Smart Meal Planning and Personalized Nutrition Tracking Using a Web-Based Application

Ponmansha. R[1], Meerasekaran. G[2], Punitha. S[3], Mariya Abinaya. V[4], Sangeetha. G[5]
[1,2,3,4,5] *II M. Sc CS Student, Department of Computer Science,Shri Nehru Maha Vidyalaya College of Arts &Science, Coimbatore.*

***Abstract*—This paper presents MealMatrix, a full-stack web application designed to help users plan, track, and manage their daily nutrition and weekly meal schedules. The system combines a Python Flask backend with a responsive HTML/CSS/JavaScript frontend to deliver a seamless, personalized nutrition management experience. Key features include OTP-based user authentication via email and SMS, user profile creation with automated daily calorie and macronutrient calculations using the Harris-Benedict Basal Metabolic Rate equation, meal selection across four categories (breakfast, lunch, dinner, snacks), an interactive weekly meal planner, and a real-time achievement dashboard. The application enforces security best practices such as rate limiting, CORS configuration, session management, and input validation. MealMatrix demonstrates how modern web technologies can be combined to build a practical, data-driven dietary assistant for everyday users.**

***Index Terms*—Flask, Harris-Benedict equation, meal planning, nutrition tracking, OTP authentication, rate limiting, SQLite, web application.**

I. INTRODUCTION

Maintaining a healthy and balanced diet is a fundamental component of overall well-being. However, many individuals find it challenging to plan meals consistently, meet daily nutritional requirements, and monitor dietary progress. Existing nutrition tools are often complex, expensive, or require a deep understanding of dietary science, making them inaccessible to the general public.

MealMatrix addresses these challenges by providing an easy-to-use, browser-based platform that automates the process of meal planning and nutrition tracking. The application guides users through a structured onboarding flow: registration with email or phone, OTP-based account verification, physical profile setup, meal preference selection, and finally, a personalized weekly meal plan with daily check-off tracking.

The system is built using Python Flask for the backend REST API, SQLite as a lightweight relational database managed through SQLAlchemy ORM, and vanilla HTML, CSS, and JavaScript for the frontend. This technology choice ensures the application is lightweight, portable, and easy to deploy.

The primary objectives of MealMatrix are: (1) to automate daily nutritional target calculation based on individual physical parameters; (2) to provide a structured interface for selecting and planning meals; (3) to track weekly meal completion and visualize user achievements; and (4) to maintain a secure and reliable user authentication system.

II. LITERATURE SURVEY

Numerous studies highlight the importance of technology in supporting dietary behavior change. Mobile and web-based nutrition applications have been shown to improve dietary self-monitoring and increase awareness of caloric intake [1]. The Harris-Benedict equation, first published in 1919 and revised by Roza and Shizgal in 1984, remains one of the most widely accepted methods for estimating Basal Metabolic Rate (BMR) and is commonly used in digital nutrition tools [2].

Research on user authentication in web applications emphasizes the need for multi-factor verification to protect personal health data. OTP-based authentication via email and SMS is recognized as an effective second factor that balances usability and security [3]. Studies on meal planning systems also demonstrate that personalized meal recommendation

engines based on user preferences and dietary constraints significantly improve user adherence compared to generic meal plans [4].

Existing commercial applications such as MyFitnessPal and Cronometer provide extensive nutrition databases but rely on user-driven food entry, which is time-consuming. MealMatrix adopts a curated meal catalogue approach, allowing users to choose from a pre-defined set of nutritionally balanced meals, simplifying the planning process while still delivering personalized macro targets.

III. SYSTEM ARCHITECTURE AND DESIGN

MealMatrix follows a client-server architecture. The frontend is composed of static HTML pages served directly by the Flask backend, while all dynamic operations are handled through REST API endpoints. This architecture separates presentation logic from business logic and makes the application straightforward to maintain and extend.

A. Backend Architecture

The Flask application is initialized through a factory function (create_app) that configures all extensions, registers blueprints, and creates the database tables on first run. Two blueprints organize the codebase: auth_bp for all authentication-related routes (prefixed /auth) and main_bp for static page delivery and general API endpoints.

Security features are applied globally: CORS is restricted to known origins (localhost and 127.0.0.1), Flask-Limiter enforces per-endpoint rate limits to prevent brute-force and spam attacks, and Flask-Session provides server-side session storage to avoid exposing session data to clients.

B. Database Schema

The database consists of four models managed by SQLAlchemy ORM:

Model	Table Name	Key Fields
User	users	id, email, phone, password_hash, is_active
OTPRecord	otp_records	user_id, otp_type, code, expires_at, used
UserProfile	user_profiles	user_id, age, gender, weight_kg, height_cm, diet_type, kcal, protein_g, carbs_g, fat_g
Weekly Plan	weekly_plans	user_id, plan_json, updated_at

Relationships are defined with cascade delete-orphan policies, ensuring that when a user account is removed, all associated records (OTP codes, profiles, and weekly plans) are automatically deleted.

C. Frontend Architecture

The frontend is composed of independent HTML pages, each self-contained with embedded CSS and JavaScript. Pages communicate with the backend exclusively through the Fetch API, receiving and sending JSON payloads. Client-side session state is maintained through server-set HTTP cookies, while page access control is enforced by checking the /auth/me endpoint on load.

IV. MODULES AND IMPLEMENTATION

A. User Authentication Module

The authentication module supports registration via either email address or phone number. Upon submission, the backend validates input format using regular expressions conforming to RFC-5322 for email and E.164 format for phone numbers. Passwords must be a minimum of eight characters and contain at least one digit, following NIST SP 800-63B guidelines.

A six-digit OTP code is generated using the Python secrets module, stored in the otp_records table with a configurable expiry time, and dispatched to the user via SMTP email or Twilio SMS. An account becomes active only after successful OTP verification. Email enumeration attacks are mitigated by returning a generic response message regardless of whether the submitted email already exists.

B. User Profile and Nutrition Calculation Module

Once authenticated, users submit their physical profile: name, age, gender, weight (kg), height (cm),

and preferred diet type. The backend validates all numeric ranges (age: 1-120, weight: 1-500 kg, height: 30-300 cm) before computing daily nutrition targets using the revised Harris-Benedict BMR equation multiplied by a moderate activity factor (1.55) to estimate Total Daily Energy Expenditure (TDEE).
Macronutrient splits are then applied based on the selected diet type. Supported diet types include balanced (25% protein, 50% carbs, 25% fat), vegetarian, vegan, keto, paleo, and high-protein, each with predefined macro ratios. Calorie-to-gram conversions use standard factors: 4 kcal/g for protein and carbohydrates, and 9 kcal/g for fat.

C. Meal Selection Module

The meal selection page presents a curated catalogue of sixteen meals across four daily categories: breakfast (oatmeal with berries, avocado toast, Greek yogurt, smoothie), lunch (chicken salad, quinoa bowl, brown rice with chicken, veggie wrap), dinner (salmon with vegetables, stir fry, tofu bowl, Buddha bowl), and snacks (mixed nuts, protein balls, yogurt parfait). Users select up to three meals per category, and their preferences are persisted as a JSON blob in the user_profiles table.

D. Weekly Meal Planner Module

The weekly planner maps the user's selected meals to each day of the week (Monday through Sunday). Users can mark individual meals as completed using checkboxes. The plan state is serialized to JSON and saved to the weekly_plans table via a POST request. On page reload, the saved state is restored via the GET endpoint, ensuring plan persistence across sessions.

E. Achievement Dashboard Module

The dashboard fetches achievement statistics from /auth/achievements, which computes the percentage of daily meal checks completed for each day of the current week. Results are returned as a seven-element array and rendered as a bar chart. Four circular progress indicators display the user's weekly achievement percentage relative to their daily kcal, protein, carbohydrate, and fat targets.

V. TECHNOLOGIES USED

Table I summarizes the complete technology stack used to build MealMatrix.

TABLE I. TECHNOLOGY STACK

Category	Technology / Tool	Purpose
Frontend	HTML5, CSS3, JavaScript	UI and client-side logic
Backend Framework	Flask (Python 3.10+)	REST API and server logic
Database	SQLite + SQLAlchemy ORM	Persistent data storage
Authentication	OTP via SMTP / Twilio SMS	Secure user verification
Session Management	Flask-Session	Server-side session handling
Security	Flask-Limiter, CORS	Rate limiting and access control
Environment Config	python-dotenv	Manage secrets and config
Typography	Google Fonts (Inter)	Frontend font rendering

All backend dependencies are declared in requirements.txt and are installable via pip. The application requires Python 3.10 or higher. The frontend has no build step and requires only a modern web browser; Google Fonts (Inter family) are loaded via CDN for typography.

VI. SYSTEM WORKFLOW

The MealMatrix user journey follows a linear onboarding flow designed to collect information progressively before generating the final personalized plan. The workflow proceeds through the following steps:

- Step 1 – Registration: The user visits the landing page and registers using either an email address or phone number along with a password.

- Step 2 – OTP Verification: A six-digit code is sent to the registered email or phone. The user enters the code on the verification page to activate their account.
- Step 3 – Profile Setup: The user provides physical details (age, gender, weight, height) and selects a diet type. The system computes and stores daily calorie and macro targets.
- Step 4 – Body Needs Review: The user views their computed nutritional targets (kcal, protein, carbohydrates, fat) before proceeding.
- Step 5 – Meal Selection: The user browses the meal catalogue and selects preferred meals across breakfast, lunch, dinner, and snack categories.
- Step 6 – Weekly Plan: The selected meals are arranged into a seven-day plan. Users check off meals as they complete them each day.
- Step 7 – Dashboard: The user reviews their weekly achievement statistics, including per-day completion percentages and overall nutrient goal progress.

All REST API endpoints return a consistent JSON envelope { "success": bool, "message": str, ...data }. Table II lists the principal API endpoints and their functions.

TABLE II. API ENDPOINT REFERENCE

Method	Endpoint	Description
POST	/auth/register/email	Register via email with OTP
POST	/auth/register/phone	Register via phone (SMS OTP)
POST	/auth/verify/email	Verify email OTP to activate account
POST	/auth/verify/phone	Verify phone OTP to activate account
POST	/auth/login	Login with email or phone + password
POST	/auth/logout	End user session
GET	/auth/me	Return current authenticated user
GET/POST	/auth/profile	Get or save user physical profile
GET	/auth/body-needs	Return daily nutrition targets
POST	/auth/meal-preferences	Save selected meal preferences
GET/POST	/auth/weekly-plan	Load or save weekly meal plan
GET	/auth/achievements	Fetch weekly achievement statistics

VII. RESULTS AND DISCUSSION

MealMatrix was tested in a local development environment running Python 3.14 on a Windows platform. The application starts with a single command and is accessible at http://127.0.0.1:5000. All database tables are created automatically on first run using Flask-SQLAlchemy'sdb.create_all() function. The SQLite database file is stored in the backend/instance/ directory.

The OTP verification flow was validated for both email (SMTP) and phone (Twilio) channels. Rate limits were verified by issuing rapid consecutive requests to sensitive endpoints; the system correctly returned HTTP 429 responses with an informative message after the configured threshold was exceeded. Input validation was tested with edge-case values (e.g., age = 0, weight = 1000 kg), confirming that the backend rejects invalid inputs and returns descriptive error messages.

The nutrition calculation module was validated against manual Harris-Benedict computations for sample profiles. For a 25-year-old male weighing 70 kg and standing 175 cm with a balanced diet, the system computed a TDEE of approximately 2,679 kcal, with macro targets of 167 g protein, 335 g carbohydrates, and 74 g fat. These results are consistent with standard nutritional guidelines.

The weekly planner correctly persists and restores state across browser sessions using server-side session cookies and database storage. The achievement dashboard renders completion statistics accurately based on the saved plan data. The frontend pages are fully responsive and functional across modern desktop browsers.

Security measures were evaluated using manual testing. CORS restrictions were confirmed to block cross-origin requests from unauthorized domains. Path traversal attempts on the /pages/ and /src/ routes returned HTTP 403 responses, confirming the protection implemented in the _safe_send helper function. Session fixation protection was verified by inspecting session regeneration on login.

VIII. CONCLUSION

This paper has presented MealMatrix, a full-stack web application for personalized meal planning and nutrition tracking. The system integrates secure OTP-based authentication, automated BMR-driven nutritional target computation, a structured meal catalogue, and a persistent weekly planner with achievement visualization into a cohesive and user-friendly platform.

The application demonstrates effective use of the Flask microframework for building RESTful APIs, SQLAlchemy for database management, and modern web technologies for a clean, responsive user interface. Security best practices including rate limiting, CORS control, server-side sessions, and input sanitization are applied throughout the system.

Future enhancements may include integration with a public nutrition database API to expand the meal catalogue, addition of a calorie-intake tracking log, implementation of a machine learning-based meal recommendation engine based on user history, mobile application development using a cross-platform framework, and support for additional diet types such as diabetic and Mediterranean diets.

ACKNOWLEDGMENT

The authors would like to thank the faculty of the Department of Computer Science at [Institute Name] for their guidance and support throughout the development of this project. The authors also acknowledge the open-source communities behind Flask, SQLAlchemy, and the Python ecosystem for their invaluable tools and documentation.

REFERENCES

[1] K. Patrick, F. Raab, M. A. Adams, L. Dillon, M. Zabinski, C. L. Rock, W. G. Griswold, and G. J. Norman, "A text message-based intervention for weight loss: Randomized controlled trial," Journal of Medical Internet Research, vol. 11, no. 1, p. e1, Jan. 2009.

[2] A. M. Roza and H. M. Shizgal, "The Harris-Benedict equation reevaluated: Resting energy requirements and the body cell mass," American Journal of Clinical Nutrition, vol. 40, no. 1, pp. 168-182, July 1984.

[3] D. Dasgupta, A. Roy, and A. Nag, "Advances in multi-factor authentication," in Advances in User Authentication, Cham: Springer, 2017, pp. 185-233.

[4] S. Ge and A. Bhatt, "Personalized meal recommendation system for dietary management," in Proc. IEEE International Conference on Big Data, Los Angeles, CA, 2019, pp. 2724-2733.

[5] M. Grinberg, Flask Web Development: Developing Web Applications with Python, 2nd ed. Sebastopol, CA: O'Reilly Media, 2018.

[6] NIST, "Digital identity guidelines: Authentication and lifecycle management," NIST Special Publication 800-63B, June 2017. [Online]. Available: https://pages.nist.gov/800-63-3/sp800-63b.html

[7] Pallets Projects, "Flask documentation (3.x)," [Online]. Available: https://flask.palletsprojects.com/en/3.0.x/

[8] SQLAlchemy, "SQLAlchemy documentation," [Online]. Available: https://docs.sqlalchemy.org/en/20/

[9] Twilio Inc., "Twilio SMS API documentation," [Online]. Available: https://www.twilio.com/docs/sms