Chapter: Project Setup & Environment (45 mins)

📌 Chapter Objectives

By the end of this chapter, you will be able to:

- Create a sample Go application using the BDD approach
- Write `.feature` files describing behavior in Gherkin syntax
- Implement corresponding Go step definitions
- Integrate and execute your tests using `godog` with `go test`

🧠 Theoretical Concepts

In a BDD (Behavior-Driven Development) workflow, we start by writing *specifications* that describe the behavior of a system using natural language (Gherkin syntax). These specifications are then mapped to **step definitions**, which contain executable test logic.

🌍 Real-World Analogy

> Imagine you're building an online banking system.
> A product owner says: *"When a user logs in with valid credentials, they should be taken to their dashboard."*
>
> In BDD, you'd write this as a `.feature` file in Gherkin and then implement Go code to verify this behavior automatically.

🛠️ Hands-On Setup

Let's build a minimal **"User Login"** feature with BDD using `godog`.

🧱 Step 1: Initialize the Project

```
mkdir user-login-bdd
cd user-login-bdd
go mod init github.com/yourname/user-login-bdd
```

📄 Step 2: Create a `.feature` File

Create a directory for Gherkin features:

```
mkdir features
touch features/login.feature
```

Paste the following into `features/login.feature`:

```
Feature: User Login

  Scenario: Successful login with valid credentials
    Given a registered user with username "john" and password "secret"
    When the user logs in with username "john" and password "secret"
    Then the login should be successful
```

Step 3: Create Step Definitions

Create a `stepdefs` folder:

```
mkdir stepdefs
touch stepdefs/login_steps.go
```

Paste this Go code into `login_steps.go`:

```
package stepdefs

import (
    "fmt"
    "github.com/cucumber/godog"
)

type user struct {
    username string
    password string
}

type loginSystem struct {
    registeredUsers map[string]string
    loginSuccess    bool
}

var system loginSystem

func aRegisteredUser(username, password string) error {
    if system.registeredUsers == nil {
        system.registeredUsers = make(map[string]string)
    }
    system.registeredUsers[username] = password
    return nil
}
```

```
func userLogsIn(username, password string) error {
    if pass, exists := system.registeredUsers[username]; exists && pass ==
password {
        system.loginSuccess = true
        return nil
    }
    system.loginSuccess = false
    return fmt.Errorf("invalid login")
}

func loginShouldBeSuccessful() error {
    if !system.loginSuccess {
        return fmt.Errorf("expected login to be successful but it failed")
    }
    return nil
}

func InitializeScenario(ctx *godog.ScenarioContext) {
    system = loginSystem{} // Reset before each scenario

    ctx.Step(`^a registered user with username "([^"]*)" and password "
([^"]*)"$`, aRegisteredUser)
    ctx.Step(`^the user logs in with username "([^"]*)" and password "
([^"]*)"$`, userLogsIn)
    ctx.Step(`^the login should be successful$`, loginShouldBeSuccessful)
}
```

🔧 Step 4: Connect It with `go test`

Create `godog_test.go` in the root directory:

```
package main

import (
    "os"
    "testing"

    "github.com/cucumber/godog"
    "github.com/cucumber/godog/colors"
    "github.com/yourname/user-login-bdd/stepdefs"
)

func TestFeatures(t *testing.T) {
    opts := godog.Options{
        Format:        "pretty",
        Paths:         []string{"features"},
        Output:        colors.Colored(os.Stdout),
        Strict:        true,
        TestingT:      t,
```

```
        StopOnFailure: true,
    }

    suite := godog.TestSuite{
        ScenarioInitializer: stepdefs.InitializeScenario,
        Options:             &opts,
    }

    if suite.Run() != 0 {
        t.Fatal("test suite failed")
    }
}
```

> 🛠️ Replace `github.com/yourname/user-login-bdd` with your actual module path from `go.mod`.

▶️ Step 5: Run the Tests

```
go test -v
```

You should see:

```
Feature: User Login

  Scenario: Successful login with valid credentials
    Given a registered user with username "john" and password "secret"
    When the user logs in with username "john" and password "secret"
    Then the login should be successful

1 scenarios (1 passed)
3 steps (3 passed)
```

🎯 Interview Questions

1. How is a BDD project structured in Go using godog?
2. What is the role of `.feature` files and step definitions?
3. How does `godog` integrate with `go test`?
4. What is the purpose of the `InitializeScenario` function?
5. Can BDD steps be reused across multiple scenarios?

📺 Curated YouTube Videos

1. BDD with Go using Godog (Intro) - Kevin Gillette
2. Writing Feature Files in Gherkin

3. Using Godog with Go for BDD
4. Intro to Cucumber BDD Testing

Resources

- godog GitHub Repo
- Gherkin Syntax Reference
- Go Modules Guide

Summary

- You now have a working BDD setup for a user login system in Go
- You've written a `.feature` file and connected it to executable Go tests
- You can now scale this structure to any number of features and scenarios

> In the next chapter, we'll explore how to organize large test suites and use scenario outlines for more flexible Gherkin files.