Lung Cancer Detecting Web Application

1. Overview of the problem

Needless to say, that one of the worst diseases these days is cancer. It is deadly and very hard if not impossible to treat. The best chance of treating it a tumor is to be diagnosed as soon as possible. My project focuses on this area. It would assist doctors in the diagnosing process the most common kinds of lung cancers. With the use of Artificial Intelligence, machine learning at that, the application would tell us the probability of someone having lung cancer based on an X-Ray, MRI, or CT scan.

Users would log into the application where they will have the possibility to upload a scan of a person's lungs. The system will include an AI, machine learning model, which can detect cancer cells on the scans. It will also have the option to save the result in a database.

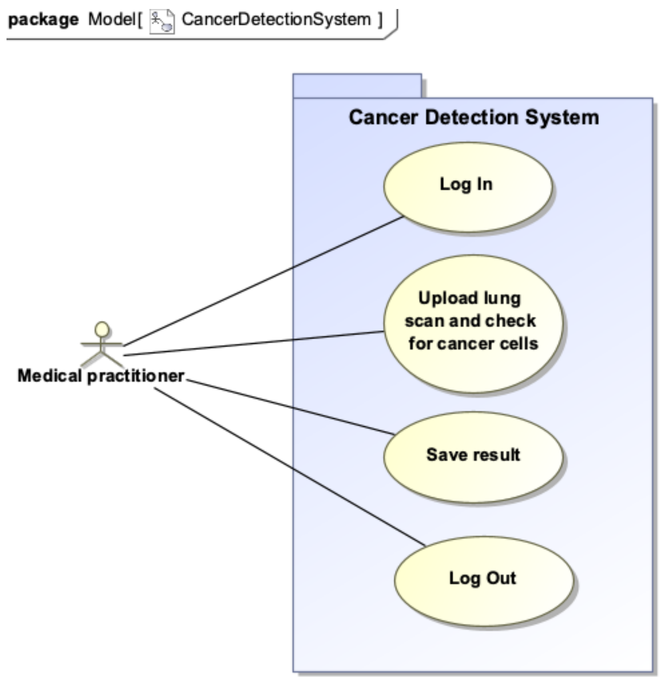
The application will be a web-based application with a backend server as well as a database.

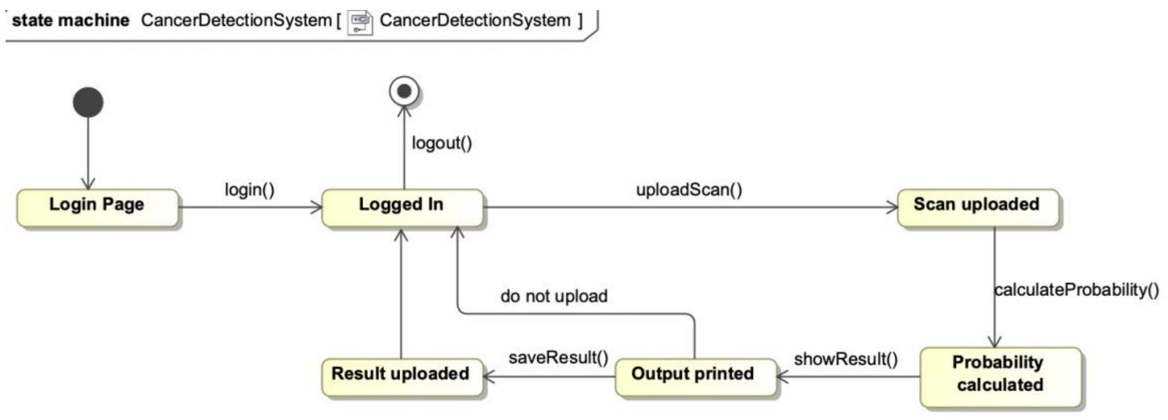
2. Usage scenario

The system will consist of a single actor, a medical practitioner/doctor (further referred to as 'user'). The system will be used in a medical environment in close proximity to X-Ray / MRI / CT scanning areas. The user will have 4 use-cases, he/she must log into the system, he/she can upload a lung scan to run the algorithm that checks for cancer prediction and probability, he/she can save the result in the database and log out of the system.

The system has in total of 6 different states it can be in. Once the application is running, the user will find himself on the "*Login Page*". In this state, the "*login*" trigger, if successful, can change the system's state into "*Logged In*". In this state a user has the possibility to call on another trigger "uploadScan" which takes it into the next state called "Scan uploaded".

Once the user clicks on the run button, triggers the event "calculateProbability" to reach the next state "Probability calculated". After that comes the "Output printed" state, which is reached by the trigger "showResult". With our last trigger "saveResult" we can reach our last state "Result Uploaded" after which the state changes back to "Logged In". The system reaches its end state with the "logout" trigger.


package Model[CancerDetectionSystem]
Cancer Detection System
Log In
Upload lung scan and check for cancer cells
Save result
Log Out
Medical practitioner



3. Requirements

The requirements are written in structured natural language while following the guidelines provided during the course.

Functional requirements

1. Users shall be able to log into the application using their unique medical ID and password.
2. Users shall be able to select a lung scan as *.jpg* or *.png* file from the computer to check whether there are cancer cells being present. *(PNG and JPG files are the most common ones and easiest to work with during the AI model creation.)*
3. The system shall provide a visual representation in form of a probability percentage of the result achieved at R2.
4. The system should recognize the 3 most common types of chest cancer types, *(Adenocarcinoma, Large cell carcinoma, Squamous cell carcinoma)* as well as differentiate them from normal/healthy lungs.
5. Users shall be able to save the scan result in the database with the fields: id, user id, patient id, age, sex, isSmoker, result and date. *(Saving the data in the database could*

prove useful in later stages of a hospital as well as help with archives. These results could also be used to improve the AI model in the future.)

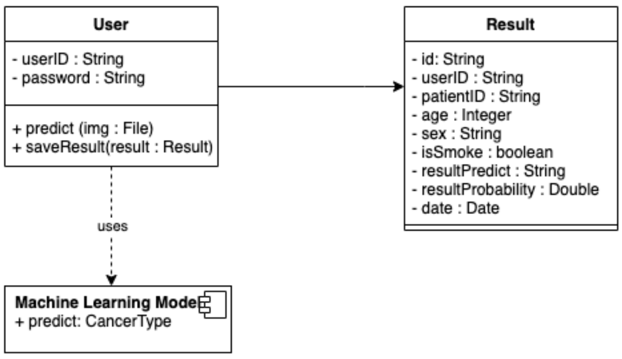
6. Users should be able to log out of the application if they do not intend to use it anymore. *(Prevent unqualified people from using it.)*

Non-functional requirements

1. The system shall recognize the presence of cancer cells with an accuracy of over 90%. *(Anything lower than 90% would render the application useless in this context.)*
2. The system should not take longer than 5 seconds to provide a cancer detection result. *(To improve the usability and user-friendliness of the application.)*
3. Retrieving and saving data (in database) shall not take longer than 1 second. *(To improve the usability and user-friendliness of the application.)*
4. The system shall be implemented with data security in mind. (*Entries in the database should not contain patient names, only their unique ID - Given the nature of the data saved in the database, it is vital to keep the entries secure.)*

4. Initial design

As for the design point of view, the system is pretty simple. In the database, there are going to be saved only the Users and the corresponding unique ID and password. The Users will be able to use the machine learning model and save the results. Another entry would be the Result with fields: id, userID, patientID, age, sex isSmoker, result and date.



Design strategies, alternatives, and tradeoffs
Machine Learning Model

The system relies on a good machine learning model. After thorough research, an unanimous choice has been made to use the python programming language to develop the model. The reasoning behind it is the rich libraries that are designed specifically for these kinds of tasks, the popularity of the programming language, the number of resources, documentation and sheer information being available. The other potential choice that was considered is the R programming language.

Frontend

There are multiple, very good and powerful frontend frameworks and libraries available as web applications keep gaining popularity. The three finalists were Angular, React and Vue. At the end the Angular framework was selected for the project because it is a full-featured framework which is in line with current technologies, built-in support for

HTTP, clearer code and personal experience with the technology. However, going with Angular I had to give up the lightweight nature of React and the fact that its programming language is JavaScript instead of Typescript.

Backend

Anyone who thinks of a traditional web application's backend probably has Java in mind. I did some research and testing; I have tried using an already trained machine learning model (developed in python) within a Java environment but have found out that it isn't as straight forward as I imagined. All the information that I could find online was outdated and most of the solutions didn't work for as multiple methods have gotten deprecated / completely changed since then. Because of my inexperience with these kinds of technologies, I have decided that spending so much time with a relatively easy task is not warranted and that the backend of the project will also be developed in python.

I have way more experience with Java while almost none with python, but this was one of the tradeoffs I had to make.

I did some research on python web frameworks and came up with two finalists: Django and flask. Unfortunately, due to time constraints, I don't have a clear-cut choice yet.

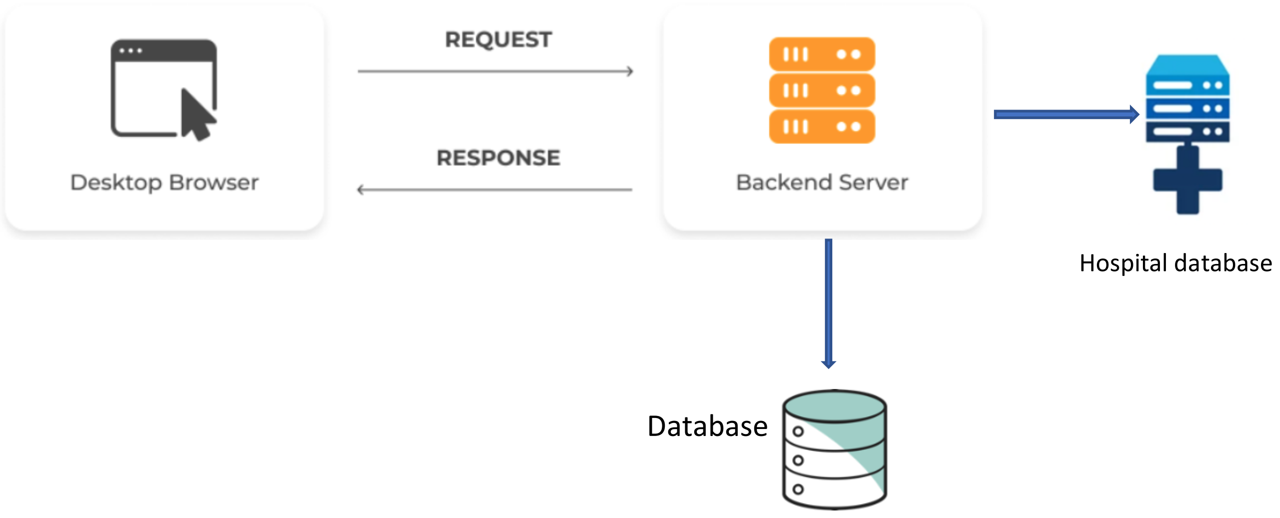
Database

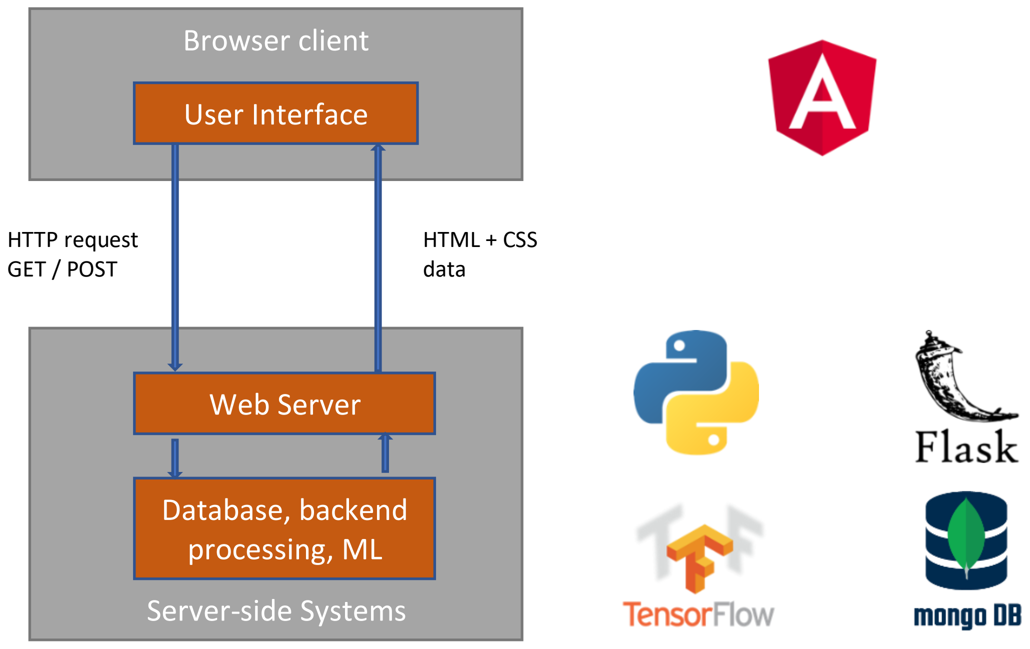
My initial idea was to use a relational database such as MySQL, but given the small size of the project, it might be a better idea to go with a no-sql database, such as MongoDB or CouchDB. Just like with the backend, I do not have a clear choice for now.

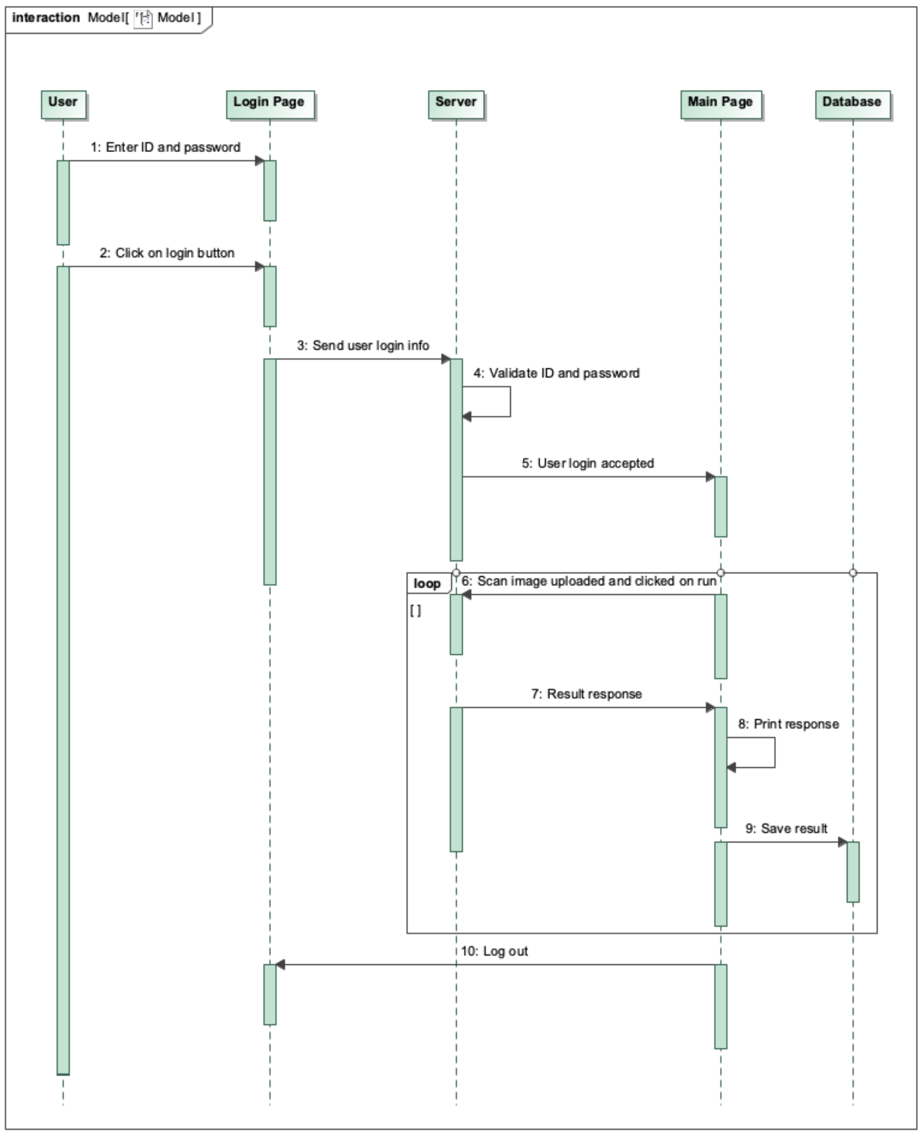
5. Initial architecture

The system should have a Client-Server architecture. Client-server architectures are well known, simple and work good in the web application field. The users will send a request to the server once they will want to login. The server sends a response, and it does or doesn't let them in (depending on if the users gave valid credentials or not).

Once inside, the clients continue to communicate with the server, while creating a request to run the cancer detection algorithm on the file they have just uploaded. The backend server gets this data, runs the algorithm and once it is finished, sends a response to the user with the probability value. A user can also send a request in case they want to save the data in the database, to which the backend responds with a simple message, success, or error.




Browser client
User Interface
HTTP request
GET / POST
HTML + CSS
data
Web Server
Database, backend
processing, ML
Server-side Systems
Flask
TensorFlow
mongo DB


interaction Model[Model]
User
Login Page
Server
Main Page
Database
1: Enter ID and password
2: Click on login button
3: Send user login info
4: Validate ID and password
5: User login accepted
loop
[ ]
6: Scan image uploaded and clicked on run
7: Result response
8: Print response
9: Save result
10: Log out