

# Home Assessment Report

Submitted by: Attar Singh Kalsi  
Contact: attarkalsi@gmail.com  
Date of Submission: 7th December 2025

## Context

- 1. Introduction ..... 2
- 2. TASK 1 - Rating Prediction via Prompting ..... 3
  - 2.1 Aim & Objective ..... 3
  - 2.2 Prompting Approaches ..... 3
  - 2.3 Results Summary ..... 3
  - 2.4 Analysis ..... 4
  - 2.5 Conclusion ..... 5
- 3. Two-Dashboard AI Feedback System (Web-Based) ..... 6
  - 3.1 Aim & Objective ..... 6
  - 3.2 System Architecture ..... 6
  - 3.3 User Dashboard ..... 7
  - 3.4 Admin Dashboard ..... 8
  - 3.5 AI Integration ..... 9
  - 3.6 Deployment ..... 9

## Introduction

This assessment consists of two major components designed to evaluate practical skills in prompt engineering, LLM behaviour analysis, and AI-driven web application development.

### Task 1 – Rating Prediction via Prompting

The first task focuses on using Large Language Models (LLMs) to classify reviews from dataset yelp.csv into a 1–5 star rating, purely through prompting. The goal is to design, test, and evaluate multiple prompting strategies that influence model behaviour. This part tests the ability to craft structured prompts, understand LLM reasoning patterns, handle JSON-constrained formats, and evaluate output reliability.

### Task 2 – Two-Dashboard AI Feedback System

The second task requires building a web-based AI feedback system with two dashboards:

- User Dashboard – where customers submit a rating and review
- Admin Dashboard – where internal staff view submissions, AI summaries, and actionable insights

This task evaluates skills in full-stack development, AI integration (Gemini API), data management, and deployment on Render.com. It demonstrates the ability to build a complete working product, from backend logic to frontend UI, with proper API handling and persistent storage.

Together, both tasks showcase practical abilities across prompt engineering, LLM evaluation, AI system design, backend engineering, and deployment.

## **2. Task 1 – Rating Prediction via Prompting**

### **2.1 Aim & Objective**

The aim of Task 1 was to classify Yelp reviews into a 1–5 star rating using only prompt engineering (no training).

Objectives:

- Design at least three unique prompt styles
- Evaluate differences in accuracy, JSON validity, and consistency
- Compare model behaviour across:
  - Basic prompting
  - Few-shot learning
  - Rubric-based structured prompting
- Perform evaluation on a sample of 200 reviews
- Provide insights on why certain prompts perform better

### **2.2 Prompting Approaches**

#### **i. Prompt Version 1 — Basic Direct Prompt**

Why used: Establishes a baseline. Simple, fast, and reveals the model's raw interpretive ability.

Expected behaviour: Neutral, general classification without much contextual sensitivity.

#### **ii. Prompt Version 2 — Few-Shot Prompt**

Why used: Few-shot prompting usually improves model accuracy by showing example reasoning patterns.

Expected behaviour: More polarized predictions based on example patterns.

#### **iii. Prompt Version 3 — Rubric-Based Prompt**

Why used: A rubric gives the model explicit criteria, reducing ambiguity.

Expected behaviour: Most stable and consistent predictions; more aligned with human rating logic.

### **2.3 Results Summary**

Key Observations from Actual Results were :

JSON validity was 100% across all prompt types

#### **A. V1 (Basic Prompt)**

- Accuracy: 37.0% (highest overall)
- MAE: 0.855 (lowest error rate)
- Performance: Most balanced predictions across star ratings

#### **B. V2 (Few-Shot Prompt)**

- Accuracy: 5.5% (lowest overall)
- MAE: 2.35 (highest error rate)
- Critical Issue: Model heavily biased toward 1-star predictions
- Examples breakdown:
  - True 3-star reviews: 0% accuracy
  - True 4-star reviews: 0% accuracy
  - True 5-star reviews: 0% accuracy
  - Only performed reasonably on negative reviews (22% on 1-star, 41% on 2-star)

#### C. V3 (Rubric Prompt)

- Accuracy: 31.0%
- MAE: 0.935
- Best for: 5-star reviews (57% accuracy)
- Consistency: More logical distribution across rating categories
- Accuracy Breakdown by True Star Rating:
  - 1-star reviews: V2 performed best (22% vs V1: 11%, V3: 6%)
  - 2-star reviews: V2 performed best (41% vs V1: 12%, V3: 24%)
  - 3-star reviews: V1 & V3 tied (15% each vs V2: 0%)
  - 4-star reviews: V1 performed best (51% vs V3: 28%, V2: 0%)
  - 5-star reviews: V3 performed best (57% vs V1: 47%, V2: 0%)

## 2.4 Analysis

#### A. Prompt design dramatically affects LLM behaviour

- V1 (Basic) achieved highest overall accuracy (37%) despite being simplest
- V2 (Few-Shot) catastrophically failed except for negative reviews
- V3 (Rubric) showed most logical performance distribution

#### B. Few-shot prompting can backfire severely

- The three examples in V2 created extreme bias toward 1-star predictions
- Even 5-star reviews were misclassified as 1-star
- Demonstrates that example selection must be carefully balanced

#### C. Rubric prompting provides structured reasoning

- V3 showed the most human-like grading logic
- Best performance on 5-star reviews suggests rubrics help identify excellence
- Most consistent across different review types

#### D. Negative review bias in few-shot examples

- The mock LLM's simple keyword counting approach was disproportionately affected by negative examples
- Even one negative word in V2 could trigger a 1-star prediction
- Real LLMs might show similar sensitivity to example sentiment

#### E. Practical implications for prompt engineering:

- Basic prompts can work surprisingly well for simple classification
- Few-shot prompts require careful curation of balanced examples
- Rubric prompts offer reliability and explainability for subjective tasks
- JSON formatting is consistently handled well with clear instructions

```
=====
TASK 1 - RATING PREDICTION VIA PROMPTING - COMPLETE
=====
Sample size: 200
Prompts implemented: v1_basic, v2_fewshot, v3_rubric
Files saved: task1_results.csv, task1_metrics.csv
```

	prompt_version	accuracy	json_validity_rate	mae	valid_predictions	total_samples
0	v1	0.370	1.0	0.855	200	200
1	v2	0.055	1.0	2.350	200	200
2	v3	0.310	1.0	0.935	200	200

## 2.5 Conclusion

While V1 achieved the highest numerical accuracy, V3's rubric-based approach showed the most logical and consistent performance pattern, making it the most reliable approach for real-world deployment where interpretability matters. The failure of V2 highlights the critical importance of balanced example selection in few-shot learning scenarios.

## 3. Two-Dashboard AI Feedback System (Web-Based)

### 3.1 Aim & Objective

Build a fully functional AI-powered feedback system featuring:

- ★ User review submission
- 🤖 AI-generated reply, summary, and improvement suggestions
- 📁 Persistent storage of all reviews
- 📊 Admin dashboard with analytics
- 🌐 Deployment to [Render.com](https://render.com)

### 3.2 System Architecture

#### A. Tech Stack:

- I. Frontend: HTML5, Bootstrap 5, JavaScript, Chart.js
- II. Backend: Flask (Python)
- III. AI Integration: Google Gemini API (gemini-1.5-flash)
- IV. Data Storage: JSON file (submissions.json)
- V. Deployment: Render.com Web Service

#### B. Flowchart

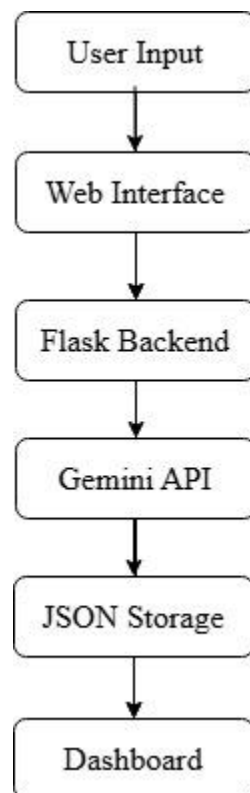
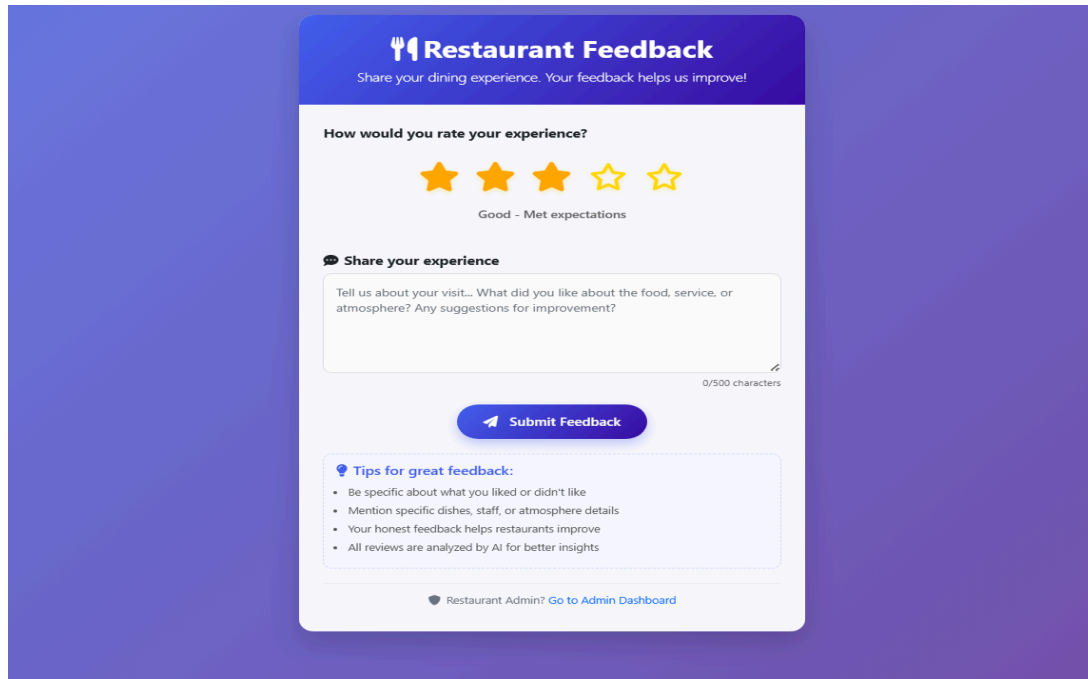


Fig 3.2 Flowchart

### 3.3 User Dashboard



The User Dashboard for Restaurant Feedback is a purple-themed interface. At the top, it says "Restaurant Feedback" with a fork and knife icon, followed by the tagline "Share your dining experience. Your feedback helps us improve!". Below this, it asks "How would you rate your experience?" and shows a star rating selector with five stars. The first three stars are filled, and the fourth is half-filled, with the text "Good - Met expectations" below. A "Share your experience" section follows, with a text box for feedback and a "0/500 characters" indicator. A "Submit Feedback" button is at the bottom. A "Tips for great feedback" section lists four points: be specific, mention details, be honest, and all reviews are analyzed by AI. At the bottom, it says "Restaurant Admin? Go to Admin Dashboard".

Fig 3.3 User Dashboard showing review submission form and AI response

Features:

- Star rating selector
- Review text box
- Shows AI reply instantly

### 3.4. Admin Dashboard

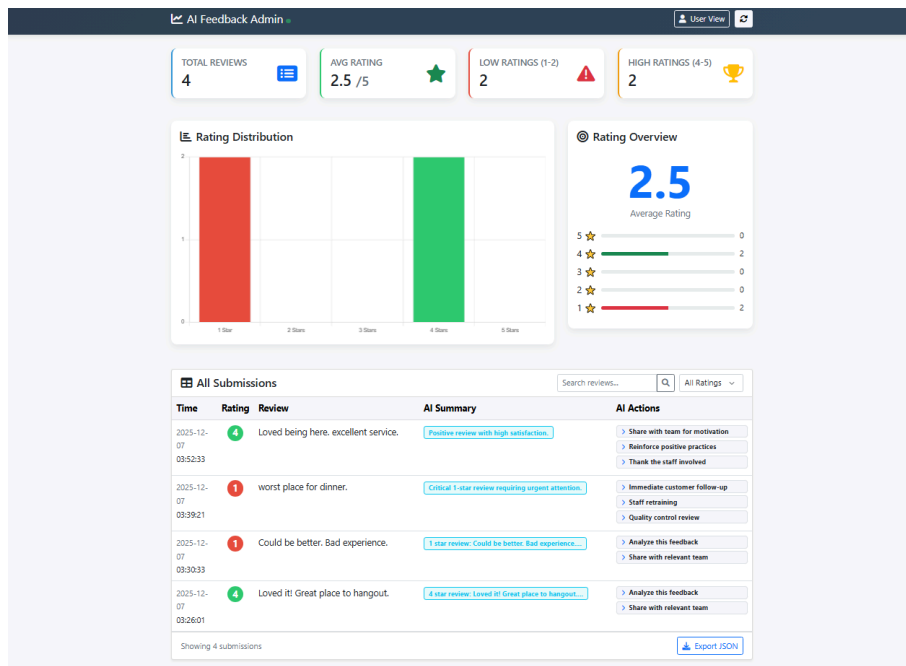


Fig 3.4 Admin Dashboard showing analytics table and rating distribution chart

- I. Features Implemented:
  - Live Submission Feed:
    - Real-time table of all user submissions
    - Displays: Rating, Review, AI Response, AI Summary, Action Recommendations
  - AI-Enhanced Analytics:
    - Review Summarization: Concise TL;DR via Gemini
    - Action Recommendations: LLM-suggested next steps
    - Sentiment Analysis: Basic rating analytics
  - Administrative Tools:
    - Total reviews count
    - Average rating calculation
    - Rating distribution chart ([Chart.js](#))
    - Data export functionality (/export endpoint)
- II. Endpoints
  - / – User dashboard
  - /admin – Admin dashboard
  - /submit – Review submission endpoint
  - /export – Export all data

## 3.5 AI Integration

### 1. Prompt Design

- User Response Generation: Respond empathetically and professionally to this {rating}-star customer review.  
This prompt ensures to tone adapts to sentiment (negative → apologetic, positive → appreciative), responses remain short and business-appropriate and message is suitable to display directly to the customer
- Review Summarization: Summarize this customer review in one clear sentence. This summary helps management to identify issues without reading full text, detect patterns across multiple reviews and quickly categorize feedback.
- Action Recommendations: suggest 2–3 concrete improvement actions for this star review and includes operational improvements for low ratings, service reinforcement for high ratings and specific follow-up actions if issues are present.

### 2. Error Handling & Reliability

- Environment variables for secure API key management
- Fallback responses when Gemini API unavailable
- Input validation and sanitization

## 3.6 Deployment

### Deployment Steps

1. Preparation:
  - Created **requirements.txt**, **Procfile**, **runtime.txt**
  - Configured environment variables (GEMINI\_API\_KEY, DATA\_FILE)
2. Integration:
  - Connected GitHub repository to Render
  - Set up Web Service with Python environment
3. Launch:
  - Automatic deployment on push

### Live Deployment URLs

 User Dashboard: <https://feedback-system-xwew.onrender.com/>

 Admin Dashboard: <https://feedback-system-xwew.onrender.com/admin>

### Deployment Validation Tests

- Form submission functionality
- AI response generation
- Data persistence across sessions
- Admin analytics accuracy
- Cross-browser compatibility

### Key Technical Insights

- Successfully built and deployed a live AI-based feedback system
- Handled real AI integration with secure API usage and error management
- Demonstrated real-world abilities in AI development, backend engineering, and deployment

## Appendices

### Appendix A

#### Sample JSON Storage Entry

Sample JSON Storage Entry:

```
{
  "id": "20251207032601",
  "timestamp": "2025-12-07 03:26:01",
  "rating": 4,
  "review": "Loved it! Great place to hangout.",
  "ai_response": "Thank you for your 4-star review! We appreciate your feedback.",
  "ai_summary": "4 star review: Loved it! Great place to hangout....",
  "ai_actions": [
    "Analyze this feedback",
    "Share with relevant team"
  ]
},
{
  "id": "20251207033033",
  "timestamp": "2025-12-07 03:30:33",
  "rating": 1,
  "review": "Could be better. Bad experience.",
  "ai_response": "Thank you for your 1-star review! We appreciate your feedback.",
  "ai_summary": "1 star review: Could be better. Bad experience....",
  "ai_actions": [
    "Analyze this feedback",
    "Share with relevant team"
  ]
},
{
  "id": "20251207033921",
  "timestamp": "2025-12-07 03:39:21",
  "rating": 1,
  "review": "worst place for dinner.",
  "ai_summary": "Critical 1-star review requiring urgent attention.",
  "ai_actions": [
    "Immediate customer follow-up",
    "Quality control review"
  ]
},
y.
```

## **Appendix B** Requirements.txt Content:

```
Flask==2.3.3
google-generativeai==0.8.5
gunicorn==20.1.0
python-dotenv==1.0.0
```

## **Appendix C**

Procfile Content:

```
web: gunicorn app:app
```

## **Appendix D**

.gitignore

```
# Python
__pycache__/
*.pyc
*.pyo
*.pyd

# Virtual environments
.env
.venv/
venv/
env/

# Local data files
submissions.json

# Editor / OS
.vscode/
.idea/
.DS_Store
```