Application API Flow Overview

I watched the video you sent, and here's the flow I've come up with for the application APIs. This includes the steps for handling products, applications, co-applications, collateral addresses, documents, and how all these pieces come together. I've also added a "Doubt Section" at the end, especially around a few things I think need more clarification (like product schema and how we link everything).

1. Login Page

- **Authentication Flow:**
    - Users authenticate via email and password.
    - If the credentials are valid, a JWT token is generated and saved in the database.
- **API:** `POST /login`
    - This returns a JWT token that will be used for subsequent requests.

2. Dashboard Page

- **What happens here:**
    - Users will see different operations based on their roles. For example, Admins will have access to manage all applications, while regular users might only see their own.
- **API:** Not a separate endpoint, but roles are managed after the login token is validated.

3. Application Page

- **What happens here:**
    - Users can create an application, which is linked to a specific product.
    - When creating an application, we first save the application data.
    - **Co-Application:** A co-application is then added using the parent application's ID. Co-applications are saved in the **CoApplication Schema**.

    - **Collateral Address:** We save the property address related to the application in the **ApplicationAddress Schema**, linking it via the parent application object ID.
    - **Documents:** We also allow document uploads (like images), which are saved in AWS S3, and the S3 path is stored in the **Document Schema** linked to the application ID.
    - **Provisional Sanction Letter:** We use Puppeteer to generate a provisional sanction letter in PDF format, pulling details from the schemas using relational object IDs.
- **API Endpoints:**
    - `POST /applications/new` – Create a new application.
    - `POST /applications/update` – Update an existing application.
    - `GET /applications/list` – List all applications.
    - `POST /documents/add` – Add documents to the system (upload them to AWS).
    - `POST /presignedurl` – Get a presigned URL for uploading files to AWS S3.
    - `POST /offerletter` – Generate a provisional sanction letter in PDF.

4. Existing Application

- **What happens here:**
    - Users can fetch existing applications they've created, using the userid object ID.
- **API:** `GET /applications/{userId}`
    - This will return the application details based on its ID.

5. My Profile Page

- **What happens here:**
    - Fetch user profile details using the user's object ID.
- **API:** `GET /user/myprofile`
    - This will return the profile data of the logged-in user.

6. Logout Page

- **What happens here:**
    - Logs the user out and deletes their refresh token from the database.

- **API:** `POST /logout`
    - This will log the user out and remove their refresh token from the system.

API List

1. **Authentication APIs:**
    a. `POST /login` – User login (returns JWT token)
    b. `POST /logout` – Logout (removes refresh token)
2. **Application Management APIs:**
    a. `POST /applications/new` – Create a new application
    b. `POST /applications/update` – Update an existing application
    c. `GET /applications/list` – List all applications
    d. `GET /applications/{userid}` – Fetch application by ID
    e. `POST /documents/add` – Upload documents (to AWS S3)
    f. `POST /presignedurl` – Get presigned URL for file uploads
    g. `POST /offerletter` – Generate provisional sanction letter PDF
3. **User Profile APIs:**
    a. `GET /user/myprofile` – Fetch user profile
4. **Other APIs:**
    a. `GET /banks/list` – Get list of banks from bank schema
    b. `GET /otherproof` – Get list of otherProof from otherProof schema

Doubt Section - Questions & Clarifications Needed

- **How is the mobile number verification done by registered mobile number in addhar card ?**
    - I've created a **Product Schema** where products are listed, and applications reference the **product object ID**.
    -
    - **MongoDB Schema Designs**

1. **Product Schema** (for product details):

```
{
  "productId": ObjectId,
  "productName": String,
  "productDetails": String,
  "createdAt": Date,
  "updatedAt": Date
}
```

2. **Application Schema** (for main application details):

```
{
  "applicationId": ObjectId,
  "userId": ObjectId,  // Reference to User Schema
  "productId": ObjectId,  // Reference to Product Schema
  "firstName": String,
  "createdAt": Date,
  "updatedAt": Date
}
```

3. **CoApplication Schema** (for co-applicant details):

```
{
  "coApplicationId": ObjectId,
  "applicationId": ObjectId,  // Reference to Parent Application
  "userId": ObjectId,  // Reference to User Schema (co-applicant)
  "createdAt": Date
}
```

4. **ApplicationAddress Schema** (for collateral property address):

```
{
  "addressId": ObjectId,
  "applicationId": ObjectId,  // Reference to Application Schema
  "street": String,
  "city": String,
  "state": String,
  "postalCode": String,
  "country": String,
  "createdAt": Date
}
```

5. **Document Schema** (for storing document details and S3 paths):

```
{
  "documentId": ObjectId,
  "applicationId": ObjectId,  // Reference to Application Schema
  "userId": ObjectId,  // Reference to User Schema
  "documentType": String,
  "awsS3Path": String,  // S3 document URL
  "createdAt": Date
}
```

6. **OtherProof Schema** (for storing other proofs like bank statements):
   ```
     "proofId": ObjectId,
     "name": ObjectId,
     "createdAt": Date
   }
   ```
7. **Bank Schema** (for storing bank details):

```
{
  "bankId": ObjectId,
  "bankName": String,

I've just created the basic attributes for the schema
```