Tumrly - Final Document

General Information about Project

Project Name: Tumrly
Supervisor(s): Fatoş Yarman Vural
Group Members: Onur OKUDUCU, Boran Alkım ERDOĞAN, İbrahim Akın YILMAZ, Enes ŞANLI, Mehmet Bora AKSÖZ

Project description

Tumrly is a mobile medical co-pilot application which is capable of;

- Detecting tumors from raw and unsegmented CT (Computed tomography) images of multiple organs such as lungs, liver, pancreas.
- Providing first level inference from the results of the CT scan using a medically fine-tuned LLM such as the seriousness of the patients conditions, next steps that the patient should take, what kind of procedures might be advised by the doctors from now on etc.
- Comparing the current and previous CT scan results of a cancer patient to see the evolution of the tumor.
- Providing a better patient observation system for doctors.

It uses MySQL for storing the patient and doctor related information, Springboot for backend, React Native for frontend, Gemini 1.0 for LLM and CLIP-Driven Universal Model for tumor segmentation.

Test Plan

1. Unit Testing
    a. Backend Server
        - Testing database CRUD operations and services through command line application. This includes doctor-patient information, meta data of unsegmented CT, segmented CT, inference, doctor review. Used at the early stages of the project.
        - Testing external file system operations(AWS) and services through command line application. This includes unsegmented CT(.nii.gz),

segmented CT(.nii.gz), inference(.pdf). Used at the early stages of the project.

- Testing authorization-authentication layer through command line application. Since our app encloses confidential information about patients, token-based(JWT) security is implemented and tested thoroughly. Used at the early stages of the project.
- Testing API endpoints through POSTMAN simulating front-end. Used continuously throughout the development process.
- Testing HTTP request services in the backend server using mock python servers (Django) that emulates our AI servers(API-wise) to anticipate possible errors and be prepared for them that may occur while integrating our services. Used at the late stages of the project.

b. Frontend Server

- The unit tests for the frontend part of the project is done for making sure that each and every component page that the user faces is able to accommodate extreme cases such as sudden disconnection from the internet, authentication on incorrect password etc. By backtracking the results of these tests and taking necessary measures, we improved the quality of services and obtained a normally functioning frontend.
- Also the connection between the frontend server and the backend is done by testing the results of ajax requests realized from frontend to the backend. The purpose of these tests were to make sure that the frontend is well and securely connected to the backend and the communication could be done flawlessly.

c. Segmentation Server

- The segmentation quality tests are done by the original developers of the model Liu et al. (see. https://arxiv.org/abs/2301.00785) and it provides state of the art success compared to the other tumor and organ segmentation models. The overall quality check of the model is done for making sure of the overall quality of the backbone result of our application.
- The functioning of the Segmentation Server in tandem with the backend server is tested using POSTMAN API. In order to achieve this, we compared the endpoints, data and the headers that are used to establish the backend-segmentation server communication and whenever we encountered an undesired result, we fixed it in an agile manner. By doing these, we made sure that the application is able to function without any problem even in the edge cases where for example the segmentation process does not return any tumors etc.

d. LLM Server
    - The functioning of the LLM Server with the application backend is tested by using POSTMAN API; once the LLM server endpoints are implemented and deployed, its desired functioning with the backend is obtained with endpoint and header optimization. We completed these tests to make sure that the backend server can properly communicate with the LLM.
    - The tests related to the image processing pipeline which obtains the size and the number of tumors in the organ is done by trying multiple hand-crafted tumor segmentation images and comparing the obtained results by ground-truth values. In these tests we saw that our image processing pipeline is capable of detecting the size of tumors when they are larger than approximately 0.3 cm square which is an acceptable error rate for our use cases. The purpose of testing the image processing pipeline is to make sure that we do not miss a tumor of significant size and properly provide information for the LLM model.

2. Integration Testing
    - Our project has four major modules and three integration joints. These joints are between frontend-backend server, backend server-segmentation server and backend server-inference(LLM) server respectively. Due to the fact that we design our project as a microservice architecture, we integrated our joints with ease. As we have done unit tests properly and soundly, this process was seamless. Used at the late stages of the project empirically.
3. System Testing
    - After integrating each joint seamlessly, system testing has flowed freely. We have made plenty of empirical trials for each workflow and even one of them has not failed. Used at the latest stage of the project.

Retro on Features

1. Personal Information and Medical History

Our app includes dedicated information pages for both doctors and patients. Patients also may see their previous CT scan results on CT uploading interface. All related data comes from the interaction between the front-end and back-end through http requests. Identity of the user is detected via JWT token mechanism. Verified at related unit testing.

2. Uploading CT Scan

Our app includes a dedicated CT scan uploading page. After the user uploads its CT scan, the files are directed to the back-end. Back-end server saves the files and its metadata and then sends the CT scan to the Segmentation server. Verified at related unit testing.

3. Segmentation of CT Scan

Our app has a dedicated segmentation server that receives an unsegmented CT scan file from the back-end server and responds to it back after identifying tumors. State of art model named CLIP-Driven Universal Model is used for the segmentation process. Segmentation server also extracts the most critical point of the CT scan as a png file that will be processed by the LLM server to make inference. Verified at related unit testing.

~~4. Chatbot for Medical Questions~~ LLM Inference on Segmentation Result

Note: We can think of our LLM server that makes inference on CT scan as a chatbot which accepts png as input.

Our app has a dedicated LLM server that receives CT scan images from the back-end server and responds with a pdf file that encloses the inference of the segmentation that makes notes on the size of the tumor (evaluation), recommendation on the condition etc. Verified at related unit testing.

5. Account Operations

Our app includes dedicated register and login pages for both doctors and patients. All related data comes from the interaction between the front-end and back-end through http requests. Security layer is achieved via the JWT token mechanism. Verified at related unit testing.

Note: There is no service for unauthenticated users in our app.

6. Email Service for Unauthenticated Users - Not available In the App.

This feature comes last in our prioritization list and we evaluated that this feature would not add a significant value to our app.

7. Portal for Doctors

Our app includes a dedicated portal for both doctors. Doctors are able to see their patients' previous scan results via search bar. All related data comes from the interaction between the front-end and back-end through http requests. Identity of the user is detected via JWT token mechanism. Verified at related unit testing.

8. Comparison of the Current and Previous CT Scan Results

Upon uploading a CT scan from the front-end with comparison feature enabled (button), after the segmentation process, the back-end finds the latest CT scan result from the database and sends it to the LLM server along with the current one. The LLM server then makes a comparison on those two images and reveals insights on the evolution of the tumor. Verified at related unit testing.

9. Recommendations based on Patient Data

See feature-4. While making recommendations, only the tumor test results are used. Not other types of data like blood analysis etc. Verified at related unit testing.

10. Reminder for Medication - Not available In the App.

At the initial stages of the project, we thought we could target all types of patients but after discussions we steered our project in a direction only for cancer patients. So this feature is discarded.

11. (Bonus) Symptom Checker - Not available In the App.

We did not have enough time left for this bonus feature.

12. (Added Later) Doctor Review System

Note: This feature is not originally in the kickoff document.

After observing the inference that our app produces, the doctor has a chance to make a review on the inference. After commenting on the result, this data is sent to the back-end and from there to the LLM server which feedbacks itself and improves later outputs. Verified at related unit testing.