Conceptualization and Integration of a Reporting Module with PACS System

Introduction

The PACS system is designed to store, manage, and retrieve medical images efficiently. These medical images are bundled together with patient demographic data which forms a DICOM structured report. These DICOM SR combined with medical analysis is converted to reports for effective diagnosis and communication with the both the patient and hospital personnel.

The reporting module interacts with the PACS System to facilitate the creation, management, and sharing of medical reports from DICOM files.

System Architecture

The Reporting Module is designed as a microservice to ensure scalability and maintainability. It comprises of the following components:

1) **Frontend Web APP**. A web interface for report creation and management using React.js or Angular frameworks.
2) **Backend Server.** Built using Django, DjangoRestFramework or FAST API: Implements and consumes PACS server RESTful API for retrieval of DICOM files. Implements server-side logic for handling report data. Handles user authentication and authorization by implementing secure login mechanisms e.g. OAuth, two-factor authentication. Implementing Role-Based Access Control i.e. assigning radiologist, technician, admin with specific permissions.
3) **Database.** Stores report templates, generated reports, and audit trails. PostgreSQL is used to store metadata, user information, and reports while the images and Reports can be stored in the same or on a NOSQL database like MongoDB.
4) **Testing Strategy**. Incorporate comprehensive unit tests for each module to ensure reliability. Build continuous Integration CI/CD pipelines for automated testing and deployment.
5) **NGINX & Kubernetes**. These are configured for deployment to ensure load balancing and auto scaling for handling large volume of requests.

System Flow and Interaction

- Image Acquisition and Transmission. Medical images from modalities are captured and transmitted to the PACS Server for indexing, storage and retrieval.
- Backend Server. The backend communicates with the PACS Server as needed. This backend also exposes an API which is consumed by the frontend app to manage reports.
- Frontend APP. The users interact with the reporting module user interface to create and manage reports which is stored in a database for future access and sharing.

Key Features

- ❈ Template-Based Reporting. The module supports predefined templates and also creation of custom report templates to adhere to each hospital's report format and standards.
- ❈ Report Management. The reporting module enables:
    - Users to manage reports i.e. CRUD operations
    - Rich text formatting and multimedia content by utilizing test editors.
    - Insertion of images and annotations from the PACS Image Viewer.
    - Collaboration tools for multi-user editing.
- ❈ Report Export.
    - Export reports in various formats (PDF, DOCX).
    - Secure sharing via email or download links.
- ❈ Audit Trail
    - Logs all changes made to reports.
    - Provides a detailed history for compliance and review.

Functionalities

1) User Management: Role-based access control for report creation and viewing.
2) Report Management: Create, edit, update and delete reports.
3) Provide collaboration tools for multi-user editing
4) Template generation: The module provides templates for different types of reports (e.g., radiology, pathology).
5) Image Embedding: Embeds images and annotations within reports.
6) Text Editor: Rich text editor for detailed report writing.
7) Automated Report Generation: Generates reports with integrated images.
8) Logs all alterations made to reports.

Testing Strategy

Write unit and integration tests to ensure

- Image Transmission performs secure and error-free transmission of images.
- Reporting Module handles report creation, editing, collaboration, and export functionalities among other necessary functions.
- Successful build and deployment of software.

Testing Tools and Frameworks

Use of frameworks like PyTest for backend server and Jest/Vitest for the frontend app to come up with effective and efficient testcases.

Use of libraries like dependabot and unittest.mock (Python) for mocking and managing project dependencies.

GitHub Actions or Jenkins for automated integration testing and deployment.

Conclusion

The Reporting Module not only enhances the functionality of the PACS system but also significantly improves the overall workflow for medical professionals. Through efficient report creation, management, and sharing, the module supports better patient care and operational efficiency in medical institutions. This integration represents a significant step towards a more connected, efficient, and effective healthcare system.