Comprehensive Project Report: Swiggy Restaurant Recommendation System

1. Approach

The objective of this project was to build a robust, content-based recommendation system capable of suggesting restaurants to users based on their specific preferences (city, budget, minimum rating, and cuisine).

The approach was divided into three core phases:

1. **Data Engineering pipeline:** A custom Python script (`data_preparation.py`) was developed to clean the raw dataset, handle missing values, and transform text-based features into a machine-readable format using One-Hot Encoding and Standard Scaling.
2. **Recommendation Engine:** Instead of generalized clustering techniques like K-Means, **Cosine Similarity** was chosen. This mathematical approach calculates the exact geometric distance between a user's multi-dimensional preference vector and the restaurant vectors, allowing for highly precise, ranked recommendations.
3. **Interactive Deployment:** A front-end application was built using **Streamlit** (`app.py`). It dynamically processes user inputs in real-time, queries the pre-computed mathematical dataset, and maps the results back to human-readable formats (names, addresses, and order links).

2. Data Analysis (EDA & Preprocessing)

During the initial exploration of the `swiggy.csv` dataset (containing over 148,000 records), several critical data challenges were identified and resolved:

- **Missing Ratings:** Over 87,000 restaurants had missing ratings (represented as `--`). Dropping these would have resulted in a massive loss of data. Instead, they were converted to `NaN` and imputed with the dataset's overall median rating (approx. 3.9). This preserved the dataset size while maintaining statistical integrity.
- **Text Heavy Columns:** The `cost` column contained string formats (e.g., "₹ 200"). Regular expressions were used to strip currency symbols and convert these values into usable numerical floats. A custom parser was also built to convert string-based rating counts (e.g., "1K+ ratings") into integers.
- **High Dimensionality Risk:** The `cuisine` column contained comma-separated lists (e.g., "Beverages, Pizzas"). To prevent an explosion of dimensions during the One-Hot Encoding phase—which would have degraded app performance—a `primary_cuisine` feature was engineered by extracting only the first listed cuisine.

3. Insights and Recommendations

Key Insights Derived from the Project

1. **The "Hidden Gem" Phenomenon:** A significant portion of the dataset consists of restaurants with low or missing rating counts. By utilizing a similarity-based engine and imputing the median rating, the system avoids the "popularity bias" trap, allowing newer or less-reviewed restaurants to still be recommended if they match the user's budget and cuisine preferences perfectly.
2. **Computational Efficiency:** Pre-filtering the encoded dataset by `city` *before* running the Cosine Similarity matrix calculation drastically reduced the computational load. This proves that geographically segmenting data is a highly effective optimization strategy for real-time recommendation apps.

Business & Technical Recommendations

1. **Hybrid Recommendation Implementation (Future Scope):** While this content-based filtering approach works perfectly for matching explicit features (cost, cuisine), Swiggy could enhance this in the future by integrating Collaborative Filtering (using historical user order data) to create a Hybrid Recommendation Engine.
2. **Dynamic Pricing Integration:** The application currently filters by a static "Maximum Cost for Two." It is recommended that future iterations of the data pipeline ingest live menu APIs to account for surge pricing or dynamic discounts.
3. **Operational Marketing:** Swiggy can use the output of this matching system to identify which cuisines are highly searched for but have low availability in specific cities, utilizing that data to incentivize new restaurant partnerships in those regions.