Health Statistics (Expanded) – **Wireframe**

**Stat-Item Boxes**

**Overall Structure:**

* **Position:** Below the Next Meal Widget, occupying the lower half of the screen.
* **Background Color:** Light to medium gray, contrasting subtly with the other sections.

**Layout:**

* **Grid Layout:**
  + **Desktop:** 2-3 columns layout.
  + **Tablet:** 2 columns layout.
  + **Mobile:** Single column layout.
  + **Spacing:** Adequate padding between boxes (approximately 10-20px) to ensure clear separation.

**Boxes:**

1. **Caloric Intake:**
   * **Position:** Top-left (or first in single column layout).
   * **Structure:**
     + **Title:** "Caloric Intake"
     + **Progress Bar:**
       - Horizontal bar with segments representing consumed vs. remaining calories.
       - **Color Scheme:** Dark gray for consumed, light gray for remaining.
     + **Numeric Display:** Current intake vs. target (e.g., "1500/2000 kcal").
   * **Interactive Elements:**
     + **Hover Effects:** Tooltip with detailed breakdown (e.g., breakfast, lunch, dinner, snacks).
     + **Tap/Press:** Opens a modal with a more detailed view.
2. **Water Consumption:**
   * **Position:** Top-middle (or second in single column layout).
   * **Structure:**
     + **Title:** "Water Consumption"
     + **Visual Indicator:**
       - Progress bar or a series of water drop icons filling up.
       - **Color Scheme:** Dark gray for consumed, light gray for remaining.
     + **Numeric Display:** Current intake vs. target (e.g., "1.5/2.5 L").
   * **Interactive Elements:**
     + **Hover Effects:** Tooltip with hourly intake details.
     + **Tap/Press:** Opens a modal with a detailed view.
3. **Activity Level:**
   * **Position:** Top-right (or third in single column layout).
   * **Structure:**
     + **Title:** "Activity Level"
     + **Progress Bar:**
       - Horizontal bar showing steps or active minutes.
       - **Color Scheme:** Dark gray for completed, light gray for remaining.
     + **Numeric Display:** Current steps vs. target (e.g., "7500/10000 steps").
   * **Interactive Elements:**
     + **Hover Effects:** Tooltip with breakdown of different activities (e.g., walking, running).
     + **Tap/Press:** Opens a modal with a detailed activity log.
4. **Sleep Duration:**
   * **Position:** Middle-left (or fourth in single column layout).
   * **Structure:**
     + **Title:** "Sleep Duration"
     + **Bar Chart:**
       - Bars representing sleep hours over the past week.
       - **Color Scheme:** Dark gray for each day's sleep duration.
     + **Numeric Display:** Average sleep duration (e.g., "7.2 hours").
   * **Interactive Elements:**
     + **Hover Effects:** Tooltip with nightly sleep details.
     + **Tap/Press:** Opens a modal with detailed sleep analysis.
5. **Nutrient Balance:**
   * **Position:** Middle-right (or fifth in single column layout).
   * **Structure:**
     + **Title:** "Nutrient Balance"
     + **Pie Chart:**
       - Segments representing proteins, carbs, and fats.
       - **Color Scheme:** Different shades of gray for each nutrient.
     + **Numeric Display:** Percentage breakdown (e.g., "Protein: 30%, Carbs: 50%, Fats: 20%").
   * **Interactive Elements:**
     + **Hover Effects:** Tooltip with nutrient details and sources.
     + **Tap/Press:** Opens a modal with detailed nutritional breakdown.

**Design:**

* **Box Background Color:** Subtle variations of light gray to medium gray.
* **Border:** Thin, solid border (1px) in a slightly darker gray for differentiation.
* **Padding:** Around 10-20px inside each box to ensure content is not cramped.
* **Font:**
  + **Title:** Sans-serif, bold, around 16-18px.
  + **Numeric Display:** Sans-serif, medium weight, around 14-16px.
* **Icons:** Grayscale icons where necessary to represent different statistics.

**Interactive Elements:**

* **Hover Effects:**
  + Tooltips providing additional information or breakdowns.
  + Tooltips have a slight fade-in effect and are positioned above or beside the relevant element.
  + **Color Scheme:** Dark gray background with white text.
* **Tap/Press Interactions:**
  + On mobile, tapping a box brings up more details in a modal.
  + Modals slide up from the bottom and cover about 70% of the screen, with a close button at the top-right.
  + **Modal Content:** Detailed view with charts, logs, and further breakdowns.

**Responsive Design:**

* **Desktop:**
  + Grid layout with 2-3 columns.
  + Elements adjust in size but maintain clarity and readability.
* **Tablet:**
  + Grid layout with 2 columns.
  + Elements resize appropriately, maintaining padding and spacing.
* **Mobile:**
  + Single column layout.
  + Elements stack vertically, ensuring clear separation and readability.
  + Interactive elements remain touch-friendly with larger tap targets.

**Visual Summary**

* **Position:** Below the Next Meal Widget, lower half of the screen.
* **Layout:** Grid layout, responsive adjustments for different screen sizes.
* **Boxes:**
  + **Caloric Intake:** Progress bar, numeric display, hover effects, tap interactions.
  + **Water Consumption:** Visual indicator, numeric display, hover effects, tap interactions.
  + **Activity Level:** Progress bar, numeric display, hover effects, tap interactions.
  + **Sleep Duration:** Bar chart, numeric display, hover effects, tap interactions.
  + **Nutrient Balance:** Pie chart, numeric display, hover effects, tap interactions.
* **Design:** Subtle borders, light to medium gray backgrounds, sans-serif fonts, grayscale icons.
* **Interactive Elements:** Tooltips, modals, hover effects, tap interactions.
* **Responsive Design:** Adapts to desktop, tablet, and mobile screens, ensuring readability and usability.

**Backend Components for Health Statistics (Expanded)**

1. **Data Collection and Storage**
   * **Database Schema:**
     + **Users Collection/Table:** Stores user profiles, including user IDs, names, and settings.
     + **Health Metrics Collection/Table:** Stores health statistics such as caloric intake, water consumption, activity level, sleep duration, and nutrient balance. Each record includes user ID, timestamp, and the specific metric data.
2. **API Endpoints**
   * **Endpoints for Data Retrieval:**
     + **GET /api/health/calories:** Retrieves the daily caloric intake for the user.
     + **GET /api/health/water:** Retrieves the daily water consumption data.
     + **GET /api/health/activity:** Retrieves the daily or weekly activity level data.
     + **GET /api/health/sleep:** Retrieves the user's sleep duration data.
     + **GET /api/health/nutrients:** Retrieves the nutrient balance data.
   * **Endpoints for Data Submission/Update:**
     + **POST /api/health/calories:** Submits or updates the user's caloric intake.
     + **POST /api/health/water:** Submits or updates the user's water consumption.
     + **POST /api/health/activity:** Submits or updates the user's activity data.
     + **POST /api/health/sleep:** Submits or updates the user's sleep data.
     + **POST /api/health/nutrients:** Submits or updates the user's nutrient intake.
3. **Data Processing and Analysis**
   * **Data Aggregation:**
     + Implement aggregation functions to calculate daily totals, weekly averages, and other necessary summaries.
     + Use cron jobs or scheduled tasks to process raw data into meaningful statistics at regular intervals.
   * **Data Analysis:**
     + Develop algorithms to analyze collected data, providing insights such as progress towards goals, patterns, and trends.
4. **Real-Time Data Sync**
   * **WebSocket Implementation:**
     + Use WebSockets to provide real-time updates to the frontend when new data is available or when existing data is updated.
   * **Polling Mechanism:**
     + Alternatively, implement a polling mechanism that periodically checks for new data and updates the frontend accordingly.
5. **Security and Data Privacy**
   * **Authentication and Authorization:**
     + Implement JWT (JSON Web Token) based authentication to ensure secure access to the API endpoints.
     + Use role-based access control (RBAC) to manage permissions and ensure that users can only access their own data.
   * **Data Encryption:**
     + Encrypt sensitive data both in transit (using HTTPS) and at rest (using database encryption features).
6. **Error Handling and Logging**
   * **Centralized Error Handling:**
     + Implement middleware to handle and log errors consistently across all API endpoints.
   * **Monitoring and Alerts:**
     + Set up monitoring tools to track API performance, error rates, and other key metrics.
     + Configure alerts to notify the development team of any critical issues.

**Detailed Backend Components Breakdown**

**1. Data Collection and Storage**

**Database Schema:**

* **Users Collection/Table:**

json

Copy code

{

"userId": "string",

"name": "string",

"email": "string",

"createdAt": "timestamp",

"updatedAt": "timestamp"

}

* **Health Metrics Collection/Table:**

json

Copy code

{

"userId": "string",

"timestamp": "timestamp",

"calories": "number",

"water": "number",

"steps": "number",

"activeMinutes": "number",

"sleepHours": "number",

"proteins": "number",

"carbs": "number",

"fats": "number"

}

**2. API Endpoints**

* **GET /api/health/calories**
  + **Description:** Retrieves the daily caloric intake for the user.
  + **Response:**

json

Copy code

{

"userId": "string",

"date": "YYYY-MM-DD",

"calories": "number"

}

* **POST /api/health/calories**
  + **Description:** Submits or updates the user's caloric intake.
  + **Request Body:**

json

Copy code

{

"userId": "string",

"date": "YYYY-MM-DD",

"calories": "number"

}

* **GET /api/health/water**
  + **Description:** Retrieves the daily water consumption data.
  + **Response:**

json

Copy code

{

"userId": "string",

"date": "YYYY-MM-DD",

"water": "number"

}

* **POST /api/health/water**
  + **Description:** Submits or updates the user's water consumption.
  + **Request Body:**

json

Copy code

{

"userId": "string",

"date": "YYYY-MM-DD",

"water": "number"

}

* **GET /api/health/activity**
  + **Description:** Retrieves the daily or weekly activity level data.
  + **Response:**

json

Copy code

{

"userId": "string",

"date": "YYYY-MM-DD",

"steps": "number",

"activeMinutes": "number"

}

* **POST /api/health/activity**
  + **Description:** Submits or updates the user's activity data.
  + **Request Body:**

json

Copy code

{

"userId": "string",

"date": "YYYY-MM-DD",

"steps": "number",

"activeMinutes": "number"

}

* **GET /api/health/sleep**
  + **Description:** Retrieves the user's sleep duration data.
  + **Response:**

json

Copy code

{

"userId": "string",

"date": "YYYY-MM-DD",

"sleepHours": "number"

}

* **POST /api/health/sleep**
  + **Description:** Submits or updates the user's sleep data.
  + **Request Body:**

json

Copy code

{

"userId": "string",

"date": "YYYY-MM-DD",

"sleepHours": "number"

}

* **GET /api/health/nutrients**
  + **Description:** Retrieves the nutrient balance data.
  + **Response:**

json

Copy code

{

"userId": "string",

"date": "YYYY-MM-DD",

"proteins": "number",

"carbs": "number",

"fats": "number"

}

* **POST /api/health/nutrients**
  + **Description:** Submits or updates the user's nutrient intake.
  + **Request Body:**

json

Copy code

{

"userId": "string",

"date": "YYYY-MM-DD",

"proteins": "number",

"carbs": "number",

"fats": "number"

}

**3. Data Processing and Analysis**

* **Aggregation Functions:**
  + Calculate daily, weekly, and monthly totals.
  + Identify trends and patterns in health data.
* **Scheduled Tasks:**
  + Use cron jobs to process raw data into meaningful statistics at regular intervals (e.g., daily summaries).

**4. Real-Time Data Sync**

* **WebSocket Implementation:**
  + Set up WebSocket connections to push real-time updates to the frontend.
* **Polling Mechanism:**
  + Implement a fallback polling mechanism to periodically check for new data.

**5. Security and Data Privacy**

* **Authentication and Authorization:**
  + Implement JWT for secure access to API endpoints.
  + Use RBAC to manage permissions.
* **Data Encryption:**
  + Encrypt sensitive data in transit (HTTPS) and at rest (database encryption).

**6. Error Handling and Logging**

* **Centralized Error Handling:**
  + Implement middleware to handle and log errors consistently across all API endpoints.
* **Monitoring and Alerts:**
  + Set up monitoring tools to track API performance and error rates.
  + Configure alerts to notify the development team of critical issues.