Project Objective: To design, develop, and deploy a recommendation system that suggests restaurants to individuals based on their demographics, medical conditions, location, food preferences, and historical behavior.

Executive Summary

This report outlines the end-to-end development and deployment of a sophisticated, multi-strategy recommendation engine. The project successfully navigated a four-stage workflow, beginning with the transformation of a complex, relational dataset into a single, feature-rich table. A deep learning Autoencoder was then leveraged for automated feature engineering, successfully creating a dense, 32-dimensional **"Taste DNA"** for each user, which captured latent behavioral patterns.

After validating the predictive power of these features, an HDBSCAN clustering algorithm performed an unsupervised segmentation of the user base, identifying **48 distinct "Taste Tribes."** This segmentation formed the basis of a functional `RecommendationEngine` that provides personalized recommendations for users in niche clusters and robust, popularity-based recommendations for mainstream users, all within a 5km geographical radius.

Finally, the entire system was successfully deployed to the cloud. The back-end API was containerized with Docker and deployed on **Render**, connected to a cloud-hosted PostgreSQL database. A decoupled, user-friendly front end was developed using HTML/CSS/JavaScript and deployed for free on **GitHub Pages**, resulting in a complete, full-stack, and publicly accessible web application.

Project Workflow and Methodology

The project was executed in a clear, logical progression, with the output of each stage serving as the foundation for the next.

Stage 1: Data Flattening & Feature Engineering

- **Objective:** To transform 8 separate, normalized source files (`Users.csv`, `Restaurants.csv`, `VisitHistory.csv`, etc.) into a single, wide table suitable for machine learning.
- **Methodology:** The script aggregated the **true historical transaction counts** from the visit history, pivoting the "long" event data into a "wide" format with 180 columns representing the order count for each of the top 15 dishes over 12 months. This was then enriched by simulating a novel **"craving"** feature, which was designed to be highly correlated with a user's real transactions but also influenced by seasonal probabilities. This process resulted in the `flattened_hybrid_craving_final.csv` file, the foundational dataset for all subsequent modeling.

Stage 2: Deep Learning Feature Extraction via Autoencoder

- **Objective:** To solve the problem of data sparsity and automatically engineer a powerful, high-level feature that represents a user's complete taste profile.
- **Methodology:** An **Autoencoder** neural network was built and trained on the 360 sparse behavioral columns. By forcing the network to compress the data down to a 32-dimensional vector and then reconstruct it, we compelled it to learn the most important latent patterns in user behavior. The predictive power of this resulting **"Taste DNA"** feature was rigorously validated by training a `RandomForestClassifier` (using the **SMOTE** technique to handle class imbalance) to

predict a user's Hypertension status, proving the feature contained a real, predictive signal. The final dataset, data_with_embeddings.csv, replaced the 360 sparse columns with the 32 dense "Taste DNA" features.

Stage 3: Unsupervised User Segmentation via Clustering

- **Objective:** To leverage the powerful "Taste DNA" feature to discover natural groupings of users with similar preferences ("Taste Tribes").
- **Methodology:** The **HDBSCAN** clustering algorithm was applied to the aggregated "Taste DNA" profiles of each of the 100,000 users. This advanced, density-based algorithm was chosen because it automatically discovers the optimal number of clusters and can identify users with unique tastes as "noise," which is a more realistic approach than forcing every user into a group.
- **Outcome:** The algorithm successfully identified **48 distinct "Taste Tribes,"** one massive "mainstream" cluster, and a significant number of unclassified "noise" users. These cluster assignments were saved to user_clusters.csv.

Stage 4: Deployment of the Full-Stack Application

- **Objective:** To take the completed models and logic and deploy them as a live, publicly accessible web application for free.
- **Methodology:** A **decoupled, cloud-native architecture** was chosen.
  1. **Back-End Deployment (API on Render):**
     - **Database Migration:** The source data was loaded into a free-tier **PostgreSQL** database hosted on Render, creating a scalable and centralized source of truth.
     - **Code Refactoring:** The RecommendationEngine was rewritten to connect to the cloud database using a secure **environment variable** for the connection URL.
     - **Containerization:** The Flask application (containing the engine logic) and all its dependencies were packaged into a portable **Docker** container.
     - **Deployment:** The Docker container was deployed as a "Web Service" on **Render**, which automatically built and hosted the live API.
  2. **Front-End Deployment (UI on GitHub Pages):**
     - **Development:** A simple but "cute" user interface was built using a single index.html file with vanilla **HTML, CSS, and JavaScript**. The JavaScript uses the fetch API to call the live back-end URL on Render.
     - **CORS Configuration:** A critical step involved updating the back-end Flask app with the Flask-Cors library to securely allow the front-end (on a github.io domain) to make requests to the back-end (on an onrender.com domain).
     - **Deployment:** The index.html file was placed in the root of the project's GitHub repository, and the **GitHub Pages** feature was enabled. This instantly provided free, reliable hosting for the front-end interface.

Final Outcome

The project successfully concluded with a fully deployed, full-stack web application. The final architecture consists of a browser-based front end that communicates with a containerized back-end API, which in turn queries a cloud database to retrieve data and serve real-time, personalized restaurant recommendations. This entire system was deployed and is running on free-tier cloud services.