Chapter: Mini Project – End-to-End BDD with Go (1.5 hours)

🎯 Objective

Build and test a minimal RESTful **User Data Management Service** using:

- Gherkin feature files
- Go step definitions with godog
- Full CRUD support

🧠 Theoretical Concepts

In a real-world microservice, we often expose REST APIs for managing resources.
BDD ensures these APIs meet the business behavior described in plain language. This approach:

- Keeps product owners involved
- Ensures test coverage is behavior-centric
- Keeps dev and QA teams aligned

🏗️ REST API Overview

We will implement a service with the following endpoints:

Method	Endpoint	Description
GET	`/users`	Fetch all users
GET	`/users/{id}`	Fetch user by ID
POST	`/users`	Create a new user
PUT	`/users/{id}`	Update a user
DELETE	`/users/{id}`	Delete user by ID
DELETE	`/users`	Delete all users

📂 Project Setup

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

✍️ Step 1: Feature Files

Create `features/user_management.feature`:

```
Feature: User Management

  Scenario: Create a new user
    When I create a user with name "John" and email "john@example.com"
    Then the response code should be 201

  Scenario: Get all users
    When I fetch all users
    Then the response code should be 200

  Scenario: Get user by ID
    When I fetch user with ID 1
    Then the response code should be 200

  Scenario: Update user by ID
    When I update user with ID 1 to name "Johnny" and email
"johnny@example.com"
    Then the response code should be 200

  Scenario: Delete user by ID
    When I delete user with ID 1
    Then the response code should be 204
```

💻 Step 2: REST Server in Go

[See main.go code in original tutorial]

🧩 Step 3: Step Definitions

[See stepdefs/user_steps.go code in original tutorial]

🧪 Step 4: Run the Tests

Create `godog_test.go`:

[See godog_test.go code in original tutorial]

Start the server in one terminal:

```
go run main.go
```

Run the BDD tests in another terminal:

```
go test -v
```

☑ Interview Questions

1. How do you connect Gherkin feature files with Go step functions?
2. How do you verify HTTP responses using BDD?
3. What's the role of `godog.Options` and `TestSuite`?
4. How can you validate structured data (JSON) in a test?
5. How would you organize large step definitions in real-world projects?

📺 Curated YouTube Videos

- Build REST API with Go (Golang)
- BDD with Cucumber and Gherkin
- Behavior Driven Development (BDD) Explained

📚 Additional Resources

- godog GitHub
- Gherkin Reference
- mux Router for Go
- Go HTTP Docs

🎯 Summary

- You built a full CRUD REST API in Go
- You defined real-world behavior in `.feature` files
- You tested API logic using Go and `godog`
- You completed an end-to-end BDD implementation

> Up next: Learn how to use Scenario Outlines and Background steps to minimize redundancy.