🧪 User API BDD Project (Golang + godog)

This project demonstrates how to implement **Behavior Driven Development (BDD)** in Go using the godog framework.
It showcases an end-to-end testable **User Data Management Service** with full REST CRUD support, verified by `.feature` files written in Gherkin syntax.

📁 Project Structure

```
user-api-bdd/
├── go.mod                  # Go module definition
├── main.go                 # REST API server implementation
├── godog_test.go           # Godog entry point for BDD test execution
├── stepdefs/
│   └── user_steps.go       # Step definitions for feature steps
└── features/
    └── user_management.feature  # Gherkin feature file describing test cases
```

📌 Features Covered

- `POST /users` — Create a new user
- `GET /users` — Get all users
- `GET /users/{id}` — Get a specific user by ID
- `PUT /users/{id}` — Update a user by ID
- `DELETE /users/{id}` — Delete a user by ID
- `DELETE /users` — Delete all users

🚀 Getting Started

1. Clone the repository

```
git clone https://github.com/yourname/user-api-bdd.git
cd user-api-bdd
```

2. Install dependencies

```
go mod tidy
```

Make sure Go version 1.18+ is installed.

3. Install godog

```
go install github.com/cucumber/godog/cmd/godog@latest
```

▶ Running the Application

In one terminal:

```
go run main.go
```

This starts the server at `http://localhost:8080`.

☑ Running BDD Tests

In another terminal:

```
go test -v
```

This executes the godog test suite, driven by the Gherkin feature in `features/user_management.feature`.

📄 Sample `.feature` Test Case

```
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
```

Each step in the above scenario is bound to a Go function inside `stepdefs/user_steps.go`.

🧪 API Example Payloads

Create User (`POST /users`)

```
{
  "name": "Alice",
  "email": "alice@example.com"
}
```

Update User (`PUT /users/1`)

```
{
  "name": "Alice Smith",
  "email": "alice.smith@example.com"
}
```

🤠 Developer Tips

- Use `fmt.Printf()` inside steps for debugging responses.
- Ensure your API server is running before executing BDD tests.
- You can group and reuse common steps using Go struct methods.
- Each feature scenario is independent; reset or manage in-memory state accordingly.

🎯 Common Gotchas

Issue	Cause	Solution
`no response received`	API server not running	Run `go run main.go` in a separate terminal
`undefined step`	Step not mapped in Go	Implement and bind using `ctx.Step(...)`
`godog run is deprecated`	Old godog usage	Use `go test -v` instead
`panic: nil pointer dereference`	Missing response/error handling	Check for `nil` response before use

---

🛠 Tech Stack

- **Language:** Golang
- **Framework:** godog (Cucumber for Go)
- **Router:** Gorilla Mux
- **Testing:** Gherkin `.feature` files + `go test`

📚 Resources

- Godog Documentation
- Cucumber Gherkin Syntax
- Go Testing
- Gorilla Mux Docs

🤝 Contributing

Feel free to fork, improve or extend the scenarios to support more advanced API contracts and validations.

📌 Author

Rahul S. Patil – @rahulspatil

📃 License

This project is licensed under the MIT License - see the LICENSE file for details.