2-Day Hands-On Training Program: BDD, Cucumber, Gherkin, and godog (with Go)

> **Audience:** Proficient in Go, beginner in BDD and associated tools
> **Objective:** By the end of Day 2, you should be able to design, implement, and run BDD scenarios using godog in a real-world Go application

---

🕘 Morning Session: Introduction and Theory

1. Welcome & Overview (30 mins)

📦 Outcome of this Session

You should now:

- Understand the purpose of this course
- Know what to expect from each session
- Recognize how BDD bridges the gap between business and tech
- Be ready to start writing Gherkin scenarios in the next chapter

Chapter: Course Structure, Goals, and Expectations

This chapter sets the tone for your journey into **Behavior Driven Development (BDD)** using **Gherkin**, **Cucumber-style syntax**, and the **godog** framework with Go. Whether you're a developer, QA engineer, or product owner, this course will help you align technical efforts with business needs more effectively.

🎯 Course Goals

By the end of this course, you will:

- ☑ Understand the **philosophy behind BDD** and how it improves software development.
- ☑ Learn how to write **clear, maintainable user scenarios** using Gherkin syntax.
- ☑ Use **godog**, a Cucumber-compatible BDD tool for Go, to automate behavior tests.
- ☑ Implement **step definitions** in Go that bind your Gherkin scenarios to executable tests.
- ☑ Build a **real-world mini-project** that demonstrates BDD from feature file to tested functionality.

🏗 Course Structure

Day	Session	Topics Covered
Day 1	Morning	What is BDD? Why use it? Gherkin Syntax
Day 1	Afternoon	Writing feature files, best practices

Day	Session	Topics Covered
Day 2	Morning	Introduction to godog, project setup
Day 2	Afternoon	Step definitions, full project, integration

Each session includes:

- **Theory (30–60 mins)**
- **Live Demo (15–30 mins)**
- **Hands-On Practice (60–90 mins)**
- **Discussion + Q&A (15–30 mins)**

🧠 What You Will Learn

1. What is BDD (Behavior Driven Development)?

BDD is a **collaborative practice** where teams define software behavior in **plain language** so that everyone—technical and non-technical—can understand and validate the system's expected behavior.

> **Real-world analogy:**
> Think of BDD like defining a restaurant order in plain English:
> *"If I order a Veg Burger, and I ask for no onions, I should receive a Veg Burger without onions."*
> The kitchen (developers) and customer (business/user) both understand the expectation clearly.

---

2. Why BDD Matters in Modern Software Projects

- 🧩 Encourages **collaboration** between developers, testers, and product owners.
- 📜 Provides **living documentation** in the form of Gherkin feature files.
- ☑️ Ensures **better test coverage** by focusing on behavior rather than code paths.
- 💡 Drives **design and development** from the user perspective.

---

3. How Does This Course Help You?

You'll learn how to:

- Speak the **language of users** using Gherkin.
- Write automation in Go with **godog**, not just traditional unit tests.
- Catch bugs **before code is written** by validating expected behavior early.

💻 Hands-On Example

Let's say you're building a **User Login system**.

Business Requirement:

> "A registered user should be able to log in with valid credentials."

Gherkin Scenario:

```
Feature: User login
  Scenario: Successful login with correct credentials
    Given a registered user with email "john@example.com" and password
"pass123"
    When the user tries to login with email "john@example.com" and password
"pass123"
    Then the login should be successful
```

Step Definition in Go (godog):

```go
func (s *loginSuite) aRegisteredUserWithEmailAndPassword(email, password
string) error {
    s.user = User{Email: email, Password: password}
    return nil
}
```

When run via godog, this test becomes a source of truth for both the developer and the stakeholder.

☑ Expectations from You

- 💻 Be hands-on: You'll write feature files, Go step definitions, and run tests.
- ❓ Ask questions: Especially when mapping business rules to test cases.
- 🗀 Organize your code: BDD projects must be cleanly structured for readability.
- ✎ Embrace TDD/BDD thinking: Write tests/scenarios before coding.

🎥 Recommended YouTube Videos

- What is BDD? - Behavior Driven Development Explained
- Gherkin Syntax Tutorial for Beginners
- Getting started with godog
- Behavior Driven Development with Go (godog)

❓ Interview Questions

- What is the difference between TDD and BDD?
- What are the main components of a Gherkin feature file?
- How does BDD improve collaboration in a team?
- How does godog relate to Cucumber?
- Can BDD tests be used for non-functional requirements like performance?
- How do you ensure your Gherkin scenarios stay maintainable over time?

- What are common pitfalls in BDD adoption?

Chapter: Why BDD is Critical in Agile Projects

🧠 What is BDD?

Behavior-Driven Development (BDD) is a software development process that encourages collaboration among developers, QA, and non-technical or business participants in a software project.

At its core, BDD helps teams:

- Focus on the **behavior** of the software from the user's perspective
- Use **natural language** (plain English) to describe system behavior
- Create a **shared understanding** between all stakeholders

📦 Why is BDD Important in Agile?

Agile encourages **rapid iterations**, **frequent feedback**, and **continuous collaboration**. BDD aligns beautifully with these principles by:

Agile Principle	How BDD Supports It
Working software over documentation	Gherkin scenarios *are* living documentation
Customer collaboration	Non-developers can write or review scenarios
Responding to change	Tests describe expected behavior, not code
Deliver working software frequently	Faster feedback through executable specs

---

🌍 Real-World Analogy

Imagine building a house.

- 🧱 **Without BDD**: The builder reads a vague blueprint, starts construction, and shows you the finished house. Oops! You wanted 3 bedrooms, not 2. Now it's costly to fix.
- 🧱 **With BDD**: You (owner), the architect, and the builder agree on a *storyboard*: "Given I open the main door, I should see a hallway that leads to three bedrooms." Everyone's expectations are aligned **before construction begins.**

👨‍💻 Simple Hands-On Example

Let's say we are building a **Login Page**.

🗣️ User Story (Business Requirement)

> As a registered user, I want to log in using my email and password so that I can access my dashboard.

☑ Without BDD

- Developer builds a form.
- Tester writes test cases after development.
- Business validates it post-facto.
- Misalignment? Too late.

☑ With BDD (Gherkin Scenario)

```
Feature: User Login

  Scenario: Successful login with valid credentials
    Given the user is on the login page
    When the user enters valid email and password
    And clicks the login button
    Then the user should be redirected to the dashboard
```

- Everyone (Business, QA, Dev) understands and agrees on this behavior.
- This scenario becomes an **automated acceptance test** using tools like Cucumber or godog.

🏆 Key Benefits in Agile Projects

1. **Shared Understanding**
   No more guessing what the client *really* meant. Scenarios spell it out.

2. **Living Documentation**
   Feature files (written in Gherkin) serve as up-to-date documentation.

3. **Shift Left Testing**
   Tests are defined **before** development starts.

4. **Faster Feedback Loop**
   Failures during development, not post-deployment.

5. **Regression Safety**
   Automated tests ensure nothing is broken in future releases.

6. **Clear Definition of Done**
   A user story is *done* when all scenarios pass.

🎥 Recommended YouTube Videos

- **What is BDD?** by Cucumber – A short official video

- **Behavior Driven Development (BDD) Explained** by GOTO Conferences – A good conceptual overview
- **BDD with Gherkin and Cucumber** by Automation Step by Step – Beginner-friendly Gherkin tutorial

---

💬 Common Interview Questions

1. **What is the difference between BDD and TDD?**

   > TDD focuses on writing tests before code. BDD focuses on defining behavior in plain language before development begins.

2. **Why is BDD preferred in Agile environments?**

   > Because it fosters collaboration, early validation, and ensures business goals are met through executable scenarios.

3. **What is Gherkin syntax and how is it used in BDD?**

   > Gherkin is a human-readable language using `Given-When-Then` to define behavior. It's the standard for writing BDD scenarios.

4. **How does BDD help reduce rework in Agile projects?**

   > By aligning developers and business stakeholders early, it prevents misunderstandings and ensures correct implementation.

5. **Which tools are used to implement BDD in Go projects?**

   > `godog` is a popular tool for BDD in Go, similar to `Cucumber` in Java.

🚀 Summary

Behavior-Driven Development brings clarity, communication, and confidence to Agile software teams. By writing expected behavior in plain language, teams ensure that the software does what the user actually wants — not what the developer *thinks* the user wants.

It helps:

- Avoid rework
- Align stakeholders
- Automate testing from the beginning
- Create meaningful, human-readable documentation

📋 Further Reading

- Cucumber BDD Overview
- godog GitHub Repository
- Agile Manifesto