**Next Meal Widget - Wireframe**

**Widget Layout**

**Overall Structure:**

* **Position:** Below the Nour Score Widget, spanning the full width of the screen.
* **Height:** Approximately 150-200px to ensure enough space for both the image and meal details.
* **Background Color:** Light gray or white to differentiate from the dark-themed background elements.

**Elements:**

1. **Meal Image:**
   * **Position:** Left half of the widget.
   * **Size:** Occupies 50% of the widget's width and full height.
   * **Design:**
     + **Image Quality:** High-quality, AI-generated image to visually represent the meal.
     + **Image Aspect Ratio:** Maintain a 1:1 or 4:3 aspect ratio to ensure consistency and visual appeal.
     + **Border:** Subtle border or shadow effect to make the image stand out.
   * **Interaction:**
     + **Hover Effects:** On desktop, hovering over the image shows a preview tooltip or a short description.
     + **Tap/Press Interactions:** On mobile, tapping the image opens a detailed view of the meal in a modal or new screen.
2. **Meal Details:**
   * **Position:** Right half of the widget, aligned vertically.
   * **Structure:**
     + **Meal Name:**
       - **Position:** At the top of the right half, prominently displayed.
       - **Font Size:** Approximately 20-24px, bold to stand out.
       - **Color:** Dark gray or black for high contrast.
     + **Nutritional Info:**
       - **Position:** Below the meal name.
       - **Content:** Displays key nutritional information such as calories, protein, carbs, and fats.
       - **Font Size:** Approximately 14-16px, regular weight.
       - **Layout:** Could be a horizontal list or vertical stack, depending on space.
       - **Icons:** Small icons next to each nutritional info item for quick visual identification.
     + **View Full Meal Button:**
       - **Position:** At the bottom right of the right half.
       - **Design:** A rectangular button with a subtle shadow effect.
       - **Text:** "View Full Meal" in bold, approximately 16px font size.
       - **Color:** Dark gray or black text on a light gray or white background to maintain the monochrome theme.
       - **Interaction:** Clicking the button opens a detailed view of the meal, either in a modal or a new screen.

**Interactive Elements:**

1. **Hover Effects:**
   * **Meal Image:** On desktop, hovering over the image shows a tooltip with a short description or additional information.
   * **Meal Details:** Hovering over the "View Full Meal" button changes its color slightly to indicate it is clickable.
2. **Tap/Press Interactions:**
   * **Meal Image:** On mobile, tapping the image opens a detailed view of the meal.
   * **Meal Name:** On mobile, tapping the meal name also opens the detailed view of the meal.
   * **View Full Meal Button:** Tapping the button navigates to the detailed meal view.

**Design:**

* **Theme:** Consistent with the overall monochrome theme of the app.
* **Colors:**
  + **Meal Image:** High-quality grayscale image if necessary to match the monochrome theme.
  + **Text:** Dark gray or black for high contrast against a light background.
  + **Button:** Light gray or white background with dark text.
* **Typography:** Sans-serif font family, consistent with the rest of the app.
* **Borders/Shadow Effects:** Subtle borders or shadows to differentiate elements without overwhelming the user.

**Responsive Design:**

* **Desktop:** The layout shows the meal image on the left and meal details on the right, both taking up equal width.
* **Tablet/Mobile:**
  + **Stacking:** The layout adapts by stacking the meal image on top and meal details below.
  + **Scaling:** Elements are resized to fit smaller screens, maintaining readability and usability.
  + **Interactions:** Tap interactions replace hover effects, ensuring all functionalities are accessible on touch devices.

**Visual Summary**

* **Widget (Full Width, 150-200px Height)**
  + **Left Half:** Meal Image (High-quality, AI-generated, 1:1 or 4:3 aspect ratio)
  + **Right Half:**
    - **Top:** Meal Name (20-24px, bold)
    - **Middle:** Nutritional Info (Calories, protein, carbs, etc., 14-16px)
    - **Bottom Right:** View Full Meal Button (Rectangular, "View Full Meal", 16px)
* **Interactive Elements:**
  + **Hover Effects:** Preview tooltip on meal image, color change on button hover
  + **Tap/Press Interactions:** Detailed view on tap for image and meal name, navigation on button press
* **Responsive Adjustments:**
  + **Desktop:** Horizontal layout with equal width for image and details
  + **Mobile/Tablet:** Vertical stacking, resized elements for clarity

**Backend Components Needed for the Next Meal Widget**

To support the functionality and integration of the Next Meal Widget in the Personalized Nutrition Application, several backend components are required. These components ensure the secure, efficient, and dynamic handling of user data and meal information.

**1. Data Management**

**a. Database Schema for Meal Data:**

* **Meal Collection/Table:** A collection in a NoSQL database like MongoDB or a table in an SQL database.
  + **Fields:**
    - mealId (unique identifier for the meal)
    - mealName (name of the meal)
    - imageURL (URL to the AI-generated image of the meal)
    - calories (caloric content of the meal)
    - protein (amount of protein in grams)
    - carbohydrates (amount of carbohydrates in grams)
    - fats (amount of fats in grams)
    - ingredients (list of ingredients)
    - preparationInstructions (steps to prepare the meal)
    - createdAt (timestamp of when the meal was added)
    - updatedAt (timestamp of when the meal was last updated)

**b. User-Specific Meal Data:**

* **User Meal Plan Collection/Table:** A collection/table to store user-specific meal plans.
  + **Fields:**
    - userId (reference to the user)
    - mealId (reference to the meal)
    - scheduledTime (time when the meal is scheduled to be eaten)
    - consumed (boolean flag indicating if the meal has been consumed)
    - rating (user rating for the meal)
    - feedback (user feedback for the meal)

**2. API Endpoints**

**a. Fetch Next Meal:**

* **Endpoint:** GET /api/meals/next
  + **Description:** Fetches the next scheduled meal for the user.
  + **Parameters:**
    - userId (required, to identify the user)
  + **Response:**
    - mealId
    - mealName
    - imageURL
    - calories
    - protein
    - carbohydrates
    - fats
    - scheduledTime
  + **Security:** Ensure the request is authenticated, and the user is authorized to access the meal data.

**b. View Full Meal Details:**

* **Endpoint:** GET /api/meals/:mealId
  + **Description:** Fetches detailed information for a specific meal.
  + **Parameters:**
    - mealId (required, to identify the meal)
  + **Response:**
    - mealName
    - imageURL
    - calories
    - protein
    - carbohydrates
    - fats
    - ingredients
    - preparationInstructions
  + **Security:** Ensure the request is authenticated.

**c. Update Meal Plan:**

* **Endpoint:** POST /api/meals/update
  + **Description:** Updates the user's meal plan, such as marking a meal as consumed or changing the scheduled time.
  + **Parameters:**
    - userId (required, to identify the user)
    - mealId (required, to identify the meal)
    - scheduledTime (optional, new scheduled time)
    - consumed (optional, boolean flag to mark as consumed)
  + **Response:**
    - Success or failure message.
  + **Security:** Ensure the request is authenticated and authorized.

**d. User Feedback:**

* **Endpoint:** POST /api/meals/feedback
  + **Description:** Allows users to provide feedback and rate their meals.
  + **Parameters:**
    - userId (required, to identify the user)
    - mealId (required, to identify the meal)
    - rating (required, user rating for the meal)
    - feedback (optional, user comments or feedback)
  + **Response:**
    - Success or failure message.
  + **Security:** Ensure the request is authenticated.

**3. AI-Generated Images Integration**

**a. Image Generation Service:**

* **Integration with AI Service:**
  + **Description:** Interface with an AI service that generates meal images based on the meal data.
  + **API Calls:**
    - Send meal data to the AI service and receive an image URL.
  + **Storage:** Store the received image URL in the meal data collection/table.

**b. Image Caching:**

* **Description:** Implement caching for meal images to reduce load times and improve performance.
  + **Strategy:** Use a CDN or in-app caching mechanisms to store and serve images efficiently.

**4. Real-time Data Sync**

**a. WebSocket/Real-Time Updates:**

* **Description:** Implement real-time updates for meal data and user interactions.
  + **WebSocket API:** Set up WebSocket connections to push updates to the client as soon as they are available.
  + **Use Cases:**
    - Notify users of upcoming meals.
    - Update meal status (e.g., consumed) in real-time.

**5. Security and Privacy**

**a. Authentication and Authorization:**

* **JWT Tokens:** Use JSON Web Tokens for secure authentication of API requests.
* **Role-Based Access Control (RBAC):** Ensure only authorized users can access and modify meal data.

**b. Data Encryption:**

* **HTTPS:** Ensure all API endpoints are accessible over HTTPS to encrypt data in transit.
* **Database Encryption:** Encrypt sensitive user data at rest in the database.

**6. Monitoring and Logging**

**a. Activity Logging:**

* **Description:** Log all API requests and interactions for monitoring and debugging.
* **Details Logged:**
  + User actions (e.g., viewing a meal, providing feedback)
  + API request and response details
  + Error logs and exceptions

**b. Performance Metrics:**

* **Description:** Monitor performance metrics for the API endpoints and database operations.
* **Tools:** Use monitoring tools like Prometheus, Grafana, or ELK stack to visualize and analyze performance data.