1. **How to Set Up and Run the Solution**

- **Node.js** and **npm** installed on your system.
- **PostgreSQL** database which is hosted on Neon. Tech followed the documentation which is provided

Steps:

1. Clone the Repository like it's a collaborated
2. Install Dependencies by below command
   npm install

3. Set Up Environment Variables to make a database connectivity

   Create a `.env` file in the root directory to store environment variables securely.

   Example: DATABASE_URL=postgres://myuser:mypassword
   @mydb.neon.tech:5432/mydatabase
   PORT = 3000 (to run locally)

4. Set Up the Database

   Create Tables: Project is depending on four tables based on relational

   Use the provided SQL script or run the SQL commands below to create the required tables.

Users table:

```
CREATE TABLE public.users (
id BIGINT NOT NULL,
fname VARCHAR(255) NOT NULL,
sname VARCHAR(255) NOT NULL,
profile_picture TEXT NOT NULL,
bio TEXT,
created_at TIMESTAMP NOT NULL,
CONSTRAINT users_pkey PRIMARY KEY (id)
);
```

Recommendations table:

```
CREATE TABLE public.recommendations (
id BIGINT NOT NULL,
user_id BIGINT NOT NULL,
title VARCHAR(255) NOT NULL,
caption TEXT,
category VARCHAR(50) NOT NULL,
created_at TIMESTAMP NOT NULL,
CONSTRAINT recommendations_pkey PRIMARY KEY (id),
CONSTRAINT recommendations_user_id_fkey FOREIGN KEY (user_id)
REFERENCES users(id)
);
```

Collections table:

```
CREATE TABLE public.collections (
id BIGSERIAL PRIMARY KEY,
user_id BIGINT REFERENCES public.users(id),
name VARCHAR(255) NOT NULL,
description TEXT,
created_at TIMESTAMP NOT NULL
);
```

Collection_recommendation table:

```
CREATE TABLE public.collection_recommendations (
id BIGSERIAL PRIMARY KEY,
collection_id BIGINT REFERENCES public.collections(id),
recommendation_id BIGINT REFERENCES public.recommendations(id)
```

);

5. Start the Server
   Run the server using below command
   node server.js

2. How to Import Sample Data into the Database

To import sample data from CSV files for users and recommendations tables, use the following commands:

psql -h hostname -U username -d database -c "\copy users FROM 'path/to/users.csv' DELIMITER ',' CSV HEADER"

psql -h hostname -U username -d database -c "\copy recommendations FROM 'path/to/recommendations.csv' DELIMITER ',' CSV HEADER"

NOTE: I have used the psql shell to manipulate the data and dumped .csv files using this shell

Please do follow the same procedure for collection and collection_recommendation tables which are in .csv files

3. Assumptions Made While Designing the System:

- **User Ownership**: Only the owner of a recommendation can adds it to their collections.
- **Cascade Deletion**: When a collection is deleted, all associated entries in collection_recommendations are removed, but recommendations themselves remain unaffected.
- **Pagination**: Default pagination is set to limit 2 records per page.
- **Error Handling**: Specific error messages are returned for cases like non-existent users, collections, or recommendations, as well as authorization checks.

4. How to Test the API Endpoints:

1. Create Collection:

Endpoint: POST/ createNewCollectionByUserSelection
CURL:

```
curl --location
'http://localhost:3000/api/collections/createNewCollectionByUse
rSelection' \
--header 'Content-Type: application/json' \
--data '{
  "user_id":3,
  "name": "Charlie Movie Collectio'\''n",
  "description": "Charlie favorite movies"
}'
```

2. **Add Recommendation to Collection**:

Endpoint: POST/collections/:collectionId/ addRecommendationsToCollection
CURL:

```
curl --location
'http://localhost:3000/api/collections/10/addRecommendationsT
oCollection' \
--header 'Content-Type: application/json' \
--data '{
  "recommendationId": 25,
  "userId": 5
}'
```

3. **Remove Recommendation from Collection:**

Endpoint:
DELETE/collections/:collectionId/removeRecommendationsFromcollection/:re
commendationId

CURL:

```
curl --location --request DELETE
'http://localhost:3000/api/collections/4/removeRecommendationsFrom
collection/8' \
--header 'Content-Type: application/json' \
--data '{
```

```
    "userId":4
}'
```

4.**View collection based on User Id:**

Endpoint: GET/collections/:userId/ viewRecommendationsFromcollections

CURL:

```
curl --location
'http://localhost:3000/api/collections/5/viewRecommendationsFromcoll
ections'
```

5.**Delete the collection and collectionRecommendation:**

Endpoint:
DELETE/collections/:userId/deletecollectionAndrecommendationsByUser/:coll
ectionId

CURL:

```
curl --location --request DELETE
'http://localhost:3000/api/collections/3/deletecollectionAndrecommend
ationsByUser/11'
```

6.**View collection in Pagination Format:**

Endpoint: GET/collections/:userId/ paginationViewAsPerUser /:collectionId/ recommendations?page={OFFSET}&limit={LIMIT}

CURL:

```
curl --location
'http://localhost:3000/api/collections/4/paginationViewAsPerUser/4/rec
ommendations?page=1&limit=3'
```