Epic Test Quest – AI Intern Task

Test Case Retrieval for RAG-Assisted Generation (2–4 hours)

Goal

We want to generate better test cases from user prompts, not from scratch, but by retrieving good examples first.

In this task, you'll simulate a **mini RAG pipeline** that takes a user prompt, retrieves relevant test cases from a small dataset, and (optionally) uses them as context to generate new ones.

You'll Receive

- **10 test cases** (structured in natural language).
- A list of **3 user prompts**

Your Task

1. **Chunk & Embed the Test Cases**
   - Load and chunk the test cases
   - Embed them using any model of your choice (e.g., all-MiniLM, sentence-transformers, etc.).
2. **Set Up Retrieval**
   - For each prompt, retrieve the **top 2–3 most relevant test cases** from the embedded set.
3. **(Optional but encouraged)**
   - Use those retrieved examples as context in an open-source LLM (e.g., HuggingFace pipeline) to generate new test cases matching the user prompt.
4. **Explain Your Process**
   - Short README or comments describing:
     - What tools you used (and why)

        - How you chunked the input
        - How you evaluated what was “relevant”

Time Limit

We don’t expect polish, just working code and clarity of thought.
Timebox: 2–4 hours max.

What to Submit

Send an email to christine@epictestquest.com within 2 days of receiving this task:

- Python script or Jupyter notebook
- README or inline explanation
  For each prompt:
    - The retrieved test cases
    - (If done) the generated test cases

You can share via GitHub, Colab, or as a zipped folder.

Questions?

Feel free to reach out if anything’s unclear,we want to see how you think, not trick you.

User Prompt:

Write test cases for checking out with a saved credit card as a logged-in user.

Generate test cases for scenarios involving invalid payment methods during checkout.

Create test cases for handling server/API errors during checkout (e.g., 500 Internal Server Error on order submission, timeout during payment gateway call). Include edge case logic and how the frontend should respond. Format as Title, Steps, Expected Result, with detailed technical coverage.

Test Case:

Title: Logged-in user completes checkout with valid credit card

Steps:

1. Navigate to product page and add an in-stock item to the cart

2. Proceed to checkout

3. Confirm shipping address and choose standard delivery

4. Select “Credit Card” as payment method

5. Enter valid credit card details (e.g., Visa ending in 4242)

6. Submit order

Expected Result: Order is successfully placed. Confirmation page is displayed with order number and summary. Confirmation email is triggered. Payment transaction is logged in backend.

Title: Server returns 500 error during final checkout submission

Steps:

1. Add product to cart and proceed to checkout

2. Fill in valid shipping and payment details

3. Submit order (API returns HTTP 500 at this step)

Expected Result: Frontend displays generic error message: 'Something went wrong, please try again'. No charge is made to the card. No order is created in the system. Option to retry is presented.

Title: Guest user completes order without creating an account

Steps:

1. Add product to cart

2. Choose 'Checkout as Guest'

3. Enter shipping and payment details

4. Submit order

Expected Result: Order confirmation page is displayed. Guest receives confirmation email. No user account is created.

Title: Reject invalid credit card entry

Steps:

1. Add item to cart and proceed to checkout

2. Enter a clearly invalid credit card number (e.g., 0000 0000 0000 0000)

3. Attempt to submit order

Expected Result: User sees an inline validation error. Cannot proceed until valid card is entered.

Title: Handle inventory mismatch during order submission

Steps:

1. Add a low-stock item (only 1 left) to cart

2. Another user purchases the item in the meantime

3. Attempt to complete checkout

Expected Result: Error message 'Item no longer available' is displayed. User is returned to cart to revise order. No payment is processed.

Title: Expired promo code rejected at checkout

Steps:

1. Add product to cart

2. Enter expired promo code (e.g., SALE2020)

3. Click 'Apply'

Expected Result: Message: 'Discount code expired'. No discount applied.

Title: User exits checkout before entering payment

Steps:

1. Add item to cart

2. Begin checkout

3. Navigate away before reaching payment

Expected Result: No order placed. Cart remains unchanged.

Title: Valid discount code applies correctly

Steps:

1. Add item to cart

2. Apply valid discount code

Expected Result: Discount applied to total.

Title: Successful PayPal checkout

Steps:

1. Add item to cart

2. Select PayPal and authenticate

3. Return and confirm order

Expected Result: Order placed via PayPal.

Title: Form validation prevents checkout with missing address

Steps:

1. Start checkout

2. Leave required address fields empty

3. Try to proceed

Expected Result: User is prompted to complete form.
