Lensy 24-Hour Prototype: Documentation Quality Auditor

What We're Building

A web-based tool that analyzes developer documentation URLs and produces quality reports with specific, actionable recommendations across five dimensions: **Relevance, Freshness, Clarity, Accuracy, and Completeness.**

Why This Matters

Generic documentation audits identify problems without solutions. Lensy provides specific fixes: "Replace `auth.login()` with `auth.authenticate()` - current code breaks in v3.0+" instead of "content needs updating."

Technical Stack

- **Frontend**: React (simple, demo-ready UI)
- **Backend**: AWS Lambda + Bedrock
- **Storage**: None needed for prototype
- **Input**: Documentation URL
- **Output**: Web dashboard with scored dimensions + recommendations

User Flow

1. URL Input

User pastes documentation URL → System fetches content

2. Structure Confirmation

If multiple topics detected:

- Shows: "Found 3 topics: Authentication (3 pages), API Reference (12 pages), Getting Started (2 pages)"
- Asks: "Does this breakdown make sense?"
- User confirms or adjusts

If single topic:

- Shows: "This appears to be about: REST API Pagination"
- Asks: "Is this categorization accurate?"
- User confirms or corrects

3. Analysis with Progress Notes

```
✓ Fetching documentation...
✓ Content extracted
⏳ Analyzing Relevance...
  → Checking target audience alignment
  → Evaluating search intent match
✓ Relevance complete (Score: 7/10)

⏳ Analyzing Freshness...
  → Checking API version references
  → Scanning code for deprecated methods
  ⚠ Snippet 1: Uses deprecated auth.login() (removed in v3.0)
✓ Freshness complete (Score: 4/10)

⏳ Analyzing Clarity...
  → Assessing readability
  ⚠ 5 code snippets missing version specs
✓ Clarity complete (Score: 8/10)

⏳ Analyzing Accuracy...
  → Validating 7 code snippets
  × Snippet 3: Syntax error line 3
  ✓ Snippet validation complete
✓ Accuracy complete (Score: 6/10)

⏳ Analyzing Completeness...
  × Failed: 404 errors on 5 linked pages
  → Skipping this dimension

✓ Analysis complete (4/5 dimensions)
```

4. Results Dashboard

Overall Score: 6.8/10 (based on 4 of 5 dimensions)

Dimension Cards:

- ✓ Relevance: 7/10
- ✓ Freshness: 4/10
- ✓ Clarity: 8/10
- ✓ Accuracy: 6/10

- ⚠ Completeness: Unable to analyze (linked pages inaccessible)

Actionable Recommendations (expandable by severity):

- **HIGH**: Fix syntax error in authentication example (line 45)
- **HIGH**: Replace deprecated `auth.login()` with `auth.authenticate()`
- **MEDIUM**: Add version badges to all 7 code examples
- **MEDIUM**: Specify "Compatible with v2.x - v2.8"
- **LOW**: Improve readability score from 8.2 to 9.0+

Five Quality Dimensions

Relevance	Target audience alignment, Search intent match and Content-to-need fit
Freshness	API version currency **Deprecated code detection** (e.g., methods removed in newer versions) **Version compatibility** (backward/forward compatibility issues) Date stamp validation
Clarity	• Readability metrics • Technical explanation depth • Code example quality • **Missing version specifications** in code examples • Compatibility notes presence
Accuracy	• **Code syntax validation** (can it be copy-pasted and run?) • **Function signatures correctness** • Factual claim verification • API endpoint documentation accuracy
Completeness	• Coverage gaps (missing parameters, edge cases) • Broken links

	• Missing code examples • Incomplete explanations

Code Verification (Distributed Across Dimensions)

Accuracy checks:

- Syntax errors
- Copy-paste viability
- Function signature correctness

Freshness checks:

- Deprecated methods
- Version compatibility (backward/forward)
- Breaking changes in newer versions

Clarity checks:

- Missing version specifications
- Absence of compatibility notes

Example output:

```
Code Snippet Analysis:
× Line 45: Syntax error (missing closing quote)
⚠ Line 78: Uses auth.login() - deprecated in v3.0
⚠ Line 112: No version specification
✓ Line 156: Valid, current syntax
```

Graceful Degradation

Core principle: Show what works, acknowledge what doesn't.

If 1+ dimensions succeed:

- Display report with available dimensions
- Mark failed dimensions with explanation
- Provide recommendations from successful analyses

If all dimensions fail:

- Show error report
- Explain what went wrong

- Don't show empty report

Partial result example:

```
Overall Score: 7.2/10 (3 of 5 dimensions)

✓ Relevance: 8/10
✓ Clarity: 9/10
⚠ Freshness: Unable to analyze
  → API changelog inaccessible
✓ Accuracy: 5/10 (syntax only)
⚠ Completeness: Partial (70% coverage)

Analysis Coverage: 60%
Confidence Level: Medium
```

Success Criteria (24 hours)

- Analyzes single documentation page/section
- Scores all 5 dimensions (or shows partial results gracefully)
- Provides 3-5 specific recommendations per dimension
- Code verification catches real syntax errors and deprecations
- Output quality suitable for sales conversations

Out of Scope (This Prototype)

- Continuous monitoring
- Full portal crawling
- CMS platform integration
- Database/persistence
- User authentication
- Multi-page batch analysis

Risk Mitigation

- **Bedrock API limits hit**: Cache responses, throttle requests
- **Poor analysis quality**: Focus on 2-3 dimensions done well vs. all 5 done poorly
- **Time pressure**: Deliver working Accuracy + Freshness (highest value) with stubs for others

Why This Validates Lensy

- Demonstrates core value: specific fixes, not generic "update this"
- Creates tangible asset for CMS Kickoff 2026 demo
- Proves technical capability with real documentation
- Tests value proposition with real users before full build