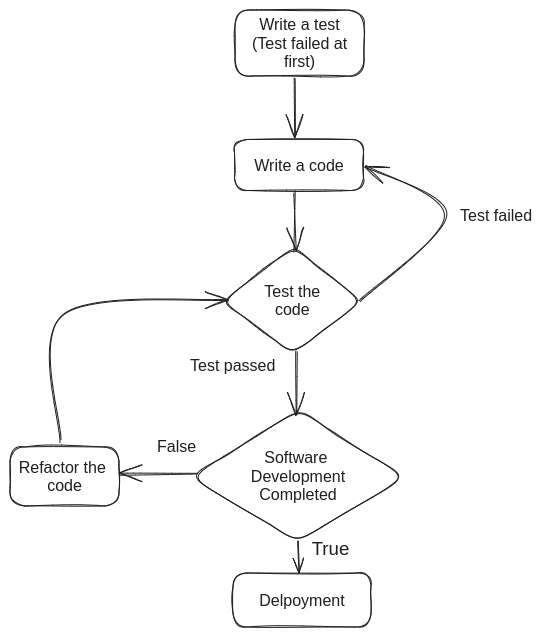
**Assignment 1: Create an infographic illustrating the Test-Driven Development (TDD) process. Highlight steps like writing tests before code, benefits such as bug reduction, and how it fosters software reliability.**

Test-Driven Development is a software development approach where tests are written before the actual code.

**Principles of TDD:**

1. **Write a test:** Before writing the actual code, a developer writes a test for the new feature or functionality. This test failed initially since the functionality has not been implemented yet.
2. **Write the code:** Now the developer writes the code for the functionality,
3. **Run the test again:** The test is run again to ensure that it passes with the newly written code. If it passes, the code is considered correct for the given test case.
4. **Refactor the code:** The developer refactors the code to improve its structure and efficiency without changing its behaviour. The test ensures that the functionality remains intact during refactoring.
5. **Repeat:** This cycle is repeated for each new feature or functionality until the software development is complete.



**Some advantages of TDD:**

* **Bug reduction:** Since tests are written for small units of code, it is easier to locate the source of an issue.
* **Comprehensive test coverage:** Ensures through testing of all code paths.
* **Software reliability:** As the test is written for small units of code, so the software becomes reliable.
* **Customer satisfaction:** Delivers higher quality, reliable software.
* **Improved Code quality:** Encourages cleaner, more maintainable code.

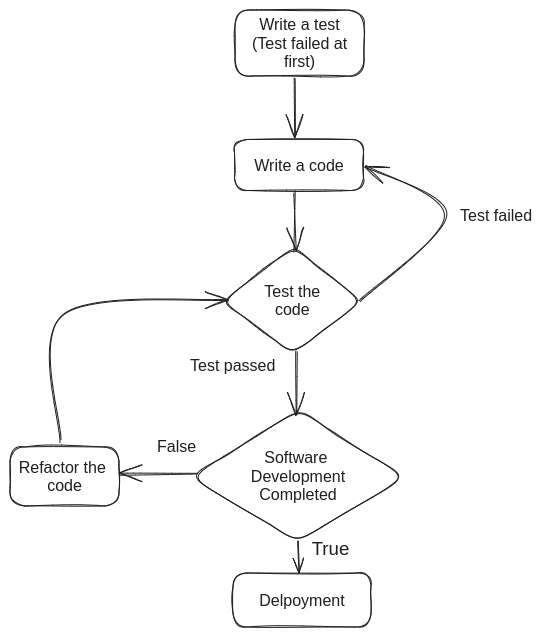
**Assignment 2: Produce a comparative infographic of TDD, BDD, and FDD methodologies. Illustrate their unique approaches, benefits, and suitability for different software development contexts. Use visuals to enhance understanding.**

1. **Test-Driven Development(TDD):**

Test-Driven Development is a software development approach where tests are written before the actual code.

**Principles of TDD:**

* **Write a test:** Before writing the actual code, a developer writes a test for the new feature or functionality. This test failed initially since the functionality has not been implemented yet.
* **Write the code:** Now the developer writes the code for the functionality.
* **Run the test again:** The test is run again to ensure that it passes with the newly written code. If it passes, the code is considered correct for the given test case.
* **Refactor the code:** The developer refactors the code to improve its structure and efficiency without changing its behaviour. The test ensures that the functionality remains intact during refactoring.
* **Repeat:** This cycle is repeated for each new feature or functionality until the software development is complete.



**Benefits of Test-Driven Development:**

* **Improved code quality:** TDD often results in cleaner, more modular code because it encourages developers to think through the design and edge cases before writing code.
* **Regression Testing:** The automated tests created during TDD serve as a safety net, catching regressions quickly when changes are made.
* **Design Documentation:** Tests act as executable specifications, documenting how the code should behave.
* **Faster Debugging:** Since issues are caught early in the development cycle, debugging is usually faster and easier.
* **Increased Confidence:** TDD encourages confidence in code changes, knowing that all tests pass, the code is likely to function correctly.

TDD is suitable in environments where there is a focus on iterative development, maintaining high quality code, adapting to changing requirements, fostering collaboration, and ensuring software reliability. It provides a systematic approach to building software that incorporates testing as an integral part of the development process, leading to more robust and maintainable applications.

1. **Behavior-Driven Development(TDD):**

It is a software development approach that aims to enhance collaboration among the team. BDD encourages writing specifications or requirements in a natural language that is understandable by all parties involved, thereby bridging the gap between technical and non-technical team members.

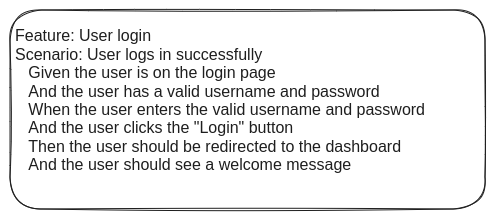
Key aspects of BDD include:

1. **Behavior:** Focuses on defining the behavior of the system from the user's perspective. This involves how the system should behave in various scenarios.
2. **Natural Language:** Specifications are often written in a structured natural language format (such as Given-When-Then) that can be understood by both technical and non-technical team members.
3. **Automation:** BDD scenarios are typically automated using specialized tools (e.g., Cucumber, SpecFlow) that execute the described behavior against the software and verify whether the actual behavior matches the expected behavior.
4. **Collaboration:** Encourages collaborations between developers, testers, domain experts right from the early stages of development, ensuring that everyone has a shared understanding of what needs to be built.
5. **Continuous Testing:** BDD scenarios often form the basis for automated tests that are continuously executed throughout the development process, helping to catch defects early and ensure that the software meets the specified requirements.

Feature: User login

The gherkin language can be used in BDD:

**Example-**

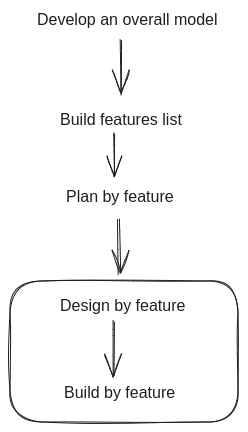
****

BDD is suitable in any software development context where there is need for clear, collaborative, and user-focused requirements validation and testing to deliver high-quality software efficiently.

1. **Feature-Driven Development(FDD):**

FDD stands for Feature-Driven Development. It is an iterative and incremental software development methodology that focuses on building features incrementally in short iterations.

FDD is known for its pragmatic approach and scalability, making it particularly suitable for large-scale projects where multiple teams collaborate on different features concurrently. It emphasizes clear communication, well-defined processes, and a focus on delivering working software incrementally and consistently.



Here are some benefits of FDD:

* **Clear Focus:** Emphasizes clear feature scope and objectives.
* **Incremental Delivery:** Releases working software iteratively.
* **Predictability:** Enhances control over timeline and budget.
* **Quality Emphasis:** Rigorous feature level design and testing.
* **Collaboration:** Promotes effective team and stakeholder communication
* **Scalability:** Adaptable to projects of varying size and complexity.
* **Adaptability:** Responsive to changing requirements and conditions.
* **Risk Reduction:** Mitigates risks by focusing on manageable feature sets.

FDD is suitable for projects where clarity, predictability, quality, collaboration, and adaptability are essential for successful delivery.

**Assignment 3: Write the Agile model principles.**

Here are key principles of Agile model:

1. **Customer Satisfaction through Early and Continuous Delivery of**

**Valuable Software:** Agile methods prioritize delivering working software

Frequently, ensuring that customers receive tangible value early and

throughout the development process.

1. **Welcome Changing Requirements, Even Late in Development:**

Agile processes are designed to accommodate changes in

requirements, recognizing that customer needs and market conditions

may evolve. Changes are seen as opportunities rather than disruptions.

1. **Deliver Working Software Frequently, in Short Timescales:** Agile

promotes short development iterations, where functional software is

Delivered regularly. This allows for continuous feedback and ensures

that the product remains aligned with stakeholders expectations.

1. **Business People and Developers Must Work Together Daily**

**throughout the Project:** Agile methodologies emphasize collaboration

between business stakeholders(Product owners, customers) and

development team.

1. **Build Project around Motivated Individuals. Give Them the**

**Environment and Support They Need, and Trust them to get the**

**Job Done:** Agile teams are empowered and self-organizing, with a

focus on creating a supportive environment where team members are

motivated to deliver high quality work.

1. **Face-to-Face Conversation:** Face-to-Face interactions promote clarity,

Empathy, and rapid problem-solving, reducing misunderstandings and

delays.

1. **Working Software is the Primary Measure of Progress:** Agile

methodologies prioritize delivering functional software increments.

progress is measured by the ability to demonstrate working features

and achieve tangible outcomes rather than completing tasks or

adhering to project plans.

1. **Agile Process Promote Sustainable Development, The Sponsor,**

**Developers, and Users Should Be Able to Maintain a Constant**

**Pace Indefinitely:** Agile fosters a sustainable work pace by promoting

managable workloads, continuous improvements practices, and a

healthy work-life balance for team members. This ensures long-term

productivity and minimizes burnout.

1. **Continuous Attention to Technical Excellence and Good Design**

**Enhances Agility:** Technical excellence enables teams to respond

quickly to change and deliver high-quality software.

1. **Simplicity is Essential:** By focusing on essential features and

minimizing complexity, teams can reduce waste, improve productivity,

and deliver value more efficiently.

1. **The Best Architectures, Requirements, and Design Emerge from**

**Self-Organizing teams:** Agile methodologies empower self-organizing

teams to make decisions collaboratively, fostering creativity, ownership,

and innovation. Team members are encouraged to leverage their

collective expertise to find optimal solutions and continuously improve

their processes.

1. **At Regular Intervals, the Team Reflects on How to Become More**

**Effective, Then Tunes and Adjusts Its Behavior Accordingly:** Agile

promotes a culture of continuous improvement through regular

retrospectives. Team reflect on their processes, collaboration, and

Outcomes, identify opportunities for enhancement, and implement

adjustments to optimize their effectiveness and satisfaction.