

Empty Feature

This feature exists to test lint warnings for empty features.

File: empty.feature

Generated: May 23, 2026

Scenarios: 0

Login

Users log in with email and password to receive a JWT access token. All errors return the same message to prevent user enumeration.

File: login.feature

Generated: May 23, 2026

Scenarios: 5

BACKGROUND

- ☐ **Given** the authentication service is running
- ☐ **And** the database is seeded with test users

SCENARIO 1 OF 5

Valid credentials return a JWT

- ☐ **Given** a user exists with email `<code><code></code>"alice@example.com"<code></code></code>` and password `<code><code></code>"secret123"<code></code></code>`
- ☐ **When** login is called with email `<code><code></code>"alice@example.com"<code></code></code>` and password `<code><code></code>"secret123"<code></code></code>`
- ☐ **Then** a JWT token is returned
- ☐ **And** the response contains the user object with email `<code><code></code>"alice@example.com"<code></code></code>`
- ☐ **And** the HTTP status is 200

SCENARIO 2 OF 5

Wrong password returns generic auth error

- ☐ **Given** a user exists with email `<code><code></code>"alice@example.com"</code></code></code>`
- ☐ **When** login is called with email `<code><code></code>"alice@example.com"</code></code></code>`
- ☐ **Then** the HTTP status is 401
- ☐ **And** the response contains error `<code><code></code>"invalid credentials"</code></code></code>`

SCENARIO 3 OF 5

Rate limited after 5 failed attempts

- ☐ **Given** a user exists with email `<code><code></code>"alice@example.com"</code></code></code>`
- ☐ **And** 5 failed login attempts have occurred in the last minute
- ☐ **When** login is called with email `<code><code></code>"alice@example.com"</code></code></code>`
- ☐ **Then** the HTTP status is 429
- ☐ **And** the response contains error `<code><code></code>"rate limited"</code></code></code>`

SCENARIO 4 OF 5

Malformed request returns validation error

- ☐ **Given** the login endpoint is available
- ☐ **When** login is called with email `<code><code></code>"not-an-email"</code></code></code>`
- ☐ **Then** the HTTP status is 400
- ☐ **And** the response contains error `<code><code></code>"validation failed"</code></code></code>`
- ☐ **And** no database query is made

Database unavailable returns 500 without crash

- ☐ **Given** the database is unavailable
- ☐ **When** login is called with email `<code><code></code>"alice@example.com"<code></code></code>` and password `<code><code></code>"secret123"<code></code></code>`
- ☐ **Then** the HTTP status is 500
- ☐ **And** the service remains running
- ☐ **And** an error is logged with level `<code><code></code>"error"<code></code></code>`

Malformed Feature

This file has intentional Gherkin errors for linting tests.

File: malformed.feature

Generated: May 23, 2026

Scenarios: 3

SCENARIO 1 OF 3

Missing Then step

- ☐ **Given** a precondition
- ☐ **When** an action is taken

SCENARIO 2 OF 3

Missing Then step

- ☐ **Given** another precondition
- ☐ **When** another action is taken

SCENARIO 3 OF 3

Empty scenario

Password Reset

Users can request a password reset link via email. The link expires after 1 hour and is single-use. Old links are invalidated when a new one is requested.

File: with_custom_blocks.feature
Generated: May 23, 2026
Scenarios: 6

SCENARIO 1 OF 6

Request reset for known email sends link

- ☐ **Given** a user exists with email `<code><code></code>"bob@example.com"<code></code></code>`
- ☐ **When** a password reset is requested for `<code><code></code>"bob@example.com"<code></code></code>`
- ☐ **Then** the HTTP status is 200
- ☐ **And** a reset email is sent to `<code><code></code>"bob@example.com"<code></code></code>`
- ☐ **And** the reset token expires in 3600 seconds

SCENARIO 2 OF 6

Request reset for unknown email returns 200 silently

- ☐ **Given** no user exists with email `<code><code></code>"ghost@example.com"<code></code></code>`
- ☐ **When** a password reset is requested for `<code><code></code>"ghost@example.com"<code></code></code>`
- ☐ **Then** the HTTP status is 200
- ☐ **And** no email is sent

SCENARIO 3 OF 6

Using a valid token sets the new password

- ☐ **Given** a valid reset token exists for `<code><code></code>"bob@example.com"<code></code></code>`
- ☐ **When** the reset endpoint is called with the token and new password `<code><code></code>"newpassword99"<code></code></code>`
- ☐ **Then** the HTTP status is 200
- ☐ **And** the user can log in with `<code><code></code>"bob@example.com"<code></code></code> and <code><code></code>"newpassword99"<code></code></code>`
- ☐ **And** the reset token is invalidated

SCENARIO 4 OF 6

Expired token is rejected

- ☐ **Given** an expired reset token exists for `<code><code></code>"bob@example.com"<code></code></code>`
- ☐ **When** the reset endpoint is called with the expired token and new password `<code><code></code>"xyz"<code></code></code>`
- ☐ **Then** the HTTP status is 400
- ☐ **And** the response contains error `<code><code></code>"token_expired"<code></code></code>`

SCENARIO 5 OF 6

Requesting a second reset invalidates the first token

- ☐ **Given** a valid reset token exists for `<code><code></code>"bob@example.com"<code></code></code>`
- ☐ **When** a password reset is requested again for `<code><code></code>"bob@example.com"<code></code></code>`
- ☐ **And** the original token is used
- ☐ **Then** the HTTP status is 400
- ☐ **And** the response contains error `<code><code></code>"invalid_token"<code></code></code>`

Password strength validation on reset

- ☐ **Given** a valid reset token exists for `<code><code></code>"bob@example.com"<code></code></code>`
- ☐ **When** the reset endpoint is called with the token and new password `<code><code></code>"<code><password></code>"<code></code></code>`
- ☐ **Then** the HTTP status is `<code><status></code>`
- ☐ **And** the response contains `<code><code></code>"<code><message></code>"<code></code></code>`

EXAMPLES

password	status	message
short	400	password_too_short
validpass1	200	ok
12345678	400	password_too_simple