Chapter: Editor Setup (30 mins)

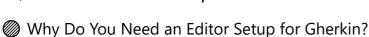


Objective

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

- Set up Visual Studio Code (VS Code) for working with .feature files written in Gherkin syntax
- Enable syntax highlighting, linting, and autocomplete
- Validate the readability and formatting of your feature files using proper tools

(2) Theoretical Concepts



Gherkin is a domain-specific language used to describe application behavior without detailing how that functionality is implemented. Since .feature files are text-based, having an IDE/editor that provides:

- Syntax highlighting
- Step suggestions
- Error detection
 - ...can significantly improve your productivity and reduce errors.

Real-World Analogy

Think of Gherkin as writing a recipe in English for someone else to cook (the developer). If your editor doesn't highlight syntax or catch typos, it's like writing a blurry, hard-to-read recipe that leads to a bad dish. A proper editor setup is like having Grammarly + Google Docs formatting features for your automation scripts.

Hands-On: Setting Up VS Code for Gherkin

Step 1: Install Visual Studio Code

Download from https://code.visualstudio.com/ and install it on your system.

Step 2: Install Gherkin/Cucumber Plugins

Recommended Extensions:

PROFESSEUR: M.DA ROS

Extension Name	Publisher	Description
Cucumber (Gherkin) Full Support	Alexander Krechik	Syntax highlight, step auto-complete, jump to step definition
Gherkin step autocomplete	Steve W	Offers step suggestions from existing steps
Prettier - Code Formatter	Prettier	Ensures clean, consistent formatting

To install:

- 1. Open VS Code.
- 2. Go to Extensions (Ctrl+Shift+X).
- 3. Search and install each of the above.

Step 3: Validate Syntax and Highlighting

1. Create a .feature file

File → New File → Save As → login.feature

2. Paste sample Gherkin code:

```
Scenario: Successful login
Given the user is on the login page
When the user enters valid credentials
Then they should be redirected to the dashboard
```

3. You should see:

- Color-coded keywords like Feature, Scenario, Given, etc.
- Warning/error markers if syntax is invalid
- Autocomplete when typing existing steps

Formatting and Readability Tips

✓ Do:

- Keep indentation consistent (2 spaces or tab)
- Use plain, non-technical language
- Group similar scenarios under one feature
- Use tags (@smoke, @login) for organization

X Avoid:

- Mixing multiple features in one file
- Using ambiguous or overly technical phrases
- Writing long scenario steps with multiple actions

Recommended YouTube Videos

- @ Watch these while setting up your editor for a complete understanding.
- VS Code Extensions for BDD and Gherkin

- How to Write Feature Files with Gherkin in VS Code
- Getting Started with Cucumber in VS Code

Common Interview Questions

- 1. Q: What is the purpose of .feature files in BDD?
 - **A:** .feature files describe test scenarios in plain language using Gherkin syntax and are interpreted by tools like Cucumber or godog.
- 2. **Q:** How do plugins improve Gherkin authoring in VS Code?
 - **A:** They provide syntax highlighting, step autocomplete, and quick navigation to definitions which improves efficiency and accuracy.
- 3. **Q:** How would you set up your local environment for Gherkin-based BDD?
 - A: I'd use VS Code with the Gherkin plugin, Prettier for formatting, and organize .feature files in a features/ directory in the project root.
- 4. **Q:** What are best practices for writing readable .feature files?
 - **A:** Use simple, business-focused language; one scenario per behavior; consistent indentation; and logical step grouping.

Quick Practice Task

Feature: User Signup

Create your own signup.feature file in VS Code with the following scenario:

```
Scenario: New user signs up with valid data
Given the user is on the signup page
When the user fills the form with valid details
And clicks on the register button
Then they should see a success message
```

Validate it in VS Code for syntax and formatting. Try renaming a keyword like Scenario to Scenarios and see how the editor helps you catch the error.

Summary

You now have a fully configured editor that:

- Highlights Gherkin syntax
- Suggests steps
- Formats your files cleanly



🔊 Pro Tip

Add this setting in your settings.json to format .feature files consistently with Prettier:

```
"[gherkin]": {
   "editor.defaultFormatter": "esbenp.prettier-vscode",
   "editor.formatOnSave": true
}
```