Fynd AI Intern - Take Home Assessment 2.0

Technical Report

Yashodip More

January 2026

Live Deployment:

User Dashboard: https://feedback-system-taupe.vercel.app

Admin Dashboard: https://feedback-system-taupe.vercel.app/admin

GitHub: github.com/yashodipmore/Fynd-AI-Intern-Take-Home-Assessment-2.0

1. Executive Summary

This assessment comprises two interconnected tasks focused on leveraging Large Language Models (LLMs) for customer feedback analysis. Task 1 explores prompt engineering techniques for rating prediction, while Task 2 implements a production-grade dual-dashboard feedback system with real-time AI-powered responses.

Key achievements include:

- 66.7% accuracy in rating prediction using Chain-of-Thought prompting
- 100% JSON validity with Zero-Shot approach for production reliability
- Fully deployed, mobile-responsive web application with real-time analytics
- Server-side LLM integration with comprehensive error handling

2. Task 1: Rating Prediction via Prompting

2.1 Objective

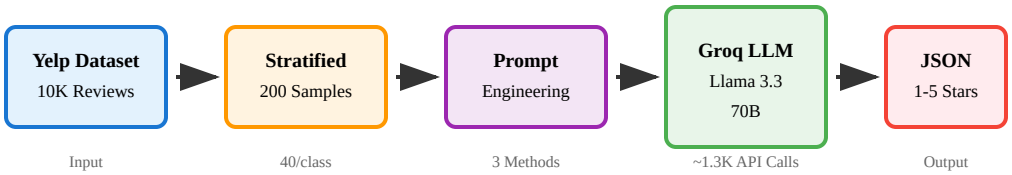
Design and evaluate prompt engineering strategies to classify Yelp reviews into 1-5 star ratings using an LLM, with structured JSON output for downstream processing.

2.2 Methodology

Dataset: 10,000 Yelp reviews with stratified sampling of 200 reviews (40 per star rating) for evaluation.

Model: Llama 3.3 70B Versatile via Groq API (chosen for fast inference and high quality).

LLM Classification Pipeline



2.3 Prompt Design Iterations

Iteration 1: Zero-Shot Direct

> **Design Philosophy:** Minimal instructions with clear rating scale definitions. Tests the model's inherent understanding without guidance.

```
Classify this Yelp review into 1-5 stars.
Rating guide: 1=Terrible, 2=Poor, 3=Average, 4=Good, 5=Excellent
Review: "{text}"
Respond ONLY with JSON: {"predicted_stars": <1-5>, "explanation": "<reason>"}
```

Iteration 2: Few-Shot with Sentiment Anchors

> **Improvement:** Added 5 calibrated examples (one per rating) plus sentiment anchor keywords to help the model understand rating boundaries.

```
Classify Yelp reviews into 1-5 stars. Examples:
1 star: "Worst ever. Cold food, rude staff." -> {"predicted_stars": 1, ...}
2 star: "Disappointing. Overcooked burger." -> {"predicted_stars": 2, ...}
3 star: "It was okay. Decent food." -> {"predicted_stars": 3, ...}
4 star: "Great pasta, friendly waiter." -> {"predicted_stars": 4, ...}
5 star: "Incredible! Best sushi ever!" -> {"predicted_stars": 5, ...}
Sentiment anchors: 1(terrible,worst) 2(disappointing) 3(okay) 4(great) 5(amazing)
```

Iteration 3: Chain-of-Thought (CoT)

> **Improvement:** Structured 4-step reasoning framework forcing the model to analyze sentiment, aspects, intensity before making a rating decision.

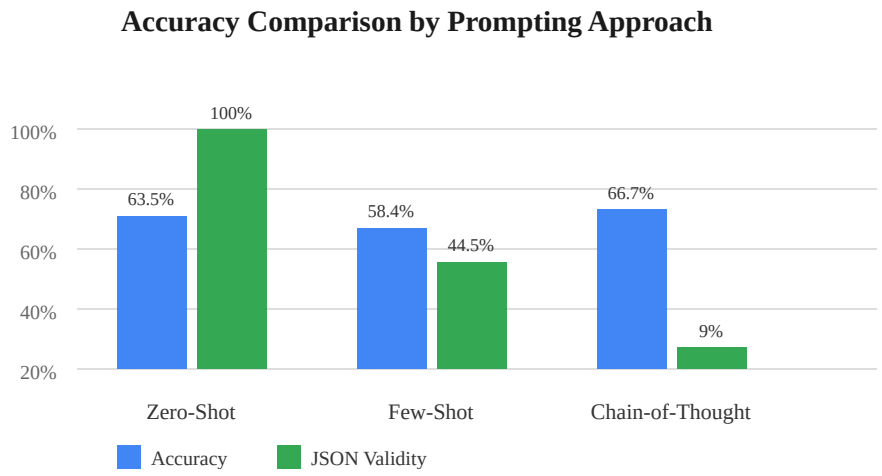
```
Analyze this Yelp review step-by-step:
STEP 1: Overall sentiment (Positive/Negative/Mixed)?
```

```
STEP 2: Aspects mentioned - Food, Service, Value, Atmosphere?
STEP 3: Intensity signals - Superlatives? Strong emotions?
STEP 4: Rating decision based on analysis
```

2.4 API Usage Statistics

> **Total Groq API Requests:** ~1,300 requests for Task 1 evaluation
> **Model:** Llama 3.3 70B Versatile
> **Dataset:** 200 reviews × 3 prompting approaches + iterations = ~1.3K calls

2.5 Evaluation Results



Metric	Zero-Shot	Few-Shot	Chain-of-Thought
JSON Validity	**100%**	44.5%	9%
Exact Accuracy	63.5%	58.4%	**66.7%**
Within ±1 Star	99%	98.9%	**100%**
Mean Absolute Error	0.38	0.43	**0.33**

2.6 Per-Class Accuracy

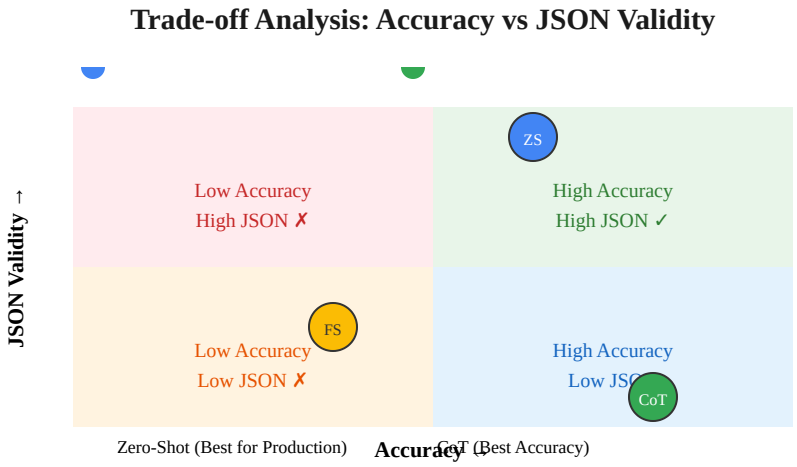
Per-Class Accuracy Heatmap

	Zero-Shot	Few-Shot	CoT
1 Star	87.5%	76.9%	100%
2 Star	47.5%	50%	50%
3 Star	45%	27.3%	60%
4 Star	40%	46.2%	25%
5 Star	97.5%	95.2%	100%

Green = High accuracy (>70%) | Yellow = Medium (40-70%) | Red = Low (<40%)

Star Rating	Zero-Shot	Few-Shot	Chain-of-Thought
1 Star	87.5%	76.9%	**100%**
2 Star	47.5%	**50%**	50%
3 Star	45%	27.3%	**60%**
4 Star	40%	**46.2%**	25%
5 Star	**97.5%**	95.2%	**100%**

2.6 Key Findings



1. **JSON Validity vs Accuracy Trade-off:** Simpler prompts produce reliable structured output, while complex reasoning prompts improve accuracy but break format compliance.

2. **Extreme Ratings are Easier:** 1-star and 5-star reviews achieve 87-100% accuracy due to strong sentiment signals. Middle ratings (2-4 stars) are harder with mixed/subtle sentiments.
3. **Within ±1 Accuracy is Excellent:** All approaches achieve 99-100%, indicating the model rarely makes large prediction errors.

2.7 Recommendations

Use Case	Recommended Approach	Reason
Production (High Volume)	Zero-Shot	100% JSON validity, acceptable accuracy
Accuracy Critical	CoT + Post-processing	Best accuracy, extract JSON via regex

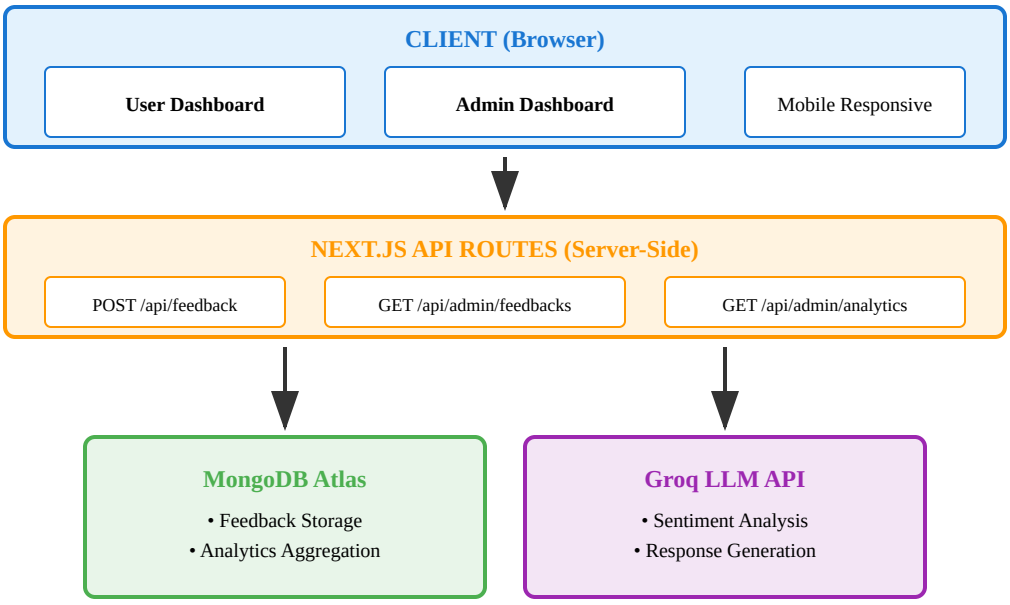
3. Task 2: Two-Dashboard AI Feedback System

3.1 System Overview

A production-grade web application featuring two dashboards: a public-facing User Dashboard for feedback submission with AI-generated responses, and an internal Admin Dashboard for analytics and feedback management.

3.2 Architecture

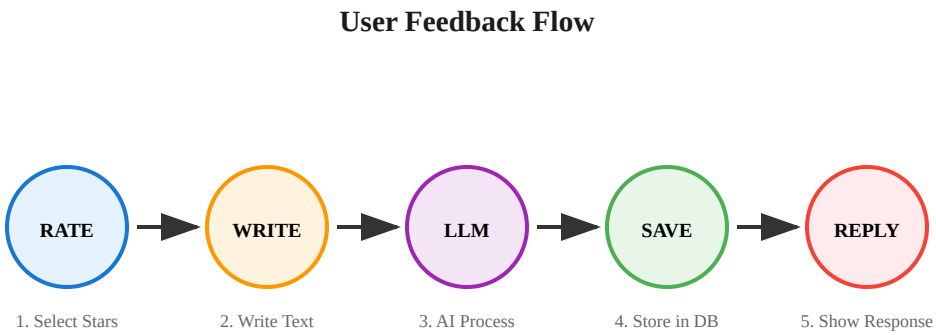
System Architecture Diagram



3.3 Technology Stack

Layer	Technology	Justification
Frontend	Next.js 14, React 18, TypeScript	Server-side rendering, type safety, modern DX
Styling	Tailwind CSS, Framer Motion	Rapid development, smooth animations
Charts	Recharts	React-native charting with responsiveness
Database	MongoDB Atlas	Flexible schema, cloud-hosted, free tier
LLM	Groq (Llama 3.3 70B)	Fast inference, high quality responses
Deployment	Vercel	Zero-config Next.js hosting, auto-scaling

3.4 Design Decisions



Server-Side LLM Calls: All LLM interactions occur on the server via API routes. This ensures API keys are never exposed to the client and allows for request validation, rate limiting, and error handling.

Structured LLM Output: The system prompts the LLM to return JSON with specific fields (sentiment, summary, actions, userResponse), enabling consistent parsing and storage.

Graceful Degradation: If the LLM API fails, the system returns pre-defined fallback responses based on the rating, ensuring users always receive feedback.

3.5 Edge Cases Handled

Scenario	Handling Strategy
Empty review submitted	AI generates response based on rating alone
Very long review (>2000 chars)	Truncated before sending to LLM
LLM API timeout/failure	Fallback response returned, feedback still saved
Invalid rating (not 1-5)	Server-side validation with error message
Database connection error	Proper error response with retry option
Invalid JSON from LLM	Regex extraction fallback for critical fields

3.6 API Design

POST /api/feedback - Submit user feedback

```
Request:  { "rating": 4, "review": "Great service!" }
Response: { "success": true, "data": { "userResponse": "Thank you...", ... } }
```

GET /api/admin/feedbacks - Fetch feedbacks with filters

```
Query: ?page=1&limit=10&rating=5&sentiment=positive
Response: { "success": true, "data": { "feedbacks": [...], "pagination": {...} } }
```

GET /api/admin/analytics - Dashboard statistics

```
Response: { "totalFeedbacks": 150, "averageRating": 4.2, "sentimentDistribution":
[...] }
```

3.7 Trade-offs and Limitations

Trade-off	Decision Made	Rationale
Response latency vs quality	Prioritized quality (Llama 3.3 70B)	Better user experience despite ~1-2s latency
Real-time vs polling	30-second polling for admin	Simpler than WebSocket, acceptable for dashboard
Authentication	Not implemented	Out of scope for assessment, but noted as limitation

Current Limitations:

- No user authentication (admin dashboard publicly accessible)
- No rate limiting on feedback submission
- Single LLM provider (no failover to alternative)
- Analytics limited to 7-day trend view

3.8 Future Improvements

- Add authentication for admin dashboard (NextAuth.js)
- Implement rate limiting per IP address
- Add LLM provider failover (Groq → OpenAI → Anthropic)
- Export analytics to PDF/Excel
- Real-time updates via WebSocket for admin

4. Conclusion

This assessment demonstrates practical application of LLM prompt engineering for classification tasks and production system design for AI-powered feedback management. Key learnings include:

1. Prompt complexity inversely correlates with output format reliability

2. Production systems require robust error handling and fallback mechanisms
3. Server-side LLM integration is essential for security and control
4. Mobile responsiveness is critical for modern web applications

The complete codebase and deployment demonstrate a production-ready approach to AI-powered customer feedback systems.

> **Repository:** github.com/yashodipmore/Fynd-AI-Intern-Take-Home-Assessment-2.0
> **User Dashboard:** https://feedback-system-taupe.vercel.app
> **Admin Dashboard:** https://feedback-system-taupe.vercel.app/admin