Psoriasis Image Submission and Clinical Feedback Web Application

Web Application Design
Christophe Soares
csoares@ufp.edu.pt

Project in Systems and Networks
Ivo Pereira
ivopereira@ufp.edu.pt

Second Semester 2023

University Fernando Pessoa

Faculty of Science and Technology

Introduction

Psoriasis is a skin condition that affects millions of people worldwide. Patients often need to share images of their skin with clinicians to receive proper treatment. This web application aims to provide a platform where patients can submit pictures of their psoriasis-affected skin, and authorized clinicians can provide feedback to the patients. The application will be implemented using React for the frontend and Golang with Gin for the backend. The use of microservices is encouraged, and you may also adopt another technologies stack for each microservice.

Application

The web application will have the following features:

1. User Registration: The users will be able to register themselves on the website using their email address and password. Once registered, the user will be authenticated and can access the features of the website.
   a. Users will navigate to the website and select the "Register" button.
   b. Users will be directed to a registration form where they will enter their email address and password.
   c. Once the user submits the registration form, the server will verify the email address is valid and not already registered.
   d. If the email address is valid and not already registered, the server will create a new user account and authenticate the user.
   e. The user will be redirected to the homepage of the website.

2. Image Submission: Authenticated users will be able to upload pictures of their skin affected by psoriasis. The images will be stored on the server-side.
   a. Authenticated users will navigate to the "Upload Image" page.
   b. Users will select an image file from their local device and provide a description and the body positions of the image (e.g., head, stomach, leg, arm, etc.).
   c. The server will encrypt the image using a secure encryption mechanism.
   d. The encrypted image and the image description will be stored in the database. e. The user will be redirected to the "My Images" page (i.e., gallery of images).

3. Image Management: Authenticated users will be able to view a list of their uploaded images, update and delete them. They will also be able to filter the images based on the date and body positions.
   a. Authenticated users will navigate to the "My Images" page.
   b. Users will be able to view a list of their uploaded images.

c. Users will be able to select an image and view a larger version of the image along with its description.
d. Users will be able to update an image's description.
e. Users will be able to delete an image.

4. Clinical Feedback: Users can authorize clinicians to view their uploaded images. Authorized clinicians can access the images and provide feedback to the patients. Multiple clinicians can be authorized to view the same patient's images. Clinicians can leave textual feedback for the patient.
   a. Users will navigate to the "Clinical Feedback" page.
   b. Users will select which clinicians they want to authorize to view their uploaded images.
   c. Clinicians will navigate to the "Clinical Feedback" page and view the authorized patient's uploaded images.
   d. Clinicians will be able to provide textual feedback for each image.
   e. Clinicians will be able to view their own feedback and feedback provided by other authorized clinicians.

5. Image Security: All images will be encrypted using a secure mechanism so that they can only be viewed by the patient and their authorized clinicians.
   a. All images will be encrypted using a secure encryption mechanism before being stored in the database.
   b. Only the patient and authorized clinicians will be able to view the images by logging into the web application.
   c. The web application will use appropriate security measures to protect user data, such as, SSL/TLS encryption, secure password storage, and two-factor authentication.

6. Future Enhancements:
   a. Add support for video submissions to allow users to capture the progression of their psoriasis over time.
   b. Implement a feature to allow users to share their images with others for research or educational purposes, with their consent
   c. Be able to relate a new image to a previous one in order to follow the clinical evolution of a given patient (timeline)

Overall, this protocol outlines the steps required to implement a secure and user-friendly web application that allows psoriasis patients to submit images of their affected skin and receive clinical feedback from authorized clinicians. By following this protocol, the web application will be robust and scalable, allowing for future enhancements to be added as required.

Architecture

The web application will be developed using React for the frontend and Golang with Gin for the backend. The backend may be fragmented into multiple microservices which can rely on multiple technologies such as, Java Spring Boot/Quarkus, PHP Laravel, Node.js, etc. The backend will use a PostgreSQL database to store the users and images information. The images will be encrypted using a secure encryption mechanism to ensure the confidentiality of the patient's information.

Continuous Integration / Continuous Delivery

Source Control Management (SCM)

Use a version control system like Git to manage the source code of your application. This will allow you to track changes, collaborate with others, and rollback changes if needed. You shall use branches and pull requests appropriately.

Static Code Analysis

Use a static code analysis tool, such as Qodana or SonarCloud, to analyze the code for errors and vulnerabilities. This will help you catch issues early in the development process and maintain code quality.

Unit Testing

Use a testing package like Testing, Testify, Httpexpect, among others, to write and execute unit tests for your application. This will help you ensure that your code works as expected and catch regressions as you make changes.

Continuous Integration

Use a CI tool, such as Github Actions, CircleCI, TeamCity to automatically build and test your application every time changes are pushed to the SCM repository. This will help you catch issues early and ensure that the code is always in a stable state.

Continuous Delivery/Deployment

Use a deployment tool, such as AWS CodeDeploy, AWS CodePipeline, among others to package and deploy your application to a staging environment for further testing. Once the application is tested in the staging environment, it can be automatically deployed to production. Frontend can be deployed in Vercel, Netlify, or others.

Integration with theoretical sessions

To ensure that the deployment process copes with the theoretical sessions given during the semester, you can create a schedule for when to trigger the CI/CD pipeline. For example, you can trigger the pipeline after each commit to ensure that any changes made are immediately tested and deployed.

Conclusion

This web application will provide a platform for psoriasis patients to submit pictures of their affected skin and receive clinical feedback from authorized clinicians. The application will be implemented using React for the frontend and Golang with Gin for the backend. The images will be encrypted using a secure mechanism to ensure the confidentiality of the patient's information.

	Topics	**Values**[1]
Web Application Design	Docker	4
	Backend	8
	Frontend	8
Project in Systems and Networks	SCM	2
	Static Code Analysis	4
	Unit Testing	4
	Continuous Integration	5
	Continuous Delivery/Deployment	5

[1] Any further information regarding the values should be obtained directly through the professor