RetiSpec

Candidate home assignment (Software)

Background:

RetiSpec is a retinal imaging company, set up to provide early diagnosis of neurodegenerative diseases, focusing on Alzheimer's disease. For that purpose, we collect a large number of retinal images and analyze them. On this assignment you will be implementing a POC for our image acquisition backend.

Description:

You will implement a RESTful service that will provide CRUD (Create, Read, Update, Delete) operations for patients and acquisitions.

The patients API will include:

- Creating a new patient. A patient should have the following fields:
  - Patient ID
  - First name
  - Last name
  - Date of birth
  - Sex
- Get a patient by ID.
- Get patients by first + last name.
- Delete a patient.

The acquisitions API will include:

- Add a new acquisition for a patient. The acquisition should include:
  - Acquisition ID
  - Eye (left or right)
  - Site name
  - Date taken
  - Operator name
  - Image data - This is the image file being uploaded
- List all patient acquisitions (by patient id). Should specify all the acquisition details besides the image itself.
- Delete an acquisition (by acquisition id).
- Download an image (by acquisition id).

You can choose the web-framework that you see fit, how to package your software and deploy it. For saving the images you can use a directory in the local filesystem.

Bonus Points:

Persist your patients and acquisition data (not the image files) on a relational database of your choosing and have the RESTful service interact with the database. Design your schema to reflect the relations between the patients and acquisitions and to support the APIs you published. Choose how to deploy the database and explain in the readme.

Notes

1. We prefer that the exercise will be implemented in python, however Java is also acceptable.
2. For the handout, please upload your code to a git repository and share the link.
3. Please provide a readme file that will explain how to run the software and use it.
4. Think about how you want to deploy your service so it would be easy to run it in the cloud.
5. Think about proper REST API design – methods, status codes, URLs, etc.