Test Driven Development

Software Engineering

Testing – Another view

Developers and testers test:

- Unit and integration tests - performed by developers before handing out builds to testers
- User acceptance testing – performed by testers
- Performance testing and UI testing – performed by both

https://www.freecodecamp.org/news/isnt-tdd-test-driven-development-twice-the-work-why-should-you-care-4ddcabeb3df9/

Testing – Another view

Descriptions:

- Unit testing – smallest testable part, faking, mocking and stubbing are indispensable
- Integration – verify that components work together
- Performance testing – ensure application performs well under expected workload

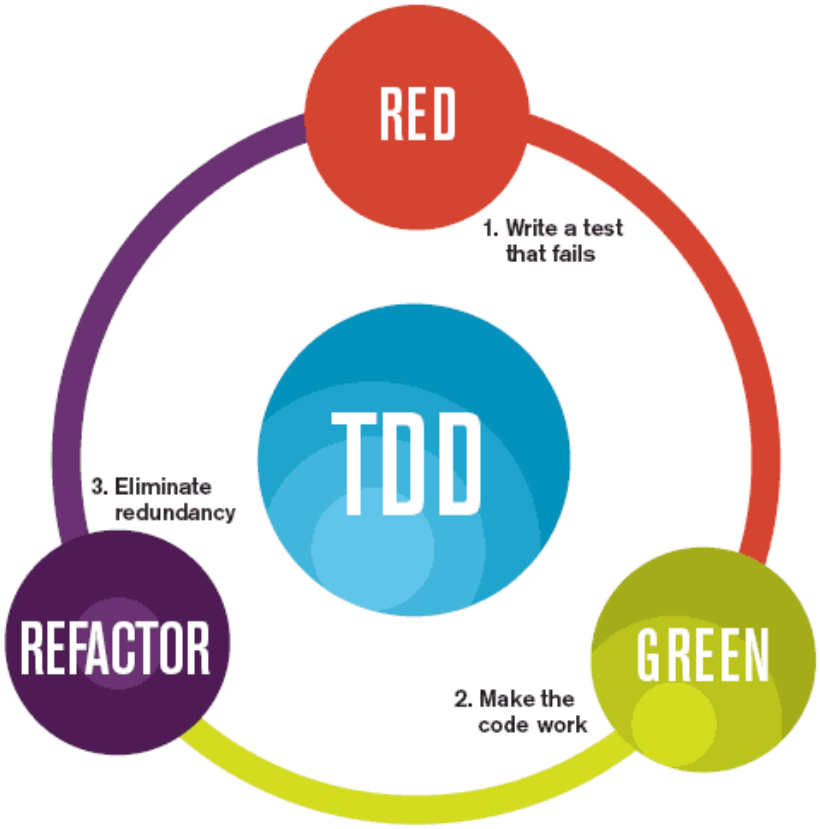
https://www.freecodecamp.org/news/isnt-tdd-test-driven-development-twice-the-work-why-should-you-care-4ddcabeb3df9/

Test-Driven Development

Test driven development:

1. Write a test case.
2. Verify that the test case fails.
3. Modify the code so that the test case succeeds. (Write the simplest code possible.)
4. Run the test to see it works. Also run previous test cases to see nothing regressed.
5. Refactor code to make it pretty

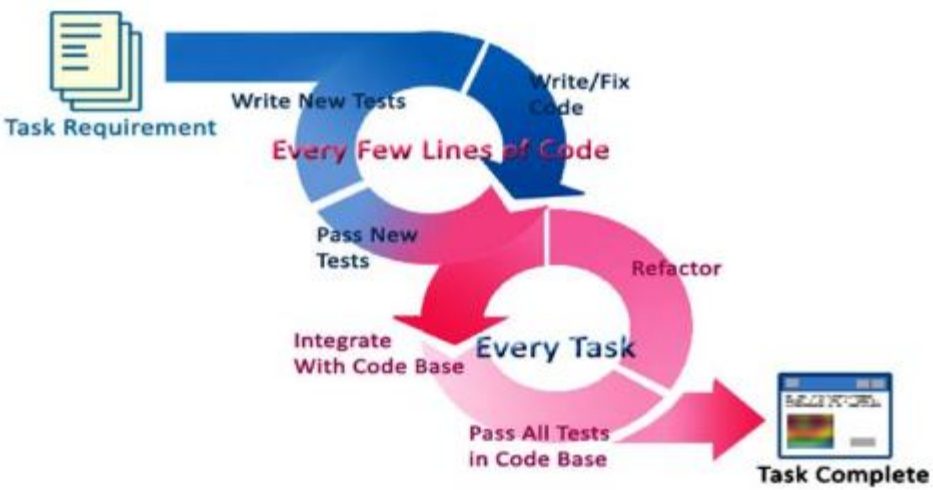
TDD



The mantra of Test-Driven Development (TDD) is "red, green, refactor."

https://www.freecodecamp.org/news/isnt-tdd-test-driven-development-twice-the-work-why-should-you-care-4ddcabeb3df9/

TDD with Integration Testing



Number of Iterations per task

https://www.freecodecamp.org/news/isnt-tdd-test-driven-development-twice-the-work-why-should-you-care-4ddcabeb3df9/

TDD Case Study

3 development teams at Microsoft & 1 at IBM used TDD and found that:

- Pre-lease defect density decreased between 40%-90% compared to projects that didn't use TDD
- Lower defect density resulted in 15%-35% increase in initial development time available due to not needing to fix old errors

https://www.freecodecamp.org/news/isnt-tdd-test-driven-development-twice-the-work-why-should-you-care-4ddcabeb3df9/

GetAllUsers Endpoint Test

```
pin test | output (ok) | run test | debug test
func TestGetUser_GoodCase_ResponseOK(t *testing.T) {
    // Arrange:
    // Create a request to send to the endpoint.
    req, err := http.NewRequest(http.MethodGet, "/users/3", nil)
    if err != nil {
        t.Fatal(err)
    }

    // Create a response recorder to capture the response
    rr := httptest.NewRecorder()
    // Create a router to route the request and url parameters
    router := mux.NewRouter()
    // Set the handler for the test
    router.HandleFunc("/users/{userID:[0-9]+}", userMockAPI.GetUser)

    // Act:
    // Perform the request.
    router.ServeHTTP(rr, req)

    // Assert:
    // Check that the response code is as expected.
    if status := rr.Code; status != http.StatusOK {
        t.Errorf("handler returned the wrong status code: got %v want %v",
            status, http.StatusOK)
    }

    // Check the response body is as expected.
    expected := `{"userID":3,"uName":"Will Turner","usernameCAS":"wturner","activeFlag":true}`
    if rr.Body.String() != expected {
        t.Errorf("handler returned unexpected body: got %v want %v",
            rr.Body.String(), expected)
    }

    // Check that headers are as expected
    if ctype := rr.Header().Get(contentType); ctype != appjson {
```

MS OUTPUT DEBUG CONSOLE TERMINAL Go Tests

GetAllUsers Endpoint Test – mock database

```go
func (mockRepo) GetAllUsers() ([]models.User, error) {
    users := []models.User{
        {
            UserID:      1,
            UName:       "Jack Sparrow",
            UsernameCAS: "jsparrow",
            ActiveFlag:  false,
        },
        {
            UserID:      2,
            UName:       "Davey Jones",
            UsernameCAS: "djones",
            ActiveFlag:  true,
        },
    }
    return users, nil
}
```

GetAllUser Model Test

user_test.go

backend > models > user_test.go > {} models > TestGetAllUsers_GoodCase

run package tests | run file tests

```go
package models

import (
    "testing"
)

var api UserRepo

pin test | run test | debug test
func TestGetAllUsers_GoodCase(t *testing.T) {
    // Perform the query and collect results.
    users, err := api.GetAllUsers()
    if err != nil {
        t.Errorf("failed to get all users from database: %v\n", err)
    }
    // Some users are expected to have been returned.
    if len(users) == 0 {
        t.Errorf("failed to get all users from database: expected a non-zero " +
            "value, got 0\n")
    }
}
```