

Customer Success Engineer Implementation Exercise

Role: Customer Success Engineer at Postman

Timeline: 4-5 hours (use AI assistants)

Presentation: 30 minutes (20-min presentation + 10-min Q&A)

Why This Exercise Matters: Show how Postman creates fast engineering value. Reduce the time to discover and successfully call an API, and make the pattern repeatable across domains.

The Situation

Customer Profile: Large financial services company, \$480K ARR, 2,000 seats

Current State (key facts):

- 72% seat adoption (1,440 active users), but low engineering value
- 413 shared workspaces; inconsistent structure and ownership
- 2,918 collections, mostly ad-hoc single requests
- 2-4 hours to discover and successfully call internal APIs
- 89% of collections have only basic status checks

Your Domain: Payment Processing

- 15 REST APIs, 14-person engineering team
- Critical workflow: Payment authorization > capture > refund
- All services on AWS (API Gateway to Lambda/ECS)
- Auth: OAuth 2.0 (external partners), JWT (internal calls)

Baseline Problem: A senior engineer spent 47 minutes integrating the refund API, jumping across 6 systems, hitting 3 dead ends, and relying on a personal workspace example.

Your Mission: Transform Postman from a compliance checkbox into an engineering accelerator. Demonstrate measurable value within 90 days to secure renewal and expansion.

Your Assignment

Create a proof-of-concept that reduces API discovery from 47 minutes to seconds, and show how the pattern scales to all 47 APIs.

Deliverable 1: Adoption Starter Kit (Working Code)

Build a working pattern for one API (Payment Refund API) that others can reuse:

Required Components:

- Script (Python/Node.js/Bash) OR GitHub Actions workflow that:
 - Creates spec in Postman Spec Hub (`POST /specs`)
 - Generates a collection from the spec (`POST /specs/{specId}/generations/collection`)
 - Sets Dev/QA/UAT/Prod environments (base URLs)
 - Implements a JWT pre-request script
- Generated Postman collection (from your script/workflow)
- README with:
 - **Business value** (problem and outcome)
 - **ROI calculation** (show your math)
 - **Scaling strategy** (apply pattern to remaining 46 APIs)
 - **Workspace consolidation** (migration plan and governance)

Deliverable 2: Value Presentation (Slides + Demo)

20 minutes to demonstrate business impact:

Required Sections:

1. **Problem Baseline:** The 47-minute discovery and its cost
2. **Solution Overview:** Your ingestion workflow (Spec Hub > Collection > Environments)
3. **Live Demo:** Run the script/workflow and show the resulting workspace structure
4. **ROI Projection:** Annual savings (14 engineers × hours saved/week × \$150/hr × 52 weeks)
5. **Scaling Roadmap:** Extend to all 15 APIs, then other domains
6. **90-Day Metrics:**
 - Discovery time (~1 hour to X sec)
 - Test coverage (11% to X%)
 - Workspace rationalization (413 to X)
 - CI/CD (static files to automated sync)

Evaluation Criteria

Criteria	Baseline (Good)	Strong (Great)	Exceptional
Technical Execution (30%)	Script/workflow imports spec and generates collection	Also supports PATCH updates and syncs linked collections	Automates spec retrieval (e.g., AWS CLI) and robust sync

Criteria	Baseline (Good)	Strong (Great)	Exceptional
Value Articulation (30%)	Basic time-savings explanation	Quantified ROI with clear metrics, exec-friendly narrative	Connects impact to renewal/expansion decision (\$480K context)
Pattern Thinking (25%)	Works for 1 API; mentions scaling	Detailed scaling approach with timeline	Repeatable framework and training plan; Domain 2 is faster
Co-Execution (15%)	Shows what you built	Explains enablement and checkpoints	Ensures independent ownership by Day 90

What Distinguishes Exceptional Candidates:

- Proactively addresses unasked problems (workspace governance, spec versioning, auth rotation)
- Executive-level narrative (connects 30-second discovery to quantified productivity gains)
- Customer empathy (builds for their constraints and success)
- Anticipates edge cases (spec changes, breaking updates, regen conflicts)

Provided Materials

Sample OpenAPI Specification:

- File: `payment-refund-api-openapi.yaml` (complete Payment Refund API spec)
- Endpoints: POST /refunds, GET /refunds, GET /refunds/{id}, POST /refunds/{id}/cancel, GET /health

Documentation & References:

- See `RESOURCES.md` for full Postman API docs and examples
- Learning Center: <https://learning.postman.com/>
- Request an API key from the hiring team

Key Postman API Endpoints:

- Create spec: POST
`https://api.getpostman.com/specs?workspaceId={{workspaceId}}`
- Generate collection: POST
`https://api.getpostman.com/specs/{{specId}}/generations/collection`

Submission Requirements

Format:

- Presentation deck (PDF or PPTX)
- Code repository (GitHub link) containing:
 - Ingestion script or GitHub Actions YAML
 - Generated Postman collection (JSON export)
 - README.md with business case and scaling strategy
 - Any configuration or environment setup instructions

Time Guidance:

- Phase 1 (working code): 2-3 hours (use AI to accelerate)
- Phase 2 (presentation): 1-2 hours (focus on value storytelling)
- Practice/refinement: 1 hour

AI Assistance Policy: Strongly encouraged. We want your strategy and communication, not manual coding. Use Claude Code, Cursor, Gemini CLI, or ChatGPT to accelerate.

What Success Looks Like

Good submission: Code works and demonstrates the core workflow. You explain what you built and cover the required sections.

Great submission: Clean structure; clear rationale and trade-offs; customer empathy; detailed, logical scaling strategy; strong communication linking tech to business.

Exceptional submission: Production-quality thinking (automation patterns); compelling story tied to quantified ROI; proactive edge-case handling; governance for preventing future sprawl; integrates components into a cohesive system; crystal-clear CSE communication and enablement focus.