Technical Design Report: HungerBusters

HungerBusters is a web-based platform focused on reducing food waste within Cornell University dining halls. The system enables students to access surplus food—whether through discounted custom orders or "surprise bags"—and incorporates an AI-driven recommendation engine to help match users with suitable food items. The platform aims to address sustainability, affordability, and food security.

1. System Architecture Overview

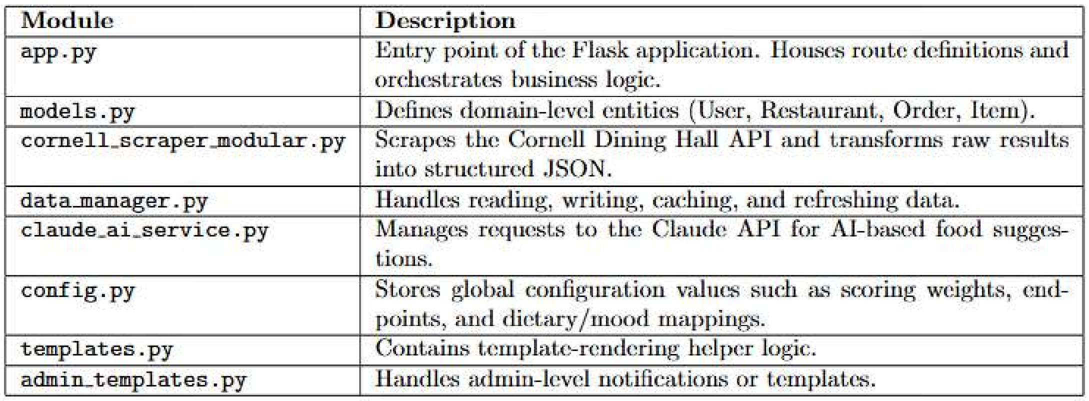
HungerBusters is designed using a modular architecture that separates data ingestion, data management, domain modeling, business logic, AI recommendations, and presentation layers.

Architecture Layers

1. **Data Ingestion Layer:** Scrapes Cornell Dining API
2. **Data Management Layer:** Loads, saves, and maintains JSON-based data.
3. **Domain Models Layer:** Defines core entities such as Users, Items, Orders, and Dining Halls.
4. **Service Layer:** Implemented primarily in app using logic
5. **AI / Recommendation Layer:** Uses Claude API to recommend users based on preferences and provide the closest possible matchings meals
6. **Presentation Layer:** Flask API routes and frontend JavaScript.
7. **Configuration Layer:** Env parameter configuration.

2. Domain Model Summary

- **User**
  - Attributes: name, login_id, preferences, dietary restrictions, mood, etc.
  - Can register, get suggestions, place custom/surprise orders.
- **Restaurant**
  - Represents each dining hall.
  - Contains inventory of surplus items.
- **Item**
  - Represents a food item.
  - Attributes include: name, ID, dietary tags, urgency level, mood mapping, pricing.
- **Order**
  - Represents either a surprise bag or custom item selection.
  - Contains list of items and associated user ID.



Module	Description
`app.py`	Entry point of the Flask application. Houses route definitions and orchestrates business logic.
`models.py`	Defines domain-level entities (User, Restaurant, Order, Item).
`cornell_scraper_modular.py`	Scrapes the Cornell Dining Hall API and transforms raw results into structured JSON.
`data_manager.py`	Handles reading, writing, caching, and refreshing data.
`claude_ai_service.py`	Manages requests to the Claude API for AI-based food suggestions.
`config.py`	Stores global configuration values such as scoring weights, endpoints, and dietary/mood mappings.
`templates.py`	Contains template-rendering helper logic.
`admin_templates.py`	Handles admin-level notifications or templates.

Table 1: Module Documentation Summary

3. Business Logic and Algorithms

3.1 Surprise Bag Generation (FREE)

- Randomized but constrained selection of surplus items.
- Considers freshness, urgency, and availability.

3.2 Custom Orders (PAID)

- Validates requested items.
- Applies discount logic (e.g., 70% off).
- Includes premium features like AI Based Recommendations.

3.3 AI Recommendations

- Powered by the Claude API.
- Inputs include:
    - User dietary preferences
    - Mood
    - Urgency of items
    - Item metadata
    - Previous ratings
- Produces ranked item suggestions.

3.4 Data Refresh

- Admin or system-triggered data refresh from Cornell API.
- The new dataset replaces the old one through DataManager.