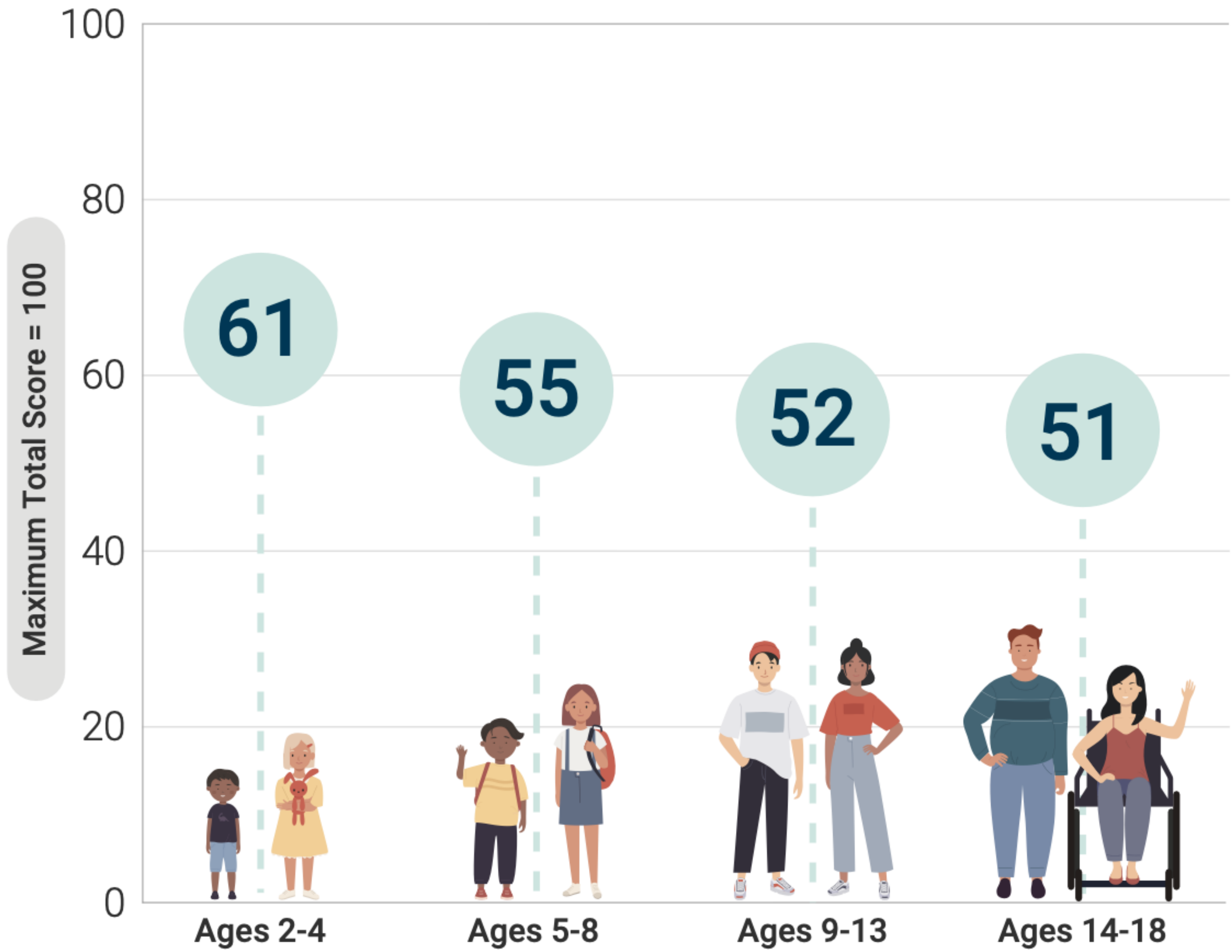
Health Aware meal recommendations using Contextual Multi -Armed Bandits

- By Group 8
- Team Members:
  - Ganesh Kumar Boini
  - Dinesh Chandra Gaddam
  - Sirisha Ginnu

THE GEORGE WASHINGTON UNIVERSITY
WASHINGTON, DC

Healthy Eating Index Scores Across Childhood and Adolescence



Data Source: Analysis of What We Eat in America, NHANES 2015-2016, ages 2 through 18, day 1 dietary intake, weighted.

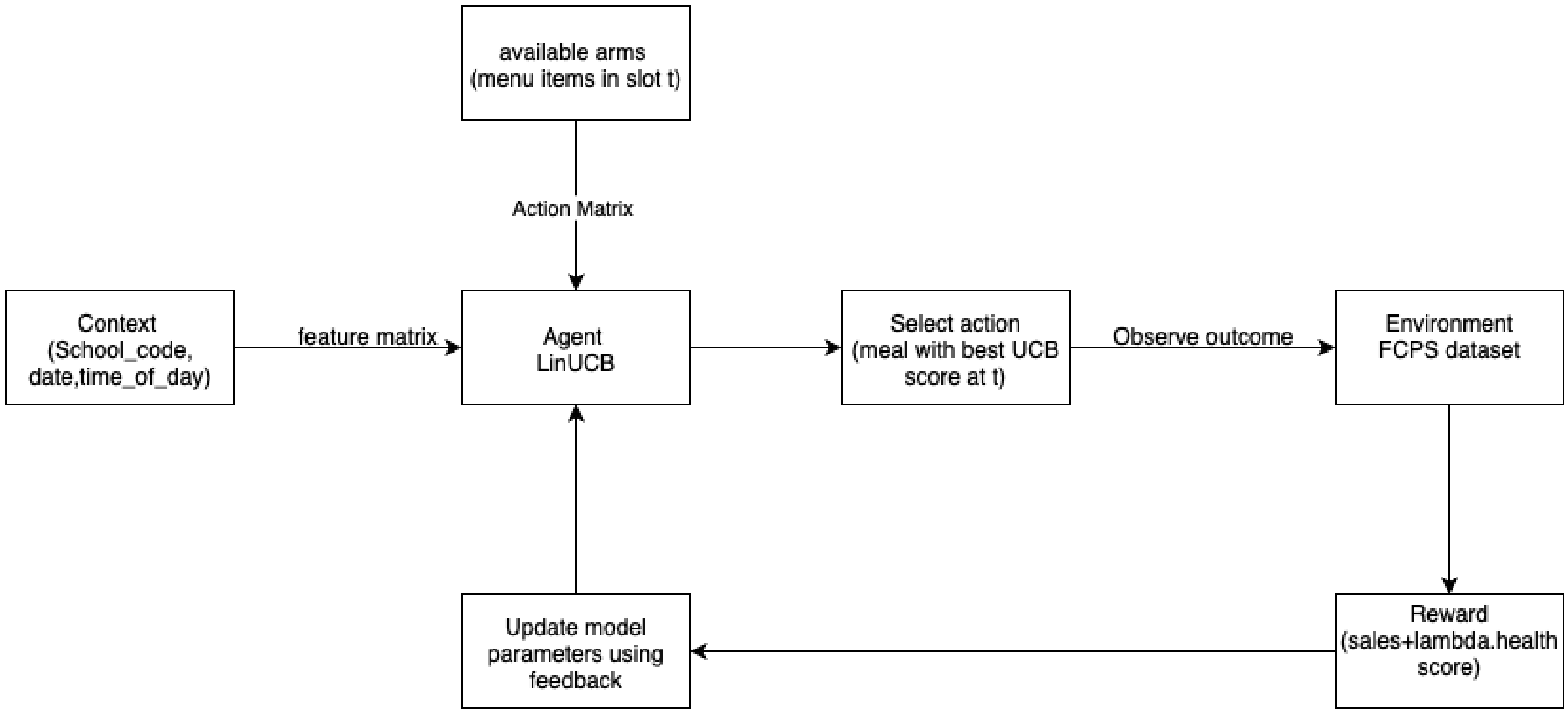
Motivation

- As seen in the earlier slide, there is a clear need for healthier meals for children and adolescents.
- What if every school meal could be both *what kids love to eat* and *what keeps them healthy*?
- Goal: Build a **school meal recommendation system** for student satisfaction that **maximizes both popularity and healthiness.**
- Objective: Learn **adaptive, data-driven meal recommendations** that balance **taste and nutrition.**
- Approach :Contextual Multi -Armed Bandits (CMAB) for decision optimization

CMAB Framework Overview

- Context: school, time_of_day, day_of_week.
- Action: meal (item_id).
- Reward: total_meals_served + $\lambda \times$ health_score.
- Focus: Balancing exploration and exploitation.

CMAB Flow for FCPS



Nutrition Data Extraction

- Reverse engineered LINQ connect API system using browser dev tools monitoring
- Found key endpoints for menu items
- Scope of extraction : full academic year coverage with all meal types and categories for all 187 FCPS institutions

Sales Dataset (FCPS)

- FCPS meal sales dataset: includes schools, timestamps, and meal details.
- Context variables: school, time, day, seasonality.
- Action variables: 160+ meal items (arms).

System design overview

- Modules:
- 1. utils/env.py → Simulates the FCPS environment
- 2. model.py → Implements LinUCB bandit model
- 3. main.py → Trains and evaluates the system

- Data Flow: Dataset → Environment → Model → Results

Environment Design (env.py)

- load_data(): Loads FCPS preprocessed CSV data.
- build_item_mapping: map a unique index to all 160 items and use it subsequently
- build_feature_matrix(): Returns contextual feature matrix.
- build_action_matrix(): Defines available meal actions per timestep.
- health_scores(): Provides healthiness scores for each meal.

LinUCB Algorithm (model.py)

- Equation: $\hat{y}_a = \theta_a^T x_t + \alpha\sqrt{(x_t^T A_a^{-1} x_t)}$
- Balances exploration (uncertainty) and exploitation (expected reward).
- Each meal (arm) has its own A_a (covariance) and b_a (reward) matrices.
- Algorithm Steps:
- 1. Estimate reward using context features
- 2. Add confidence bound (exploration term)
- 3. Choose action with highest upper bound
- 4. Update model with observed reward

Training & Update Mechanism

- train(): Iteratively selects meals, observes rewards, updates model.
- action(): Chooses arm maximizing upper confidence bound.
- update(): Updates A_a and b_a using observed reward.
- reset() & save(): Manage experiment lifecycle and persistence.

Reward Design & Health Factor (λ)

- Reward = total_meals_served + λ × healthiness_score.
- λ tunes the trade-off between popularity and nutrition.
- Larger λ → healthier meals favored

Evaluation Metrics (metrics.py)

- Regret: Difference between optimal and chosen reward.
- Cumulative Reward: Tracks long-term performance.
- Plots used for comparing exploration strategies.
- Example: Cumulative Reward vs Time.

Benchmarking Experiments (benchmark.py)

- Tested multiple λ values for reward balancing.
- Compared LinUCB vs baseline (random, popularity-based).
- Saved results via bench_results_to_csv().
- Metrics: Mean cumulative reward, total regret, and recommendation stability.

Results Visualization (plot.py)

- plot_top_meals(): Shows most recommended items.
- plot_recommendations(): Trends over time or by context.
- Example visuals: Heatmaps, bar charts, and cumulative reward plots.

Key Findings

Future Work

- • Extend to full RL (state transitions, delayed rewards).
- • Implement Thompson Sampling or Neural Bandits.
- • Integrate model with real-time FCPS meal systems.

Thank You / Q&A

- Questions or feedback welcome!