Assignment-1: Build Something

Template version 1.4 – March 12, 2020

PA2577: Introduction to Cloud Computing and Big Data

November 30, 2023

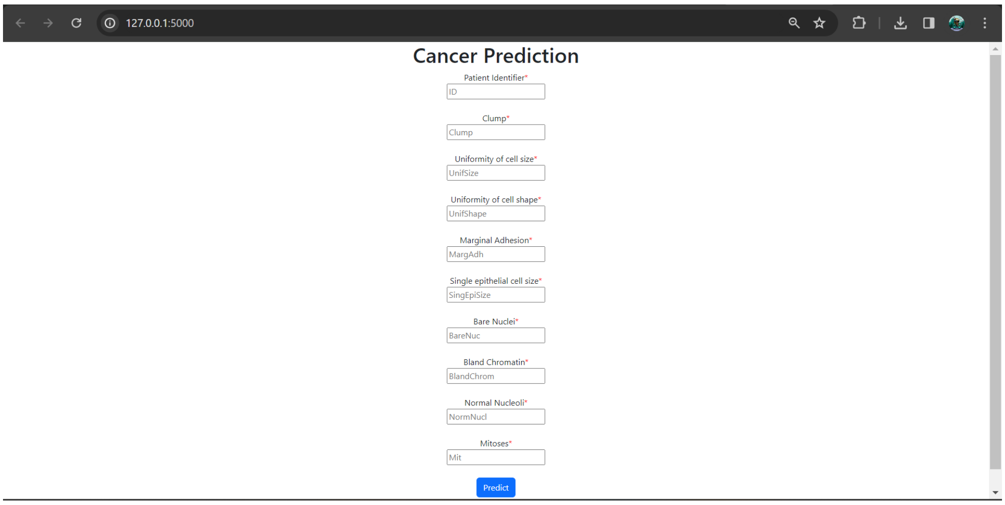
Title	Cancer prediction web application
Supervisor(s)	Mikael Svahnberg

Student 1	Name	Venkata Surya Akash Mernedi
	E-Mail	veme22@student.bth.se
	Program	Masters Qualification Plan in Computer Science

Submission/change history (latest entry on top)	
Date	**Changes made**
2023-12-01	Initial submission.

1 Software Application Idea

I have developed a web application that predicts the likelihood of cancer in patients using various parameters. This tool is designed to aid doctors in better analyzing their patients. It utilizes machine learning algorithms to analyze the input data and provide users with a risk assessment. The backend application is developed using Flask, and MongoDB is used as the database to store and manage data securely.



127.0.0.1:5000

Cancer Prediction

Patient Identifier*
ID

Clump*
Clump

Uniformity of cell size*
UnifSize

Uniformity of cell shape*
UnifShape

Marginal Adhesion*
MargAdh

Single epithelial cell size*
SingEpiSize

Bare Nuclei*
BareNuc

Bland Chromatin*
BlandChrom

Normal Nucleoli*
NormNucl

Mitoses*
Mit

Predict

Figure 1: Cancer prediction web application

2 Architecture design decisions in application

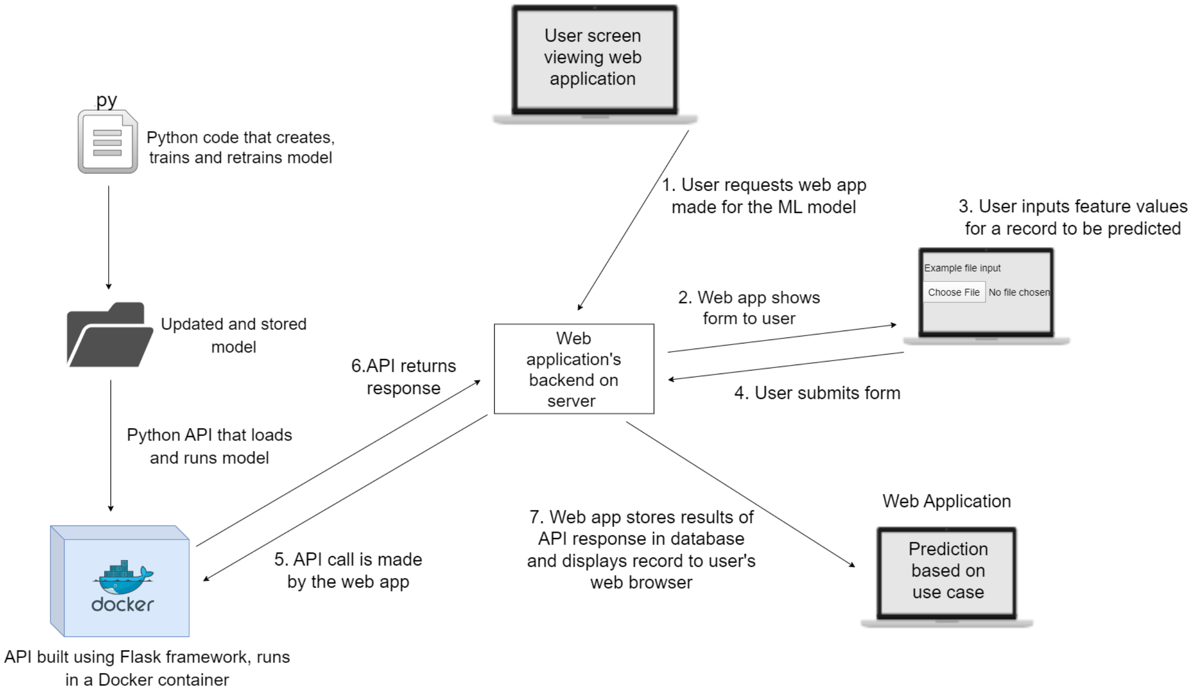

User screen viewing web application
py
Python code that creates, trains and retrains model
1. User requests web app made for the ML model
3. User inputs feature values for a record to be predicted
Example file input
Choose File
No file chosen
Updated and stored model
2. Web app shows form to user
Web application's backend on server
6.API returns response
4. User submits form
Python API that loads and runs model
Web Application
7. Web app stores results of API response in database and displays record to user's web browser
5. API call is made by the web app
Prediction based on use case
docker
API built using Flask framework, runs in a Docker container

Figure 2: Design of web application

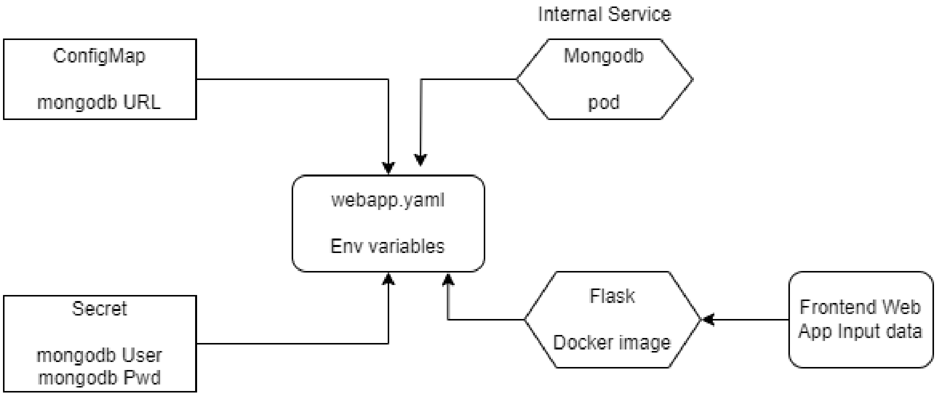


Figure 3: Architecture of Kubernetes Orchestration

3 Business implications of the architecture design

1. **Scalability:** Microservices enable scaling of specific components based on demand, enhancing the application's ability to handle increased user traffic.
2. **Agility:** The modular architecture of the flask allows easy updates and additions to specific components without affecting the entire web application.
3. **Data Flexibility:** MongoDB's schema flexibility accommodates changes in data structures, supporting the incorporation of new medical data types.

4 Interaction between microservices

1. **Data Preprocessing:** Validates and preprocesses incoming data before passing it to the ML prediction service. This is
2. **ML Prediction:** Utilizes the trained model to predict cancer likelihood based on preprocessed data. Used the pickle module in python to perform this operation.
3. **Database Management Microservice:** Manages the storage and retrieval of patient data in MongoDB.

5 Details about deployment

1. **Docker Containers:** Each microservice is containerized using Docker for consistency across development and production environments. The docker image created is pushed into the docker hub and the same is pulled by Kubernetes for orchestration.
2. **Kubernetes Orchestration:** Kubernetes manages container deployment, scaling, and load balancing, ensuring high availability and efficient resource utilization.
3. **Flask:** The Flask application effectively interacts with the database and serves the static web pages. The database is in StatefulSet with 1 replica and the web pages are implemented using the basic HTML scripts.

6 Security issues identified/mitigated

1. **Data Encryption:** Data transmitted between microservices and the database is encrypted to prevent unauthorized access.
2. **Container Security:** Docker containers are scanned to check the vulnerabilities and the package versions are updated accordingly.
3. **Kubernetes secrets:** The confidential information like username and password are encoded with base 64 and followed the referencing method while configuration and performing the kubernetes orchestration.