NutritionLedger Client Interaction History

1. **Consultation Logs**
    - Notes from each consultation session
    - Diet plans, workout plans, and other recommendations made

Breakdown the sub features into granular sub features

Interaction History

1. Consultation Logs

- **Session Details**
    - Date and time of consultation
    - Mode of consultation (in-person, online, phone)
    - Duration of consultation
- **Notes from Consultation**
    - Free-text field for general observations
    - Structured fields for tracking specific health metrics (e.g., weight, BMI, blood pressure)
    - Attachments to include any supporting documents or images
- **Diet Plans Provided**
    - Custom diet plans created during the consultation
    - Pre-made diet templates chosen
    - Modifications to existing diet plans

- **Workout Plans Provided**
  - Custom workout plans created during the consultation
  - Pre-made workout templates chosen
  - Modifications to existing workout plans
- **Additional Recommendations**
  - Supplement recommendations
  - Lifestyle changes (e.g., sleep, hydration)
  - Referrals to other healthcare providers
- **Follow-up Actions**
  - Schedule next consultation
  - Assign tasks or goals before next meeting
  - Notification reminders for follow-up
- **Client Feedback**
  - Client's immediate reactions or feedback
  - Any questions or concerns raised by the client
- **Status**
  - Mark consultation as completed, pending, or cancelled
  - Flag any urgent actions or notes
- **Search and Filter**
  - Ability to search past consultation logs by date, keyword, etc.
  - Filter options by status, client, or other categories
- **Export and Share**
  - Option to export consultation log as a PDF or other document formats
  - Share consultation log securely with the client or other healthcare providers

Identify models

Models

1. `ConsultationLog`

- `date_time` : DateTime
- `mode` : Enum (in-person, online, phone)
- `duration` : Time Duration (or Integer for minutes)
- `status` : Enum (completed, pending, cancelled)
- `urgent_flag` : Boolean
- `client_id` : Foreign Key (links to a Client model)

2. `SessionNote`

- `consultation_log_id` : Foreign Key (links to ConsultationLog)
- `text` : Text (for general observations)

3. ~~`HealthMetric`~~

- ~~`consultation_log_id` : Foreign Key (links to ConsultationLog)~~
- ~~`metric_type` : Enum (weight, BMI, blood pressure, etc.)~~
- ~~`value` : Float/String~~

4. ~~`Attachment`~~

- ~~`consultation_log_id` : Foreign Key (links to ConsultationLog)~~
- ~~`file` : FileField (or similar)~~

5. ~~`DietPlan`~~

- `consultation_log_id` : Foreign Key (links to ConsultationLog)
- `type` : Enum (custom, pre-made)
- `modifications` : Text (or JSON)

6. `WorkoutPlan`

- `consultation_log_id` : Foreign Key (links to ConsultationLog)
- `type` : Enum (custom, pre-made)
- `modifications` : Text (or JSON)

7. `AdditionalRecommendation`

- `consultation_log_id` : Foreign Key (links to ConsultationLog)
- `recommendation_type` : Enum (Supplement, Lifestyle, Referral)
- `description` : Text

8. `FollowUpAction`

- `consultation_log_id` : Foreign Key (links to ConsultationLog)
- `action_type` : Enum (Schedule next consultation, Assign task, Notification)
- `details` : Text (or DateTime for next consultation)

9. `ClientFeedback`

- `consultation_log_id` : Foreign Key (links to ConsultationLog)
- `feedback_text` : Text
- `questions_or_concerns` : Text

consultation_log

session_note

consultation_follow_ups

consultation_feedbacks

Perform normalization

Normalization in the context of database design is a process used to minimize data redundancy and dependency by organizing fields and table of a database. Your given models already appear to be fairly well-normalized as per the requirements you've laid out. They've been separated into distinct entities where each entity seems to represent one "thing" (e.g., a consultation log, a health metric, an attachment), and there's no obvious redundancy in the data being stored.

Here is how these models are well-designed in terms of normalization:

1. **ConsultationLog**: This is the central model and includes attributes that are solely about the consultation itself. The `client_id` field acts as a foreign key that connects it to a specific client, indicating who the consultation is for.
2. **SessionNote, HealthMetric, Attachment, DietPlan, WorkoutPlan, AdditionalRecommendation, FollowUpAction, ClientFeedback**: All of these are supporting models to `ConsultationLog`. They each contain a `consultation_log_id`, acting as a foreign key connecting them to a specific consultation. These models are designed to hold data that could occur multiple times for a single consultation, which is why they are separated out. This aligns well with the First Normal Form (1NF).
3. **Use of Enums**: Using enums for fields like `mode` in `ConsultationLog`, `metric_type` in `HealthMetric`, `type` in `DietPlan`, and so on helps to restrict the data to a fixed set, reducing the chance for anomalies.
4. **Separate Models for Different Concerns**: Different types of data are broken down into separate models (Second Normal Form, 2NF). For instance, `HealthMetric` is separate from `ConsultationLog`, making it easier to manage health metrics independently of consultations.

5. **No Multivalued Dependencies**: Your models don't have sets of fields that depend on another field, which would violate the Third Normal Form (3NF).
6. **Data Types**: The choice of data types also seems appropriate. For example, you've used DateTime for date and time, Enums for fields that should only have specific values, and Boolean for binary flags. This helps to ensure that the data stored will be of the expected type, enhancing data integrity.

Identify database constraints and indexes

Database constraints are essential for ensuring data integrity. Here are some constraints you can consider for your models when using PostgreSQL:

ConsultationLog

1. `date_time`: NOT NULL
2. `mode`: NOT NULL, CHECK (mode IN ('in-person', 'online', 'phone'))
3. `duration`: NOT NULL, CHECK (duration > 0)
4. `status`: NOT NULL, CHECK (status IN ('completed', 'pending', 'cancelled'))
5. `urgent_flag`: NOT NULL, DEFAULT false
6. `client_id`: NOT NULL, FOREIGN KEY REFERENCES Client(id)

SessionNote

1. `consultation_log_id`: NOT NULL, FOREIGN KEY REFERENCES ConsultationLog(id)
2. `text`: NOT NULL

HealthMetric

1. `consultation_log_id`: NOT NULL, FOREIGN KEY REFERENCES ConsultationLog(id)

2. `metric_type`: NOT NULL, CHECK (metric_type IN ('weight', 'BMI', 'blood pressure', ...))
3. `value`: NOT NULL

Attachment

1. `consultation_log_id`: NOT NULL, FOREIGN KEY REFERENCES ConsultationLog(id)
2. `file`: NOT NULL

DietPlan

1. `consultation_log_id`: NOT NULL, FOREIGN KEY REFERENCES ConsultationLog(id)
2. `type`: NOT NULL, CHECK (type IN ('custom', 'pre-made'))
3. `modifications`: NOT NULL

WorkoutPlan

1. `consultation_log_id`: NOT NULL, FOREIGN KEY REFERENCES ConsultationLog(id)
2. `type`: NOT NULL, CHECK (type IN ('custom', 'pre-made'))
3. `modifications`: NOT NULL

AdditionalRecommendation

1. `consultation_log_id`: NOT NULL, FOREIGN KEY REFERENCES ConsultationLog(id)
2. `recommendation_type`: NOT NULL, CHECK (recommendation_type IN ('Supplement', 'Lifestyle', 'Referral'))
3. `description`: NOT NULL

FollowUpAction

1. `consultation_log_id`: NOT NULL, FOREIGN KEY REFERENCES ConsultationLog(id)

2. `action_type`: NOT NULL, CHECK (action_type IN ('Schedule next consultation', 'Assign task', 'Notification'))
3. `details`: NOT NULL

ClientFeedback

1. `consultation_log_id`: NOT NULL, FOREIGN KEY REFERENCES ConsultationLog(id)
2. `feedback_text`: NOT NULL
3. `questions_or_concerns`: NOT NULL

Write out the migrations for the models

```
# Create ConsultationLogs Table
class CreateConsultationLogs < ActiveRecord::Migration[6.1]
  def change
    create_table :consultation_logs do |t|
      t.datetime :date_time, null: false
      t.string :mode, null: false
      t.integer :duration, null: false
      t.string :status, null: false
      t.boolean :urgent_flag, default: false, null: false
      t.references :client, null: false, foreign_key: true

      t.timestamps
    end
  end
end

# Create SessionNotes Table
class CreateSessionNotes < ActiveRecord::Migration[6.1]
  def change
```

```
    create_table :session_notes do |t|
      t.text :text, null: false
      t.references :consultation_log, null: false, foreign_ke
y: true

      t.timestamps
    end
  end
end

# Create HealthMetrics Table
class CreateHealthMetrics < ActiveRecord::Migration[6.1]
  def change
    create_table :health_metrics do |t|
      t.string :metric_type, null: false
      t.float :value, null: false
      t.references :consultation_log, null: false, foreign_ke
y: true

      t.timestamps
    end
  end
end

# Create Attachments Table
class CreateAttachments < ActiveRecord::Migration[6.1]
  def change
    create_table :attachments do |t|
      t.string :file, null: false
      t.references :consultation_log, null: false, foreign_ke
y: true

      t.timestamps
    end
  end
end
```

```
# Create DietPlans Table
class CreateDietPlans < ActiveRecord::Migration[6.1]
  def change
    create_table :diet_plans do |t|
      t.string :type, null: false
      t.text :modifications, null: false
      t.references :consultation_log, null: false, foreign_ke
y: true

      t.timestamps
    end
  end
end

# Create WorkoutPlans Table
class CreateWorkoutPlans < ActiveRecord::Migration[6.1]
  def change
    create_table :workout_plans do |t|
      t.string :type, null: false
      t.text :modifications, null: false
      t.references :consultation_log, null: false, foreign_ke
y: true

      t.timestamps
    end
  end
end

# Create AdditionalRecommendations Table
class CreateAdditionalRecommendations < ActiveRecord::Migrati
on[6.1]
  def change
    create_table :additional_recommendations do |t|
      t.string :recommendation_type, null: false
      t.text :description, null: false
```

```
      t.references :consultation_log, null: false, foreign_ke
y: true

      t.timestamps
    end
  end
end

# Create FollowUpActions Table
class CreateFollowUpActions < ActiveRecord::Migration[6.1]
  def change
    create_table :follow_up_actions do |t|
      t.string :action_type, null: false
      t.text :details, null: false
      t.references :consultation_log, null: false, foreign_ke
y: true

      t.timestamps
    end
  end
end

# Create ClientFeedbacks Table
class CreateClientFeedbacks < ActiveRecord::Migration[6.1]
  def change
    create_table :client_feedbacks do |t|
      t.text :feedback_text, null: false
      t.text :questions_or_concerns, null: false
      t.references :consultation_log, null: false, foreign_ke
y: true

      t.timestamps
    end
  end
end
```

Consultation log management feature

Consultation Log Management:

1. **Create New Consultation Log**
    - Add new consultation log with all necessary details.
2. **View Consultation Log**
    - See the details of a specific consultation log.
3. **Update/Edit Consultation Log**
    - Modify existing consultation log details.
4. **Delete Consultation Log**
    - Remove an existing consultation log.
5. **Search for Consultation Log**
    - Use keywords, dates, or other filters to find specific logs.
6. **Sort Consultation Logs**
    - Sort logs based on date, status, client name, etc.
7. **List All Consultation Logs**
    - View a list of all consultation logs.
8. **Export Consultation Log**
    - Download log details as PDF, Excel, or other formats.
9. **Share Consultation Log**
    - Share the log details securely with clients or other healthcare providers.
10. **Print Consultation Log**

- Print the details of the log directly from the software.

11. **Duplicate Consultation Log**
    - Create a new log based on an existing one.
12. **Change Status**
    - Update the status of the log to "Pending", "Completed", or "Cancelled".
13. **Set Urgent Flag**
    - Mark the consultation log as urgent for immediate action or review.
14. **Attach Files**
    - Add attachments related to the consultation like reports, images, etc.
15. **Add Health Metrics**
    - Record health metrics like weight, BMI, and blood pressure.
16. **Add Session Notes**
    - Write or edit notes from the consultation session.
17. **Assign Diet Plans**
    - Add or edit diet plans directly linked to the consultation log.
18. **Assign Workout Plans**
    - Add or edit workout plans directly linked to the consultation log.
19. **Additional Recommendations**
    - Add or modify additional advice, such as supplement recommendations, lifestyle changes, or referrals.
20. **Follow-Up Actions**
    - Set reminders, assign tasks or schedule next consultations.
21. **Client Feedback**
    - Record immediate reactions or feedback from the client, as well as any questions or concerns they may have.

Title: Creation of a new Consultation Log

"As a nutritionist, I want to create a new Consultation Log so that I can keep detailed records of client interactions, plans, and recommendations."

Acceptance Criteria:

1. **Basic Consultation Information**
    - Given I am an authenticated nutritionist
      When I visit the "New Consultation Log" page
      Then I should see a form with fields: "Date and Time", "Mode of Consultation", "Duration", "Status", "Urgent Flag", and a "Save Log" button.
2. **Adding Session Notes**
    - Given I am filling out a new Consultation Log
      When I scroll to the "Session Notes" section
      Then I should see a text area where I can add general observations and specific health metrics.
3. **Health Metrics**
    - Given I am filling out a new Consultation Log
      When I scroll to the "Health Metrics" section
      Then I should be able to add structured fields for tracking specific health metrics like weight, BMI, and blood pressure.
4. **Attachments**
    - Given I am filling out a new Consultation Log
      When I scroll to the "Attachments" section
      Then I should be able to upload supporting documents or images.
5. **Diet and Workout Plans**
    - Given I am filling out a new Consultation Log
      When I scroll to the "Diet and Workout Plans" section
      Then I should see options to either select pre-made templates or create custom plans.

6. **Additional Recommendations**
    - Given I am filling out a new Consultation Log
      When I scroll to the "Additional Recommendations" section
      Then I should be able to add recommendations for supplements, lifestyle changes, or referrals to other healthcare providers.
7. **Follow-up Actions and Client Feedback**
    - Given I am filling out a new Consultation Log
      When I scroll to the "Follow-up Actions" and "Client Feedback" sections
      Then I should be able to schedule the next consultation, assign tasks, and capture immediate client reactions or feedback.
8. **Saving the Consultation Log**
    - Given I have filled in all the necessary fields in the Consultation Log
      When I click on the "Save Log" button
      Then the Consultation Log should be saved to the database
      And I should be redirected to the Consultation Log's details page
      And I should see a confirmation message "Consultation Log was successfully created."

Title: Updating an Existing Consultation Log

"As a nutritionist, I want to update an existing Consultation Log so that I can make changes or add new information to client interactions, plans, and recommendations."

Acceptance Criteria:

1. **Access to Edit Mode**
    - Given I am an authenticated nutritionist
      When I am viewing an existing Consultation Log

Then I should see an "Edit" button that takes me to the "Edit Consultation Log" page.

2. **Basic Consultation Information**
    - Given I am in edit mode for a Consultation Log
      When I look at the form fields
      Then I should see the existing details pre-filled in the fields like "Date and Time", "Mode of Consultation", "Duration", "Status", "Urgent Flag", etc.
3. **Updating Session Notes**
    - Given I am editing a Consultation Log
      When I scroll to the "Session Notes" section
      Then I should be able to update the existing text area for general observations and specific health metrics.
4. **Updating Health Metrics**
    - Given I am editing a Consultation Log
      When I scroll to the "Health Metrics" section
      Then I should be able to update or add new structured fields for specific health metrics like weight, BMI, and blood pressure.
5. **Updating Attachments**
    - Given I am editing a Consultation Log
      When I scroll to the "Attachments" section
      Then I should be able to update or upload new supporting documents or images.
6. **Updating Diet and Workout Plans**
    - Given I am editing a Consultation Log
      When I scroll to the "Diet and Workout Plans" section
      Then I should be able to update the existing diet and workout plans, or select new pre-made templates or create new custom plans.
7. **Updating Additional Recommendations**
    - Given I am editing a Consultation Log
      When I scroll to the "Additional Recommendations" section

Then I should be able to update or add new recommendations for supplements, lifestyle changes, or referrals to other healthcare providers.

8. **Updating Follow-up Actions and Client Feedback**
    - Given I am editing a Consultation Log
      When I scroll to the "Follow-up Actions" and "Client Feedback" sections
      Then I should be able to update the schedule for the next consultation, update assigned tasks, and capture or update immediate client reactions or feedback.
9. **Saving the Updated Consultation Log**
    - Given I have made the desired updates in the Consultation Log
      When I click on the "Save Changes" button
      Then the Consultation Log should be updated in the database
      And I should be redirected back to the Consultation Log's details page
      And I should see a confirmation message "Consultation Log was successfully updated."

Deleting an Existing Consultation Log

"As a nutritionist, I want to have the ability to delete a Consultation Log so that I can remove records that are no longer needed, erroneous, or have been created by mistake."

Acceptance Criteria:

1. **Access to Delete Option**
    - Given I am an authenticated nutritionist
      When I am viewing an existing Consultation Log
      Then I should see a "Delete" button.
2. **Confirmation Prompt**
    - Given I have clicked the "Delete" button
      When I am prompted to confirm the action
      Then I should see a message asking if I am sure I want to delete the

Consultation Log
And the message should have "Confirm" and "Cancel" options.

3. **Cancel Deletion**
   - Given the confirmation prompt is displayed
     When I click on "Cancel"
     Then the Consultation Log should not be deleted
     And I should remain on the Consultation Log's details page.
4. **Confirm Deletion**
   - Given the confirmation prompt is displayed
     When I click on "Confirm"
     Then the Consultation Log should be permanently removed from the database
     And I should be redirected to the list of remaining Consultation Logs.
5. **Deletion Confirmation Message**
   - Given I have confirmed the deletion of the Consultation Log
     When I am redirected to the list of remaining Consultation Logs
     Then I should see a confirmation message that states "Consultation Log was successfully deleted."
6. **Check Database**
   - Given the Consultation Log has been deleted
     When I search for the same Consultation Log in the list or database
     Then I should not be able to find it, confirming it has been permanently deleted.

Viewing All Consultation Logs for a Specific Client

"As a nutritionist, I want to view all consultation logs for a specific client so that I can have a comprehensive understanding of our consultation history and make

more informed recommendations."

Acceptance Criteria:

1. **Given** I am an authenticated nutritionist

   When I visit the "Client Profile" page for a specific client

   Then I should see a section or tab labeled "Consultation Logs."
2. **Given** I am on the "Consultation Logs" section of a specific client's profile

   When I look at the section

   Then I should see a list of all consultation logs related to that client, sorted by date by default.
3. **Given** I am viewing the list of consultation logs for a specific client

   When I click on a consultation log entry

   Then I should be taken to a detailed view of that specific consultation log.
4. **Given** I am viewing the list of consultation logs for a specific client

   When I use the search or filter options

   Then I should be able to find consultation logs based on date, status, or other available filters.
5. **Given** I am viewing the list of consultation logs for a specific client

   When I look at each log entry in the list

   Then I should see summarized information such as date, status, and any urgent flags.
6. **Given** I am viewing the list of consultation logs

When I look at the list

Then there should be an option to export or print the list for offline review.

Viewing a Specific Consultation Log for a Client

"As a nutritionist, I want to view a specific consultation log for a particular client so that I can understand all the details of that consultation, including notes, health metrics, and plans, to make future consultations more effective."

Acceptance Criteria:

1. **Given** I am an authenticated nutritionist

   When I visit the "Client Profile" page of a specific client

   Then I should see a section or tab labeled "Consultation Logs."
2. **Given** I am on the "Consultation Logs" section of a specific client's profile

   When I click on a specific consultation log entry

   Then I should be redirected to a detailed view of that specific consultation log.
3. **Given** I am viewing a specific consultation log for a client

   When I look at the page

   Then I should see all the details of the consultation:

   - Date and time
   - Mode of consultation
   - Duration

- Status and any urgent flags
- Notes
- Health Metrics
- Diet Plans
- Workout Plans
- Additional Recommendations
- Follow-Up Actions
- Client Feedback

4. **Given** I am on the page for a specific consultation log

   When I look at the details

   Then all information should be clearly organized in respective sections or tabs for easy navigation.
5. **Given** I am on the page for a specific consultation log

   When I look at the page

   Then I should have the option to edit or delete the consultation log, as well as to export the data for offline use.

Searching for a Specific Consultation Log for a Client

"As a nutritionist, I want to search for a specific consultation log for a particular client so that I can quickly access relevant information for review, analysis, and future planning."

Acceptance Criteria:

1. **Given** I am an authenticated nutritionist

   When I visit the "Client Profile" page of a specific client

   Then I should see a section or tab labeled "Consultation Logs."
2. **Given** I am on the "Consultation Logs" section of a specific client's profile

   When I look at the page

   Then I should see a search bar or a search icon.
3. **Given** I see the search bar or search icon

   When I click on it

   Then I should have options to input or select search criteria, such as:

   - Date range
   - Mode of consultation (in-person, online, phone)
   - Status (completed, pending, cancelled)
   - Keywords (from notes, diet plans, etc.)
4. **Given** I have entered one or multiple search criteria

   When I execute the search

   Then I should see a list of consultation logs that match the criteria.
5. **Given** I see a list of matching consultation logs

   When I click on a specific entry from the list

   Then I should be redirected to the detailed view of that specific consultation log.

Sorting Consultation Logs for a Specific Client

"As a nutritionist, I want to sort the consultation logs for a specific client so that I can better organize and review their historical data for effective healthcare management."

Acceptance Criteria:

1. **Given** I am an authenticated nutritionist

   When I visit the "Client Profile" page of a specific client

   Then I should see a section or tab labeled "Consultation Logs."
2. **Given** I am on the "Consultation Logs" section of a specific client's profile

   When I look at the list of consultation logs

   Then I should see a sorting option or icon near the top of the list.
3. **Given** I see the sorting option or icon

   When I click on it

   Then I should have options to sort the consultation logs by:
   - Date and time of consultation (ascending or descending)
   - Mode of consultation (in-person, online, phone)
   - Status (completed, pending, cancelled)
   - Urgency flag (urgent or not)
4. **Given** I have selected one or multiple sorting criteria

   When I execute the sort

Then the list of consultation logs should rearrange according to the selected criteria.

5. **Given** the list has been sorted

   When I browse through it

   Then the sorting should be clearly indicated, such as highlighted column headers or sorting icons.

Dataflow for consultation log features

```
// Pseudocode representing data flow in a Rails application for

1. Routes (config/routes.rb)

   Define route for "/clients/:client_id/consultation_logs/new"
   Define route for POST "/clients/:client_id/consultation_logs"
   Define route for GET "/clients/:client_id/consultation_logs/
   Define route for PATCH "/clients/:client_id/consultation_logs
   Define route for DELETE "/clients/:client_id/consultation_log
   Define route for GET "/clients/:client_id/consultation_logs/
   Define route for GET "/clients/:client_id/consultation_logs/s
   Define route for POST "/clients/:client_id/consultation_logs/

2. When a user visits "/clients/:client_id/consultation_logs/new

   The Router:
   - Matches the URL to the "consultation_logs#new" route defini
   - Calls the `new` action in the ConsultationLogsController.
```

```
   ConsultationLogsController:

   Procedure new
       Create a new ConsultationLog instance and assign it to @c
       Render the "new" view
   End Procedure

3. The "new" view (app/views/consultation_logs/new.html.erb)

   // Main Form

   Display form for ConsultationLog and a "Bulk Import" button:
   Use @consultation_log to pre-fill form fields (if needed)
     Include a "session notes" section for adding structure fiel
   Include a "Health Metrics" section for adding structured fiel
   Include an "Attachments" section for uploading files
   Include a "Diet and Workout Plans" section for templates or c
   Include an "Additional Recommendations" section for adding sp
   Include a "Follow-up Actions" and "Client Feedback" section f
   Submitting the form sends a POST request to "/clients/:client

4. When the form is submitted with a POST request to "/clients/

   The Router:
   - Matches the POST request to the "consultation_logs#create"
   - Calls the `create` action in the ConsultationLogsControlle

   ConsultationLogsController:

   Procedure create
       Create a new ConsultationLog instance with data from the
       If @consultation_log saves successfully
           Save associated Health Metrics, Attachments, Diet and
           Redirect to the consultation log's show page
           Show a confirmation message "Consultation Log was suc
       Else
```

```
            Render the "new" view with error messages
        End If
    End Procedure

5. If there are errors

   The "new" view is rendered again:
   Display the form for ConsultationLog, now with error messages
   Use @consultation_log to pre-fill form fields and show errors

5. If there are errors

   The "new" view is rendered again:
   Display the form for consultation log, now with error message
   Use @consultation_log to pre-fill form fields and show errors

6. Edit/Update

   When a user visits "/clients/:client_id/consultation_logs/:id

   The Router:
   - Matches the URL to the "consultation_logs#edit" route defin
   - Calls the `edit` action in the ConsultationLogsController.

   ConsultationLogsController:

   Procedure edit
       Find the ConsultationLog by id and assign it to @consulta
       Render the "edit" view
   End Procedure

   When the form is submitted with a PATCH request to "/clients/

   The Router:
   - Matches the PATCH request to the "consultation_logs#update"
   - Calls the `update` action in the ConsultationLogsControlle
```

```
   ConsultationLogsController:

   Procedure update
       Find the ConsultationLog by id and assign it to @consulta
       Update @consultation_log with new data from the form
       If @consultation_log updates successfully
           Redirect to the consultation log's show page for the
       Else
           Render the "edit" view with error messages
       End If
   End Procedure

7. Destroy

   When a user triggers a DELETE request to "/clients/:client_id

   The Router:
   - Matches the DELETE request to the "consultation_logs#destro
   - Calls the `destroy` action in the ConsultationLogsControlle

   ConsultationLogsController:

   Procedure destroy
       Find the ConsultationLog by id and assign it to @consulta
       Delete @consultation_log
       Redirect to the consultation logs index page for the spec
   End Procedure


8. Search for Consultation Log

   Update Routes (config/routes.rb):

   Define route for GET "/clients/:client_id/consultation_logs/s
```

```
   When a user visits "/clients/:client_id/consultation_logs"

   Add a Search Form:
   - Include a form with a text field for search queries and a '
   - Submitting the form sends a GET request to "/clients/:clier

   The Router:
   - Matches the GET request to the "consultation_logs#search" r
   - Calls the `search` action in the ConsultationLogsControlle

   ConsultationLogsController:

   Procedure search
       Take the search query from the GET params
       Use the query to search through ConsultationLog records a
       Render the "search_results" view
   End Procedure

9. The "search_results" view (app/views/consultation_logs/searcl

   Display the search results:
   Loop through @search_results to display each matching Consult
   Include links to view/edit/delete each ConsultationLog.

10. Sort Consultation Logs

    Update Routes (config/routes.rb):

    Define route for GET "/clients/:client_id/consultation_logs/

    When a user visits "/clients/:client_id/consultation_logs"

    Add Sort Options:
    - Include dropdown or buttons to sort by different fields (e
    - Selecting a sort option sends a GET request to "/clients/
```

```
    The Router:
    - Matches the GET request to the "consultation_logs#sort" ro
    - Calls the `sort` action in the ConsultationLogsController

    ConsultationLogsController:

    Procedure sort
        Take the sort option from the GET params
        Use the sort option to order ConsultationLog records and
        Render the "sorted_logs" view
    End Procedure

11. The "sorted_logs" view (app/views/consultation_logs/sorted_l

    Display the sorted logs:
    Loop through @sorted_logs to display each sorted Consultatio
    Include links to view/edit/delete each ConsultationLog.

12. Export Consultation Logs

    Update Routes (config/routes.rb):

    Define route for GET "/clients/:client_id/consultation_logs/

    When a user visits "/clients/:client_id/consultation_logs"

    Add Export Button:
    - Include a button to export logs
    - Clicking the button sends a GET request to "/clients/:clie

    The Router:
    - Matches the GET request to the "consultation_logs#export"
    - Calls the `export` action in the ConsultationLogsControlle

    ConsultationLogsController:
```

```
    Procedure export
        Fetch all ConsultationLog records for the specific clien
        Generate a downloadable file (CSV, PDF, etc.) from @cons
        Send the generated file to the user
    End Procedure

13. Share Consultation Log via Email

    Update Routes (config/routes.rb):

    Define route for POST "/clients/:client_id/consultation_logs

    When a user is viewing a specific Consultation Log:

    Add Share via Email Button:
    - Include a "Share via Email" button.
    - Clicking the button opens a form to enter the email addres
    - Submitting the form sends a POST request to "/clients/:cli

    The Router:
    - Matches the POST request to the "consultation_logs#share"
    - Calls the `share` action in the ConsultationLogsControlle

    ConsultationLogsController:

    Procedure share
        Fetch the ConsultationLog record by its id and assign it
        Fetch the email address from the form parameters
        Use a mailer to send the @consultation_log details to th
        If the email sends successfully
            Redirect back to the consultation log's show page
            Show a confirmation message "Consultation log was su
        Else
            Show an error message "Unable to share consultation
        End If
    End Procedure
```

```
14. Duplicate Consultation Log

    Update Routes (config/routes.rb):

    Define route for POST "/clients/:client_id/consultation_logs

    When a user is viewing a specific Consultation Log:

    Add Duplicate Button:
    - Include a "Duplicate" button.
    - Clicking the button sends a POST request to "/clients/:cli

    The Router:
    - Matches the POST request to the "consultation_logs#duplica
    - Calls the `duplicate` action in the ConsultationLogsContro

    ConsultationLogsController:

    Procedure duplicate
        Fetch the ConsultationLog record by its id and assign it
        Create a new instance of ConsultationLog and copy attrib
        If the new instance saves successfully
            Redirect to the new consultation log's show page
            Show a confirmation message "Consultation log was su
        Else
            Redirect back to the original consultation log's sho
            Show an error message "Unable to duplicate consultat
        End If
    End Procedure

15 Import consultation log

        Define route for POST "/clients/:client_id/consultation_

    When the "Import" button is clicked with a POST request to "
```

```
   The Router:
   - Matches the POST request to the "consultation_logs#import"
   - Calls the `import` action in the ConsultationLogsControlle

   ConsultationLogsController:

   Procedure import
       Read the imported file
       For each row in the file
           Create a new ConsultationLog instance with data from
           If @consultation_log saves successfully
               Continue
           Else
               Log the error and continue
           End If
       End For
       Redirect to the consultation logs list page
       Show a confirmation message "Consultation Logs were succe
   End Procedure


16. Bulk import consultation logs

     When the "Bulk Import" button is clicked with a POST reques

   The Router:
   - Matches the POST request to the "consultation_logs#bulk_imp
   - Calls the `bulk_import` action in the ConsultationLogsContr

   ConsultationLogsController:

   Procedure bulk_import
       Read the imported bulk file (CSV, Excel, etc.)
       For each row in the file
           Create a new ConsultationLog instance with data from
```

```
            If @consultation_log saves successfully
                Continue
            Else
                Log the error and continue
            End If
        End For
        Redirect to the consultation logs list page
        Show a confirmation message "Consultation Logs were succe
    End Procedure


17. Bulk Export Consultation Logs

    Update Routes (config/routes.rb):

    Define route for POST "/clients/:client_id/consultation_logs

    When a user is viewing the list of Consultation Logs for a c

    Add Checkbox:
    - Include a checkbox next to each consultation log.

    Add Bulk Export Button:
    - Include a "Bulk Export" button.
    - Clicking the button sends a POST request to "/clients/:cli

    The Router:
    - Matches the POST request to the "consultation_logs#bulk_ex
    - Calls the `bulk_export` action in the ConsultationLogsCont

    ConsultationLogsController:

    Procedure bulk_export
        Fetch the selected ConsultationLog records by their IDs
        Generate a CSV/PDF (or any other format) from @consultat
```

```
        Send the generated file to the client
    End Procedure
```

Models pseudocode for consultation log feature:

```
// Pseudocode for ConsultationLog model in a Rails application

Model ConsultationLog

    // Attributes
    Declare date_time as DateTime
    Declare mode as String
    Declare duration as Integer
    Declare status as String
    Declare urgent_flag as Boolean, default: false
    Declare client_id as Integer

    // Associations
    A ConsultationLog belongs to a Client
    A ConsultationLog has many SessionNotes
    A ConsultationLog has many HealthMetrics
    A ConsultationLog has many Attachments
    A ConsultationLog has many DietPlans
    A ConsultationLog has many WorkoutPlans
    A ConsultationLog has many AdditionalRecommendations
    A ConsultationLog has many FollowUpActions
    A ConsultationLog has many ClientFeedbacks

    // Validations
    date_time must be present
```

```
    mode must be present and one of ['in-person', 'online', 'pho
    duration must be present and greater than 0
    status must be present and one of ['completed', 'pending',
    urgent_flag must be present
    client_id must be present

End Model

// Similar pseudocode can be written for each of the other model

Model SessionNote

    // Attributes
    Declare consultation_log_id as Integer
    Declare text as Text

    // Associations
    A SessionNote belongs to a ConsultationLog

    // Validations
    consultation_log_id must be present
    text must be present

End Model

Model HealthMetric

    // Attributes
    Declare consultation_log_id as Integer
    Declare metric_type as String
    Declare value as String

    // Associations
    A HealthMetric belongs to a ConsultationLog

    // Validations
```

```
    consultation_log_id must be present
    metric_type must be present and one of ['weight', 'BMI', 'bl
    value must be present

End Model

// Pseudocode for Attachment model in a Rails application

Model Attachment

    // Attributes
    Declare consultation_log_id as Integer
    Declare file as File

    // Associations
    An Attachment belongs to a ConsultationLog

    // Validations
    consultation_log_id must be present
    file must be present

End Model

// Pseudocode for DietPlan model in a Rails application

Model DietPlan

    // Attributes
    Declare consultation_log_id as Integer
    Declare type as String
    Declare modifications as Text

    // Associations
    A DietPlan belongs to a ConsultationLog

    // Validations
```

```
    consultation_log_id must be present
    type must be present and one of ['custom', 'pre-made']
    modifications must be present

End Model

// Pseudocode for WorkoutPlan model in a Rails application

Model WorkoutPlan

    // Attributes
    Declare consultation_log_id as Integer
    Declare type as String
    Declare modifications as Text

    // Associations
    A WorkoutPlan belongs to a ConsultationLog

    // Validations
    consultation_log_id must be present
    type must be present and one of ['custom', 'pre-made']
    modifications must be present

End Model

// Pseudocode for AdditionalRecommendation model in a Rails appl

Model AdditionalRecommendation

    // Attributes
    Declare consultation_log_id as Integer
    Declare recommendation_type as String
    Declare description as Text

    // Associations
    An AdditionalRecommendation belongs to a ConsultationLog
```

```
    // Validations
    consultation_log_id must be present
    recommendation_type must be present and one of ['Supplement
    description must be present

End Model

// Pseudocode for FollowUpAction model in a Rails application

Model FollowUpAction

    // Attributes
    Declare consultation_log_id as Integer
    Declare action_type as String
    Declare details as Text

    // Associations
    A FollowUpAction belongs to a ConsultationLog

    // Validations
    consultation_log_id must be present
    action_type must be present and one of ['Schedule next consu
    details must be present

End Model

// Pseudocode for ClientFeedback model in a Rails application

Model ClientFeedback

    // Attributes
    Declare consultation_log_id as Integer
    Declare feedback_text as Text
    Declare questions_or_concerns as Text
```

```
    // Associations
    A ClientFeedback belongs to a ConsultationLog

    // Validations
    consultation_log_id must be present
    feedback_text must be present
    questions_or_concerns must be present

End Model
```

Form for user input

```
// Pseudocode for Consultation Log Form in a Rails Application V

Form for @consultation_log do |f|

    // Date and Time Field
    Label "Date and Time"
    f.datetime_field :date_time

    // Mode of Consultation Field
    Label "Mode"
    f.select :mode, options: ['in-person', 'online', 'phone']

    // Duration Field
    Label "Duration (in minutes)"
    f.number_field :duration, min: 1

    // Status Field
    Label "Status"
    f.select :status, options: ['completed', 'pending', 'cancel
```

```
    // Urgent Flag Checkbox
    Label "Mark as Urgent"
    f.check_box :urgent_flag

    // Client ID (this assumes a dropdown; could be implemented
    Label "Client"
    f.collection_select :client_id, @clients, :id, :name

    // Session Note Text
    Label "Session Notes"
    Textarea for @session_note[:text]

    // Health Metrics
    Label "Health Metrics"
    // Logic to dynamically add metric fields (e.g., type and va

    // Attachments
    Label "Attachments"
    // Logic to upload and attach files

    // Diet Plan Fields
    Label "Diet Plan"
    f.select :diet_plan_type, options: ['custom', 'pre-made']
    Textarea for @diet_plan[:modifications]

    // Workout Plan Fields
    Label "Workout Plan"
    f.select :workout_plan_type, options: ['custom', 'pre-made']
    Textarea for @workout_plan[:modifications]

    // Additional Recommendations
    Label "Additional Recommendations"
    // Logic to dynamically add recommendations (e.g., type and

    // Follow Up Actions
```

```
    Label "Follow-Up Actions"
    // Logic to dynamically add follow-up actions (e.g., type ar

    // Client Feedback
    Label "Client Feedback"
    Textarea for @client_feedback[:feedback_text]
    Textarea for @client_feedback[:questions_or_concerns]

    // Submit Button
    f.submit "Save Consultation Log"

End Form
```

Rails terminal commands to implement this

1. Generate the `ConsultationLog` model:

```
rails generate model ConsultationLog date_time:datetime mode:string duration:integer status:string urgent_flag:boolean client:references
```

Generate the `SessionNote` model:

```
rails generate model SessionNote consultation_log:references text:text
```

Generate the `HealthMetric` model:

```
rails generate model HealthMetric consultation_log:references metric_type:string value:string
```

Generate the `Attachment` model:

```
rails generate model Attachment consultation_log:references file:string
```

Generate the `DietPlan` model:

```
rails generate model DietPlan consultation_log:references type:string modifications:text
```

Generate the `WorkoutPlan` model:

```
rails generate model WorkoutPlan consultation_log:references type:string modifications:text
```

Generate the `AdditionalRecommendation` model:

```
rails generate model AdditionalRecommendation consultation_log:references
recommendation_type:string description:text
```

Generate the `FollowUpAction` model:

```
rails generate model FollowUpAction consultation_log:references action_type:string
details:text
```

Generate the `ClientFeedback` model:

```
rails generate model ClientFeedback consultation_log:references feedback_text:text
questions_or_concerns:text
```

1. **Run Migrations**
   rails db:migrate

2. **Generate ConsultationLog Controller**
   rails generate controller ConsultationLogs