**Backend System Documentation**

**Part 1: Core Application Structure and Configuration**

**Main Application (main.py)**

**Overview:  
The main application module initializes and configures the Flask application with necessary middleware, logging, and blueprints for different functionality modules.**

**Key Components:**

* **Application Creation (create\_app()):**
  + **Initializes Flask application**
  + **Configures logging levels (INFO for app, WARNING for werkzeug and pymongo)**
  + **Initializes application configuration and constants**
  + **Exports constants to frontend**
  + **Sets up error handlers for 404 and 500 errors**
* **Registered Blueprints:**
  + **Food Routes (food\_routes)**
  + **Authentication Routes (auth\_routes)**
  + **Doctor Routes (doctor\_routes)**
  + **Patient Routes (patient\_routes)**
  + **Test Routes (test\_routes)**
  + **Meal Insulin Management (meal\_insulin\_bp)**
  + **Activity Tracking (activity\_bp)**
  + **Blood Sugar Monitoring (blood\_sugar\_bp)**
  + **Medication Routes (medication\_routes)**
* **Error Handling:**
  + **404 Not Found: Returns JSON response for resource not found**
  + **500 Internal Server Error: Logs error and returns JSON response**

**Configuration:**

* **Debug mode: Disabled in production**
* **Host: 0.0.0.0**
* **Port: 5000**

**Application Configuration (config.py)**

**Overview:  
Handles application configuration, database connection, and logging setup.**

**Key Components:**

* **Database Configuration:**
  + **MongoDB connection via PyMongo**
  + **URI: mongodb://localhost:27017/native\_new**
* **CORS Configuration:**
  + **Allowed Origins: http://localhost:3000**
  + **Allowed Methods: GET, POST, PUT, DELETE, OPTIONS**
  + **Supports Credentials: True**
* **Application Settings:**
  + **Secret Key Configuration**
  + **Timezone: UTC**
  + **Token Expiry: 24 hours**
* **Logging Configuration:**
  + **Base Level: INFO**
  + **PyMongo Logging: WARNING**
  + **MongoDB Logging: WARNING**

**Constants Management (constants.py)**

**Overview:  
Manages application-wide constants and configuration values, supporting patient-specific customization.**

**Core Components:**

* **Measurement Systems:**
  + **Volume Measurements: cup, half\_cup, quarter\_cup, tablespoon, teaspoon, bowl, plate**
  + **Weight Measurements: palm, handful, fist, plate, grams, kilograms**
* **Activity Levels:**
  + **Sleep (-2): -0.2 impact**
  + **Very Low Activity (-1): -0.1 impact**
  + **Normal Activity (0): 0 impact**
  + **High Activity (1): 0.1 impact**
  + **Vigorous Activity (2): 0.2 impact**
* **Meal Types:**
  + **Breakfast, Lunch, Dinner, Snack**
* **Patient Constants:**
  + **Base Calculations: insulin\_to\_carb\_ratio, correction\_factor, target\_glucose, protein\_factor, fat\_factor**
  + **Timing Factors: Meal-specific adjustments, time-of-day considerations, disease-specific modifications, medication impacts**

**Part 2: Authentication System (auth\_routes.py)**

**Overview:  
Handles user authentication, registration, and token management for both patients and doctors.**

**Key Functions:**

* **Login: Authenticates users and provides JWT token.**
* **Register: Creates new user accounts and sets up default patient constants.**
* **Hash Password: Hashes user passwords securely.**
* **Verify Password: Verifies password input against stored hash.**
* **Generate JWT Token: Generates JWT tokens with expiration.**

**Part 3: Doctor Routes (doctor\_routes.py)**

**Overview:  
Manages doctor-specific functionalities including patient management and constant adjustments.**

**Key Functions:**

* **Get Patients: Retrieves list of all patients.**
* **Get Patient Constants: Retrieves patient-specific constants.**
* **Update Patient Constants: Updates patient constants.**
* **Update Patient Conditions: Updates patient's active conditions.**
* **Update Patient Medications: Updates patient's active medications.**
* **Log Medication: Records medication administration.**

**Part 4: Patient Routes (patient\_routes.py)**

**Overview:  
Handles patient-specific functionalities and data access.**

**Key Functions:**

* **Get Patient Constants: Retrieves patient-specific settings and constants.**

**Part 5: Medication Routes (medication\_routes.py)**

**Overview:  
Manages medication schedules, tracking, and administration records.**

**Key Functions:**

* **Get Medication Schedule: Retrieves active medication schedules.**
* **Create or Update Medication Schedule: Creates/updates medication schedules.**
* **Create Initial Medication Logs: Creates initial medication logs for new schedules.**
* **Validate Time Format: Validates time formats (HH:MM).**
* **Handle Timezone Conversion: Handles timezone conversions.**
* **Check Schedule Overlap: Manages schedule overlaps.**

**Part 6: Food Management System**

**Food Data Models (models/food\_data.py)**

**Overview:  
Defines a comprehensive food database with standardized measurements and nutritional information across different categories.**

**Key Functions:**

* **Define Food Categories: Defines food categories and subcategories.**
* **Structure Food Data: Structures food data for use in APIs.**
* **Validate Food Entry: Validates new or custom food entries.**

**Food Routes (routes/food\_routes.py)**

**Key Functions:**

* **Get Food Categories: Returns all food categories and measurements.**
* **Search Food: Queries and returns matching foods with details.**
* **Create Custom Food: Adds a new custom food entry.**
* **Get Nutritional Summary: Calculates total nutrition for meal items.**

**Part 7: Blood Sugar Management (blood\_sugar.py)**

**Overview:  
Manages blood sugar readings, tracking, and analysis for patients and doctors with support for different measurement units and target ranges.**

**Key Functions:**

* **Blood Sugar Recording: Records new blood sugar readings.**
* **Blood Sugar Retrieval: Retrieves historical readings.**
* **Doctor's Patient View: Accesses patient blood sugar history.**
* **Status Classification: Classifies blood sugar status (low, normal, high).**

**Part 8: Meal and Insulin Management (meal\_insulin.py)**

**Overview:  
Calculates insulin doses based on meals, activities, and health factors using machine learning and reinforcement learning principles.**

**Key Functions:**

* **Health Factor Calculation: Calculates disease impact, medication timing, and dynamic adjustments.**
* **Activity Impact Analysis: Calculates total activity impact coefficient.**
* **Meal Timing Factors: Determines timing-based adjustments for insulin.**
* **Insulin Calculation: Calculates insulin needs based on carbs, protein, fat, and activity.**

**Part 9: Activity Tracking (activity.py)**

**Overview:  
Manages physical activity tracking and its impact on insulin requirements, integrating with the meal and insulin calculation system.**

**Key Functions:**

* **Activity Recording: Records expected and completed activities.**
* **Activity History: Retrieves activity records.**
* **Activity Impact System: Calculates activity impact coefficients.**

**Part 10: Authentication and Error Handling**

**Authentication (utils/auth.py)**

**Key Functions:**

* **Token Required: Protects routes with JWT authentication.**
* **Get User from Token: Extracts and verifies user from JWT token.**

**Error Handling (utils/error\_handler.py)**

**Key Functions:**

* **API Error Handler: Catches and logs API exceptions.**
* **Log Error: Logs error details for debugging.**
* **Format Error Response: Formats standardized error responses.**

**Part 11: Application Initialization and Utilities**

**Overview:  
The application initialization module provides a factory pattern for creating and configuring the Flask application with appropriate settings and dependencies.**

**Factory Function:**

* **Creates and configures the Flask application instance. It accepts an optional configuration name (e.g., 'testing') and ensures the application is initialized with the correct settings, middleware, and routes.**

**Configuration Settings:**

1. **Default Configuration:**
   * **MongoDB URI for data storage**
   * **Secret key for secure operations**
   * **Application timezone set to UTC**
   * **Token expiry defined in hours**
   * **Allowed origins for Cross-Origin Resource Sharing (CORS)**
2. **Logging Configuration:**
   * **Application: Logs at INFO level**
   * **PyMongo: Logs at WARNING level**
   * **MongoDB: Logs at WARNING level**
3. **CORS Configuration:**
   * **Enabled for specified origins**
   * **Applies to all resource patterns**

**Testing Routes:**

* **Health Check Endpoint (/test):**
  + **Returns a simple JSON message confirming the backend is operational.**
  + **Status code: 200 OK**

**Part 12: Security Implementation**

**JWT Token Security:**

* **Token Structure: Includes user ID, user type, and expiration time.**
* **Token Handling: Proper expiration and secure validation methods.**

**Database Security:**

* **Connection: Secure and authenticated MongoDB connection.**
* **User Data Protection: Password hashing and sensitive data encryption.**

**API Security:**

* **CORS Protection: Restricted origins and controlled methods.**
* **Request Validation: Sanitization, type checking, and size limitations for inputs.**
* **Error Handling: Prevents sensitive data exposure and ensures proper logging.**