Create an API system for CRUD operations on patient details for a medical-centric application. The way this data will be captured from the client side is in a multi-screen form.

Form 1 (Credential data):

1. Email (string)
2. Password (string)
3. Repeat Password (string)

Form 2 (Personal data):

1. First name (string)
2. Second name (string)
3. Mobile number (number)
4. Date of birth (datetime)
    a. Calculate age and store it in the DB (this won't be captured by UI)
5. Weight (float)
6. Height (string) (e.g. **6-3** signifies 6 feet and 3 inches)
    a. Calculate BMI and store it in the DB (this won't be captured by UI. Formula to calculate BMI is available online)
7. Country of origin (string)
8. Are you diabetic or pre-diabetic? (boolean)
9. Have you suffered any cardiac related issues in the past or are suffering currently? (boolean)
10. Do you have concerns with your blood pressure? (boolean)
11. Get disease type and description from the user.

Form 3 (Family data):

1. Father's name (string)
2. Father's age (number)
3. Father's country of origin (string)
4. Mother's name (string)
5. Mother's age (number)
6. Mother's country of origin (string)
7. Is any of your parents diabetic or pre-diabetic? (boolean) (This should be multiple disease)
8. Have any of your parents suffered any cardiac related issues in the past or are suffering currently? (boolean) (This should be multiple disease)
9. Are any of your parents concerned with their blood pressure? (boolean) (This should be multiple disease)

Form 4 (Upload documents):

1. Upload aadhar card (front)
2. Upload aadhar card (back)
3. Upload medical insurance card (front)
4. Upload medical insurance card (back)

Use cases to consider:

1. Encrypt password before storing it in the DB
2. Create different tables in the DB to store each form data (this will help you explore aggregation)
3. Store data into the DB after each form is filled and the user is moved to another step.
4. If the user does not complete the account creation process and chooses to close the app on form 3 (that means data till form 2 is already stored), then ask the user to complete the remaining forms when he/she chooses to login later.
5. Create a JWT token for each user for session management.
6. Capture this token as an access-header in update profile APIs.
7. Update profile scenarios:
    a. Users should be able to update or remove only his/her own profile, unless he/she is an admin.
    b. Add some sample profiles as an admin.
    c. An admin can update or remove any profile.
8. Upload document
    a. Upload the files on any file-hosting service (e.g. S3, MediaFire, MEGA, etc)
    b. Get the file link from these file hosting services and store those links in the DB.
9. Login with email and social login
10. Email Verification with email template

APIs to be developed:

1. POST: Sign up new patient
2. GET: Get all patients (return personal data only), Use pagination
3. PUT: Update patient details by patient ID
4. DELETE: Remove patient data by patient ID
5. POST: Patient log in