a direct link between the data (model) and the UI components (view).  
  
Task 3:  
What do you know about continuous development?  
is a software practice where code is built, tested, and released frequently and automatically, instead of waiting for big releases.  
It’s like delivering small updates to an app regularly daily or weekly instead of giving users one big update after months.  
  
Task 4:  
What are the conditions for polymorphism?  
Overloading → method name should be same , different parameters with type and value also.  
Overriding → method name has to same with same parameters but different class with inheritance.  
  
Task 05:  
What is, why is it used , where is it used..   
TDD and BDD approach..  
  
**TDD- Test Driven Development**  
A software development process where you write tests first, then the code that makes those tests pass.  
Write unit tests first (using JUnit), then write code to pass the tests.  
Why it’s used  
Ensures fewer bugs from the start.  
Gives confidence when changing or refactoring code.  
Improves code quality and design

Where it’s used  
Backend logic development.  
API and service-level programming.  
When reliability and correctness are critical (e.g., banking, healthcare, ecommerce).  
  
  
**BDD- Behavior Driven Development**  
An extension of TDD that focuses on system behavior in plain language (like English) and Uses Given-When-Then format to describe how software should behave.  
Describe system behavior in **natural language** (using Gherkin), and automate those behaviors using tools like **Cucumber**.

Why it’s used  
Improves collaboration between developers, testers, and business teams.  
Makes requirements clear before coding starts.  
Bridges the gap between technical and non-technical stakeholders.

Where it’s used  
Frontend UI testing.  
End-to-end system testing.  
Projects where communication with non-technical stakeholders is important.  
  
Task006:  
list of the Manual and Automation Testing:  
Automation :  
Selenium – Web automation  
Cypress – Modern web app testing  
Playwright – Browser automation  
TestComplete – UI test automation  
Ranorex – Desktop, web, and mobile automation  
Katalon Studio – Web, API, and mobile automation  
  
Manual:  
Jira – Bug tracking and project management  
TestRail – Test case management  
Quality Center (HP ALM) – Test planning and execution  
Bugzilla – Defect tracking  
Mantis – Bug/issue tracking  
qTest – Test management  
PractiTest – End-to-end test management  
Zephyr – Test management plugin for Jira