Flask CRUD Operations – Edge Cases

Test Case 1: Missing Required Fields (Create)

Practice Task: Create a record without mandatory fields

Input:
POST /items

Steps:
• Open Postman
• Send POST request without required fields
• Submit the request

Expected Output:
The system should reject the request and return a validation error.

Actual Output:
The request reaches the Flask API.
Validation detects missing required fields.
The record is not created in the database.
A 400 Bad Request response is returned.

Test Case 2: Empty Request Body

Practice Task: Send POST request with empty body

Input:
POST /items

Steps:
• Open Postman
• Send POST request without body
• Submit request

Expected Output:
The system should reject the empty request.

Actual Output / Answer:
Flask receives an empty request body.
JSON parsing fails or returns None.
No data is processed.
A 400 Bad Request response is returned.

Test Case 3: Invalid JSON Format

Practice Task: Send malformed JSON data

Input:
POST /items

Steps:
• Send malformed JSON payload
• Submit request

Expected Output:
The system should show a JSON parsing error.

Actual Output / Answer:
Flask fails to parse the JSON payload.
A JSON decode error occurs.
The request is rejected.
A 400 error response is returned.

Test Case 4: Duplicate Record Creation

Practice Task: Create a record with duplicate unique field

Input:
POST /items

Steps:
• Create a record
• Send another POST with same unique value

Expected Output:
The system should prevent duplicate entry.

Actual Output / Answer:
Database detects duplicate data.
Insert operation fails.
Record is not duplicated.
A conflict or error response is returned.

Test Case 5: Extra Fields in Request

Practice Task: Send unexpected fields in request body

Input:
POST /items

Steps:
• Add extra fields in JSON
• Send request

Expected Output:
The system should ignore or reject extra fields.

Actual Output / Answer:
Flask receives extra fields.
Schema validation filters or rejects them.
Only valid fields are processed.
A controlled response is returned.

Test Case 6: Incorrect Data Type

Practice Task: Send wrong data type for a field

Input:
POST /items

Steps:
• Send string instead of integer
• Submit request

Expected Output:
The system should reject invalid data types.

Actual Output / Answer:
Validation detects data type mismatch.
Database insert fails.
Record is not created.
An error response is returned.

Test Case 7: Large Payload Size

Practice Task: Send very large JSON payload

Input:
POST /items

Steps:
• Send large payload
• Submit request

Expected Output:
The system should restrict payload size.

Actual Output:
Request size exceeds limit.
Server rejects the request.
No processing occurs.
Payload size error is returned.

Test Case 8: Missing Content-Type Header

Practice Task: Send JSON without Content-Type

Input:
POST /items

Steps:
• Remove Content-Type header
• Send request

Expected Output:
The system should reject unsupported media type.

Actual Output / Answer:
Flask cannot parse request body.
Request data is treated as invalid.
Processing fails.
415 Unsupported Media Type is returned.

Test Case 9: Get Non-Existing Record

Practice Task: Fetch record using invalid ID

Input:
GET /items/999

Steps:
• Send GET request with invalid ID

Expected Output:
The system should return not found.

Actual Output / Answer:
Database query returns no data.
Flask detects missing record.
No data is returned.
404 Not Found response is sent.

Test Case 10: Invalid ID Format

Practice Task: Pass string instead of numeric ID

Input:
GET /items/abc

Steps:
• Send request with invalid ID format

Expected Output:
The system should reject invalid ID.

Actual Output / Answer:
Flask routing or validation fails.
Type conversion error occurs.
Request is rejected.
Validation error response is returned.

Test Case 11: Update Non-Existing Record

Practice Task: Update record that does not exist

Input:
PUT /items/999

Steps:
• Send update request for invalid ID

Expected Output:
The system should return record not found.

Actual Output / Answer:
Database update affects zero rows.
No record is updated.
Flask handles empty result.
404 response is returned.

Test Case 12: Partial Update Missing Fields

Practice Task: Update record with partial data

Input:
PATCH /items/1

Steps:
• Send only one field to update

Expected Output:
The system should update only provided fields.

Actual Output / Answer:
Flask processes partial data.
Unchanged fields remain intact.
Update succeeds.
Success response is returned.