**1. Generate Palette API**

1. **Request Received**:
   * The backend receives a POST request with user inputs, such as keywords, the number of colors, and optionally the user ID.
2. **Process Input**:
   * If keywords are provided:
     + The backend uses an algorithm (AI API?) to generate colors based on the keywords and number of colors.
   * If only the number of colors is provided:
     + The backend generates a random palette.
3. **Generate Palette**:
   * A palette is created by selecting RGB values for the colors.
4. **Save Palette**:
   * The palette is saved in the database with a unique paletteId and creation timestamp.
   * The action ("Generated palette with keywords...") is added to the palette’s history.
5. **Response Sent**:
   * The backend returns a JSON response containing:
     + The paletteId and paletteName.
     + The list of colors (their RGB values).
     + The action history.

**2. Revert Palette API**

**Flow:**

1. **Request Received**:
   * The backend receives a POST request containing the paletteId and versionId to revert to.
2. **Retrieve Palette**:
   * The backend queries the database for the palette with the given paletteId.
   * It checks the history array to find the details of the specified versionId.
3. **Revert to Previous Version**:
   * The backend restores the palette’s colors to match the specified version.
   * It also updates the current palette state and appends a new history entry (e.g., "Reverted to version v1").
4. **Save Updated Palette**:
   * The reverted palette is saved back into the database.
5. **Response Sent**:
   * The backend returns the updated palette, including:
     + The reverted colors.
     + The updated history with the latest action.

**3. Load User API**

**Flow:**

1. **Request Received**:
   * The backend receives a GET request with either:
     + The user’s IP address (for unauthenticated users) or
     + The user ID (for logged-in users).
2. **Authenticate User**:
   * If an IP address is provided:
     + The backend matches the IP with its records to find the associated user.
   * If a user ID is provided:
     + The backend directly retrieves the user profile.
3. **Retrieve User Data**:
   * The backend fetches:
     + The user’s recent palettes (palette IDs and names).
4. **Response Sent**:
   * The backend sends a JSON response containing:
     + The user ID and username.
     + A list of recent palettes.