**Assignment on 30-07-2024**

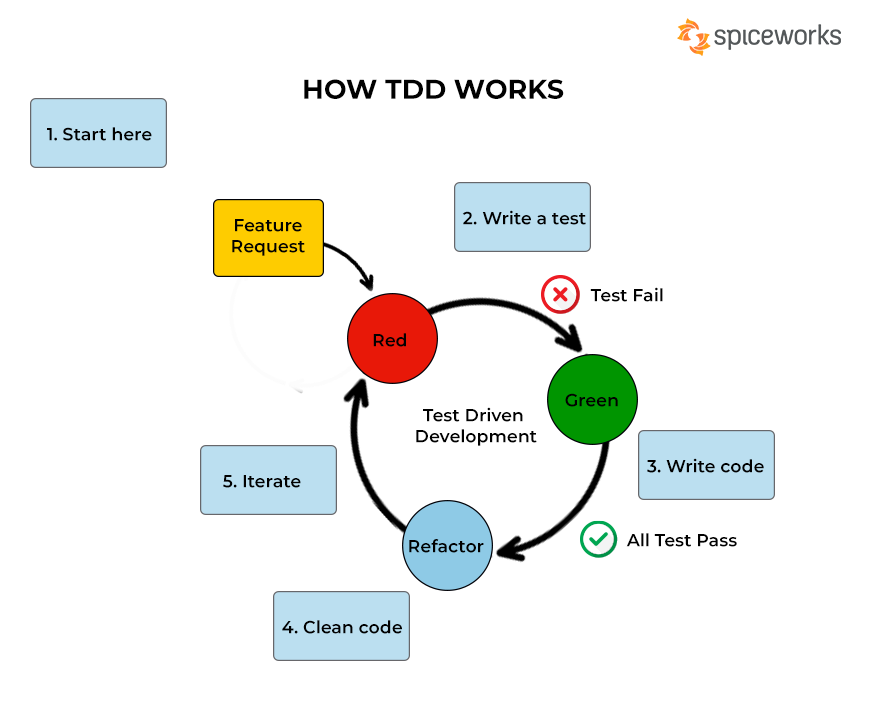
**Assignment 1:** Create an infographic illustrating the Test-Driven Development (TDD) process. Highlight steps like writing tests before code, benefits such as bug reduction, and how it fosters software reliability.

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

**Concept:**  
TDD is a software development approach where tests are written before the actual code. The process involves writing a test for a small piece of functionality, writing the minimal code required to pass the test, and then refactoring the code.

**Process:**

1. **Write a Test:** Define a test for a new feature or functionality.
2. **Run the Test:** Check if the new test fails (since the feature isn't implemented yet).
3. **Write Code:** Implement the minimal amount of code to pass the test.
4. **Run Tests:** Ensure all tests pass.
5. **Refactor Code:** Improve the code without changing its behavior.
6. **Repeat:** Continue the cycle for additional features.

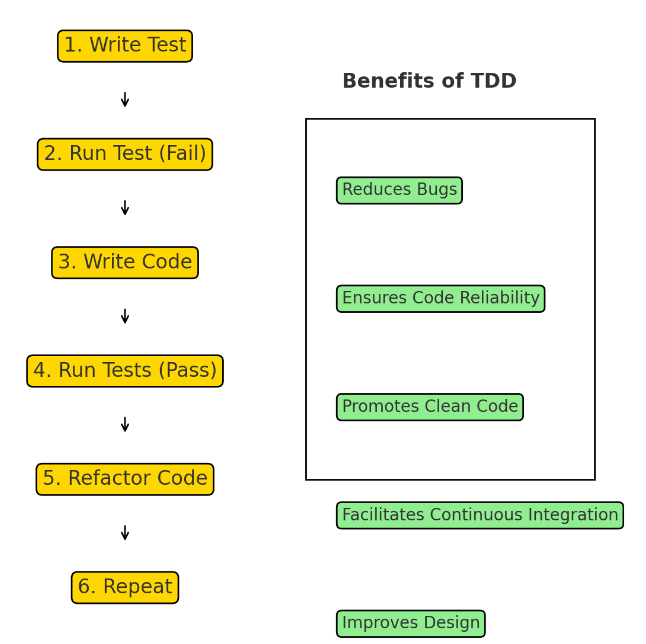


**Steps:**

1. **Write Test:** Define a test for a specific functionality before writing the code.
2. **Run Test (Fail):** Initially, the test should fail since the functionality isn't implemented yet.
3. **Write Code:** Develop the minimum amount of code required to pass the test.
4. **Run Tests (Pass):** Execute the tests to verify that the code works as expected.
5. **Refactor Code:** Improve the code structure and quality without changing its behavior.
6. **Repeat:** Continue this cycle for each new functionality or feature.

**Benefits of TDD:**

* **Reduces Bugs:** Catch issues early by writing tests first.
* **Ensures Code Reliability:** Regular testing ensures the code works correctly.
* **Promotes Clean Code:** Encourages refactoring and maintaining a clean codebase.
* **Facilitates Continuous Integration:** Integration becomes smoother with frequent testing.
* **Improves Design:** Encourages thoughtful design and modularity through testable units.



**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 underst

Let's create a comparative infographic for TDD (Test-Driven Development), BDD (Behavior-Driven Development), and FDD (Feature-Driven Development), focusing on their approaches, key benefits, and typical use cases.

| **Aspect** | **Test-Driven Development (TDD)** | **Behavior-Driven Development (BDD)** | **Feature-Driven Development (FDD)** |
| --- | --- | --- | --- |
| **Approach** | Write tests before code; focus on unit testing. | Define behavior using natural language, aligning with user stories. | Model-driven, feature-centric development. |
| **Primary Focus** | Ensuring code correctness and reliability. | Ensuring that the software behavior matches business requirements. | Developing and delivering functional features iteratively. |
| **Test Format** | Unit tests are written in programming languages. | Scenarios are written in a natural language (e.g., Gherkin). | Focuses on features rather than tests; progress tracked via feature lists. |
| **Collaboration** | Primarily between developers and testers. | Involves collaboration between developers, testers, and non-technical stakeholders. | Collaboration mainly among development teams; stakeholders provide feature lists. |
| **Benefits** | Early bug detection, continuous integration support, improved code quality. | Improved stakeholder communication, ensures alignment with business needs, supports automated acceptance testing. | Focus on delivering working software, scalable for large teams, clear progress tracking. |
| **Challenges** | Can be time-consuming to write and maintain tests. | Requires clear definition of behaviors, potential complexity in maintaining scenarios. | Requires detailed planning and design, can be complex for smaller teams. |
| **Suitability** | Projects where code correctness is crucial and test coverage is prioritized. | Projects with frequent requirement changes and significant stakeholder involvement. | Large-scale projects with well-defined features and need for modular development. |

Here's the comparative infographic for TDD, BDD, and FDD methodologies:

* **Test-Driven Development (TDD):**
  + **Approach:** Tests are written before the code, focusing on small units of functionality.
  + **Benefits:** Early bug detection, improved code quality, supports continuous integration.
  + **Suitability:** Best for projects where code correctness is crucial and test coverage is prioritized.
* **Behavior-Driven Development (BDD):**
  + **Approach:** Focuses on behavior specifications written in natural language, aligning with user stories.
  + **Benefits:** Improves communication with non-technical stakeholders, ensures alignment with business requirements, supports automated acceptance tests.
  + **Suitability:** Ideal for projects with significant stakeholder involvement and frequent requirement changes.
* **Feature-Driven Development (FDD):**
  + **Approach:** A model-driven approach where features are developed iteratively and incrementally.
  + **Benefits:** Focus on delivering features, clear progress tracking, scalable for large teams.
  + **Suitability:** Suitable for large-scale projects with well-defined feature lists and modular development. ​

**Assignment 1**: Agile Project Planning - Create a one-page project plan for a new software feature using Agile planning techniques. Include backlog items with estimated story points and a prioritized list of user stories.

**Project Plan: New Software Feature**

**Feature: User Authentication and Authorization**

**Sprint Duration:** 2 weeks  
**Team Members:** 5 (2 Developers, 1 QA, 1 UI/UX Designer, 1 Product Owner)

**Backlog Items and User Stories**

| **Priority** | **User Story** | **Description** | **Acceptance Criteria** | **Story Points** |
| --- | --- | --- | --- | --- |
| 1 | **As a user, I want to register for an account** | Allow new users to create an account using email and password. | Users can register with a unique email; receive a confirmation email. | 8 |
| 2 | **As a user, I want to log in to my account** | Enable users to log in using their credentials. | Users can log in with registered email and password. | 5 |
| 3 | **As a user, I want to reset my password if forgotten** | Provide a way for users to reset their password through email verification. | Users can request a password reset and update their password. | 8 |
| 4 | **As a user, I want to log out of my account** | Implement a logout feature for users to securely exit their accounts. | Users can log out from their account securely. | 3 |
| 5 | **As an admin, I want to assign roles to users** | Admins can assign roles (user, admin, etc.) to different accounts for access control. | Admins can assign and change roles; role-based access control works. | 13 |
| 6 | **As a user, I want to update my account details** | Allow users to update their profile information, such as name, email, and password. | Users can update profile information and receive confirmation. | 5 |
| 7 | **As a user, I want to be notified of suspicious login attempts** | Notify users via email if suspicious login attempts are detected on their account. | Users receive notifications of unusual login attempts. | 8 |
| 8 | **As an admin, I want to view a list of all registered users** | Provide an admin panel feature to list all registered users and their roles. | Admins can view a list of users with their roles and registration date. | 8 |
| 9 | **As a user, I want to enable two-factor authentication (2FA) for my account** | Add an additional layer of security by allowing users to enable 2FA using a mobile app or email verification. | Users can enable 2FA and are prompted during login. | 13 |

**Prioritized User Stories for Sprint 1**

1. **User Registration** - 8 Story Points
2. **User Login** - 5 Story Points
3. **Password Reset** - 8 Story Points
4. **Logout** - 3 Story Points
5. **Profile Update** - 5 Story Points

**Total Story Points for Sprint 1:** 29

**Sprint Goals**

* Implement basic user account functionality: registration, login, logout, and password management.
* Ensure a simple and intuitive user interface for these features.
* Set up basic security measures, including email verification and password strength validation.

**Sprint Planning and Execution**

* **Sprint Planning Meeting:** Define tasks and assign them to team members.
* **Daily Stand-ups:** Track progress, discuss blockers, and adjust tasks as needed.
* **Sprint Review:** Demonstrate the implemented features to stakeholders.
* **Sprint Retrospective:** Reflect on what went well and areas for improvement for the next sprint.

This plan includes the key elements of Agile project planning, focusing on user stories, prioritization, and iterative development. The story points help estimate the effort required, while the prioritized list ensures the most critical features are developed first.

**Assignment 2:** Daily Standup Simulation - Write a script for a Daily Standup meeting for a development team working on the software feature from Assignment 1. Address a common challenge and incorporate a solution into the communication flow

**Daily Standup Script**

**Context:** The team is developing a new feature for a software application based on Assignment 1. Let's assume the feature is a real-time collaboration tool for document editing.

**Scrum Master (Alex):** Good morning, everyone! Let's kick off our daily standup. Remember to keep it short and sweet. We'll go over what we did yesterday, what we're working on today, and any blockers. Let's start with the back-end team.

**Backend Developer (Jamie):**  
**Yesterday:** I finished implementing the API endpoints for real-time updates.  
**Today:** I'll be working on optimizing the data synchronization process to reduce latency.  
**Blockers:** We experienced some unexpected delays in the WebSocket connections under high load. I plan to collaborate with the DevOps team to diagnose and address these performance issues.

**Scrum Master (Alex):** Thanks, Jamie. Let's have the DevOps team take a look at that. Next up, front-end.

**Frontend Developer (Taylor):**  
**Yesterday:** I completed the UI for the collaborative editing toolbar and started integrating it with the WebSocket service.  
**Today:** I'll be focusing on implementing real-time cursor positioning and handling concurrent text edits.  
**Blockers:** There's a slight lag when multiple users edit the same document, causing the UI to jump occasionally. I’ll sync with Jamie to see if the backend optimization can help.

**Scrum Master (Alex):** Good point, Taylor. Let's ensure there's a clear communication line between front-end and back-end for real-time issues. Now, onto QA.

**QA Tester (Morgan):**  
**Yesterday:** I ran test cases for document synchronization and identified a few edge cases where changes weren't reflected correctly.  
**Today:** I'll be verifying fixes for those cases and expanding our test scenarios to include stress testing for high user loads.  
**Blockers:** No blockers from QA at the moment. But, I'd like to flag that we need more robust error handling for network interruptions.

**Scrum Master (Alex):** Thanks, Morgan. Let's prioritize those error handling improvements. Finally, UX.

**UX Designer (Jordan):**  
**Yesterday:** I worked on finalizing the user flow for inviting collaborators and the notification system for changes.  
**Today:** I'll be reviewing the user feedback from the last usability test and refining the UI based on the latest feature integrations.  
**Blockers:** None for now, but I need to sync with the front-end team to ensure our design changes are feasible within the current sprint.

**Scrum Master (Alex):** Great, Jordan. Let's make sure you and Taylor coordinate on those design changes.

**Common Challenge Discussion:**  
Before we wrap up, let's address a common challenge we've been facing: the slight lag and synchronization issues when multiple users are editing a document simultaneously.

**Proposed Solution:**  
To tackle this, we'll have Jamie and Taylor work closely together to fine-tune both the back-end data synchronization and the front-end rendering logic. Additionally, Morgan will assist with stress testing to better simulate high user loads and identify potential bottlenecks.

**Scrum Master (Alex):** Let's aim to have a preliminary fix by the end of the day so we can test it thoroughly. Any final thoughts or concerns?

**(No additional concerns)**

**Scrum Master (Alex):** Excellent. Let's stay focused and keep communication open. We'll reconvene tomorrow for another check-in. Have a productive day, everyone!

This script covers a typical daily standup flow, highlighting daily updates, addressing a common challenge, and proposing a solution to ensure the team works collaboratively towards resolving issues.