Claims API - Test Case Suite

Table of Contents

1. Authentication

Endpoint: POST /auth/login

Description: Log in using username and password to obtain JWT tokens.

Preconditions: Valid user credentials should be there.

Test Steps:

1. Send POST /auth/login with valid JSON body {"username":"user","password":"pass"}.

2. Observe response status code.

3. Validate accessToken is present and has non-zero length.

4. Validate expiresIn is a positive integer (if present).

Expected Results:

• Status Code: 200 OK

• Response Body: JSON matching LoginResponse schema. Required field: accessToken.

• Tokens should be usable for authenticated endpoints (Bearer).

TC1 - Create Claim (Happy Path)

Endpoint: POST /claims

Create a new claim with valid required fields.

Preconditions:

• Authenticated user with valid Bearer token.

Request Example:

```
{
  "policyNumber": "P-10001",
  "claimantName": "Jane Doe",
  "damageDate": "2025-11-10",
  "lossDescription": "Water damage in kitchen"
}
```

Expected Status: 201 Created

Expected Response Example / Schema:

```
{
  "id": "c001",
  "policyNumber": "P-10001",
  "claimantName": "Jane Doe",
  "damageDate": "2025-11-10",
  "lossDescription": "Water damage in kitchen",
  "images": [],
  "status": "OPEN",
  "createdAt": "2025-11-10T12:00:00Z",
  "updatedAt": "2025-11-10T12:00:00Z"
}
```

Validation Points:

• Response status is 201 Created.

• id is autogenerated and a non-empty string.

• status defaults to OPEN.

• createdAt and updatedAt are ISO-8601 date-time strings.

• Response conforms to Claim schema (all required fields present).

TC2 - Missing Required Field (policyNumber)

Endpoint: POST /claims

Submit claim creation request without policyNumber to trigger validation error.

Preconditions:

• Authenticated user.

Request Example:

```
{
  "claimantName": "John Doe",
  "damageDate": "2025-11-10",
  "lossDescription": "Car accident"
}
```

Expected Status: 400 Bad Request

Expected Response Example / Schema:

```
{
  "code": "BAD_REQUEST",
  "message": "Missing required field: policyNumber",
  "details": null
}
```

Validation Points:

• No claim record is created.

• Error response follows Error schema (code, message).

• The error message clearly identifies the missing field.

TC3 - Invalid damageDate Format

Endpoint: POST /claims

Provide damageDate in wrong format to validate date parsing.

Preconditions:

• Authenticated user.

Request Example:

```
{
  "policyNumber": "P-10002",
  "claimantName": "Jane Smith",
  "damageDate": "11-10-2025",
  "lossDescription": "Glass door broken"
}
```

Expected Status:400 Bad Request

Expected Response Example / Schema:

```
{
  "code": "BAD_REQUEST",
  "message": "Invalid date format for damageDate. Expected YYYY-MM-DD",
  "details": null
}
```

Validation Points:

• API validates 'damageDate' against date format (YYYY-MM-DD).

• No record created on invalid date format.

• Error uses Error schema.

TC4 - Retrieve Claim by ID

Endpoint: GET /claims/{id}

Fetch an existing claim by its ID and validate the full response.

Preconditions:

• A claim with id 'c001' exists.

• Authenticated user.

Expected Status:200 OK

Expected Response Example / Schema:

```
{
  "id": "c001",
  "policyNumber": "P-10001",
  "claimantName": "Jane Doe",
  "damageDate": "2025-11-10",
  "lossDescription": "Water damage in kitchen",
  "images": [],
  "status": "OPEN",
  "createdAt": "2025-11-10T12:00:00Z",
  "updatedAt": "2025-11-10T12:00:00Z"
}
```


Validation Points:

• Returned id matches requested id.

• All required fields per Claim schema are present and types match.

• Timestamps are valid ISO-8601 date-time strings.

TC-005 - Update Claim Status Step-by-Step

Endpoint: PATCH /claims/{id}

Perform sequential status updates: OPEN -> IN_REVIEW -> APPROVED -> PAID.

Preconditions:

• Claim exists with status OPEN.

• Authenticated user.

Request Example:

{ "status": "IN_REVIEW" }

Expected Status:200 OK

Expected Response Example / Schema:


```
{
  "id": "c001",
  "status": "IN_REVIEW",
  "...": "other claim fields..."
}
```

Validation Points:

• Each PATCH returns 200 OK and the updated claim object.

• Status transitions are validated (disallowed transitions return 400).

• updatedAt timestamp changes after each update.

TC6 - List All Claims (Pagination)

Endpoint: GET /claims?page=1&pageSize;=10

Retrieve paginated list of claims with default sorting.

Preconditions:

• Authenticated user.

• Multiple claims exist.

Expected Status:200 OK

Expected Response Example / Schema:

```
[
  { "id": "c001", "policyNumber": "P-10001", "status": "OPEN", "createdAt":
"2025-11-10T12:00:00Z"
  { "id": "c002", "policyNumber": "P-10002", "status": "APPROVED", "createdAt":
"2025-11-09T09:00:
]
```

Validation Points:

• Response is an array of Claim objects.

• Header 'X-Total-Count' returns total matching records as integer.

• page and pageSize parameters validated (page>=1, 1<=pageSize<=100).

• Invalid pagination (pageSize=0 or >100) returns 400, 0

TC7 - Filter Claims by Status

Endpoint: GET /claims?status=APPROVED

Return only claims with status APPROVED.

Preconditions:

• Authenticated user.
• There are approved claims in the system.

Expected Status:200 OK

Expected Response Example / Schema:

```
[
  { "id": "c002", "policyNumber": "P-10002", "status": "APPROVED", "createdAt":
"2025-11-09T09:00:
]
```

Validation Points:

• All returned items have status == APPROVED.

• Unknown status value (e.g., INVALID) returns 400 Bad Requests.

TC8 - Check Health Endpoint

Endpoint: GET /health

API health check endpoint returns basic status.

Expected Status:200 OK

Expected Response Example / Schema:{ "status": "ok" }

Validation Points:

• Response is { "status": "ok" } and status code 200.

10. Negative & Validation Scenarios

Invalid login credentials

Action: POST /auth/login with wrong password

Expected: 401 Unauthorized

```
{ "code":"UNAUTHORIZED", "message":"Invalid username or password" }
```

Expired token

Action: Use expired Bearer token on any secured endpoint

Expected: 401 Unauthorized

```
{ "code":"UNAUTHORIZED", "message":"Token expired" }
```

Malformed token

Action: Use 'Bearer randomstring' on secured endpoint

Expected: 401 Unauthorized

```
{ "code":"UNAUTHORIZED", "message":"Missing or invalid Bearer token." }
```

Claim not found

Action: GET /claims/c99999

Expected: 404 Not Found

```
{ "code":"NOT_FOUND", "message":"Claim with ID 'c99999' not found." }
```

Illegal status change

Action: PATCH /claims/{id} with status OPEN (from PAID)

Expected: 400 Bad Request

```
{ "code":"BAD_REQUEST", "message":"Invalid status transition from PAID to
OPEN." }
```

Invalid claim ID format

Action: GET /claims/123

Expected: 400 Bad Request

```
{ "code":"BAD_REQUEST", "message":"Invalid claim ID format. Expected string
pattern: c[0-9]+" }
```

11. Error Code Standards

Invalid claim ID format

HTTP Code: 400 Bad Request

Example message: Invalid claim ID format. Expected string pattern: c[0-9]+

Claim not found

HTTP Code: 404 Not Found

Example message: Claim with ID 'c99999' not found.

Duplicate Claim

HTTP Code: 409 Conflict

Example message: Similar claim for this policyNumber 'P-10001' already exists.

If we have the same description, damage date, policy number for a claim, the response should return a 409

Invalid date format

HTTP Code: 400 Bad Request

Example message: Invalid date format for damageDate. Expected YYYY-MM-DD.

Missing token

HTTP Code: 401 Unauthorized

Example message: Missing or invalid Bearer token.

12. Appendix: Schemas Claim (required fields shown):

```
{
  "id": "string",
  "policyNumber": "string",
  "claimantName": "string",
  "damageDate": "YYYY-MM-DD (date)",
  "lossDescription": "string",
  "images": ["uri", "..."],
  "status": "OPEN | IN_REVIEW | APPROVED | PAID",
  "createdAt": "ISO-8601 date-time",
  "updatedAt": "ISO-8601 date-time"
}
```

Error schema:

```
{
  "code": "string (e.g., BAD_REQUEST, NOT_FOUND)",
  "message": "string",
  "details": { "fieldErrors": {...} } | null
}
```