Low-Level Design: API Endpoint Definitions

This document outlines the API endpoints provided by the BioSight FastAPI application.

---

1. Authentication Routes (`/client-side/biosight/routes/auth.py`)

These endpoints handle user registration, login, and logout.

1.1 Register User

- **HTTP Method:** `POST`
- **Path:** `/api/register`
- **Function:** `register(user_data: UserCreate)`
- **Request:**
  - Body: JSON object matching the `UserCreate` Pydantic model (likely contains `email` , `password` , `name`).
- **Response:**
  - Success (200 OK): JSON object matching the `User` Pydantic model (excluding password).
  - Error (400 Bad Request): If email is already registered.
  - Error (503 Service Unavailable): If database connection is unavailable.
  - Error (500 Internal Server Error): For other database errors or if user creation fails unexpectedly.
- **Description:** Creates a new user account in the database after hashing the password.

1.2 Login User

- **HTTP Method:** `POST`
- **Path:** `/api/login`
- **Function:** `login(response: Response, form_data: OAuth2PasswordRequestForm = Depends())`
- **Request:**
  - Body: Form data (`application/x-www-form-urlencoded`) with `username` (expected to be the user's email) and `password` .
- **Response:**
  - Success (200 OK): JSON object `{"message": "Login successful"}` . Sets an `access_token` cookie (HttpOnly).

    - Error (401 Unauthorized): If email is not found or password does not match.
    - Error (503 Service Unavailable): If database connection is unavailable.
    - Error (500 Internal Server Error): For other database errors.
- **Description:** Authenticates a user based on email and password. If successful, it creates a JWT access token and sets it in an HttpOnly cookie.

1.3 Logout User

- **HTTP Method:** `POST`
- **Path:** `/api/logout`
- **Function:** `logout(response: Response)`
- **Request:**
    - Body: None. Expects a valid `access_token` cookie from a previous login.
- **Response:**
    - Success (200 OK): JSON object `{"message": "Logout successful"}`. Clears the `access_token` cookie by setting its expiration to the past.
- **Description:** Logs out the user by clearing the authentication cookie.

---

2. Main Application Routes (`/client-side/biosight/app.py`)

These endpoints handle the main application logic, including UI rendering, file uploads, and data management.

2.1 Get Login Page

- **HTTP Method:** `GET`
- **Path:** `/login`
- **Function:** `login_page(request: Request)`
- **Request:**
    - None.
- **Response:**
    - Success (200 OK): `HTMLResponse` rendering `login.html`.
- **Description:** Serves the HTML page for user login.

2.2 Get Registration Page

- **HTTP Method:** `GET`
- **Path:** `/register`
- **Function:** `register_page(request: Request)`
- **Request:**
    - None.

- **Response:**
  - Success (200 OK): `HTMLResponse` rendering `register.html`.
- **Description:** Serves the HTML page for user registration.

2.3 Get Home Page

- **HTTP Method:** `GET`
- **Path:** `/`
- **Function:** `home(request: Request, current_user: Optional[User] = Depends(get_current_user_optional))`
- **Request:**
  - Requires authentication (optional): Checks for a valid `access_token` cookie via `get_current_user_optional` dependency.
- **Response:**
  - Success (200 OK): `HTMLResponse` rendering `index.html` if the user is authenticated (passes `user` object to template).
  - Redirect (307 Temporary Redirect): Redirects to `/login` if the user is not authenticated.
- **Description:** Serves the main application page if the user is logged in; otherwise, redirects to the login page.

2.4 Get Prometheus Metrics

- **HTTP Method:** `GET`
- **Path:** `/metrics`
- **Function:** `metrics()`
- **Request:**
  - None.
- **Response:**
  - Success (200 OK): Plain text response formatted for Prometheus scraping, generated by `prometheus_client.generate_latest`.
- **Description:** Exposes application metrics (counters, histograms) for monitoring by Prometheus.

2.5 Upload Image Files

- **HTTP Method:** `POST`
- **Path:** `/upload/`
- **Function:** `upload_files(request: Request, files: list[UploadFile] = File(...), current_user: User = Depends(get_current_user))`
- **Request:**
  - Requires authentication: Uses `get_current_user` dependency.

    - Body: Form data (`multipart/form-data`) containing one or more files under the key `files`.
- **Response:**
    - Success (200 OK): `HTMLResponse` rendering `result.html` with classification results and a link to a zip file.
    - Error (400 Bad Request): If no files are provided or if any file has an invalid type.
    - Error (500 Internal Server Error): If there's an error saving metadata to the database.
- **Description:** Handles uploads of multiple image files. For each valid file, it saves the file, predicts its class using the ML model, organizes the file into a class-specific folder, saves metadata (including prediction, paths, user email, timestamp) to the database, and increments monitoring counters. Finally, it renders a results page.

2.6 Download Organized Images Zip

- **HTTP Method:** `GET`
- **Path:** `/download-zip/`
- **Function:** `download_zip(current_user: User = Depends(get_current_user))`
- **Request:**
    - Requires authentication: Uses `get_current_user` dependency.
- **Response:**
    - Success (200 OK): `FileResponse` delivering the zip archive (`application/zip`).
    - Error (404 Not Found): If the zip file doesn't exist and cannot be created.
- **Description:** Creates (if needed) and serves a zip file containing all images currently in the organized folders.

2.7 Delete Image

- **HTTP Method:** `DELETE`
- **Path:** `/delete-image/{predicted_class}/{filename}`
- **Function:** `delete_image(predicted_class: str, filename: str, current_user: User = Depends(get_current_user))`
- **Request:**
    - Requires authentication: Uses `get_current_user` dependency.
    - Path Parameters: `predicted_class` (string), `filename` (string - the randomized name).
- **Response:**
    - Success (200 OK): JSON object `{"message": "Image deleted successfully"}`.
    - Error (404 Not Found): If the image file does not exist at the specified path.
    - Error (500 Internal Server Error): If there's an error deleting the file or during database interaction.

- **Description:** Deletes the specified image file from its organized folder and attempts to delete its corresponding metadata entry from the database.

2.8 Update Image Class

- **HTTP Method:** `PUT`
- **Path:** `/update-class/{old_class}/{filename}`
- **Function:** `update_image_class(old_class: str, filename: str, new_class: str, current_user: User = Depends(get_current_user))`
- **Request:**
  - Requires authentication: Uses `get_current_user` dependency.
  - Path Parameters: `old_class` (string - the class the UI *thinks* the image is in), `filename` (string - the randomized name).
  - Query Parameter: `new_class` (string - the target class to move the image to).
- **Response:**
  - Success (200 OK): JSON object containing `message`, `old_class`, `new_class`, `is_updated`, `original_class`.
  - Error (404 Not Found): If the image metadata or the image file itself cannot be found.
  - Error (500 Internal Server Error): If there's an error moving the file or updating the database. Includes rollback attempt on DB error.
- **Description:** Moves an image file from its current class folder to a new class folder. Updates the `predicted_class` and `is_updated` fields in the database metadata. Handles cases where the file might not be in the expected `old_class` folder by searching.

2.9 Health Check

- **HTTP Method:** `GET`
- **Path:** `/health`
- **Function:** `health_check()`
- **Request:**
  - None.
- **Response:**
  - Success (200 OK): JSON object with `{"status": "ok", "mongodb": "ok", "model": "ok"}` if all checks pass.
  - Error (503 Service Unavailable): JSON object with `{"status": "error", ...}` and details if MongoDB connection fails or the model is not loaded.
- **Description:** Provides a basic health status check for the application, verifying database connectivity and model loading state.

---

3. Static File Serving

- `/static/...` : Serves static assets (CSS, JS) from the static directory.
- `/uploads/...` : Serves files directly from the `UPLOAD_FOLDER` (defined in config). *Note: This might be a security risk if not intended.*
- `/organized/...` : Serves files from the `ORGANIZED_FOLDER` (defined in config), allowing access to classified images.
