Brain tumour detection using TensorFlow flask

ABOUT PROJECT

This project is web application that takes input that is MRI image of brain and tells the user if he/she has brain tumour or not. At the core of this project is pretrained machine learning model that is accurate up to 99% and is powered by TensorFlow. We also provide each user unique login sessions and also store their sessions in cookies so that same user does not need to log in again and again.

USER MODULE / CORE INTERFACE

There are 5 overall html pages in this program that provide interapp connectivity

1. Base.html
2. Login.html
3. Signup.html
4. Upload.html
5. Complete.html

As we are using flask, we get a lot bigger advantage while making web pages and hence we have avoided using JavaScript.

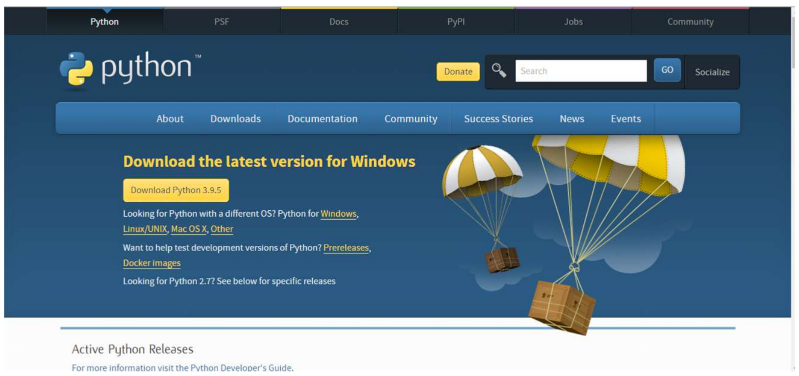
Hardware and software requirements

1. Microprocessor Intel Pentium IV (2.4 GHz.)
2. Primary Storage (RAM) –1GB
3. Secondary Storage (Hard Disk) –4GB

Other basic configuration includes display and other input output devices

For software we be will require to install Python and then anaconda as our virtual environment and then install some libraries

To install python got to - https://www.python.org/downloads/ and download the latest version



Download the latest version for Windows

Looking for Python with a different OS? Python for Windows, Linux/UNIX, Mac OS X, Other

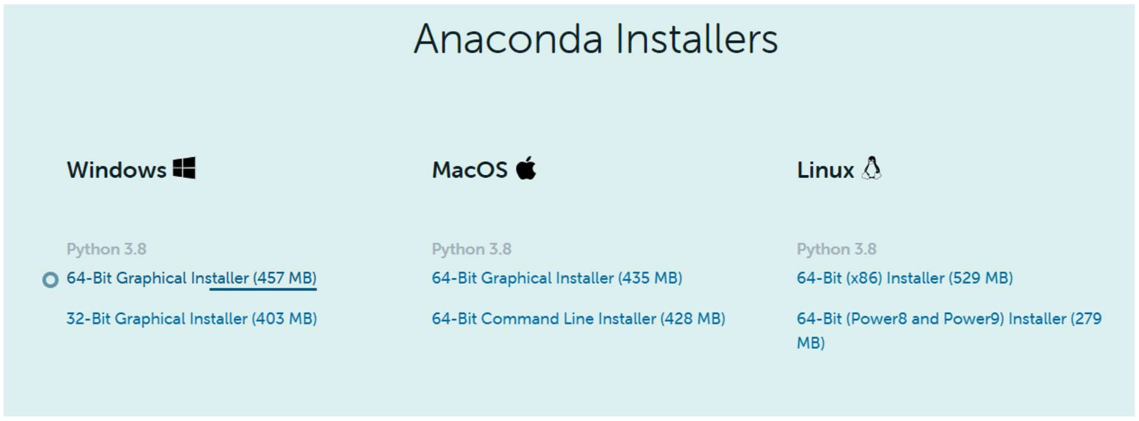
Want to help test development versions of Python? Prereleases, Docker images

Looking for Python 2.7? See below for specific releases

Active Python Releases

For more information visit the Python Developer's Guide.

To install anaconda, go to - https://www.anaconda.com/products/individual

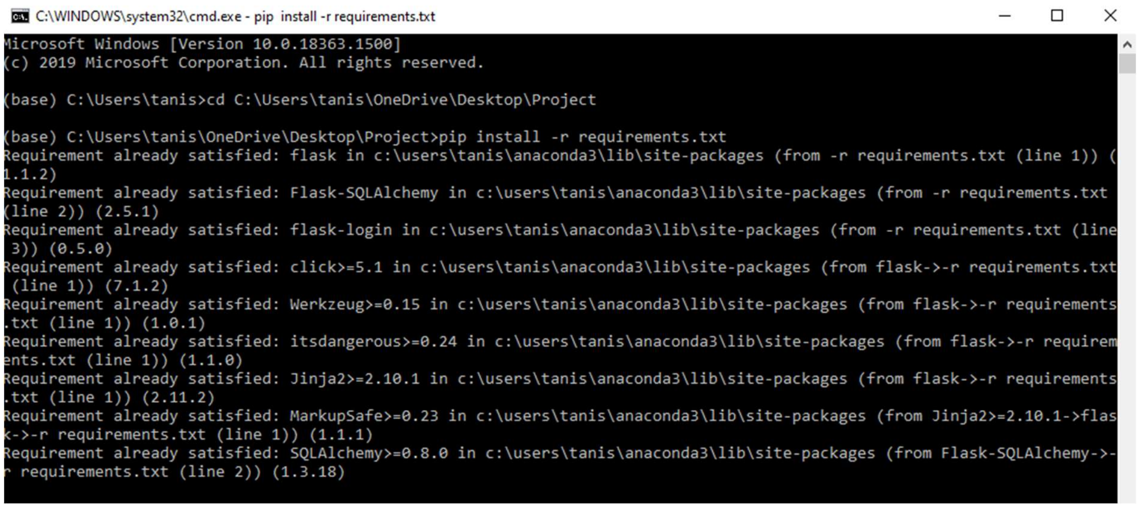

Anaconda Installers
Windows
Python 3.8
64-Bit Graphical Installer (457 MB)
32-Bit Graphical Installer (403 MB)
MacOS
Python 3.8
64-Bit Graphical Installer (435 MB)
64-Bit Command Line Installer (428 MB)
Linux
Python 3.8
64-Bit (x86) Installer (529 MB)
64-Bit (Power8 and Power9) Installer (279 MB)

After installation, I recommend using an IDE like VSCode to make sure the project work correctly, but this part is optional:

Installing important libraries used in our project –

Open anaconda navigator and the open cmd prompt the change location to your project and write the following cmd:

pip install -r requirements.txt


C:\WINDOWS\system32\cmd.exe - pip install -r requirements.txt

```
Microsoft Windows [Version 10.0.18363.1500]
(c) 2019 Microsoft Corporation. All rights reserved.

(base) C:\Users\tanis>cd C:\Users\tanis\OneDrive\Desktop\Project

(base) C:\Users\tanis\OneDrive\Desktop\Project>pip install -r requirements.txt
Requirement already satisfied: flask in c:\users\tanis\anaconda3\lib\site-packages (from -r requirements.txt (line 1)) (
1.1.2)
Requirement already satisfied: Flask-SQLAlchemy in c:\users\tanis\anaconda3\lib\site-packages (from -r requirements.txt
(line 2)) (2.5.1)
Requirement already satisfied: flask-login in c:\users\tanis\anaconda3\lib\site-packages (from -r requirements.txt (line
 3)) (0.5.0)
Requirement already satisfied: click>=5.1 in c:\users\tanis\anaconda3\lib\site-packages (from flask->-r requirements.txt
 (line 1)) (7.1.2)
Requirement already satisfied: Werkzeug>=0.15 in c:\users\tanis\anaconda3\lib\site-packages (from flask->-r requirements
.txt (line 1)) (1.0.1)
Requirement already satisfied: itsdangerous>=0.24 in c:\users\tanis\anaconda3\lib\site-packages (from flask->-r requirem
ents.txt (line 1)) (1.1.0)
Requirement already satisfied: Jinja2>=2.10.1 in c:\users\tanis\anaconda3\lib\site-packages (from flask->-r requirements
.txt (line 1)) (2.11.2)
Requirement already satisfied: MarkupSafe>=0.23 in c:\users\tanis\anaconda3\lib\site-packages (from Jinja2>=2.10.1->flas
k->-r requirements.txt (line 1)) (1.1.1)
Requirement already satisfied: SQLAlchemy>=0.8.0 in c:\users\tanis\anaconda3\lib\site-packages (from Flask-SQLAlchemy->-
r requirements.txt (line 2)) (1.3.18)
```

Later we will see all important libraries in much more detail.

Python

Python is one of the languages that is witnessing incredible growth and popularity year by year. In 2017, StackOverflow calculated that python would beat all other programming languages by 2020 as it has become the fastest-growing programming language in the world.

It is also considered one of the best programming languages for machine learning.

Python language is incredibly easy to use and learn for new beginners and newcomers. The python language is one of the most accessible programming languages available because it has simplified syntax and not complicated, which gives more emphasis on natural language. Due to its ease of learning and usage, python codes can be easily written and executed much faster than other programming languages.

When **Guido van Rossum** was creating python in the 1980s, he made sure to design it to be a general-purpose language. One of the main reasons for the popularity of python would be its simplicity in syntax so that it could be easily read and understood even by amateur developers also.

One can also quickly experiment by changing the code base of python because it is an interpreted language which makes it even more popular among all kinds of developers.

Due to its corporate sponsorship and big supportive community of python, python has excellent libraries that you can use to select and save your time and effort on the initial cycle of development. There are also lots of cloud media services that offer cross-platform support through library-like tools, which can be extremely beneficial.

Libraries with specific focus are also available like nltk for natural language processing or scikit-learn for machine learning applications.
With its great potential python is being used in each and every field.

TensorFlow

Created by the Google Brain team, TensorFlow is an open-source library for numerical computation and large-scale machine learning. TensorFlow bundles together a slew of machine learning and deep learning (aka neural networking) models and algorithms and makes them useful by way of a common metaphor. It uses Python to provide a convenient front-end API for building applications with the framework, while executing those applications in high-performance C++.

TensorFlow can train and run deep neural networks for handwritten digit classification, image recognition, word embeddings, recurrent neural networks, sequence-to-sequence models for machine translation, natural language processing, and PDE (partial differential equation) based simulations. Best of all, TensorFlow supports production prediction at scale, with the same models used for training. TensorFlow applications can be run on most any target that's convenient: a local machine, a cluster in the cloud, iOS and Android devices, CPUs or GPUs. If you use Google's own cloud, you can run TensorFlow on Google's custom TensorFlow Processing Unit (TPU) silicon for further acceleration. The resulting models created by TensorFlow, though, can be deployed on most any device where they will be used to serve predictions.

For this project we will a take a pretrained model and the run it on our local machine which will check that brain tumour is present or not.

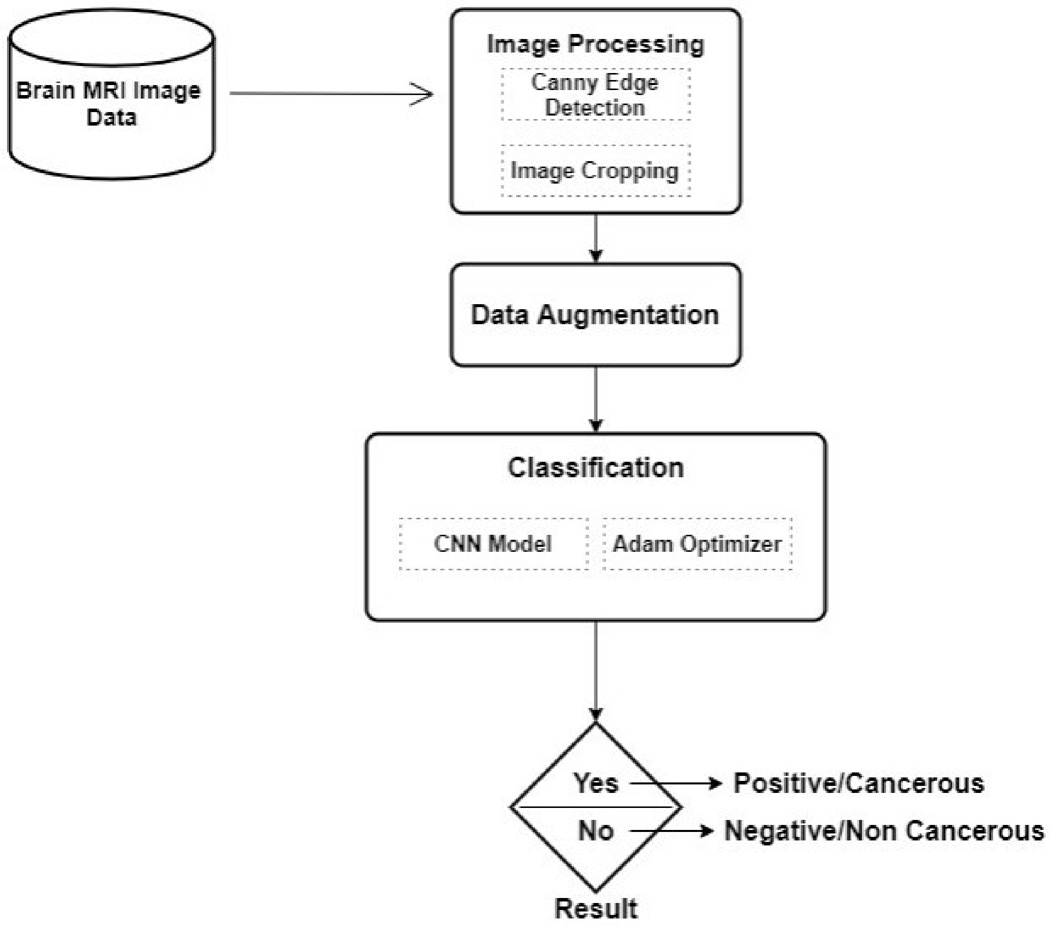
About the model

We are using *Brain tumour classification in MRI image using convolutional neural network full reference available at* https://www.aimspress.com/article/doi/10.3934/mbe.2020328?viewType=HTML

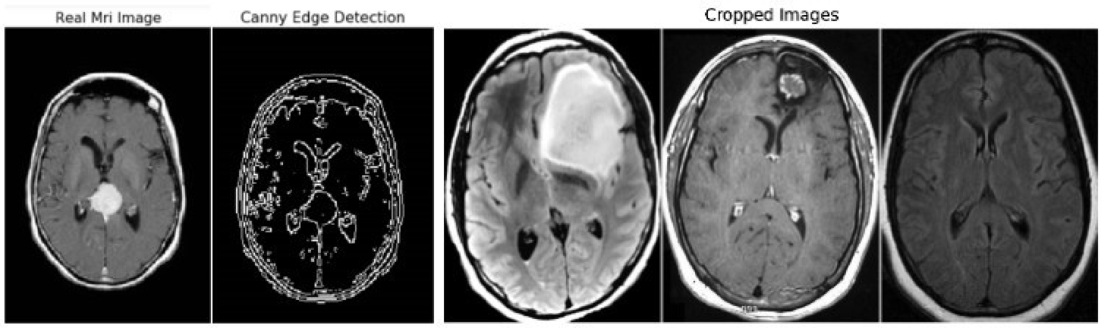
Brain tumor is a severe cancer disease caused by uncontrollable and abnormal partitioning of cells. Recent progress in the field of deep learning has helped the health industry in Medical Imaging for Medical Diagnostic of many diseases. For Visual learning and Image Recognition, task CNN is the most prevalent and commonly used machine learning algorithm. Similarly, in our paper, we introduce the convolutional neural network (CNN) approach along with Data Augmentation and Image Processing to categorize brain MRI scan images into cancerous and non-cancerous. Using the transfer learning approach we compared the performance of our scratched CNN model with pre-trained VGG-16, ResNet-50, and Inception-v3 models. As the experiment is tested on a very small dataset but the experimental result shows that our model accuracy result is very effective and have very low complexity rate by achieving 100% accuracy, while VGG-16 achieved 96%, ResNet-50 achieved 89% and Inception-V3 achieved 75% accuracy. Our model requires very less computational power and has much better accuracy results as compared to other pre-trained models.

In the past few years because of AI and Deep learning, significant advancement has been made in the medical science like Medical Image processing technique which helps doctors to the diagnose disease early and easily, before that, it was tedious and time-consuming. So to resolve such kind of limitations computer-aided technology is much needed because Medical Field needs efficient and reliable techniques to diagnose life-threatening diseases like cancer, which is the leading cause of mortality globally for patients. So in our study with the help of Brain MRI Images, we provide a method for classification of brain tumors into cancerous and non-cancerous using data augmentation technique and convolutional neural network model.

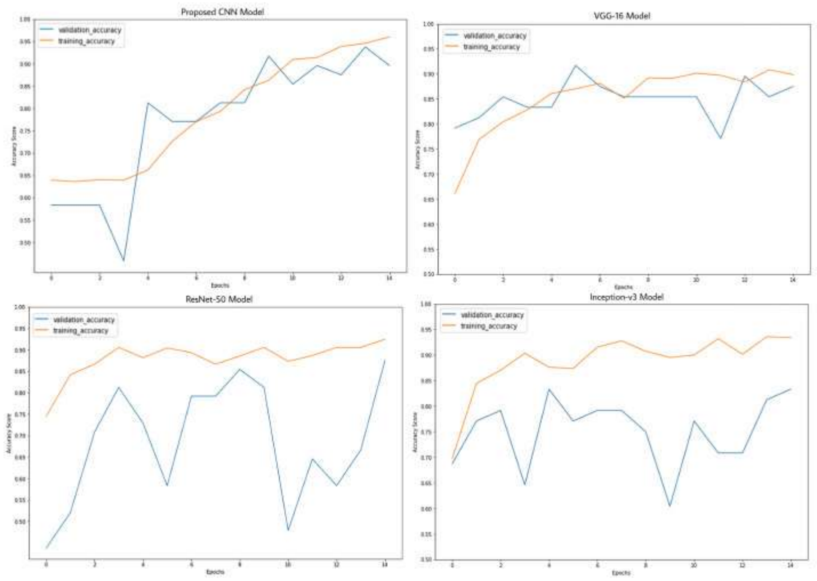
Proposed methods



3.1. Image processing



Multiple pretrained model with their accuracy are



Here CNN (Convolution neural network) is showing the best results on untested data

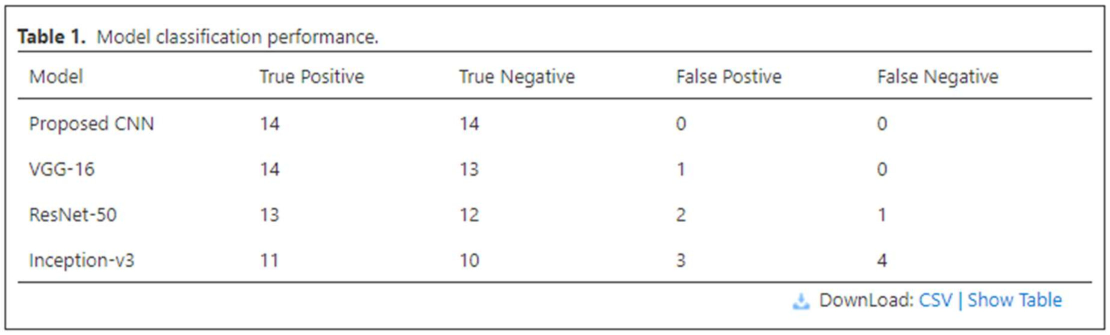


Table 1. Model classification performance.

Model	True Positive	True Negative	False Postive	False Negative
Proposed CNN	14	14	0	0
VGG-16	14	13	1	0
ResNet-50	13	12	2	1
Inception-v3	11	10	3	4

DownLoad: CSV | Show Table

Ant there we are using this pertained model which gives us the most accurate results.

For this project we eliminated the machine learning aspect by using a pretrained model. Now we will discuss the architecture of our project and its work flow then we will see about backend and our project in action.

Flask

Flask is a lightweight Web Server Gateway Interface (WSGI) web application framework.

A micro web framework that has minimal dependencies on external libraries, written in Python, which was formed for a faster and easier use, and also has the ability to scale up to complex applications.

A developer is provided the required code by Flask when building a web application. It lets you start up a server, handle requests, templates and much more.

As a beginner in web development, you want to know more about Flask. Flask is considered the best framework for beginners due to its flexibility and gives you the opportunity to learn. Here are the advantages and disadvantages of the Flask web framework.

While Flask is simple enough to get started with, you should know the Python language before starting.

Advantages of the Flask framework

Flask is considered the best framework for light web application serving, it is a lightweight framework and can also be useful to the developer if he or she chooses a web interface to the default system based UI.

Easy to understand development
The Flask framework is easy to understand, that is why it is best for beginners. Its simplicity gives you the opportunity to understand it better and learn from it. There are interesting features to use in the framework. The simplicity in the flask framework enables the developer to navigate around and create the application easily.

Unlike other web application frameworks, flask let you be in total control in web development taking full creative control of the application and web development. The developer has the chance of being "in the drivers seat", taking charge of what you want to do like adding external features.

It is very flexible and easy.
There are only a handful of parts of flask that cannot be changed or altered because of its simplicity and minimalism. This means that almost all the parts of flask are open to change, unlike some other web frameworks.

Flask comes with a template engine that lets you use the same user interface for multiple pages. Python can insert variables into the templates.

Testing
Using Flask for web development allows for unit testing through its integrated support, built-in development server, fast debugger, and restful request dispatching. It is lightweight to enable you to transit into a web framework easily with some extension.

SDLC or Software Development Life Cycle

The Systems Development Life Cycle (SDLC), or Software Development Life Cycle in systems engineering and software engineering, is the process of creating or altering systems, and the models and methodologies that people use to develop these systems. The concept generally refers to computer or information systems. In software engineering the SDLC concept underpins many kinds of software development methodologies. These methodologies form the framework for planning and controlling the creation of an information system the software development process. Systems Development Life Cycle (SDLC) is a logical process used by a systems analyst to develop an information system, including requirements, validation, training, and user (stakeholder) ownership. Any SDLC should result in a high-quality system that meets or exceeds customer expectations, reaches completion within time and cost estimates, works effectively and efficiently in the current and planned Information Technology infrastructure, and is inexpensive to maintain and cost-effective to enhance.

Computer systems are complex and often (especially with the recent rise of Service Oriented Architecture) link multiple traditional systems potentially supplied by different software vendors. To manage this level of complexity, a number of SDLC models have been created: "waterfall"; "fountain"; "spiral"; "build and fix"; "rapid prototyping"; "incremental"; and "synchronize and stabilize". SDLC models can be described along a spectrum of agile to iterative to sequential. Agile methodologies, such as XP and Scrum, focus on light-weight processes which allow for rapid changes along the development cycle. Iterative methodologies, such as Rational Unified Process and Dynamic Systems Development Method, focus on limited project scopes and expanding or improving products by multiple iterations. Sequential or big-design-upfront (BDUF) models, such as Waterfall, focus on complete and correct planning to guide large projects and risks to successful and predictable results. Some agile and iterative proponents confuse the term SDLC with sequential or "more traditional" processes; however, SDLC is an umbrella term for all methodologies for the design, implementation, and release of software. 1. Initiation/planning

2. Requirements gathering and analysis

3. Design

4. Build or coding

5. Testing

Testing

• **Unit testing**: unit testing is a software verification and validation method in which a programmer tests if individual units of source code are fit for use. A unit is the smallest testable part of an application

• **Integration testing**: The purpose of integration testing is to detect any inconsistencies between the software units that are integrated together (called assemblages) or between any of the assemblages and the hardware.

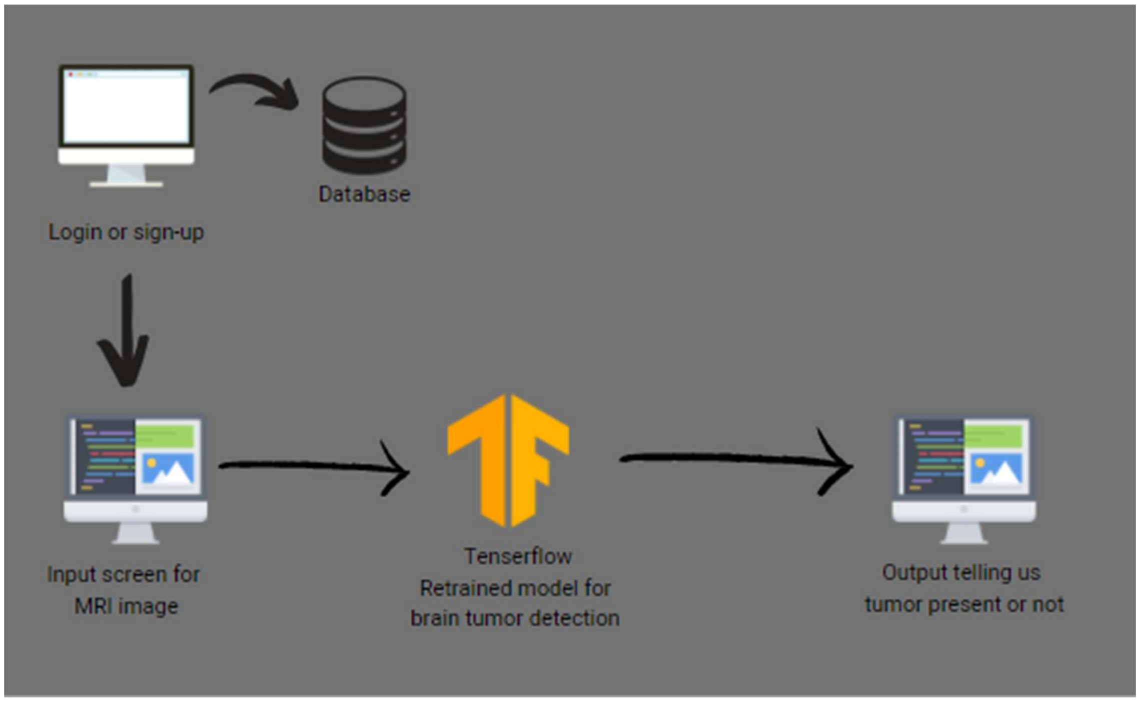
• **Black box testing**: Black-box testing uses external descriptions of the software, including specifications, requirements, and designs to derive test cases.

• **White box testing**: White box testing (a.k.a. clear box testing, glass box testing, and transparent box testing or structural testing) uses an internal perspective of the system to design test cases based on internal structure.

• **Regression testing**: It is any type of software testing that seeks to uncover software errors by partially retesting a modified program.

• **System testing**: System testing of software or hardware is testing conducted on a complete, integrated system to evaluate the system's compliance with its specified requirements.

Working of our web application


Login or sign-up
Database
Input screen for MRI image
Tenserflow Retrained model for brain tumor detection
Output telling us tumor present or not

The working of our application is simple

First, we have a database for login and signup purpose which are linked to login and signup page respectively.

Then we have a page where user can upload MRI images of that image is then passed to our tensorflow model which then generates output as True or False which is then passed to our final output screen and changes of web pages are also interlinked.

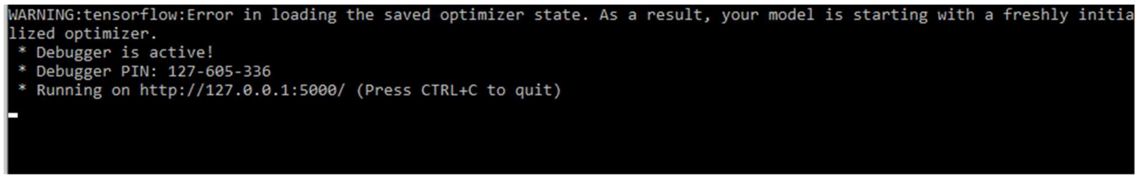
The data pipeline is also similar to above working.

Outputs and work flow

To run the program, go to open anaconda, then open command prompt.exe

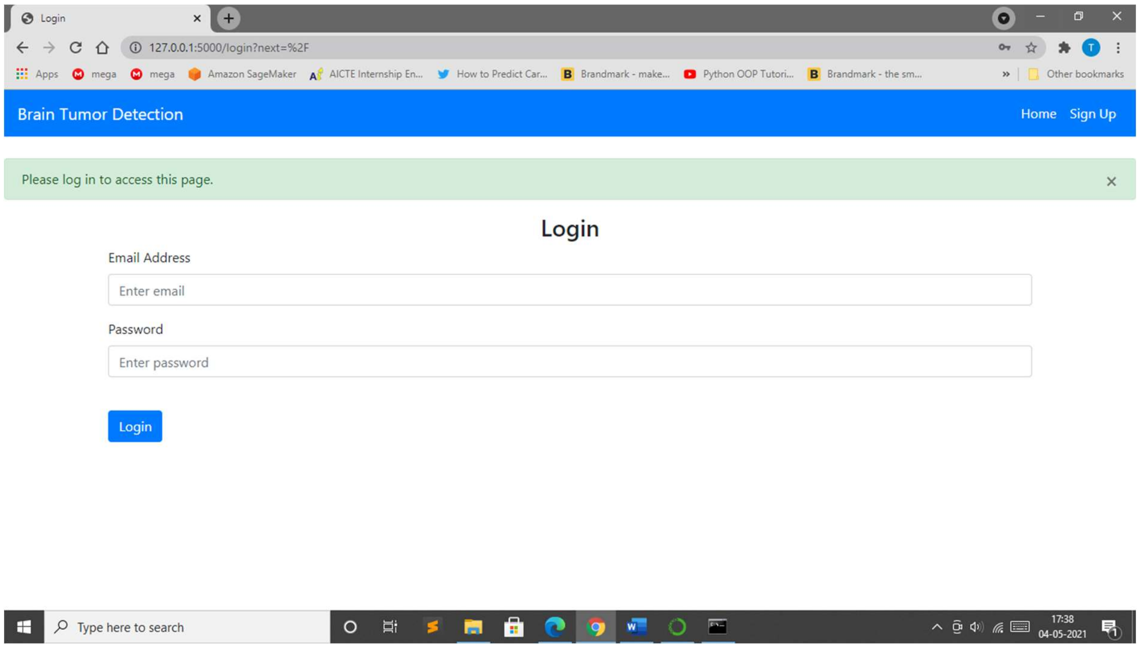
Then open the folder using cd command

To run the program, use command "python main.py"



```
WARNING:tensorflow:Error in loading the saved optimizer state. As a result, your model is starting with a freshly initia
lized optimizer.
 * Debugger is active!
 * Debugger PIN: 127-605-336
 * Running on http://127.0.0.1:5000/ (Press CTRL+C to quit)
```

Now go to http://127.0.0.1:5000/ on your browser



Please log in to access this page.

Login

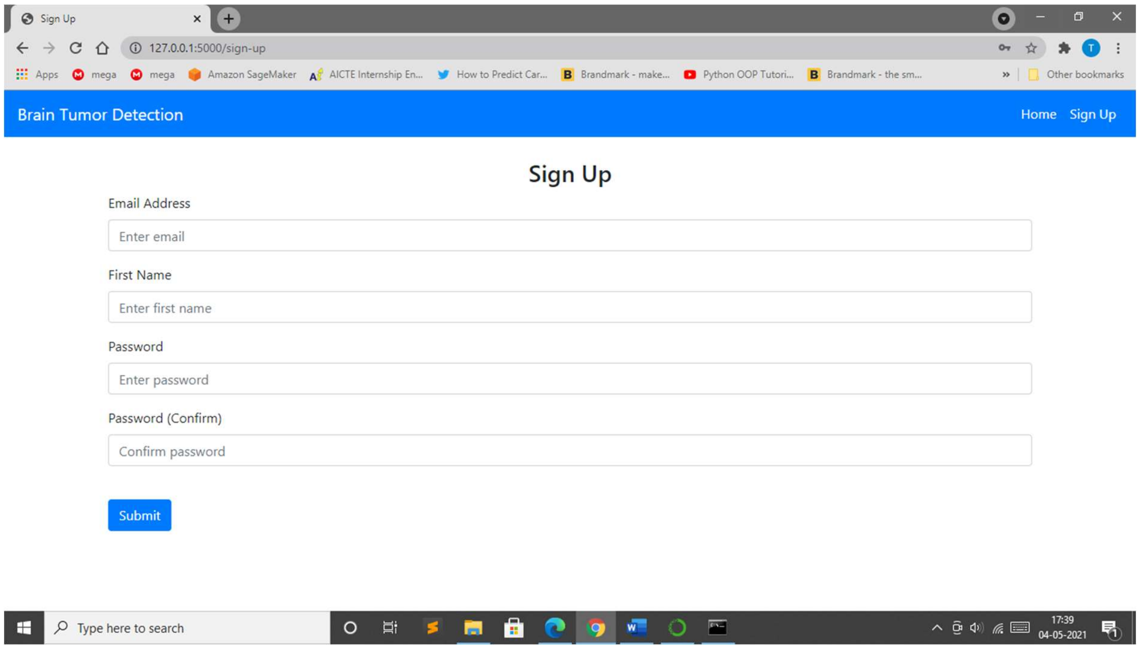
Email Address

Enter email

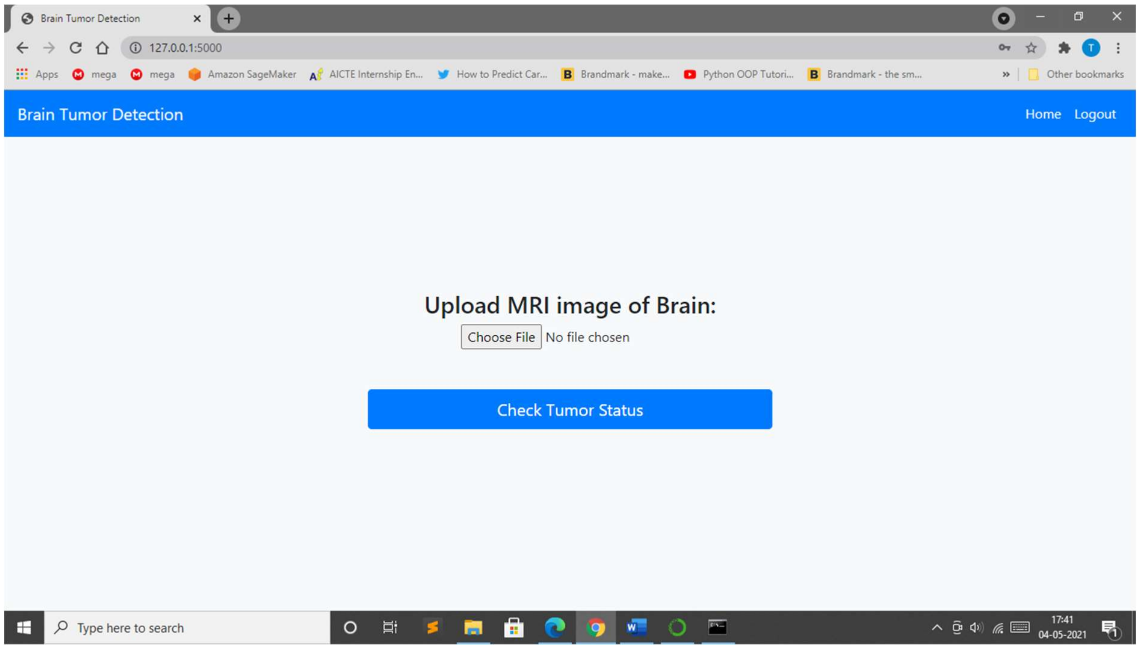
Password

Enter password

Now go to top right corner and sign up

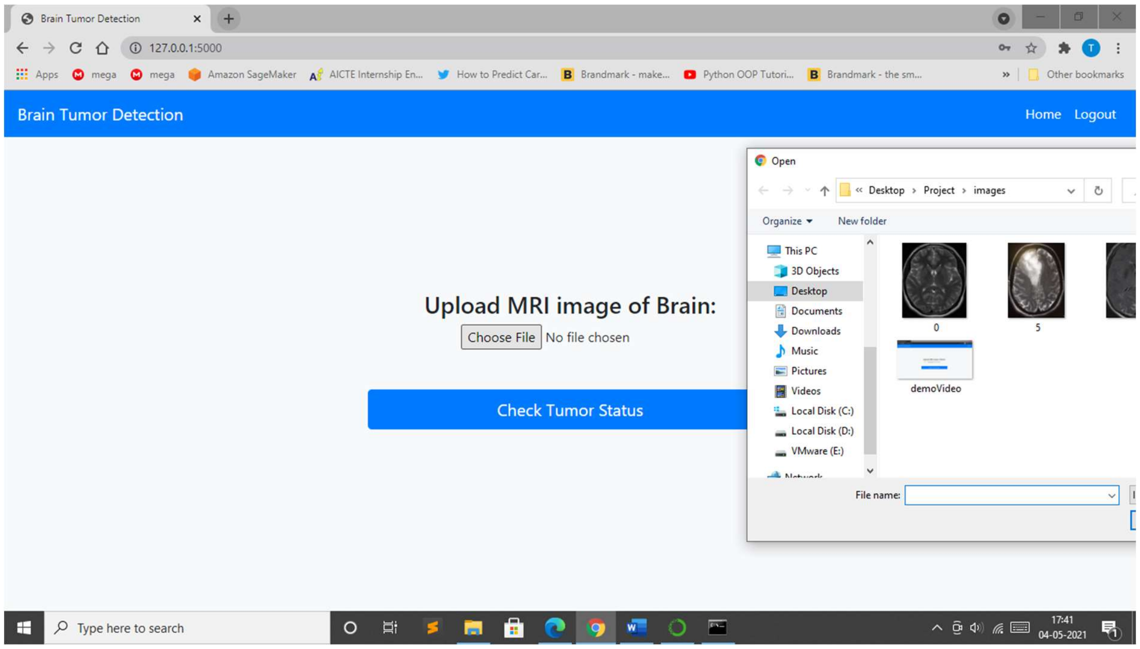


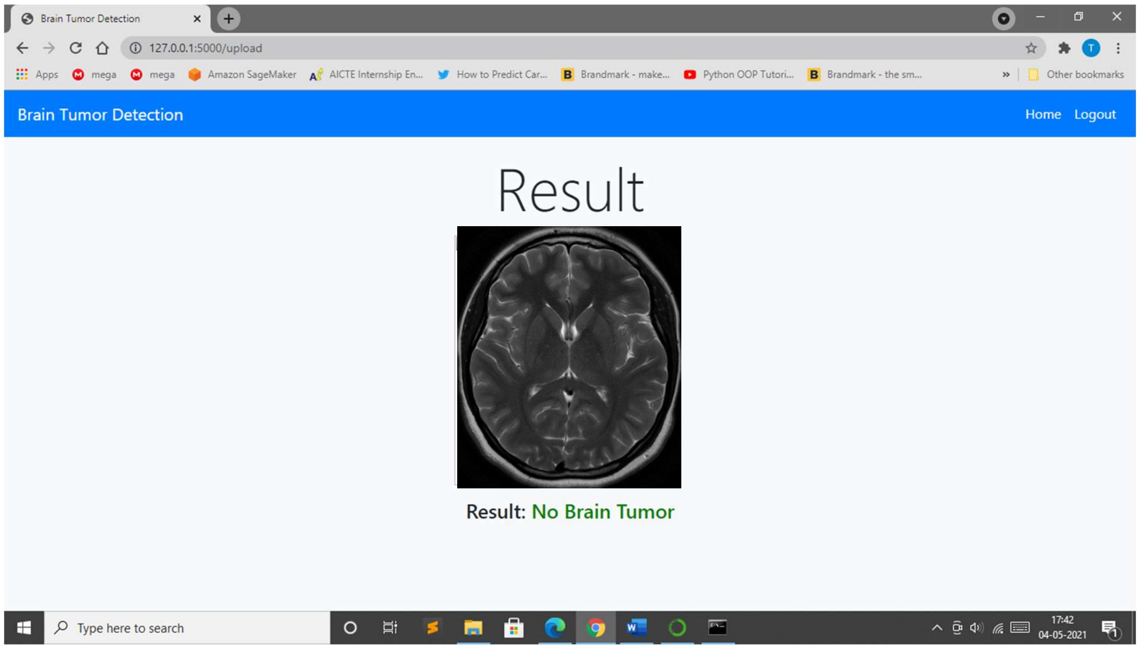
And the fill the details



Click on choose image

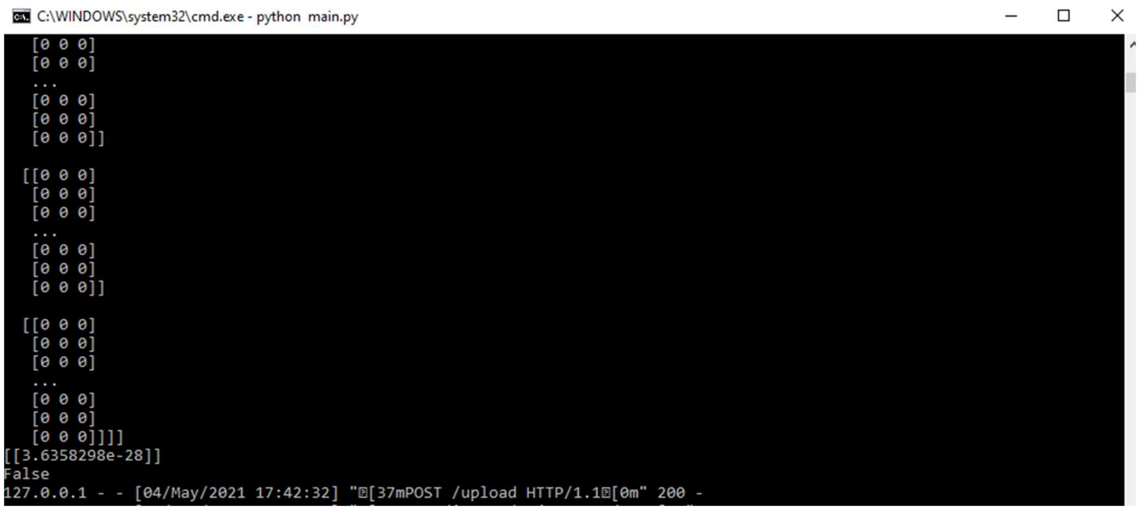
And then select an image of MRI





We can go to cmd.exe and see how this project is running

Here we have output as False as there is no Brain tumour present

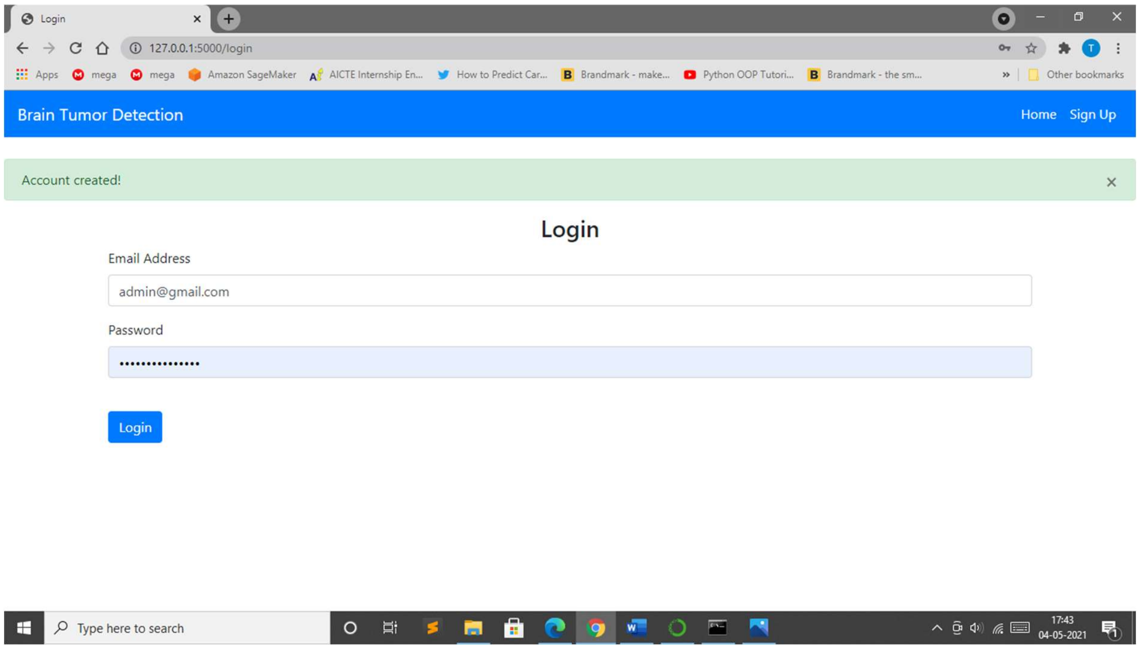


```
  [0 0 0]
  [0 0 0]
  ...
  [0 0 0]
  [0 0 0]
  [0 0 0]]

 [[0 0 0]
  [0 0 0]
  [0 0 0]
  ...
  [0 0 0]
  [0 0 0]
  [0 0 0]]

 [[0 0 0]
  [0 0 0]
  [0 0 0]
  ...
  [0 0 0]
  [0 0 0]
  [0 0 0]]]]
[[3.6358298e-28]]
False
127.0.0.1 - - [04/May/2021 17:42:32] "[37mPOST /upload HTTP/1.1[0m" 200 -
```

After checking the result you can logout



Account created!

Login

Email Address

admin@gmail.com

Password

•••••••••••••••

Coding

In our program main.py is just used to call all functions/programs so that flow is maintained properly

Main.py

```
from website import create_app

app = create_app()

if __name__ == '__main__':
    app.run(debug=True)
```

now we will define __init__.py which will be used to verify login credentials form user input

__init__.py

```
from flask import Flask
from flask_sqlalchemy import SQLAlchemy
from os import path
from flask_login import LoginManager

db = SQLAlchemy()
DB_NAME = "database.db"


def create_app():
    app = Flask(__name__)
    app.config['SECRET_KEY'] = 'hjshjhdjah kjshkjdhjs'
    app.config['SQLALCHEMY_DATABASE_URI'] = f'sqlite:///{DB_NAME}'
    db.init_app(app)

    from .views import views
    from .auth import auth

    app.register_blueprint(views, url_prefix='/')
    app.register_blueprint(auth, url_prefix='/')

    from .models import User, Note

    create_database(app)

    login_manager = LoginManager()
    login_manager.login_view = 'auth.login'
    login_manager.init_app(app)

    @login_manager.user_loader
    def load_user(id):
        return User.query.get(int(id))

    return app


def create_database(app):
    if not path.exists('website/' + DB_NAME):
        db.create_all(app=app)
        print('Created Database!')
```

To authenticate the login details from __init__.py page we will use the following code

auth.py

```
from flask import Blueprint, render_template, request, flash, redirect, url_fo
r
from .models import User
from werkzeug.security import generate_password_hash, check_password_hash
from . import db
from flask_login import login_user, login_required, logout_user, current_user


auth = Blueprint('auth', __name__)


@auth.route('/login', methods=['GET', 'POST'])
def login():
    if request.method == 'POST':
        email = request.form.get('email')
        password = request.form.get('password')

        user = User.query.filter_by(email=email).first()
        if user:
            if check_password_hash(user.password, password):
                flash('Logged in successfully!', category='success')
                login_user(user, remember=True)
                return redirect(url_for('views.home'))
            else:
                flash('Incorrect password, try again.', category='error')
        else:
            flash('Email does not exist.', category='error')

    return render_template("login.html", user=current_user)


@auth.route('/logout')
@login_required
```

```
def logout():
    logout_user()
    return redirect(url_for('auth.login'))


@auth.route('/sign-up', methods=['GET', 'POST'])
def sign_up():
    if request.method == 'POST':
        email = request.form.get('email')
        first_name = request.form.get('firstName')
        password1 = request.form.get('password1')
        password2 = request.form.get('password2')

        user = User.query.filter_by(email=email).first()
        if user:
            flash('Email already exists.', category='error')
        elif len(email) < 4:
            flash('Email must be greater than 3 characters.', category='error'
)
        elif len(first_name) < 2:
            flash('First name must be greater than 1 character.', category='er
ror')
        elif password1 != password2:
            flash('Passwords don\'t match.', category='error')
        elif len(password1) < 7:
            flash('Password must be at least 7 characters.', category='error')
        else:
            new_user = User(email=email, first_name=first_name, password=gener
ate_password_hash(
                password1, method='sha256'))
            db.session.add(new_user)
            db.session.commit()
            login_user(new_user, remember=True)
            flash('Account created!', category='success')
            return redirect(url_for('views.home'))

    return render_template("sign_up.html", user=current_user)
```

to store data in database that we just created let us use models.py

```
from . import db
from flask_login import UserMixin
from sqlalchemy.sql import func


class Note(db.Model):
    id = db.Column(db.Integer, primary_key=True)
    data = db.Column(db.String(10000))
    date = db.Column(db.DateTime(timezone=True), default=func.now())
    user_id = db.Column(db.Integer, db.ForeignKey('user.id'))


class User(db.Model, UserMixin):
    id = db.Column(db.Integer, primary_key=True)
    email = db.Column(db.String(150), unique=True)
    password = db.Column(db.String(150))
    first_name = db.Column(db.String(150))
    notes = db.relationship('Note')
```

then we will use our tensorflow pretrained model to find out that given image has brain tumour or not

We define predictor.py

```
import numpy as np
from keras.preprocessing import image
from tensorflow.keras.models import load_model
saved_model = load_model("model/VGG_model.h5")
status = True


def check(input_img):
    print(" your image is : " + input_img)
    print(input_img)

    img = image.load_img("images/" + input_img, target_size=(224, 224))
    img = np.asarray(img)
    print(img)

    img = np.expand_dims(img, axis=0)

    print(img)
    output = saved_model.predict(img)

    print(output)
    if output[0][0] == 1:
        status = True
    else:
        status = False

    print(status)
    return status
```

And now lastly, we will take user input and then pass to all the above modules that we just created

This is gonna be backbone of our program

```
from flask import Blueprint, render_template, request, flash, jsonify
from flask_login import login_required, current_user
from .models import Note
from . import db
import json
import os
from flask import Flask, render_template, request
from predictor import check

APP_ROOT = os.path.dirname(os.path.abspath(__file__))
views = Blueprint('views', __name__)


@views.route('/', methods=['GET', 'POST'])
@login_required
def home():
    return render_template('upload.html')


@views.route('/upload', methods=['GET', 'POST'])
def upload():
    target = os.path.join(APP_ROOT, 'images/')
    print(target)

    if not os.path.isdir(target):
        os.mkdir(target)

    for file in request.files.getlist('file'):
        print(file)
        filename = file.filename
        print(filename)
        dest = '/'.join([target, filename])
        print(dest)
        file.save(dest)

    status = check(filename)

    return render_template('complete.html', image_name=filename, predvalue=sta
tus)

if __name__ == "main":
    app.run(port=4555, debug=True)
```

Code for webpages HTML and CSS

As we are using flask and hence, we can use base html module and inherit it into other flask modules so

base.html

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1" />
    <link
      rel="stylesheet"
      href="https://stackpath.bootstrapcdn.com/bootstrap/4.4.1/css/bootstrap.m
in.css"
      integrity="sha384-
Vkoo8x4CGsO3+Hhxv8T/Q5PaXtkKtu6ug5TOeNV6gBiFeWPGFN9MuhOf23Q9Ifjh"
      crossorigin="anonymous"
    />
    <link
      rel="stylesheet"
      href="https://stackpath.bootstrapcdn.com/font-awesome/4.7.0/css/font-
awesome.min.css"
      crossorigin="anonymous"
    />

    <title>{% block title %}Home{% endblock %}</title>
  </head>
  <body>
    <nav class="navbar navbar-expand-lg navbar-dark bg-primary">
      <!-- <a class="navbar-
brand" href="#">Cancer Predictor and Detector</a> -->
      <a class="navbar-brand" href="#">Brain Tumor Detection</a>
      <button class="navbar-toggler" type="button" data-
toggle="collapse" data-target="#navbarSupportedContent" aria-
controls="navbarSupportedContent" aria-expanded="false" aria-
label="Toggle navigation">
          <span class="navbar-toggler-icon"></span>
      </button>

      <div class="collapse navbar-collapse" id="navbarSupportedContent">
          <ul class="navbar-nav ml-auto">
              <li class="nav-item active">
                  <a class="nav-link" href="./">Home <span class="sr-
only">(current)</span></a>
              </li>
              <li class="nav-item active">
                <a class="nav-item nav-link" id="signUp" href="/sign-
up">Sign Up</a>
```

```
              </li>
          </ul>
      </div>
  </nav>
        <span class="navbar-toggler-icon"></span>
      </button>
      <div class="collapse navbar-collapse" id="navbar">
        <div class="navbar-nav">
          {% if user.is_authenticated %}
          <a class="nav-item nav-link" id="home" href="/">Home</a>
          <a class="nav-item nav-
link" id="logout" href="/logout">Logout</a></div>
          {% else %}
          <nav class="navbar navbar-expand-lg navbar-dark bg-primary">
            <!-- <a class="navbar-
brand" href="#">Cancer Predictor and Detector</a> -->
            <a class="navbar-brand" href="#">Brain Tumor Detection</a>
            <button class="navbar-toggler" type="button" data-
toggle="collapse" data-target="#navbarSupportedContent" aria-
controls="navbarSupportedContent" aria-expanded="false" aria-
label="Toggle navigation">
                <span class="navbar-toggler-icon"></span>
            </button>

          <a class="nav-item nav-link" id="login" href="/login">Login</a>
          <a class="nav-item nav-link" id="signUp" href="/sign-up">Sign Up</a>
          {% endif %}
        </div>
      </div>
    </nav>

    {% with messages = get_flashed_messages(with_categories=true) %} {% if
    messages %} {% for category, message in messages %} {% if category ==
    'error' %}
    <div class="alert alert-danger alter-dismissable fade show" role="alert">
      {{ message }}
      <button type="button" class="close" data-dismiss="alert">
        <span aria-hidden="true">&times;</span>
      </button>
    </div>
    {% else %}
    <div class="alert alert-success alter-dismissable fade show" role="alert">
      {{ message }}
      <button type="button" class="close" data-dismiss="alert">
        <span aria-hidden="true">&times;</span>
      </button>
    </div>
    {% endif %} {% endfor %} {% endif %} {% endwith %}
    <div class="container">{% block content %} {% endblock %}</div>
    <script
      src="https://code.jquery.com/jquery-3.2.1.slim.min.js"
      integrity="sha384-
KJ3o2DKtIkvYIK3UENzmM7KCkRr/rE9/Qpg6aAZGJwFDMVNA/GpGFF93hXpG5KkN"
      crossorigin="anonymous"
    ></script>
    <script
      src="https://cdnjs.cloudflare.com/ajax/libs/popper.js/1.12.9/umd/popper.
min.js"
      integrity="sha384-
ApNbgh9B+Y1QKtv3Rn7W3mgPxhU9K/ScQsAP7hUibX39j7fakFPskvXusvfa0b4Q"
      crossorigin="anonymous"
    ></script>
    <script
      src="https://maxcdn.bootstrapcdn.com/bootstrap/4.0.0/js/bootstrap.min.js
"
      integrity="sha384-
JZR6Spejh4U02d8jOt6vLEHfe/JQGiRRSQQxSfFWpi1MquVdAyjUar5+76PVCmYl"
      crossorigin="anonymous"
    ></script>

    <script
      type="text/javascript"
      src="{{ url_for('static', filename='index.js') }}"
    ></script>
  </body>
</html>
```

Login.html

```
{% extends "base.html" %} {% block title %}Login{% endblock %} {% block conten
t
%}
<form method="POST">
  <h3 align="center">Login</h3>
  <div class="form-group">
    <label for="email">Email Address</label>
    <input
      type="email"
      class="form-control"
      id="email"
      name="email"
      placeholder="Enter email"
    />
  </div>
  <div class="form-group">
    <label for="password">Password</label>
    <input
      type="password"
      class="form-control"
      id="password"
      name="password"
      placeholder="Enter password"
    />
  </div>
  <br />
  <button type="submit" class="btn btn-primary">Login</button>
</form>
{% endblock %}
```

Signup.html

```
{% extends "base.html" %} {% block title %}Sign Up{% endblock %} {% block
content %}
<form method="POST">
  <h3 align="center">Sign Up</h3>
  <div class="form-group">
    <label for="email">Email Address</label>
    <input
      type="email"
      class="form-control"
      id="email"
      name="email"
      placeholder="Enter email"
    />
  </div>
  <div class="form-group">
    <label for="firstName">First Name</label>
    <input
      type="text"
      class="form-control"
      id="firstName"
      name="firstName"
      placeholder="Enter first name"
    />
  </div>
  <div class="form-group">
    <label for="password1">Password</label>
    <input
      type="password"
      class="form-control"
      id="password1"
      name="password1"
      placeholder="Enter password"
    />
  </div>
  <div class="form-group">
    <label for="password2">Password (Confirm)</label>
    <input
      type="password"
      class="form-control"
      id="password2"
      name="password2"
      placeholder="Confirm password"
    />
  </div>
  <br />
  <button type="submit" class="btn btn-primary">Submit</button>
</form>
{% endblock %}
```

Upload.html

```
<!doctype html>
<html lang="en">

<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, user-
scalable=no, initial-scale=1.0, maximum-scale=1.0, minimum-scale=1.0">
    <meta http-equiv="X-UA-Compatible" content="ie=edge">
    <title>Brain Tumor Detection</title>
    <link type="text/css" href="{{ url_for('static', filename='css/bootstrap.m
in.css') }}" rel="stylesheet" />
    <link rel="stylesheet" href="https://stackpath.bootstrapcdn.com/bootstrap/
4.4.1/css/bootstrap.min.css" integrity="sha384-
Vkoo8x4CGsO3+Hhxv8T/Q5PaXtkKtu6ug5TOeNV6gBiFeWPGFN9MuhOf23Q9Ifjh" crossorigin=
"anonymous">
</head>

<body class="bg-light">

    <nav class="navbar navbar-expand-lg navbar-dark bg-primary">
        <!-- <a class="navbar-
brand" href="#">Cancer Predictor and Detector</a> -->
        <a class="navbar-brand" href="#">Brain Tumor Detection</a>
        <button class="navbar-toggler" type="button" data-
toggle="collapse" data-target="#navbarSupportedContent" aria-
```

```
controls="navbarSupportedContent" aria-expanded="false" aria-
label="Toggle navigation">
            <span class="navbar-toggler-icon"></span>
        </button>

        <div class="collapse navbar-collapse" id="navbarSupportedContent">
            <ul class="navbar-nav ml-auto">
                <li class="nav-item active">
                    <a class="nav-link" href="./">Home <span class="sr-
only">(current)</span></a>
                </li>
                <li class="nav-item active">
                    <a class="nav-
link" href="./logout">Logout <span class="sr-only">(current)</span></a>
                </li>
            </ul>
        </div>
    </nav>
    <div class="container mt-auto" style="padding-top: 10%;">

        <form id="upload-
form" action="{{ url_for('views.upload') }}" method="post" enctype="multipart/
form-data">
            <div class="row mt-5">
                <div class="col-md-2"></div>
                <div class="col-md-8 justify-content-center center">

                    <h3 class="display-5 text-
center">Upload MRI image of Brain: </h3>
                    <!-- <input id="file-
picker" type="file" name="file" accept="image/*" multiple> -->
                    <input id="file-
picker" type="file" name="file" accept="image/*" class="file-control col-md-
6 d-flex justify-content-center" style="display: block; margin-
left: auto; margin-right: auto; width: 40%;" required>

                    <div id="msg"></div>
                    <div class="mt-5">
                        <button type="submit" class="btn btn-primary btn-
lg btn-block col-md-8 mr-auto ml-auto" id="upload-
button">Check Tumor Status</button>
                    </div>
        </form>
        </div>
        <div class="col-md-2"></div>
        </div>

    </div>
    <script src="https://code.jquery.com/jquery-
3.4.1.slim.min.js" integrity="sha384-
J6qa4849blE2+poT4WnyKhv5vZF5SrPo0iEjwBvKU7imGFAV0wwj1yYfoRSJoZ+n" crossorigin=
"anonymous"></script>
    <script src="https://cdn.jsdelivr.net/npm/popper.js@1.16.0/dist/umd/popper
.min.js" integrity="sha384-
Q6E9RHvbIyZFJoft+2mJbHaEWldlvI9IOYy5n3zV9zzTtmI3UksdQRVvoxMfooAo" crossorigin=
"anonymous"></script>
    <script src="https://stackpath.bootstrapcdn.com/bootstrap/4.4.1/js/bootstr
ap.min.js" integrity="sha384-
wfSDF2E50Y2D1uUdj0O3uMBJnjuUD4Ih7YwaYd1iqfktj0Uod8GCExl3Og8ifwB6" crossorigin=
"anonymous"></script>
</body>

</html>
```

Complete.html

```
<!doctype html>
<html lang="en">

<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, user-
scalable=no, initial-scale=1.0, maximum-scale=1.0, minimum-scale=1.0">
    <meta http-equiv="X-UA-Compatible" content="ie=edge">
    <title>Brain Tumor Detection</title>
    <link type="text/css" href="{{ url_for('static', filename='css/bootstrap.m
in.css') }}" rel="stylesheet" />
    <link rel="stylesheet" href="https://stackpath.bootstrapcdn.com/bootstrap/
4.4.1/css/bootstrap.min.css" integrity="sha384-
Vkoo8x4CGsO3+Hhxv8T/Q5PaXtkKtu6ug5TOeNV6gBiFeWPGFN9MuhOf23Q9Ifjh" crossorigin=
"anonymous">
</head>

<body class="bg-light">

    <nav class="navbar navbar-expand-lg navbar-dark bg-primary">
        <a class="navbar-brand" href="#">Brain Tumor Detection</a>
        <button class="navbar-toggler" type="button" data-
toggle="collapse" data-target="#navbarSupportedContent" aria-
controls="navbarSupportedContent" aria-expanded="false" aria-
label="Toggle navigation">
            <span class="navbar-toggler-icon"></span>
        </button>

        <div class="collapse navbar-collapse" id="navbarSupportedContent">
            <ul class="navbar-nav ml-auto">
```

```
                <li class="nav-item active">
                    <a class="nav-link" href="./">Home <span class="sr-
only">(current)</span></a>
                </li>
                <li class="nav-item active">
                    <a class="nav-
link" href="./logout">Logout <span class="sr-only">(current)</span></a>
                </li>
              </ul>
        </div>
    </nav>
    <div class="container">

        <h3 class="text-center display-3 mt-3 mb-3">Result</h3>
        <img src="../images/{{image_name}}" height="300px" class="image-
responsive align-center" style="display: block; margin-left: auto; margin-
right: auto; width: 25%;">

        {% if predvalue %}
        <h4 class="text-left display-5 text-center pt-
3">Result: <span style="color:red;">Brain Tumor detected!!!</span></h4> {% els
e %}
        <h4 class="text-left display-5 text-center pt-
3">Result: <span style="color:green;">No Brain Tumor</span></h4> {% endif %}

    </div>
    <script src="https://code.jquery.com/jquery-
3.4.1.slim.min.js" integrity="sha384-
J6qa4849blE2+poT4WnyKhv5vZF5SrPo0iEjwBvKU7imGFAV0wwj1yYfoRSJoZ+n" crossorigin=
"anonymous"></script>
    <script src="https://cdn.jsdelivr.net/npm/popper.js@1.16.0/dist/umd/popper
.min.js" integrity="sha384-
Q6E9RHvbIyZFJoft+2mJbHaEWldlvI9IOYy5n3zV9zzTtmI3UksdQRVvoxMfooAo" crossorigin=
"anonymous"></script>
    <script src="https://stackpath.bootstrapcdn.com/bootstrap/4.4.1/js/bootstr
ap.min.js" integrity="sha384-
wfSDF2E50Y2D1uUdj0O3uMBJnjuUD4Ih7YwaYd1iqfktj0Uod8GCExl3Og8ifwB6" crossorigin=
"anonymous"></script>
</body>

</html>
```

Future scope and improvements

This project has a lot of potential as it uses machine learning and transforms it into a flask web application. There is scope of improvement as some more functionality can be added such as database for each user and his input and output, the website can also be dynamic and show multiple results at the same time

The data pipeline that is used is very effective and flexible for future changes.

The web application can be directly published to cloud flask apps and made live.

Also changing some code and then we can put the program on AWS for public purpose the app is flexible and clearly written which is easy to understand.

Conclusion

This app can be integrated with MRI machines and the direct result that tumour is present or not can be answered. This is useful because sometimes other disorders also create some mis pattern in MRI images and can be mis interpreted as a tumour This application gives accuracy up to 99% and avoid all possible misconceptions about brain tumour.