A Gesture Based Tool for Sterile Browsing of Radiology Images

Project Report

Submitted by

Team PNT2022TMID53060

SSN College of Engineering

Role	Member	Register Number
Team Leader	Karthik D	195001047
Team Members	A Anirudh	195001015
	Pugalarasu K	195001306
	Nestor Ingarshal J	195001069

Table of Contents

Table of Contents **2**

1. INTRODUCTION **5**

1.1 Project Overview 5

1.2 Purpose 5

2. LITERATURE SURVEY **6**

2.1 Existing problem 6

2.2 References 7

2.3 Problem Statement Definition 7

3. IDEATION & PROPOSED SOLUTION **8**

3.1 Empathy Map Canvas 8

3.2 Ideation & Brainstorming 9

Brainstorming 9

Group the Ideas 10

Prioritize the Ideas 11

3.3 Proposed Solution 12

3.4 Problem Solution fit 13

4. REQUIREMENT ANALYSIS **14**

4.1 Functional requirements 14

4.2 Non-Functional requirements 14

5. PROJECT DESIGN **15**

5.1 Data Flow Diagrams 15

5.2 Solution & Technical Architecture 16

5.3 User Stories 17

6. PROJECT PLANNING & SCHEDULING 19

6.1 Sprint Planning & Estimation 19

6.2 Sprint Delivery Schedule 20

6.3 Reports from JIRA 21

Sprint 1 Jira Files 21

Sprint 2 Jira Files 22

Sprint 3 Jira Files 24

Sprint 4 Jira Files 25

7. CODING & SOLUTIONING 26

7.1 Feature - Registration and Login 26

7.2 Feature - Dashboard 27

7.3 Feature - Account Settings 27

7.4 Feature - Help 27

7.5 Feature - Gesture Prediction 27

Database Schema 28

8. TESTING 29

8.1 Test Cases 29

8.2 User Acceptance Testing 31

9. RESULTS 32

9.1 Performance Metrics 32

Model Summary 32

Accuracy 33

10. ADVANTAGES & DISADVANTAGES 33

Advantages 33

Disadvantages 34

11. CONCLUSION 34

12. FUTURE SCOPE 34

13. APPENDIX 36

Source Code 36

Project Codebase Structure 36

App/templates/about.html 36

App/templates/change_password.html 40

App/templates/help.html 43

App/templates/landingpage.html 47

App/templates/login.html 51

App/templates/message_flash.html 54

App/templates/prediction.html 54

App/templates/registration_form_helper.html 57

{% macro render_field(field) %} 57

App/templates/registration.html 58

App/app.py 61

App/database.py 72

Model/Hand-Gesture-Classification.ipynb: 77

Model Training on the Cloud 79

Downloading Model Weights 81

GitHub & Project Demo Link 82

1. INTRODUCTION

1.1 Project Overview

The following interactions must be supported by the system:

- User interacts with the UI (User Interface) to upload the image as input.
- Depending on the different gesture inputs different operations are applied to the input image.
- Once the model analyzes the gesture, the prediction with operation applied on the image is showcased on the UI.

To accomplish this, we have completed all the activities and tasks listed below:

- Data Collection.
    - Collect the dataset
- Data Preprocessing.
    - Import the *ImageDataGenerator* library
    - Configure *ImageDataGenerator* class
    - Apply *ImageDataGenerator* functionality to Train Set and Test Set
- Import the model building Libraries
- Initializing the model
- Model Building

1.2 Purpose

Humans are able to recognize body and sign language easily. This is possible due to the combination of vision and synaptic interactions that were formed along brain development . In order to replicate this skill in computers, some problems need to be solved: how to separate objects of interest in

images and which image capture technology and classification techniques are more appropriate, among others.

In this project Gesture based Desktop automation ,First the model is trained pre-trained on the images of different hand gestures, such as showing numbers with fingers as 1, 2, 3, 4. This model uses the integrated webcam to capture the video frame. The image of the gesture captured in the video frame is compared with the pretrained model and the gesture is identified. If the gesture prediction is 1 then images are blurred; 2, image is resized; 3, image is rotated etc.

2. LITERATURE SURVEY

2.1 Existing problem

A comprehensive literature survey was performed to understand and critically evaluate the current state of research on gesture recognition software for non-intrusive image browsing. The findings are listed below.

[1] Analyzed and classified hand gestures for HCI applications using computer vision-based techniques. Supervised feed-forward neural network with backpropagation for classifying hand gestures.

[2] Introduced the skin model, applied the above to the calibrated position and orientation of the hand used to classify gestures. Gaussian Mixture Model (GMM), filters out non-skin colors of an image and removes lighting bias.

[3] Classified hand gestures using skin color and hand feature cues, used optical flow techniques for hand tracking Image segmentation model based on skin color, finger and palm features. Image segmentation model based on skin color, finger and palm features.

[4] Implemented a vision-based gesture recognition system by clustering hand features, and deployed the model sterile browsing of images. Haar features to represent the hand, fuzzy c-means clustering for classifying the gestures.

[5] Superposed the hand skeletons for each posture into a single dynamic signature for the image, relative positions of features used for gesture classification. Two-dimensional skeleton representation of the human hand, Baddeley's distance as a measure of dissimilarities between model parameters.

2.2 References

[1] G.R.S Murthy, R.S.Jadon, “Hand gesture recognition using neural networks”

[2] Hsien-I Lin, Ming-Hsiang Hsu, and Wei-Kai Chen, “Hand gesture recognition using neural networks”

[3] Y. Fang, K. Wang, J. Cheng and H. Lu, "A Real-Time Hand Gesture Recognition Method"

[4] Wachs, J. et al. “A Real-Time Hand Gesture Interface for Medical Visualization Applications”

[5] Ionescu, B. et al., “Dynamic Hand Gesture Recognition Using the Skeleton of the Hand”

2.3 Problem Statement Definition

With a wave of our hand, there are a lot of things that we can achieve in our daily lives. But there is only a limit that is possible with technology. This is because Humans can recognize sign language with the combination of vision and synaptic interactions with the brain. Computers are unable to do the same thus limiting their capabilities. We wish to push the boundary. With the help of information such as shape, alignment and position of the palm we can obtain certain information.

The gestures are of 2 types namely static and dynamic. By examining the contour of the hand, static hand movements can be determined. Analysis of hand motions yields dynamic hand gestures.

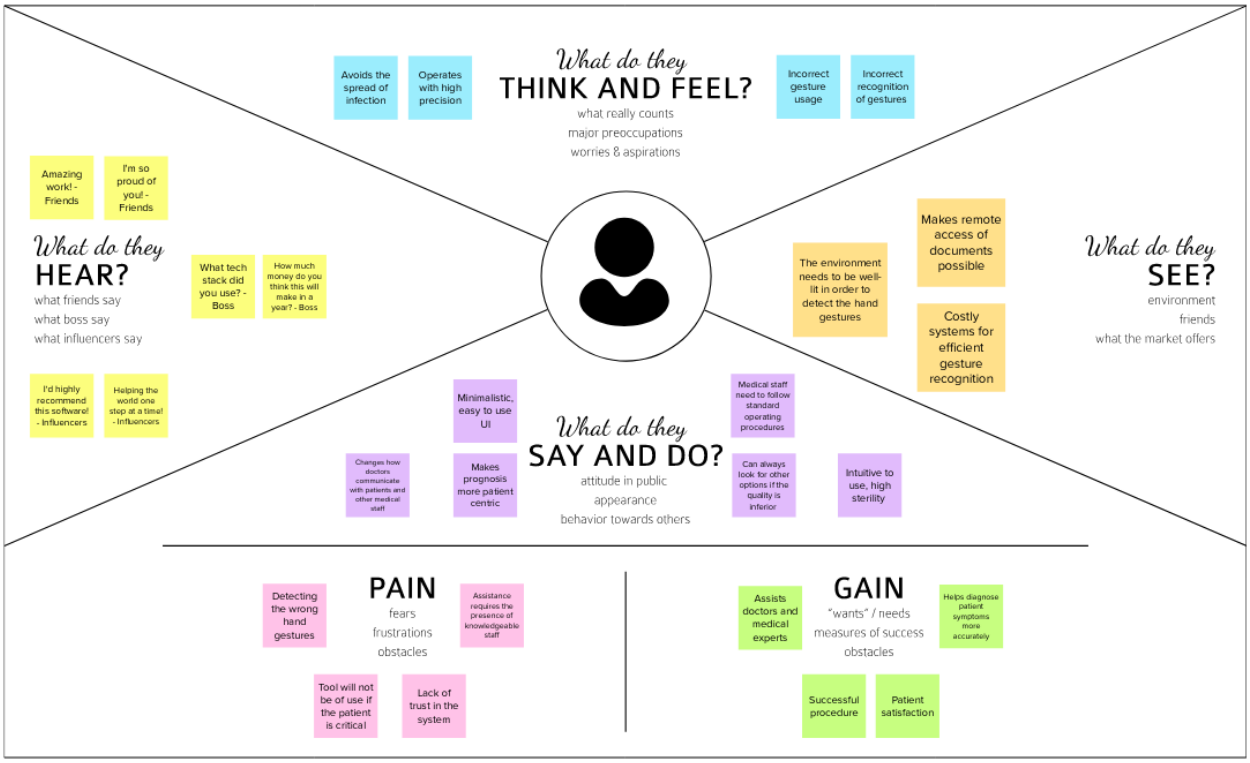
The issue is the inability to instantly recognise motions without a pause in hand motion. We solve these issues using real-time hand gesture detection is real-time hand gesture detection makes use of various identification algorithms, processing speed, and picture processing approaches. In this project, the model is first trained on pictures of various hand motions, such as showing the numbers 1, 2, 3, and 4 with the fingers. The video frame is captured by this model using the built-in webcam.

The gesture is recognised by comparing the image captured in the video frame with the pretrained model. We are going to use a robust deep learning model capable of making predictions quickly. Our system shall be able to recognize gestures at various hand angles. It shall also work whether the hand is close to or far away from the camera. The system as a whole shall be very responsive.

3. IDEATION & PROPOSED SOLUTION

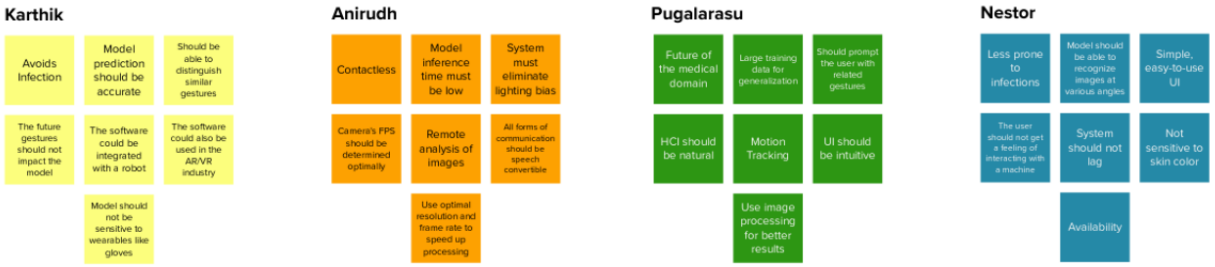
3.1 Empathy Map Canvas

An empathy map is a simple, easy-to-digest visual that captures knowledge about a user's behaviors and attitudes. It is a useful tool to help teams better understand their users. Creating an effective solution requires understanding the true problem and the person who is experiencing it. The exercise of creating the map helps participants consider things from the user's perspective along with his or her goals and challenges.

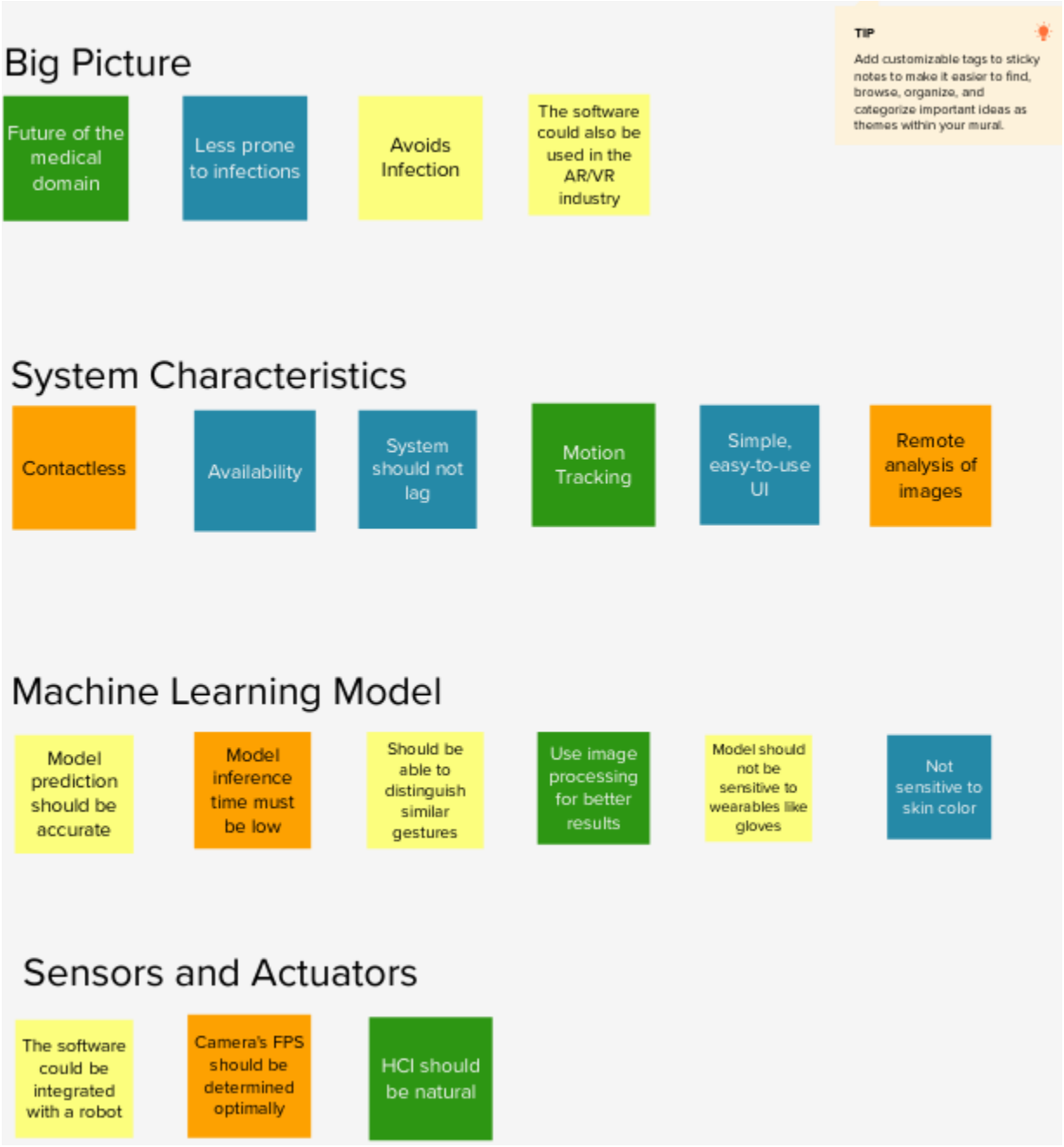


3.2 Ideation & Brainstorming

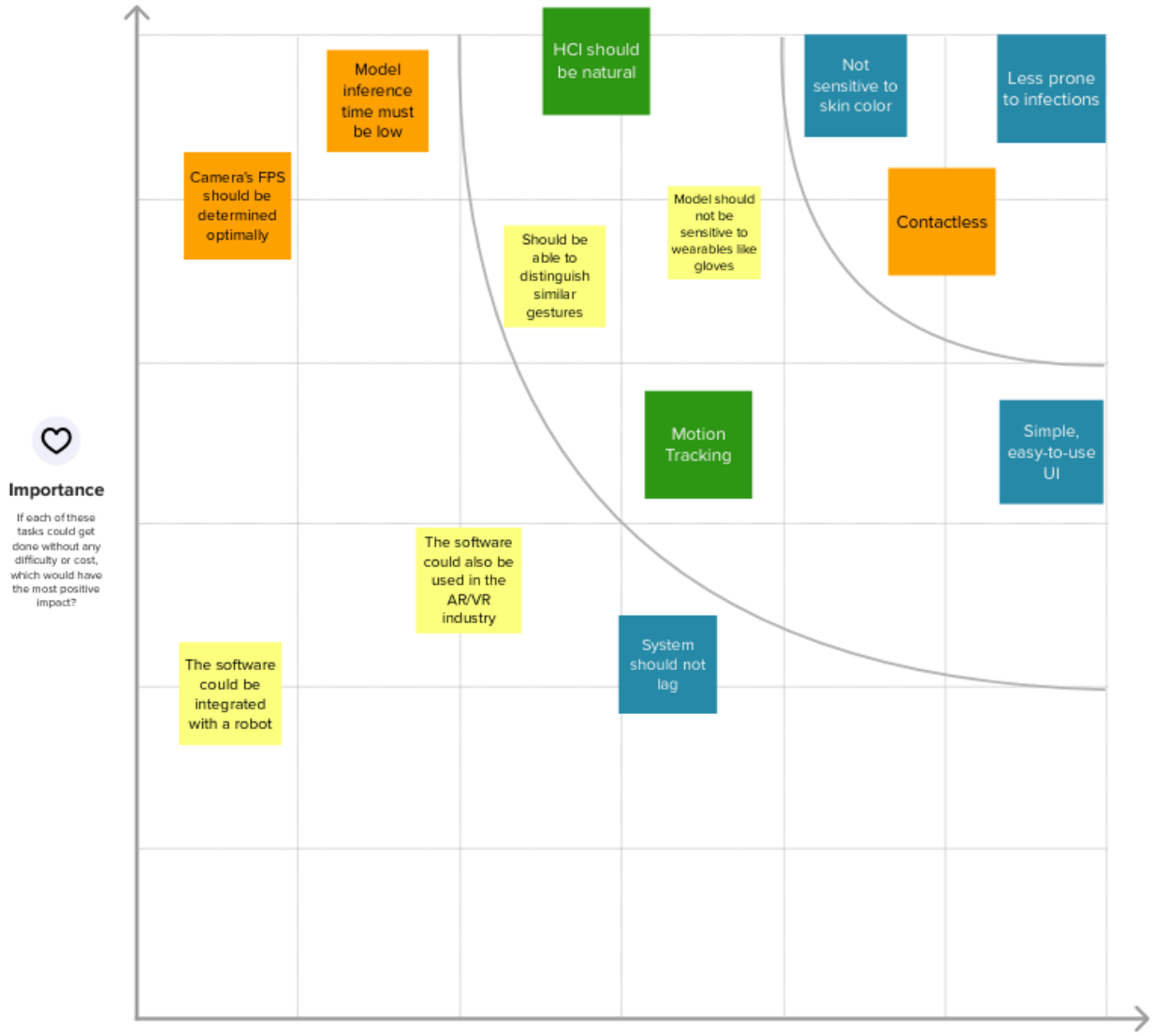
Brainstorming



Group the Ideas



Prioritize the Ideas



3.3 Proposed Solution

S.No.	Parameter	Description
1.	Problem Statement (Problem to be solved)	To design an artificial intelligence tool to recognise and classify hand gestures for sterile browsing of radiology images.
2.	Idea / Solution description	A deep learning model shall be trained on images of hand gestures. The model will be deployed on the cloud for real-time classification.
3.	Novelty / Uniqueness	An end-to-end web application for hand gesture recognition shall be developed. The system shall allow users with limited technical knowledge to use the system with ease. The model shall be able to detect six different types of hand gestures. It shall display the inference in an understandable way to the user.
4.	Social Impact / Customer Satisfaction	The system is aimed at helping doctors and medical staff interact with radiology images in a non-intrusive manner. It helps prevent infections and disease.
5.	Business Model (Revenue Model)	The system can be sold to clients in two ways. It can either be used as a standalone application or on a monthly/yearly subscription basis.
6.	Scalability of the Solution	The system can be used for analyzing other medical images in a non-intrusive manner viz. X-Ray, MRI. It can also be refactored for use in the AR/VR domain.

3.4 Problem Solution fit

A Gesture-Based Tool for Sterile Browsing of Radiology Images | **Problem Solution Fit** | **Team ID:** PNT2022TMID53060

Define CS, fit into CC

1. CUSTOMER SEGMENT(S)

- Used by medical professionals to eliminate sharing indirect physical contact with infected persons through radiology tools.

6. CUSTOMER CONSTRAINTS

- Predefined gestures needs to be remembered by the users of the application.
- Tradeoff between the choice of camera (controls gesture identification accuracy), and the network connectivity and bandwidth requirements.

6. AVAILABLE SOLUTIONS

- Medical professionals may use the tools directly, causing infection spread.
- Professionals may use other control devices like keyboards and joysticks to browse images, but these devices also require contact.

Explore AS, differentiate

Focus on J&P, tap into BE

2. JOBS-TO-BE-DONE / PROBLEMS

- Helpful tooltips and prompts to remind users about the various available gestures, based on the current state of browsing.
- Ensure that ambiguous gestures are discerned properly by targeted training of related samples.

9. PROBLEM ROOT CAUSE

- Browsing control requires the user to memorize specific gestures.
- System is subject to processing delays in gesture detection and implementing corresponding action.
- Unclear and closely related gestures cause ambiguity.

7. BEHAVIOR

- Users need to be provided training to adapt to the gesture based tool.
- User manual and live reference should be made available to assist users.
- Model should function with minimal quality of images.

Focus on J&P, tap into BE

Focus on J&P, tap into BE, understand RC

3. TRIGGERS

- The buzzing use of AI in the medical industry will inspire professionals to adopt the tool.

4. EMOTIONS: BEFORE / AFTER

- Professionals are assured of safety from physically disseminated diseases when browsing radiology images, ensuring sterility.

10. YOUR SOLUTION

- Prevent physically communicable diseases when browsing medical images.
- Convenient and contact-free use of gestures to control viewing devices makes for a quick and efficient operating room environment.
- Helpful in novel situations such as the COVID-19 pandemic, when social distancing was required and physical contact had to avoided.

8. CHANNELS OF BEHAVIOR

ONLINE

- Stable network connection is required for the active functioning of the gesture-based control system.

OFFLINE

- Power availability for display device and gesture capturing camera.
- Offline manuals for quick reference to relevant getures.

Identify Strong TR & EM

4. REQUIREMENT ANALYSIS

4.1 Functional requirements

FR No.	Functional Requirement (Epic)	Sub Requirement (Story / Sub-Task)
FR-1	User Experience	A responsive UI/UX shall be designed to help users interact with the system by using a cohort of hand gestures.
FR-2	Cloud Deployment	The system shall deploy the trained CNN on the cloud.
FR-3	Hand Gesture Identification	The system shall be able to classify the images of hand gestures captured by a camera.
FR-4	Application Domain Pertinence	The CNN used by the system shall be trained on data that is relevant to the application domain.

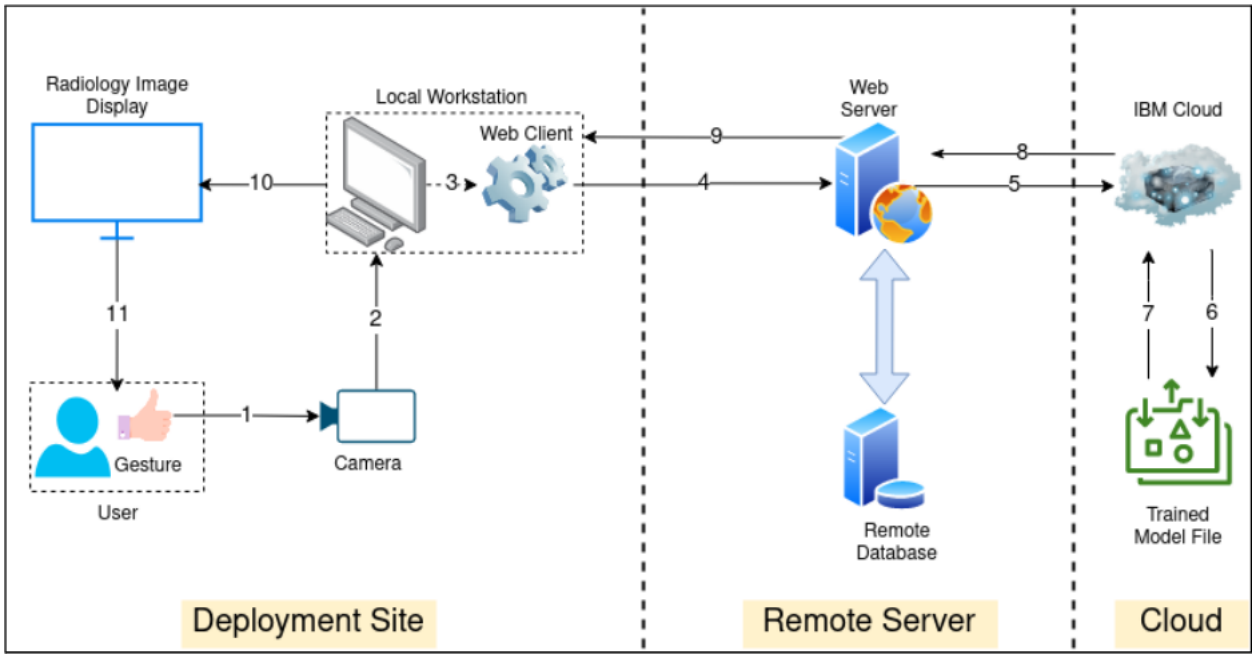
4.2 Non-Functional requirements

NFR No.	Non-Functional Requirement	Description
NFR-1	Usability	The system shall act as an arbitrator between the user and the deep learning model.
NFR-2	Security	The system shall only permit authorized users to access the system. The system shall prevent unauthorized users from entering the system.
NFR-3	Reliability	The system shall be in the operational mode for at least 300 days in a year. In case of failure, the system shall be able to recover in under 3 seconds.
NFR-4	Performance	The system shall be able to respond to a user gesture in under 5 seconds.
NFR-5	Availability	The model shall be available for public use as long as it remains operational.
NFR-6	Scalability	The system shall be accessible to over 1,000,000 concurrent users without any loss of performance.

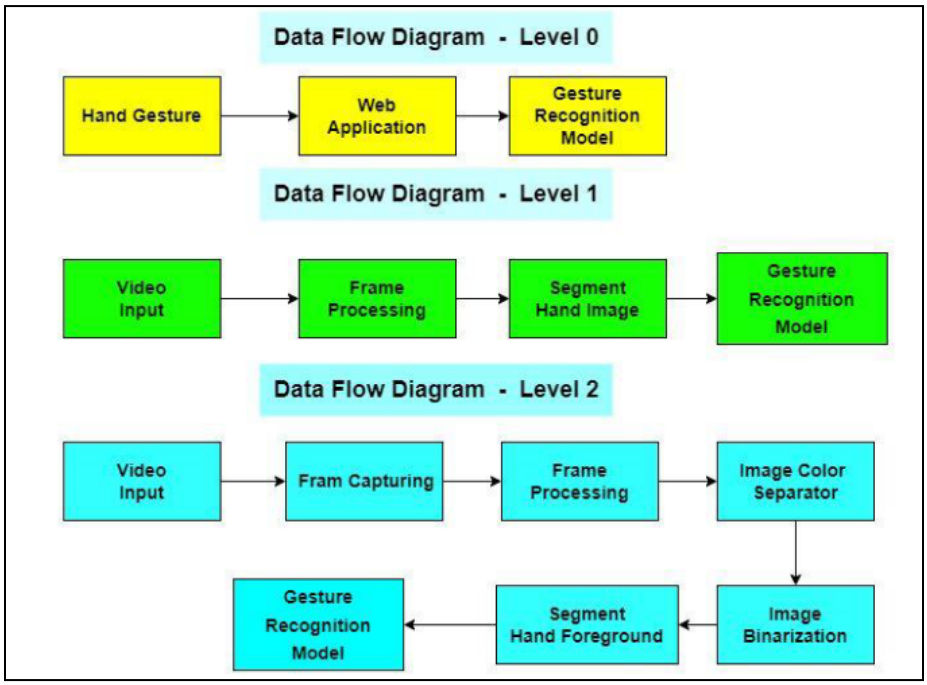
5. PROJECT DESIGN

5.1 Data Flow Diagrams

The following data flow diagram depicts an overview of the proposed system.



The data flow diagrams at levels zero and one are represented below.

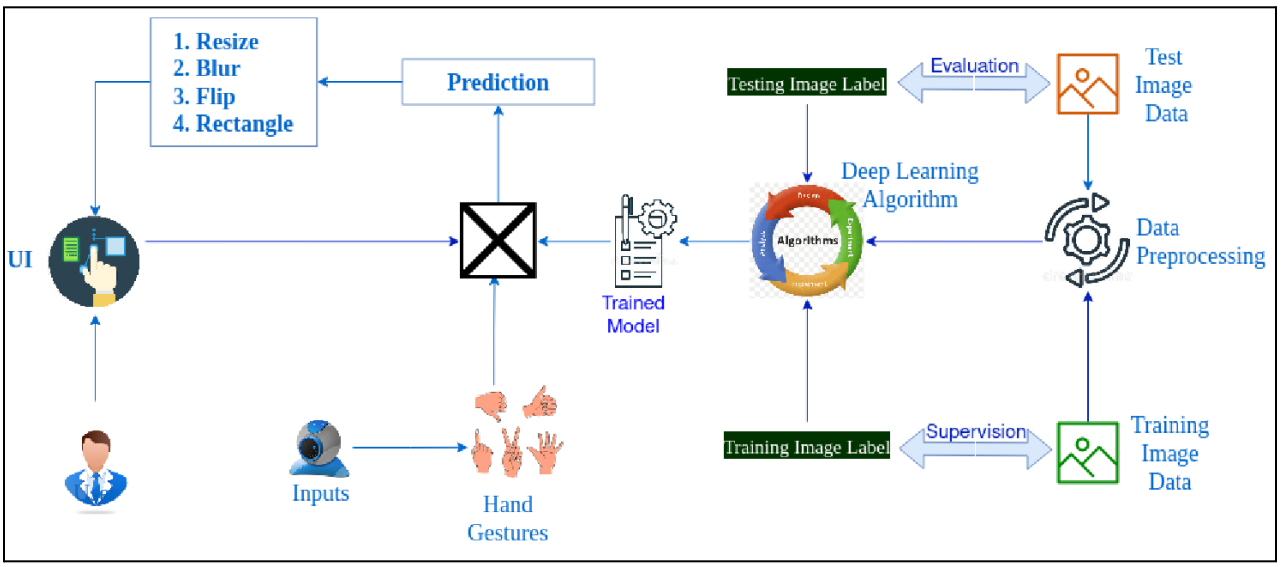


5.2 Solution & Technical Architecture

MVP Idea: A web application that maps hand gestures to transformations to modify static images.

The requirements of the final technical product are as follows:

1. Initialize user all-time access details on signing up.
2. Modify registered users' account details viz. password.
3. Provide a help page to let users get acquainted with the system.
4. Provide a platform for users to upload and modify images according to the recognized gestures.
5. Provide error checking wherever necessary:
    a. During registration, empty values, and password verification are performed.
    b. During sign-in, account verification is performed.
    c. During gesture prediction, ensure that camera access is provided, and that uncorrupted images are uploaded.



5.3 User Stories

User Type	Functional Requirement (Epic)	User Story Number	User Story / Task	Acceptance criteria	Priority	Release
Customer (Web User)	Registration	USN-1	As a user, I can register for the application by entering my email, password, and confirming my password.	I can access my account/dashboard.	High	Sprint-1
Customer (Web User)	Registration	USN-2	As a user, I will receive confirmation email once I have registered for the application	I can receive a confirmation email.	High	Sprint-1
Customer (Web User)	Dashboard	USN-3	As a user, I can log in and access my dashboard page.	I can use the sign-in button and log in to my personalized dashboard.	High	Sprint-1
Customer (Web User)	User Action	USN-4	As a user, I can upload an image of my hand for gesture classification.	I can access my dashboard to upload an image.	High	Sprint-2
Customer (Web User)	User Action	USN-5	As a user, I can access my webcam and take a snapshot/image of my hand for gesture classification.	I can enable webcam access and take a photo.	Medium	Sprint-2
Customer (Web User)	Dashboard	USN-6	As a user, I can view my dashboard to see the classified gesture after uploading an image to the application.	I can view the classified gesture after uploading an image to the application.	High	Sprint-2
Administrator	Model Enhancement	USN-7	As a user, I need a Deep Learning model that can recognize hand gestures with a low error.	The accuracy of the new model should be better than the old one.	Medium	Sprint-2
Administrator	Cloud Deployment	USN-8	As a user, I need the application to be accessible worldwide.	I can access the application from anywhere in the world and at any time.	High	Sprint-3
Administrator	Cloud Deployment	USN-9	As a user, I need the deep learning model to be accessible worldwide.	I can run predictions from anywhere in the world and at any time.	High	Sprint-3
Customer (Web User)	User Action	USN-10	As a user, I should be able to access my dashboard to change my password.	I can change my password and login with my new credentials.	Low	Sprint-3

User Type	Functional Requirement (Epic)	User Story Number	User Story / Task	Acceptance criteria	Priority	Release
Customer Care Executive	Warning	USN-11	As a user, I should be alerted in case: 1. The application can't access my webcam 2. The image I took/uploaded is corrupted	I should be alerted in case my files are corrupted.	Medium	Sprint-4
Customer (Web User)	Launch Application	USN-12	As a user, I can launch the application and upload images of hands for gesture recognition.	I can access the application from anywhere in the world and at any time.	High	Sprint-4
Customer (Web User)	Prediction	USN-13	As a user, I can get the predicted results from the model deployed in the cloud.	I can run predictions from anywhere in the world and at any time.	High	Sprint-4

6. PROJECT PLANNING & SCHEDULING

6.1 Sprint Planning & Estimation

Sprint	Functional Requirement (Epic)	User Story Number	User Story / Task	Story Points	Priority	Team Members
Sprint-1	Registration	USN-1	As a user, I can register for the application by entering my email, password, and confirming my password.	2	High	A Anirudh & Nestor Ingarshal
Sprint-1	Registration	USN-2	As a user, I will receive a confirmation email once I have registered for the application	2	High	D Karthik & K Pugalarasu
Sprint-1	Dashboard	USN-3	As a user, I can login and access my dashboard page.	3	High	A Anirudh & Nestor Ingarshal
Sprint-2	User Action	USN-4	As a user, I can upload an image of my hand for gesture classification.	2	High	D Karthik & K Pugalarasu
Sprint-2	User Action	USN-5	As a user, I can access my webcam and take a snapshot/image of my hand for gesture classification.	1	Medium	A Anirudh & Nestor Ingarshal
Sprint-2	Dashboard	USN-6	As a user, I can view my dashboard to see the classified gesture after uploading an image to the application.	3	High	D Karthik
Sprint-2	Model Enhancement	USN-7	As a user, I need a Deep Learning model that can recognize hand gestures with a low error	2	Medium	A Anirudh
Sprint-3	Cloud Deployment	USN-8	As a user, I need the application to be accessible all over the world.	5	High	A Anirudh & D Karthik
Sprint-3	Cloud Deployment	USN-9	As a user, I need the deep learning model to be accessible all over the world.	3	High	K Pugalarasu & Nestor Ingarshal
Sprint-3	User Action	USN-10	As I user, I should be able to access my dashboard to change my password.	1	Low	A Anirudh & D Karthik

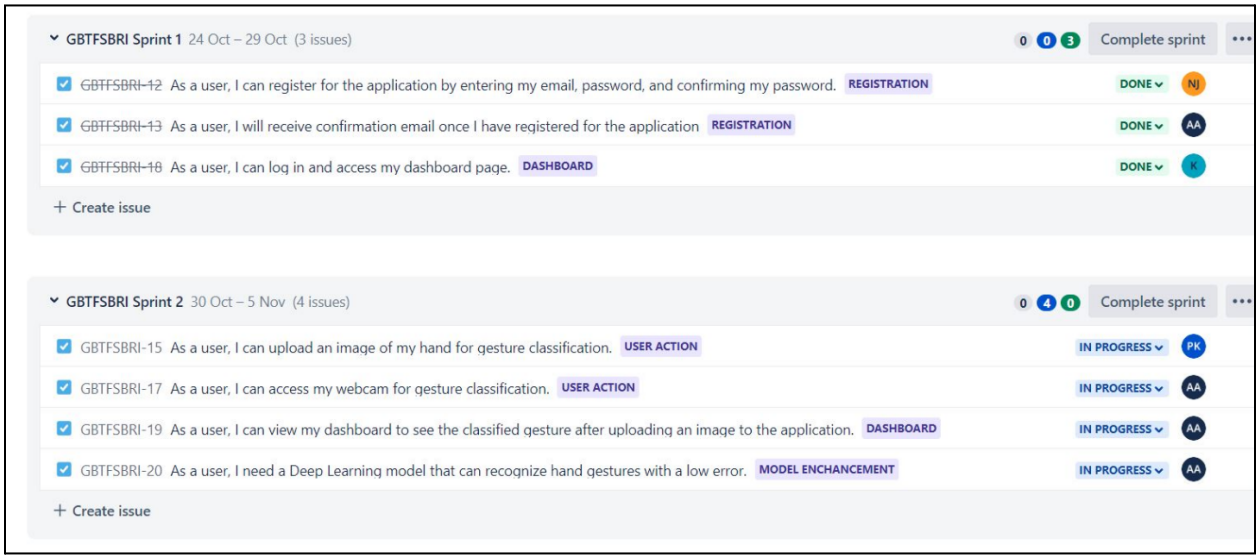
Sprint	Functional Requirement (Epic)	User Story Number	User Story / Task	Story Points	Priority	Team Members
Sprint-4	Warning	USN-11	As a user, I should be alerted in case the application can't access my webcam, or if the image I took/uploaded is corrupted	2	Medium	K Pugalarasu & Nestor Ingarshal
Sprint-4	Launch Application	USN-12	As a user, I can launch the application and upload images of hands for gesture recognition.	3	High	A Anirudh & D Karthik & K Pugalarasu & Nestor Ingarshal
Sprint-4	Prediction	USN-13	As a user, I can get the predicted results from the model deployed in the cloud.	3	High	A Anirudh & D Karthik & K Pugalarasu & Nestor Ingarshal

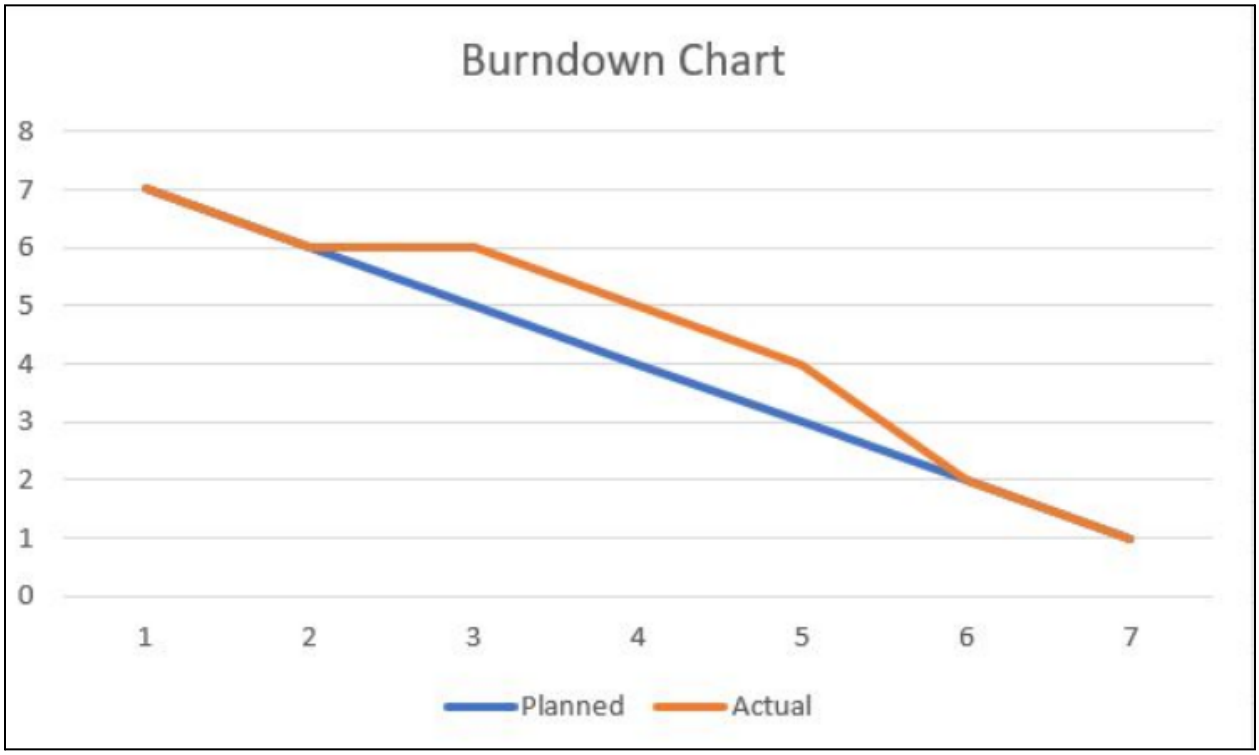
6.2 Sprint Delivery Schedule

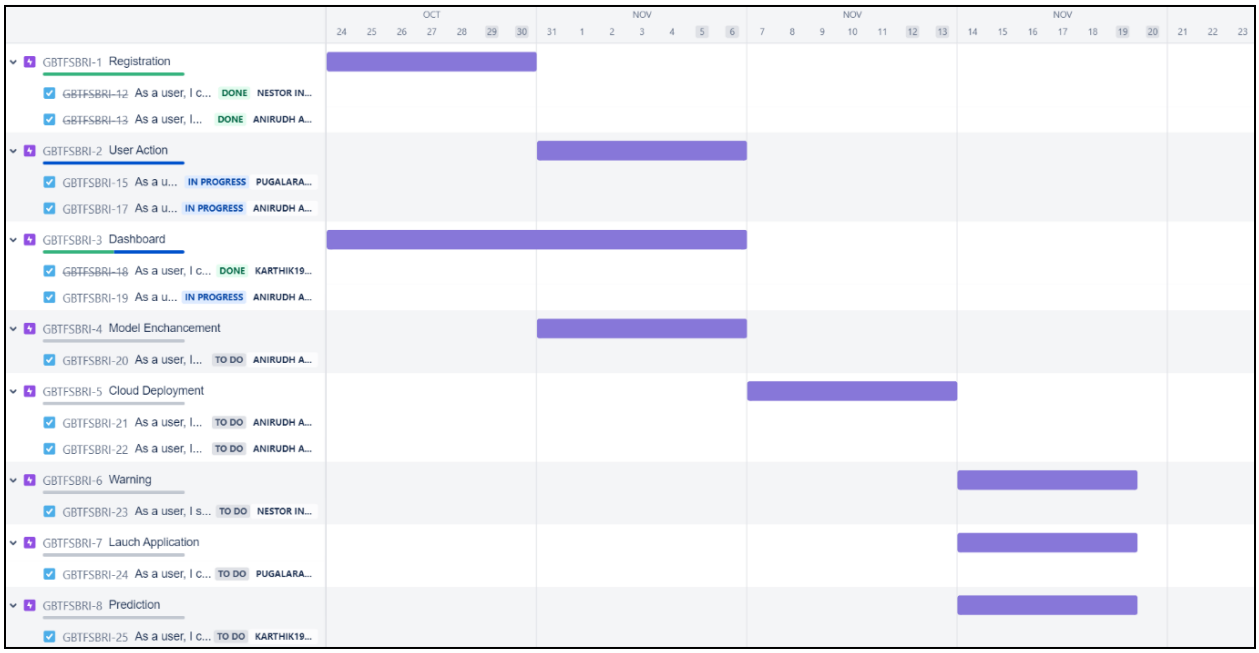
Sprint	Total Story Points	Duration	Sprint Start Date	Sprint End Date (Planned)	Story Points Completed	Sprint Release Date (Actual)
Sprint-1	7	6 Days	24 Oct 2022	29 Oct 2022	7	(Meet Planned Date)
Sprint-2	8	6 Days	31 Oct 2022	05 Nov 2022	8	(Meet Planned Date)
Sprint-3	9	6 Days	07 Nov 2022	12 Nov 2022	9	(Meet Planned Date)
Sprint-4	8	6 Days	14 Nov 2022	19 Nov 2022	8	(Meet Planned Date)

6.3 Reports from JIRA

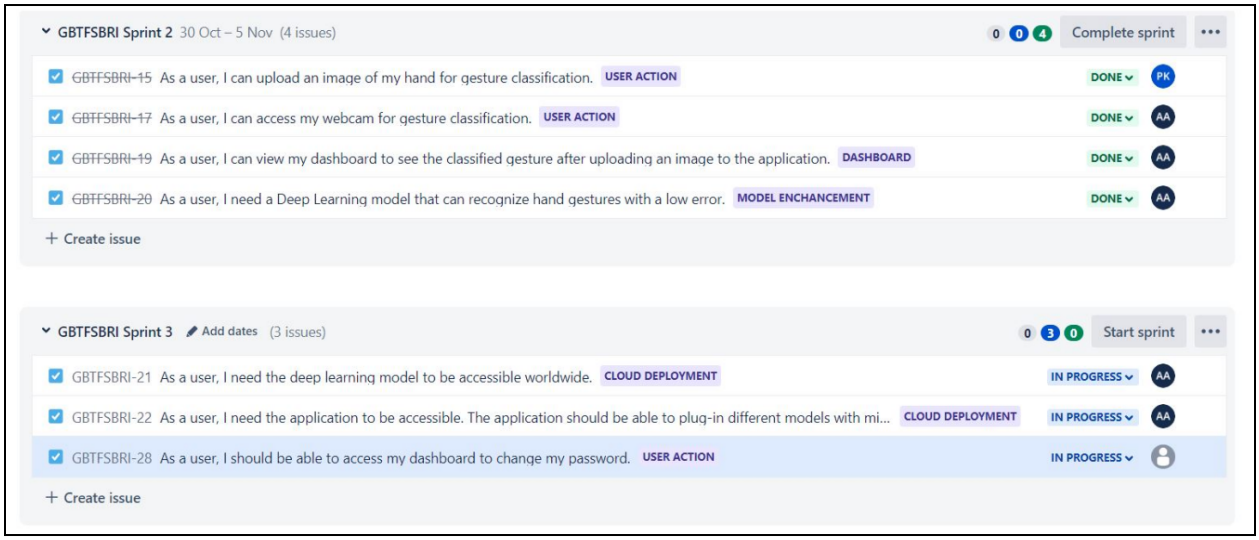
Sprint 1 Jira Files

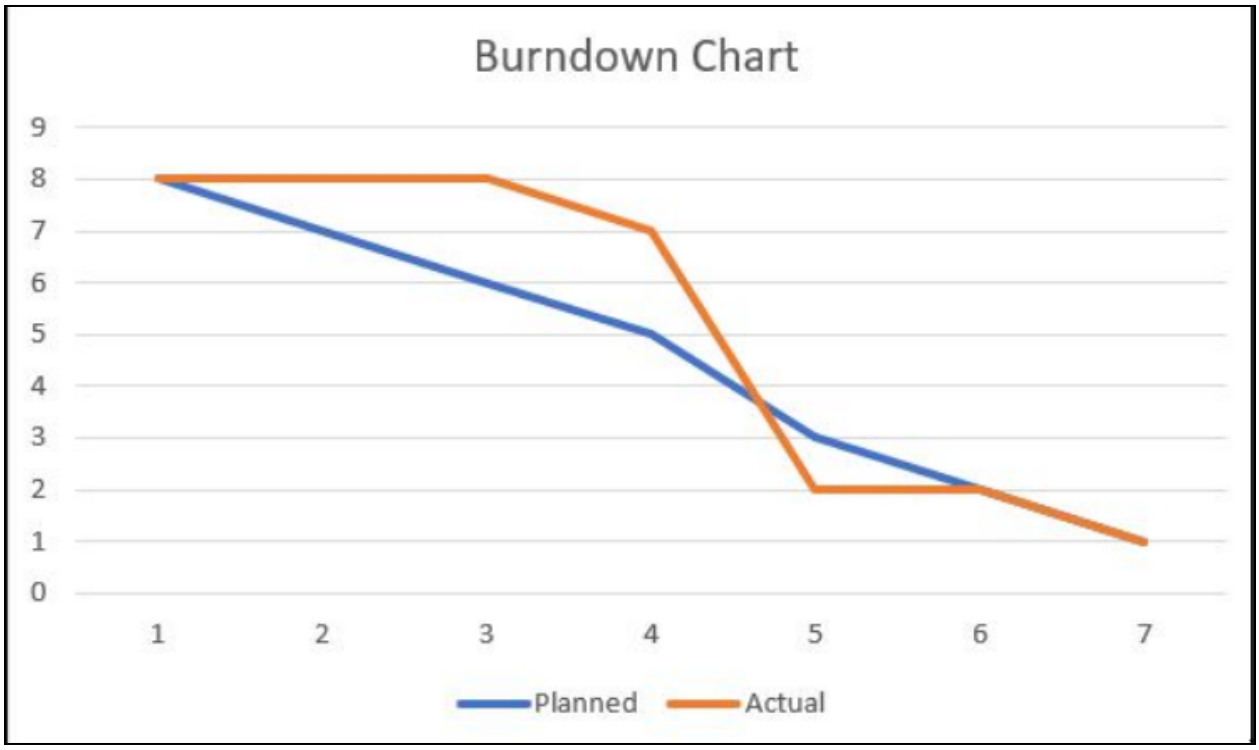


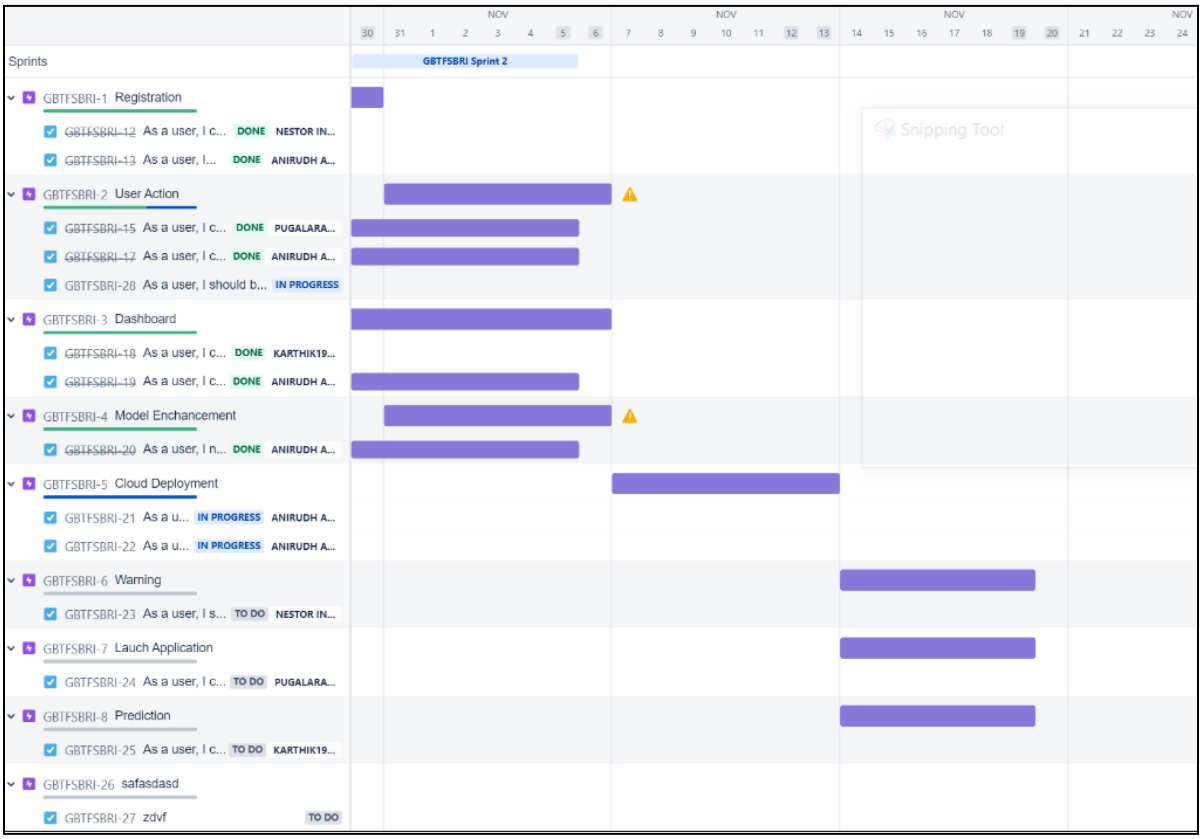




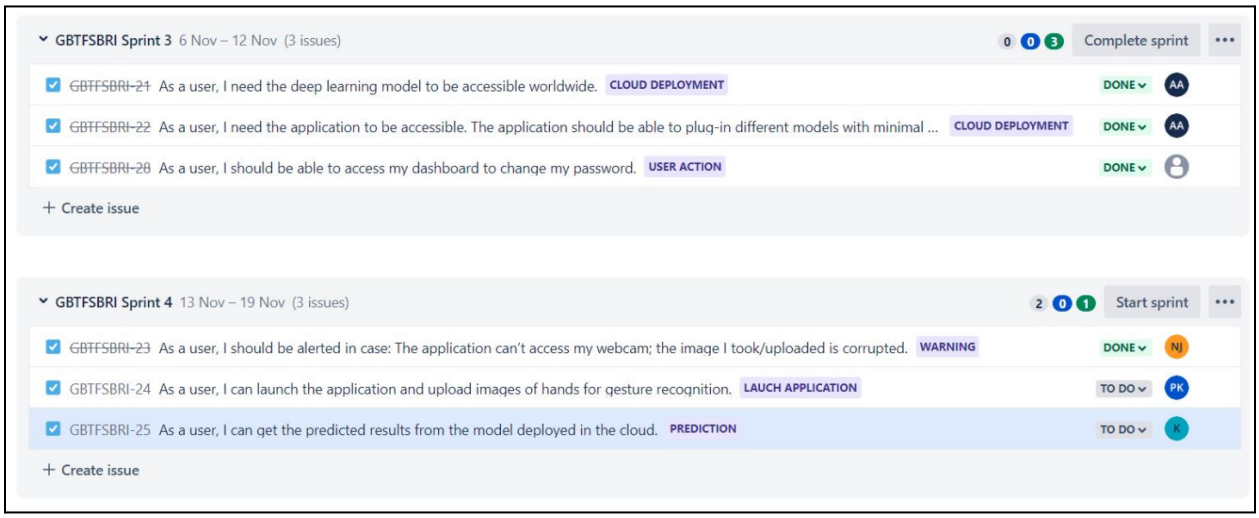
Sprint 2 Jira Files

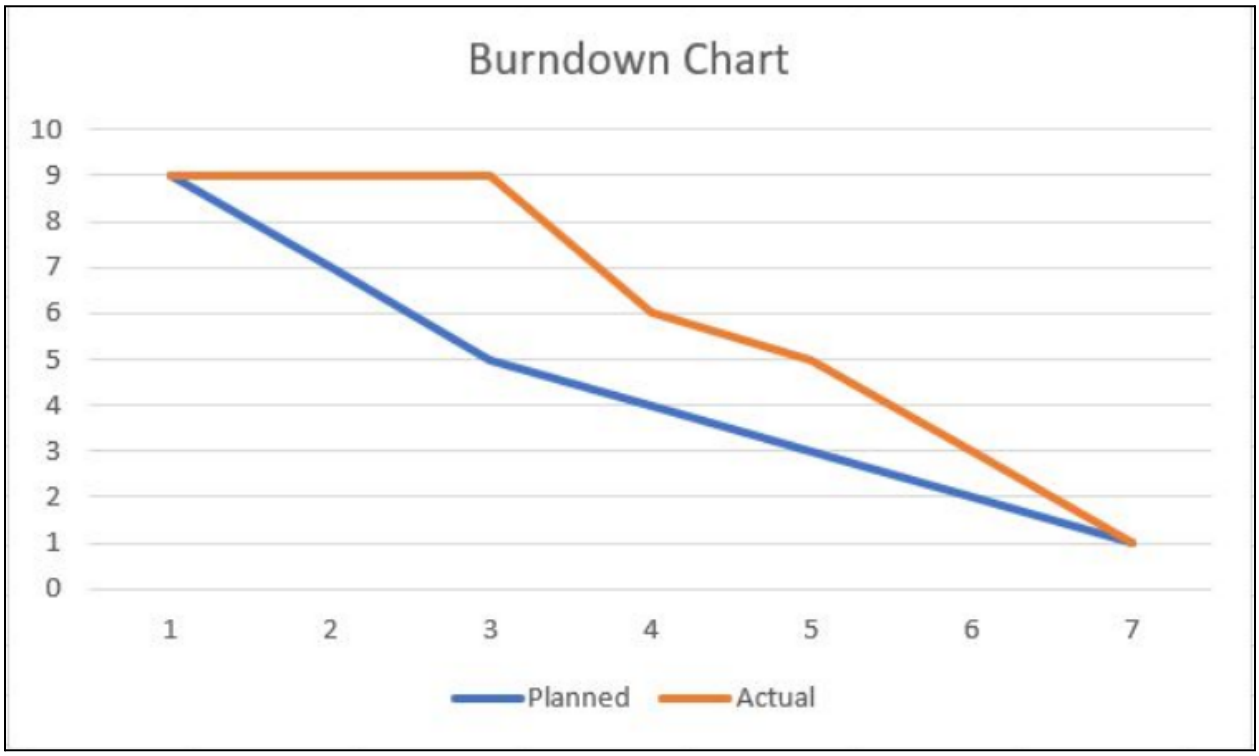


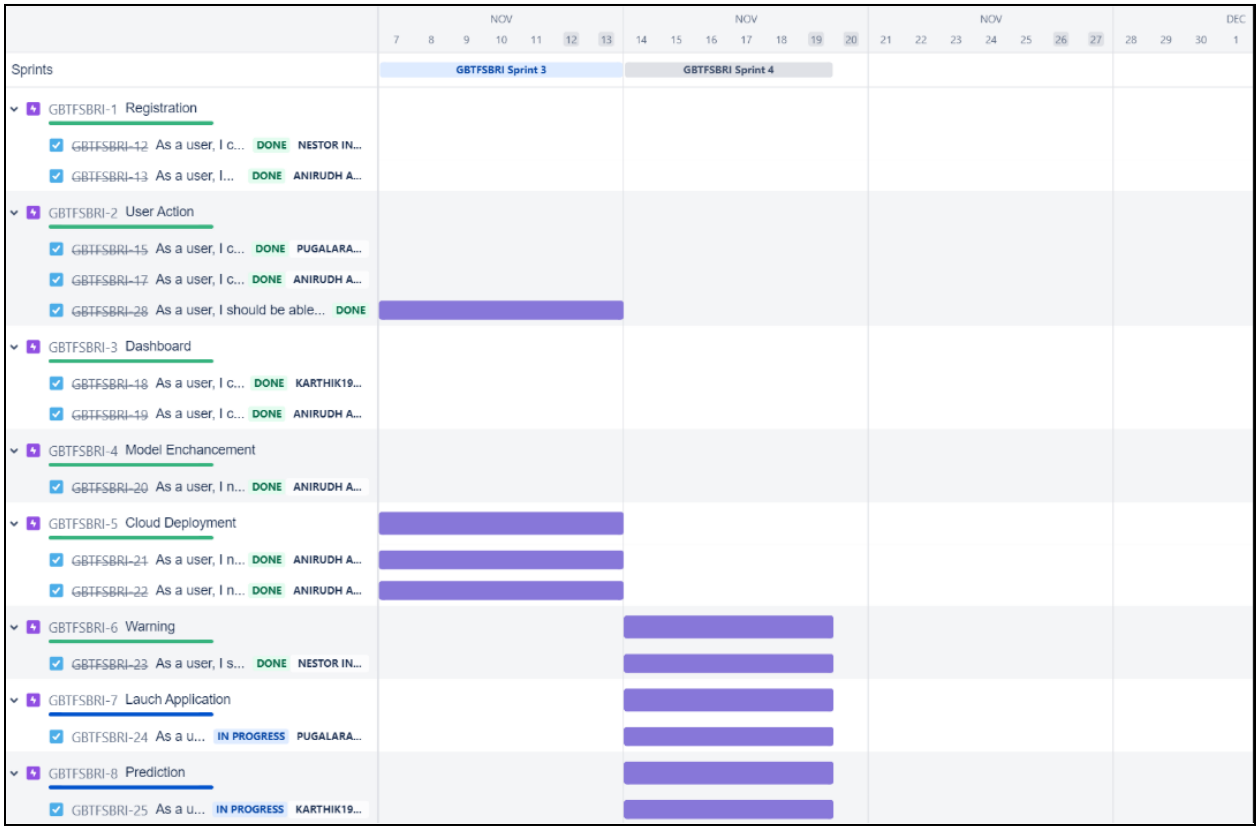

Burndown Chart
9
8
7
6
5
4
3
2
1
0
1
2
3
4
5
6
7
Planned
Actual


NOV
30 31 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24
Sprints
GBTFSBRI Sprint 2
GBTFSBRI-1 Registration
GBTFSBRI-12 As a user, I c... DONE NESTOR IN...
GBTFSBRI-13 As a user, I... DONE ANIRUDH A...
GBTFSBRI-2 User Action
GBTFSBRI-15 As a user, I c... DONE PUGALARA...
GBTFSBRI-17 As a user, I c... DONE ANIRUDH A...
GBTFSBRI-28 As a user, I should b... IN PROGRESS
GBTFSBRI-3 Dashboard
GBTFSBRI-18 As a user, I c... DONE KARTHIK19...
GBTFSBRI-19 As a user, I c... DONE ANIRUDH A...
GBTFSBRI-4 Model Enchancement
GBTFSBRI-20 As a user, I n... DONE ANIRUDH A...
GBTFSBRI-5 Cloud Deployment
GBTFSBRI-21 As a u... IN PROGRESS ANIRUDH A...
GBTFSBRI-22 As a u... IN PROGRESS ANIRUDH A...
GBTFSBRI-6 Warning
GBTFSBRI-23 As a user, I s... TO DO NESTOR IN...
GBTFSBRI-7 Lauch Application
GBTFSBRI-24 As a user, I c... TO DO PUGALARA...
GBTFSBRI-8 Prediction
GBTFSBRI-25 As a user, I c... TO DO KARTHIK19...
GBTFSBRI-26 safasdasd
GBTFSBRI-27 zdvf TO DO
Snipping Tool

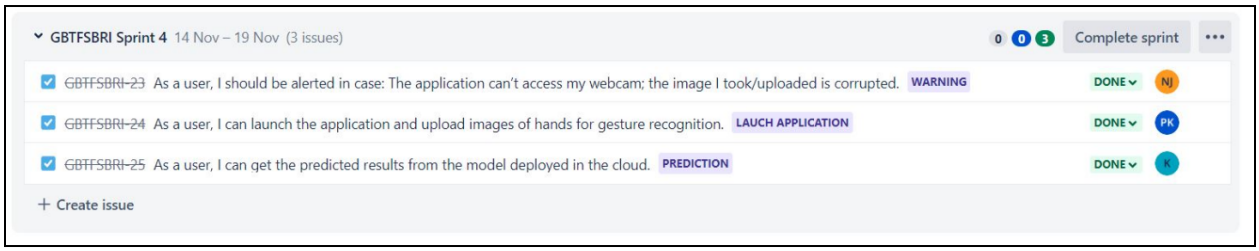
Sprint 3 Jira Files

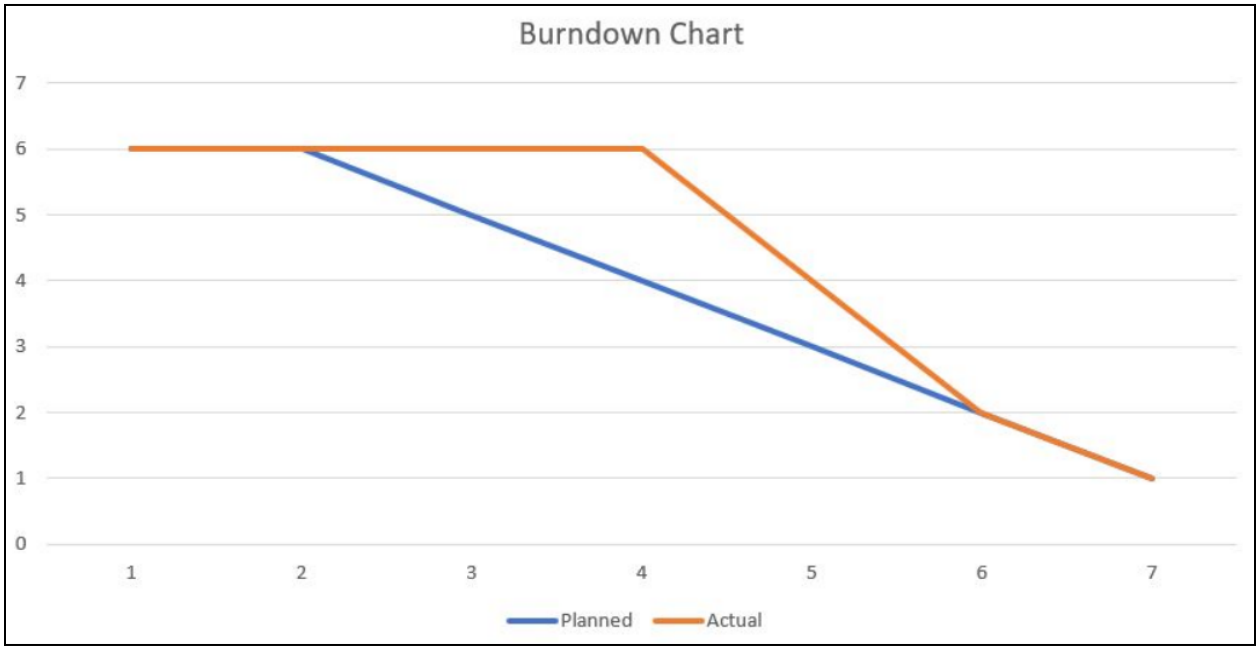


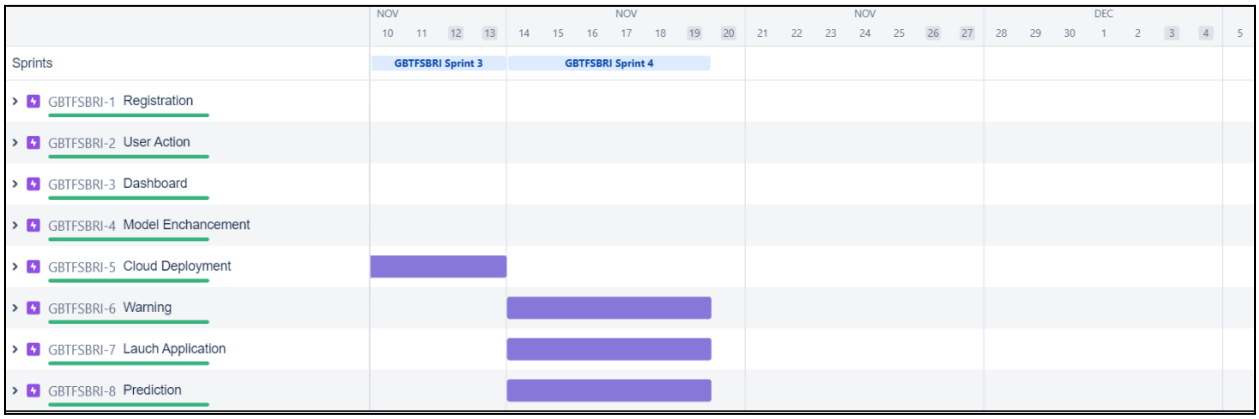




Sprint 4 Jira Files







7. CODING & SOLUTIONING

7.1 Feature - Registration and Login

This feature allows users to sign up to use the application & sign in to the application if they already have an account. The registration and login forms also provide some basic validation like checking if the fields are not empty and user authentication.

7.2 Feature - Dashboard

Registered users will be redirected to a landing page which provides them with information about the number of times they have used the application and when their current session started. The landing page also provides users with quick links to other pages like the help page, gesture prediction page and account settings page.

7.3 Feature - Account Settings

This feature lets users change their account details. For the time being, changing passwords has been supported.

7.4 Feature - Help

This feature walks users through the process of using the gesture prediction tool. It provides users with an easy-to-understand manual to help them get acquainted with the application.

7.5 Feature - Gesture Prediction

This feature is at the heart of the application. It allows users to perform geometric transformations to any kind (file type and dimension) of image. It allows users to upload an image, and use their integrated webcam/external camera to provide six different types of gestures for image modification.

Database Schema

Table definition

USERS

No statistics available.

Name	Data type	Nullable	Length	Scale
name	VARCHAR	Y	50	0
username	VARCHAR	Y	25	0
email	VARCHAR	N	50	0
password	VARCHAR	N	100	0

Table definition

USESSION

No statistics available.

Name	Data type	Nullable	Length	Scale
email	VARCHAR	N	50	0
sessions	VARCHAR	N	100	0

8. TESTING

8.1 Test Cases

- Verify that the user is able to login successfully on entering appropriate credentials.
- The UI elements, such as the card for the login, the button to submit etc are functional and are rendered properly in all devices.
- Registered users who enter an incorrect password will not be redirected to the landing page.
- Unregistered users who attempt to log in to the application will not be redirected to the landing page.
- Verify that the users are able to register themselves to the application.
- The UI elements, such as the card for the registration, the button to submit etc are functional and are rendered properly in all devices.
- The user should not be able to register successfully if any of the fields are left empty.
- The application should prevent the user from re-registering with the same email ID.
- Post successful registration, the user should receive an email from anirudh19015@cse.ssn.edu.in with the user's username and password.
- The application should check if a user's email ID exists before completing the registration process.
- The UI part of the dashboard, the side navbar, the sign out option, help page link and gesture prediction link must be functioning and rendered properly.
- The user should be able to view the number of times he/she has logged in to the application as well as when the current session started. The value for all-time logins for new users should be one.
- The user should be able to modify his/her account details by clicking on the user icon at the bottom-left of the landing page.
- The application should reject a password change if the new password is the same as the current password.
- Users who do not know how to use an application should be able to navigate to the help page. The UI elements on the help page, like cards and images should be rendered

properly.

- The application should render the UI elements like cards, buttons, checkboxes and file upload features on the gesture prediction page properly.
- The application should only access the camera if permission is enabled by the user.
- The application should only begin the gesture recognition process if the user has uploaded an uncorrupted image.
- The application should open a new window with a live webcam stream. The webcam stream should have a region of interest(ROI) for the user to show various gestures.
- Verify that the application is able to recognize gesture zero.
- Verify that the application is able to recognize gesture one.
- Verify that the application is able to recognize gesture two.
- Verify that the application is able to recognize gesture three.
- Verify that the application is able to recognize gesture four.
- Verify that the application is able to recognize gesture five.
- Verify that the user is able to log out of the application.

8.2 User Acceptance Testing

2. Defect Analysis

This report shows the number of resolved or closed bugs at each severity level, and how they were resolved

Resolution	Severity 1	Severity 2	Severity 3	Severity 4	Subtotal
By Design	0	0	0	0	0
Duplicate	0	0	0	0	0
External	2	0	0	0	2
Fixed	0	1	0	0	1
Not Reproduced	0	0	0	0	0
Skipped	0	0	0	0	0
Won't Fix	0	0	0	0	0
Totals	2	1	0	0	3

3. Test Case Analysis

This report shows the number of test cases that have passed, failed, and untested

Section	Total Cases	Not Tested	Fail	Pass
Login	4	0	0	4
Registration	6	0	1	5
Landing Page	2	0	0	2
Account Details	2	0	1	1
Help	1	0	0	1
Gesture Prediction	10	0	0	10
Sign Out	1	0	0	1

9. RESULTS

9.1 Performance Metrics

Model Summary

```
classifier.summary()
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d (Conv2D)             (None, 62, 62, 32)        320

 max_pooling2d (MaxPooling2D  (None, 31, 31, 32)       0
 )

 conv2d_1 (Conv2D)           (None, 29, 29, 32)        9248

 max_pooling2d_1 (MaxPooling  (None, 14, 14, 32)       0
 2D)

 flatten (Flatten)           (None, 6272)              0

 dense (Dense)               (None, 128)               802944

 dense_1 (Dense)             (None, 6)                 774

=================================================================
Total params: 813,286
Trainable params: 813,286
Non-trainable params: 0
_________________________________________________________________
```

Accuracy

```
Epoch 1/20
119/119 [==============================] - 5s 42ms/step - loss: 1.4090 - accuracy: 0.4259 - val_loss: 0.6351 - val_accuracy: 0.8000
Epoch 2/20
119/119 [==============================] - 5s 38ms/step - loss: 0.7041 - accuracy: 0.7003 - val_loss: 0.5916 - val_accuracy: 0.8000
Epoch 3/20
119/119 [==============================] - 5s 41ms/step - loss: 0.5020 - accuracy: 0.7963 - val_loss: 0.6768 - val_accuracy: 0.7333
Epoch 4/20
119/119 [==============================] - 5s 41ms/step - loss: 0.3939 - accuracy: 0.8519 - val_loss: 0.4509 - val_accuracy: 0.8667
Epoch 5/20
119/119 [==============================] - 5s 40ms/step - loss: 0.3088 - accuracy: 0.8889 - val_loss: 0.3790 - val_accuracy: 0.8667
Epoch 6/20
119/119 [==============================] - 5s 40ms/step - loss: 0.2642 - accuracy: 0.8973 - val_loss: 0.4686 - val_accuracy: 0.8667
Epoch 7/20
119/119 [==============================] - 5s 40ms/step - loss: 0.1891 - accuracy: 0.9343 - val_loss: 0.3799 - val_accuracy: 0.9333
Epoch 8/20
119/119 [==============================] - 5s 41ms/step - loss: 0.1654 - accuracy: 0.9360 - val_loss: 0.6095 - val_accuracy: 0.8667
Epoch 9/20
119/119 [==============================] - 5s 40ms/step - loss: 0.1182 - accuracy: 0.9579 - val_loss: 0.4162 - val_accuracy: 0.9333
Epoch 10/20
119/119 [==============================] - 5s 40ms/step - loss: 0.1253 - accuracy: 0.9680 - val_loss: 0.4763 - val_accuracy: 0.9000
Epoch 11/20
119/119 [==============================] - 5s 39ms/step - loss: 0.1078 - accuracy: 0.9646 - val_loss: 0.5120 - val_accuracy: 0.9000
Epoch 12/20
119/119 [==============================] - 5s 39ms/step - loss: 0.0657 - accuracy: 0.9764 - val_loss: 0.2290 - val_accuracy: 0.9667
Epoch 13/20
119/119 [==============================] - 5s 41ms/step - loss: 0.1008 - accuracy: 0.9680 - val_loss: 0.2593 - val_accuracy: 0.9667
Epoch 14/20
119/119 [==============================] - 5s 41ms/step - loss: 0.0969 - accuracy: 0.9663 - val_loss: 0.2971 - val_accuracy: 0.9667
Epoch 15/20
119/119 [==============================] - 5s 40ms/step - loss: 0.0698 - accuracy: 0.9731 - val_loss: 0.2917 - val_accuracy: 0.9667
Epoch 16/20
119/119 [==============================] - 5s 40ms/step - loss: 0.0492 - accuracy: 0.9832 - val_loss: 0.2443 - val_accuracy: 0.9333
Epoch 17/20
119/119 [==============================] - 5s 40ms/step - loss: 0.0212 - accuracy: 0.9949 - val_loss: 0.2986 - val_accuracy: 0.9667
Epoch 18/20
119/119 [==============================] - 5s 42ms/step - loss: 0.0190 - accuracy: 0.9933 - val_loss: 0.1804 - val_accuracy: 0.9333
Epoch 19/20
119/119 [==============================] - 5s 40ms/step - loss: 0.0799 - accuracy: 0.9646 - val_loss: 0.2960 - val_accuracy: 0.9667
Epoch 20/20
119/119 [==============================] - 5s 40ms/step - loss: 0.0567 - accuracy: 0.9848 - val_loss: 0.2684 - val_accuracy: 0.9667
<keras.callbacks.History at 0x7fb22655a880>
```

At the end of 20 training epochs, the model reached an accuracy of ***98.47% on the training data***, and ***96.67% on the validation data***.

10. ADVANTAGES & DISADVANTAGES

Advantages

- Lets users perform transformations to static images according to recognized gestures without needing to know what happens under the hood.
- Lets users upload images of any file type and of any dimension.
- Provides users with a help page to get them acquainted with the application.
- Lets users modify their account details as and when required.

- Gives users information about the number of times they have accessed the application for billing purposes.

Disadvantages

- Email verification before registration is not supported.
- The frame processing rate is higher than expected.
- The model can predict gestures very accurately only if the background is not cluttered and the arm isn't very far away from the camera.
- The model sometimes confuses the gestures for two and three.

11. CONCLUSION

A Gesture Based Tool for Sterile Browsing of Radiology Images supporting myriad features has been designed and implemented using IBM's cloud based tools viz. IBM DB2 Cloud Database, IBM Object Storage, IBM Watson Studio and IBM Cloud Pak for Watson Studio. Simple Mail Transfer Protocol was used to send emails to registered users. The application was deployed locally by downloading the weights for the deep learning model using the IBM Watson Machine Learning Client.

12. FUTURE SCOPE

Future avenues of work in this application would include,

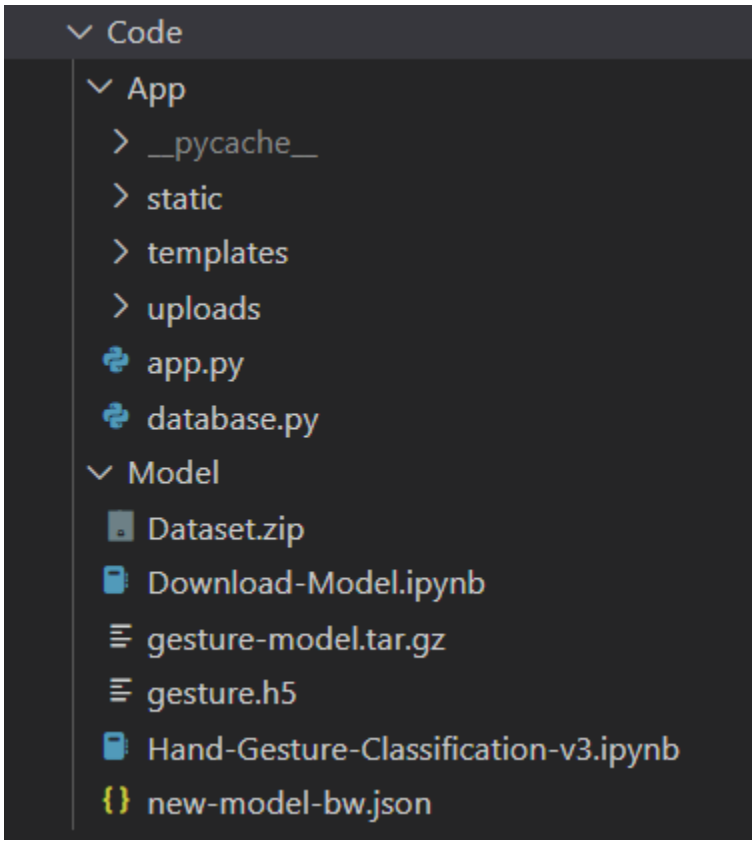
1. Deploying the entire application on the cloud by using IBM's cloud services.
2. Developing more complex deep learning models for gesture recognition. These models would take into account various environmental conditions, user limitations and physical anatomy of the hand.

3. Supporting an increased application inference time by using multi-threading or other methods to speed up model inference and frame processing.
4. Converting the tool into an enterprise solution by billing users according to their usage of the tool.

13. APPENDIX

Source Code

Project Codebase Structure



App/templates/about.html

```
<!DOCTYPE html>

<html lang="en">



<head>

    <!-- Required Meta Tags -->

    <meta charset="UTF-8">
```

```
    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <link rel="stylesheet"
href="https://cdn.jsdelivr.net/npm/bootstrap@5.2.2/dist/css/bootstrap.min.
css">

    <title>A Gesture Based Tool for Sterile Browsing of Radiology
Images</title>

</head>



<body style="overflow-x: hidden; background-color: #e3e6e4;">

    <script
src="https://cdn.jsdelivr.net/npm/bootstrap@5.2.2/dist/js/bootstrap.min.js
"></script>



    <nav class="navbar bg-dark">

        <div class="container-fluid">

          <a class="navbar-brand"
href="https://smartinternz.com/guided-project/a-gesture-based-tool-for-ste
rile-browsing-of-radiology-images-cnn-and-open-cv" target="_blank"
style="color:white">

            <img src="{{url_for('static', filename='tool-icon.png')}}"
alt="Logo" width="23" height="24" class="d-inline-block align-text-top">

            IBM Nalaiya Thiran - PNT2022TMID53060

        </a>

        <ul class="nav nav-pills">

         <li class="nav-item">

            <a class="nav-link" href="#scrollspyHeading1"
style="color:white" >Sign Up</a>
```

```
          </li>

          <li class="nav-item">

            <a class="nav-link" href="#scrollspyHeading2"
style="color:white">Sign In</a>

          </li>

        </ul>

    </div>

    </nav>



    <div class="container-fluid text-center" style="margin: 40px;">

        <div class="jumbotron text-center">

            <h1>A Gesture Based Tool for Sterile Browsing of Radiology
Images</h1>

        </div>

    </div>



    <div class="container">

    <div class="col d-flex justify-content-center">

    <div class="card  justify-content-center" style="width: 80%;">

        <img class="card-img-top rounded mx-auto d-block"
src="{{url_for('static', filename='about-photo.jpg')}}" alt="Card image
cap" style="max-width: 40%;max-height: 60%; margin: 10px;">

        <div class="card-body">

          <p class="card-text" align="justify">Humans are able to
recognize body and sign language easily. This is possible due to the
```

```
combination of vision and synaptic interactions that were formed along
brain development. In order to replicate this skill in computers, some
problems need to be solved: how to separate objects of interest in images
and which image capture technology and classification technique are more
appropriate, among others.</p>

          <p class="card-text" align="justify">In this project Gesture
based Desktop automation, first the model is trained pre trained on the
images of different hand gestures, such as a showing numbers with fingers
as 1,2,3 and 4. This model uses the integrated webcam to capture the video
frame. The image of the gesture captured in the video frame is compared
with  the Pre-trained model and the gesture is identified. If the gesture
predictes is 1 then images is blurred; 2 image is resized; 3 image is
rotated etc.</p>

        </div>

      </div>

    </div>

    </div>



    <div class="d-flex justify-content-center" style="margin: 20px;">

    <div class="row"  style="width: 50%;">

        <div class="col-sm-6" id="scrollspyHeading1">

          <div class="card">

            <div class="card-body">

              <h5 class="card-title">Sign Up</h5>

              <p class="card-text">Don't have an account? Sign up
below!</p>

              <a class="btn btn-primary"
href="{{url_for('register')}}">Sign Up</a>
```

```
            </div>
          </div>
        </div>
        <div class="col-sm-6" id="scrollspyHeading2">
          <div class="card">
            <div class="card-body">
              <h5 class="card-title">Sign In</h5>
              <p class="card-text">Existing User? Sign in here!</p>
              <a class="btn btn-primary" href="{{url_for('login')}}">Sign
in</a>
            </div>
          </div>
        </div>
      </div>
    </div>


</body>
</html>
```

App/templates/change_password.html

```
<!DOCTYPE html>
<html lang="en">
<head>
```

```
    <meta charset="UTF-8">

    <meta http-equiv="X-UA-Compatible" content="IE=edge">

    <link rel="stylesheet"
href="https://cdn.jsdelivr.net/npm/bootstrap@5.2.2/dist/css/bootstrap.min.
css">

    <title>A Gesture Based Tool for Sterile Browsing of Radiology
Images</title>

</head>

<body style="background-color: #f0f5f2;">

    <script
src="https://cdn.jsdelivr.net/npm/bootstrap@5.2.2/dist/js/bootstrap.min.js
"></script>

    {% from "registration_form_helper.html" import render_field %}

    {% include "message_flash.html" %}



    <nav class="navbar navbar-expand-lg navbar-dark bg-dark"
style="padding: 10px">

        <a class="navbar-brand">{{session.username}}'s Account</a>

        <button class="navbar-toggler" type="button"
data-toggle="collapse" data-target="#navbarText"
aria-controls="navbarText" aria-expanded="false" aria-label="Toggle
navigation">

          <span class="navbar-toggler-icon"></span>

        </button>

        <div class="collapse navbar-collapse" id="navbarText">

          <ul class="navbar-nav mr-auto">
```

```
            <li class="nav-item">
                <a class="nav-link"
href="{{url_for('landingpage')}}">Home</a>
              </li>
              <li class="nav-item">
                <a class="nav-link" href="{{url_for('help')}}">Help</a>
              </li>
            <li class="nav-item">
              <a class="nav-link" href="{{url_for('predict')}}">Gesture
Prediction</a>
            </li>
          </ul>
        </div>
    </nav>


    <div class="card text-white bg-dark mb-3 mx-auto" style="max-width:
80%; margin-top: 5%;">
        <div class="card-header">Account Settings</div>
        <div class="card-body">
          <h5 class="card-title">Change Password</h5>
          <p class="card-text">Change your account's password here. Use
the first text box to enter the new password. Confirm it by typing in the
same password in the text box below.</p>


        <form class="form-group" method="post" action="">
```

```
            <div class="form-group" style="padding: 10px;">

                {{render_field(form.new_password, class="form-control")}}

            </div>



            <div class="form-group" style="padding: 10px;">

                {{render_field(form.new_confirm, class="form-control")}}

            </div>

            <div class="card-footer text-center">

                <p><input type="submit" value="Submit" class="btn
btn-primary" style="margin-top: 20px;"></p>

            </div>

        </form>

        </div>

    </div>



</body>

</html>
```

App/templates/help.html

```
<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">
```

```
    <meta http-equiv="X-UA-Compatible" content="IE=edge">

    <link rel="stylesheet"
href="https://cdn.jsdelivr.net/npm/bootstrap@5.2.2/dist/css/bootstrap.min.
css">

    <title>A Gesture Based Tool for Sterile Browsing of Radiology
Images</title>

</head>

<body style="overflow-x: hidden; background-color: #e3e6e4;">



    <script
src="https://cdn.jsdelivr.net/npm/bootstrap@5.2.2/dist/js/bootstrap.min.js
"></script>



    <nav class="navbar navbar-expand-lg navbar-dark bg-dark"
style="padding: 10px">

        <a class="navbar-brand">{{session.username}}'s Account</a>

        <button class="navbar-toggler" type="button"
data-toggle="collapse" data-target="#navbarText"
aria-controls="navbarText" aria-expanded="false" aria-label="Toggle
navigation">

          <span class="navbar-toggler-icon"></span>

        </button>

        <div class="collapse navbar-collapse" id="navbarText">

          <ul class="navbar-nav mr-auto">

            <li class="nav-item">

                <a class="nav-link"
href="{{url_for('landingpage')}}">Home</a>
```

```
              </li>

            <li class="nav-item">

              <a class="nav-link" href="{{url_for('predict')}}">Gesture
Prediction</a>

            </li>

          </ul>

        </div>

    </nav>



    <div class="row justify-content-center" style="margin-top: 20px;">

      <div class="card" style="width: 80%;">

          <img class="card-img-top mx-auto" src="{{url_for('static',
filename='access-webcam.jpg')}}" alt="Card image cap" style="max-width:
40%; height: auto; padding-top: 10px;">

          <div class="card-body">

          <h5 class="card-title">Step I</h5>

          <p class="card-text">First, you need to allow the application to
access your webcam. The application uses this video stream from the webcam
to capture and map hand gestures.</p>

          </div>

      </div>

  </div>



    <div class="row justify-content-center" style="margin-top: 20px;">

        <div class="card" style="width: 80%;">
```

```
            <img class="card-img-top mx-auto" src="{{url_for('static',
filename='upload-image.png')}}" alt="Card image cap" style="padding-top:
10px; max-width: 40%; height: auto; ">

            <div class="card-body">

            <h5 class="card-title">Step II</h5>

            <p class="card-text">Second, you need to upload an image. You
can do so by clicking on the upload image icon. This image will be
modified according to the recognized hand gestures.</p>

            </div>

        </div>

    </div>



    <div class="row justify-content-center" style="margin-top: 20px;
margin-bottom: 20px;">

        <div class="card" style="width: 80%;">

            <img class="card-img-top mx-auto" src="{{url_for('static',
filename='gestures.png')}}" alt="Card image cap" style="max-width: 40%;
height: auto; padding-top: 10px;">

            <div class="card-body">

            <h5 class="card-title">Step III</h5>

            <p class="card-text">Finally, you can run the deep learning
model for gesture recognition! All you need to do is to click the
`Predict` button on the page.</p>

            </div>

        </div>

    </div>
```

```
    <div class="d-flex justify-content-center">
    <a id="back-to-top" href="#" class="btn btn-dark btn-lg back-to-top" role="button" style="margin-bottom: 20px;">Scroll to Top</a>
    </div>


</body>
</html>
```

App/templates/landingpage.html

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <link rel="stylesheet" href="https://cdn.jsdelivr.net/npm/bootstrap@5.2.2/dist/css/bootstrap.min.css">
    <title>A Gesture Based Tool for Sterile Browsing of Radiology Images</title>
</head>
<body style="background-color: #e3e6e4;">
    <script src="https://cdn.jsdelivr.net/npm/bootstrap@5.2.2/dist/js/bootstrap.min.js"></script>
```

```
    <div class="container-fluid">

        <div class="row flex-nowrap">

            <div class="col-auto col-md-3 col-xl-2 px-sm-2 px-0 bg-dark">

                <div class="d-flex flex-column align-items-center
align-items-sm-start px-3 pt-2 text-white min-vh-100">

                    <a class="d-flex align-items-center pb-3 mb-md-0
me-md-auto text-white text-decoration-none">

                        <span class="fs-5 d-none d-sm-inline">Menu</span>

                    </a>

                    <ul class="nav nav-pills flex-column mb-sm-auto mb-0
align-items-center align-items-sm-start" id="menu">

                        <li>

                            <a href="{{url_for('help')}}" class="nav-link
px-0 align-middle">

                                <i class="fs-4 bi-people"></i> <span
class="ms-1 d-none d-sm-inline">Help</span> </a>

                        </li>

                        <li>

                            <a href="{{url_for('predict')}}"
class="nav-link px-0 align-middle">

                                <i class="fs-4 bi-people"></i> <span
class="ms-1 d-none d-sm-inline">Gesture Prediction</span> </a>

                        </li>

                        <li>

                            <a href="{{url_for('signout')}}"
class="nav-link px-0 align-middle">
```

```
                                <i class="fs-4 bi-people"></i> <span
class="ms-1 d-none d-sm-inline">Sign Out</span> </a>

                        </li>



                    </ul>

                    <hr>

                    <div class="dropdown pb-4" data-toggle="tooltip"
title="Account Settings">

                        <a href="{{url_for('changepass')}}" class="d-flex
align-items-center text-white text-decoration-none" id="dropdownUser1"
\aria-expanded="false">

                            <img src="{{url_for('static',
filename='user-photo.jpg')}}" alt="user photo" width="30" height="30"
class="rounded-circle">

                            <span class="d-none d-sm-inline
mx-1">{{session.name}}</span>

                        </a>

                    </div>

                </div>

            </div>



            <div class="col py-3" style="margin-top: 20px;">

                <div class="card">

                    <h5 class="card-header">Welcome!</h5>

                    <div class="card-body">
```

```
                    <h5 class="card-title">Hi
<small>{{session.username}}</small>! Weclome back to the application!</h5>

                    <p class="card-text">Number of accesses to the
application(All Time): <small>{{session.number_of_access}}</small></p>

                    <p class="card-text">Latest Session:
<small>{{session.last}}</small></p>

                    <a href="{{url_for('help')}}" class="btn
btn-primary">Get to know how to use the application</a>

                  </div>

                </div>

                <h4 style="margin-top: 40px;">

                      Technical Architecture

                      <small class="text-muted">Learn how different
parts of our application interact!</small>

                </h4>

                <img src="{{url_for('static',
filename='technical-architecture.jpg')}}" class="mx-auto d-block"
style="padding-top: 10px;">

            </div>

        </div>

    </div>



</body>

</html>
```

App/templates/login.html

```
<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

    <meta http-equiv="X-UA-Compatible" content="IE=edge">

    <link rel="stylesheet"
href="https://cdn.jsdelivr.net/npm/bootstrap@5.2.2/dist/css/bootstrap.min.
css">

    <title>A Gesture Based Tool for Sterile Browsing of Radiology
Images</title>



    <style>

      .bottom-left {

        margin-top: 5%;

        margin-left: 2%;

      }

    </style>



  </head>



<body style="background-color: #e3e6e4;">

    {% include 'message_flash.html' %}
```

```
    <script
src="https://cdn.jsdelivr.net/npm/bootstrap@5.2.2/dist/js/bootstrap.min.js
"></script>

    <div class="container-fluid" style="width: 80%; margin-top: 30px;">

      <div class="card text-center" >

            <div class="card-header">

              <ul class="nav nav-tabs card-header-tabs">

                <li class="nav-item">

                    <a class="nav-link active"
href="{{url_for('login')}}">Sign In</a>

                </li>

                <li class="nav-item">

                  <a class="nav-link" href="{{url_for('register')}}">Sign
Up</a>

                </li>

              </ul>

            </div>

            <div class="card-body">

              <h5 class="card-title">Sign In to your Account</h5>



              <form class="form-group" method="post" action="">

                <div class="form-group" style="padding: 10px;">

                    <label>Email</label>

                    <input type="text" name="email" class="form-control"
value="{{request.form.email}}">
```

```
                </div>

                <div class="form-group" style="padding: 10px;">

                    <label>Password</label>

                    <input type="password" name="password"
class="form-control" value="{{request.form.password}}">

                </div>

                <p><input type="submit" value="Submit" class="btn
btn-primary" style="margin-top: 20px;"></p>

             </form>

           </div>

         </div>

    </div>

    <a type="button" class="btn btn-primary bottom-left"
href="{{url_for('about')}}">

      <svg xmlns="http://www.w3.org/2000/svg" width="16" height="16"
fill="currentColor" class="bi bi-arrow-return-left" viewBox="0 0 16 16">

        <path fill-rule="evenodd" d="M14.5 1.5a.5.5 0 0 1 .5.5v4.8a2.5 2.5
0 0 1-2.5 2.5H2.707l3.347 3.346a.5.5 0 0 1-.708.708l-4.2-4.2a.5.5 0 0 1
0-.708l4-4a.5.5 0 1 1 .708.708L2.707 8.3H12.5A1.5 1.5 0 0 0 14 6.8V2a.5.5
0 0 1 .5-.5z"/>

      </svg> Go Back</a>

</body>

</html>
```

App/templates/message_flash.html

```
{% with messages = get_flashed_messages (with_categories=true) %}

    {% if messages %}

            {% for category, message in messages %}

                <div class="alert alert-{{ category }}">{{ message
}}</div>

        {% endfor %}

    {% endif %}

{% endwith %}



{% if error %}

  <div class="alert alert-danger">{{error}}</div>

{% endif %}



{% if msg %}

  <div class="alert alert-success">{{msg}}</div>

{% endif %}
```

App/templates/prediction.html

```
<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">
```

```
    <meta http-equiv="X-UA-Compatible" content="IE=edge">

    <link rel="stylesheet"
href="https://cdn.jsdelivr.net/npm/bootstrap@5.2.2/dist/css/bootstrap.min.
css">

    <title>A Gesture Based Tool for Sterile Browsing of Radiology
Images</title>

</head>

<body style="overflow-x: hidden; background-color: #e3e6e4;">

    <script
src="https://cdn.jsdelivr.net/npm/bootstrap@5.2.2/dist/js/bootstrap.min.js
"></script>

    {% include "message_flash.html" %}




    <nav class="navbar navbar-expand-lg navbar-dark bg-dark"
style="padding: 10px">

        <a class="navbar-brand">{{session.username}}'s Account</a>

        <button class="navbar-toggler" type="button"
data-toggle="collapse" data-target="#navbarText"
aria-controls="navbarText" aria-expanded="false" aria-label="Toggle
navigation">

          <span class="navbar-toggler-icon"></span>

        </button>

        <div class="collapse navbar-collapse" id="navbarText">

          <ul class="navbar-nav mr-auto">

            <li class="nav-item">
```

```
                <a class="nav-link"
href="{{url_for('landingpage')}}">Home</a>

              </li>

            <li class="nav-item">

              <a class="nav-link" href="{{url_for('help')}}">Help</a>

            </li>

          </ul>

        </div>

    </nav>



    <div class="card mx-auto" style="width: 80%; margin-top: 10%;">

        <form class="form-group" action="predict" method="post"
enctype="multipart/form-data">

                <ul class="list-group">

                <li class="list-group-item">

                    <b>Step I: Access Webcam</b>

                    <p>Click the checkbox below to enable the application
to access your camera.

                        <div class="form-check">

                            <input class="form-check-input"
type="checkbox" value="checked" name="video" id="video">Allow this
application to access my camera

                        </div>

                    </p>

                </li>
```

```
                <li class="list-group-item">

                    <b>Step II: Upload Image</b>

                    <p>Upload an image from your computer to the
application by clicking on the `Choose File` buttom below.</p>

                    <input type="file" class="form-control" id="userfile"
name="userfile" style="margin-top: 10px; margin-bottom: 10px; width:
30%;"/>

                </li>

                <li class="list-group-item mx-auto"><p><input
type="submit" value="Predict!" class="btn btn-primary" style="margin-top:
20px;"></p></li>

                </ul>

        </form>

    </div>

</body>

</html>
```

App/templates/registration_form_helper.html

```
{% macro render_field(field) %}
    {{ field.label }}
    {{ field(**kwargs)|safe }}
    {% if field.errors %}

        {% for error in field.errors %}
            <span class="help-inline" style="color:red">{{ error }}</span>
        {% endfor %}
    {% endif %}
{% endmacro %}
```

App/templates/registration.html

```
<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <link rel="stylesheet"
href="https://cdn.jsdelivr.net/npm/bootstrap@5.2.2/dist/css/bootstrap.min.
css">

    <title>A Gesture Based Tool for Sterile Browsing of Radiology
Images</title>

    <style>

        .bottom-left {

          margin-top: 2%;

          margin-left: 2%;

        }

      </style>

</head>



<body style="background-color: #e3e6e4">

    <script
src="https://cdn.jsdelivr.net/npm/bootstrap@5.2.2/dist/js/bootstrap.min.js
"></script>

    {% from "registration_form_helper.html" import render_field %}

    {% include "message_flash.html" %}
```

```
    <div class="container" style="width: 80%; margin-top: 30px;" >

        <div class="card">

            <div class="card-header text-center" style="font-weight:
bold;">

                Registration Page

            </div>

            <form method="post" action="">

                <div class="form-group" style="padding: 10px;">

                    {{render_field(form.name, class="form-control")}}

                </div>



                <div class="form-group" style="padding: 10px;">

                    {{render_field(form.email, class="form-control")}}

                </div>



                <div class="form-group" style="padding: 10px;">

                    {{render_field(form.username, class="form-control")}}

                </div>



                <div class="form-group" style="padding: 10px;">

                    {{render_field(form.password, class="form-control")}}

                </div>
```

```
                <div class="form-group" style="padding: 10px;">

                    {{render_field(form.confirm, class="form-control")}}

                </div>

                <div class="card-footer text-center">

                    <p><input type="submit" value="Submit" class="btn
btn-primary" style="margin-top: 20px;"></p>

                </div>

            </form>

        </div>

    </div>



    <a type="button" class="btn btn-primary bottom-left"
href="{{url_for('about')}}">

        <svg xmlns="http://www.w3.org/2000/svg" width="16" height="16"
fill="currentColor" class="bi bi-arrow-return-left" viewBox="0 0 16 16">

          <path fill-rule="evenodd" d="M14.5 1.5a.5.5 0 0 1 .5.5v4.8a2.5
2.5 0 0 1-2.5 2.5H2.707l3.347 3.346a.5.5 0 0 1-.708.708l-4.2-4.2a.5.5 0 0
1 0-.708l4-4a.5.5 0 1 1 .708.708L2.707 8.3H12.5A1.5 1.5 0 0 0 14
6.8V2a.5.5 0 0 1 .5-.5z"/>

        </svg> Go Back</a>



</body>

</html>
```

App/app.py

```
from flask import Flask, render_template, url_for, flash, redirect,
session, logging, request

from wtforms import Form, StringField, TextAreaField, PasswordField,
validators

from flask_mail import Mail, Message

import os

import cv2

import operator

from time import sleep

import numpy as np

from tensorflow.keras.models import load_model

from werkzeug.utils import secure_filename

from datetime import datetime

import database



app = Flask(__name__)

mail = Mail(app)

model = load_model('../Model/gesture.h5')

print("Model Loaded!")



app.config['MAIL_SERVER'] = 'smtp.gmail.com'

app.config['MAIL_PORT'] = 465

app.config['MAIL_USERNAME'] = 'XXX'
```

```
app.config['MAIL_PASSWORD'] = 'XXX'

app.config['MAIL_USE_TLS'] = False

app.config['MAIL_USE_SSL'] = True

mail = Mail(app)

class RegisterForm(Form):

    name = StringField('Name', [validators.Length(min=1, max=50)])

    username = StringField('Username', [validators.Length(min=4, max=25)])

    email = StringField('Email', [validators.Length(min=6, max=50)])

    password = PasswordField('Password', [

        validators.DataRequired(),

        validators.EqualTo('confirm', message='Passwords do not match')

    ])

    confirm = PasswordField('Confirm Password')



class ChangePasswordForm(Form):

    new_password = PasswordField('Password', [

        validators.DataRequired(),

        validators.EqualTo('new_confirm', message='Passwords do not
match')

    ])

    new_confirm = PasswordField('Confirm Password')
```

```python
@app.route('/')

@app.route('/about')

def about():

    return render_template('about.html')



@app.route('/landingpage')

def landingpage():

    return render_template('landingpage.html')



@app.route('/help')

def help():

    return render_template('help.html')



@app.route('/changepass', methods=["GET", "POST"])

def changepass():



    form = ChangePasswordForm(request.form)

    if request.method == "POST" and form.validate():

        form_password = form.new_password.data

        res = database.update_password(session['email'], form_password)

        flash('Your password has been changed successfully!', 'success')



    return render_template('change_password.html', form=form)
```

```
@app.route('/predict', methods=["GET", "POST"])
def predict():

    if request.method == "POST":

        if request.form.get("video") is None:
            error = "Please enable camera access!"
            return render_template('prediction.html', error=error)

        if  request.files["userfile"].filename == "":
            error = "Please upload an image!"
            return render_template('prediction.html', error=error)


        file = request.files['userfile']
        basepath = os.path.dirname(__file__)
        file_path =
os.path.join(basepath,'uploads',secure_filename(file.filename))
        file.save(file_path)
        print('Image saved successfully!')
        image1 = cv2.imread(file_path)
```

```
        if request.form.get("video") is not None:

            cap = cv2.VideoCapture(0)

            while True:

                _, frame = cap.read()

                frame = cv2.flip(frame, 1)



                x1 = int(0.5*frame.shape[1])

                y1 = 10

                x2 = frame.shape[1]-10

                y2 = int(0.5*frame.shape[1])



                cv2.rectangle(frame, (x1-1, y1-1), (x2+1, y2+1), (255,0,0)
,1)

                roi = frame[y1:y2, x1:x2]



                roi = cv2.resize(roi, (64, 64))

                roi = cv2.cvtColor(roi, cv2.COLOR_BGR2GRAY)

                _, test_image = cv2.threshold(roi, 120, 255,
cv2.THRESH_BINARY)

                cv2.imshow("test", test_image)



                result = model.predict(test_image.reshape(1, 64, 64, 1))
```

```
                prediction = {'Zero': result[0][0],
                              'One': result[0][1],
                              'Two': result[0][2],
                              'Three': result[0][3],
                              'Four': result[0][4],
                              'Five': result[0][5]}


                prediction = sorted(prediction.items(),
key=operator.itemgetter(1), reverse=True)


                cv2.putText(frame, prediction[0][0], (10, 120),
cv2.FONT_HERSHEY_PLAIN, 1, (0,255,255), 1)
                cv2.imshow("Frame", frame)


                image1=cv2.imread(file_path)


                if prediction[0][0]=='Zero':


                    cv2.rectangle(image1, (480, 170), (650, 420), (0, 0,
255), 2)
                    cv2.imshow("Rectangle", image1)
                    key=cv2.waitKey(3000)
                    if (key & 0xFF) == ord("0"):
```

```
                        cv2.destroyWindow("Gesture 0 - Fixed Rectangle
Drawing")



                elif prediction[0][0]=='One':



                    resized = cv2.resize(image1, (200, 200))

                    cv2.imshow("Gesture 1 - Fixed Resizing I(200,200)",
resized)

                    key=cv2.waitKey(3000)

                    if (key & 0xFF) == ord("1"):

                        cv2.destroyWindow("Gesture 1 - Fixed Resizing
I(200,200)")




                elif prediction[0][0]=='Two':

                    (h, w, d) = image1.shape

                    center = (w // 2, h // 2)

                    M = cv2.getRotationMatrix2D(center, -45, 1.0)

                    rotated = cv2.warpAffine(image1, M, (w, h))

                    cv2.imshow("Gesture 2 - OpenCV Rotation", rotated)

                    key=cv2.waitKey(3000)

                    if (key & 0xFF) == ord("2"):

                        cv2.destroyWindow("Gesture 2 - OpenCV Rotation")
```

```
                elif prediction[0][0]=='Three':
                    blurred = cv2.GaussianBlur(image1, (21, 21), 0)
                    cv2.imshow("Gesture 3 - Blurring", blurred)
                    key=cv2.waitKey(3000)
                    if (key & 0xFF) == ord("3"):
                        cv2.destroyWindow("Gesture 3 - Blurring")

                elif prediction[0][0]=='Four':

                    resized = cv2.resize(image1, (400, 400))
                    cv2.imshow("Gesture 4 - Fixed Resizing II(400,400)",
resized)
                    key=cv2.waitKey(3000)
                    if (key & 0xFF) == ord("4"):
                        cv2.destroyWindow("Gesture 4 - Fixed Resizing
II(400,400)")

                elif prediction[0][0]=='Five':
                    gray = cv2.cvtColor(image1, cv2.COLOR_RGB2GRAY)
                    cv2.imshow("Gesture 5 - Grayscaling", gray)
                    key=cv2.waitKey(3000)
                    if (key & 0xFF) == ord("5"):
                        cv2.destroyWindow("Gesture 5 - Grayscaling")
```

```
                interrupt = cv2.waitKey(10)
                if interrupt & 0xFF == 27: # esc key
                    break

            cap.release()
            cv2.destroyAllWindows()

    return render_template('prediction.html')



@app.route("/register", methods=["GET", "POST"])
def register():
    form = RegisterForm(request.form)
    if request.method == "POST" and form.validate():
        form_name = form.name.data
        form_email = form.email.data
        form_username = form.username.data
        form_password = form.password.data

        email_check = database.check_email_exists(form_email)
        if email_check == True:
```

```
            error = 'The email you entered already exists in the
database!'

            return render_template('registration.html', error=error,
form=form)



        res = database.users_insert(form_name, form_username, form_email,
form_password)



        flash('You are now registered and can login!', 'success')

        msg = Message('Registration Verfication',
sender="anirudh19015@cse.ssn.edu.in", recipients=[form_email])

        msg.body = (f'Congratulations! Welcome user! Your Credentials are
Username: {form_username} Password: {form_password}')

        mail.send(msg)

        print("Email Sent!")



    return render_template('registration.html', form=form)



@app.route("/login", methods=["GET", "POST"])

def login():

    if request.method == "POST":

        email = request.form["email"]

        candidate_password = request.form["password"]

        #print("Email is", email)

        result_set = database.users_fetch_by_email(email)
```

```
        if type(result_set) != type(False):
            if result_set['password'] != candidate_password:
                error = 'Invalid login!'
                return render_template('login.html', error=error)

            database.usession_insert(email)
            user_sessions = database.usession_fetch_by_email(email)
            print(user_sessions)

            session['logged_in'] = True
            session['email'] = email
            session['name'] = result_set['name']
            session['username'] = result_set['username']
            session['number_of_access'] = len(user_sessions)
            session['last'] =  user_sessions[-1]

            return redirect(url_for('landingpage'))


        else:
            error = 'User not found! Please enter the correct username
and/or password.'
```

```
            return render_template('login.html', error=error)


    return render_template('login.html')


@app.route('/signout')
def signout():
    session.clear()
    flash('You are now logged out', 'success')
    return redirect(url_for('login'))


if __name__ == "__main__":
    app.secret_key="secret123"
    app.run(debug=True)
```

App/database.py

```
from datetime import date, datetime

import ibm_db



dsn_hostname =
"2f3279a5-73d1-4859-88f0-a6c3e6b4b907.c3n41cmd0nqnrk39u98g.databases.appdo
main.cloud"

dsn_uid = "jlr06722"

dsn_password = "VlN0yoUMB4fzESl4"
```

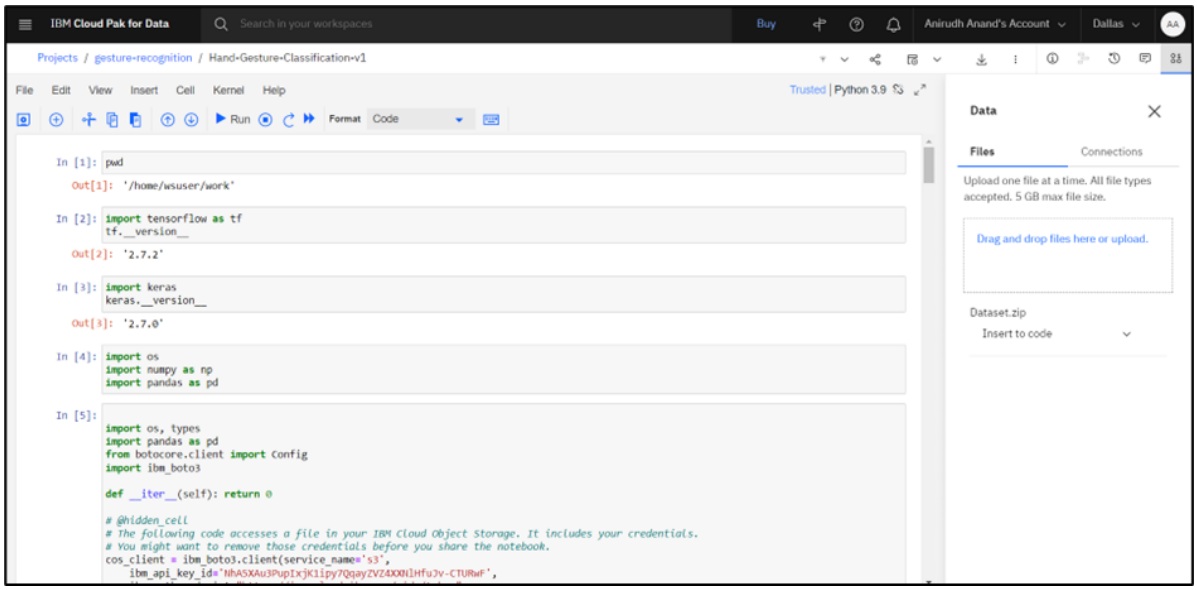
```
dsn_driver = "{IBM DB2 ODBC DRIVER}"

dsn_database = "BLUDB"

dsn_port = "30756"

dsn_protocol = "TCPIP"



dsn = (

    "DRIVER={0};"

    "DATABASE={1};"

    "HOSTNAME={2};"

    "PORT={3};"

    "PROTOCOL={4};"

    "UID={5};"

    "PWD={6};"

    "SECURITY=SSL"

).format(dsn_driver, dsn_database, dsn_hostname, dsn_port, dsn_protocol,
dsn_uid, dsn_password)



def connect_db():

    conn = ibm_db.connect(dsn, "", "")

    return conn



def users_insert(name, username, email, password):

    conn = connect_db()
```

```
    sql = "INSERT INTO USERS(\"name\", \"username\", \"email\",
\"password\") VALUES(?, ?, ?, ?)"

    stmt = ibm_db.prepare(conn, sql)

    params = (name, username, email, password)

    res = ibm_db.execute(stmt, params)

    return res



def users_fetch_by_email(email):

    conn = connect_db()

    sql = "SELECT \"name\", \"password\", \"username\" FROM USERS WHERE
\"email\" = ?"

    stmt = ibm_db.prepare(conn, sql)

    ibm_db.bind_param(stmt, 1, email)

    ibm_db.execute(stmt)

    result_set =ibm_db.fetch_assoc(stmt)

    return result_set



def usession_insert(email):

    conn = connect_db()



    today = datetime.now()

    sessions = today.strftime("%d/%m/%Y %H:%M:%S")
```

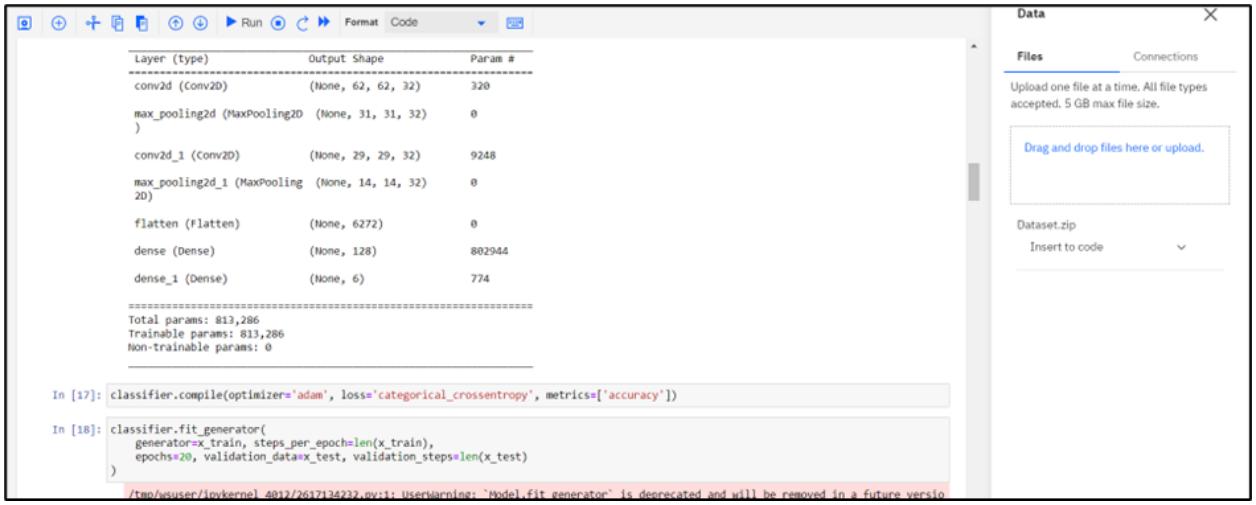
```
    sql = "INSERT INTO USESSION(\"email\", \"sessions\") VALUES(?, ?)"
    stmt = ibm_db.prepare(conn, sql)
    ibm_db.bind_param(stmt, 1, email)
    ibm_db.bind_param(stmt, 2, sessions)
    res = ibm_db.execute(stmt)
    return res


def usession_fetch_by_email(email):
    conn = connect_db()
    sql = "SELECT \"sessions\" FROM USESSION WHERE \"email\" = ?"
    stmt = ibm_db.prepare(conn, sql)
    ibm_db.bind_param(stmt, 1, email)
    ibm_db.execute(stmt)
    times = []
    while ibm_db.fetch_row(stmt) != False:
        times.append(ibm_db.result(stmt, 'sessions'))
    return times


def update_password(email, password):
    conn = connect_db()
    sql = "UPDATE USERS SET \"password\" = ? where \"email\" = ?"
    stmt = ibm_db.prepare(conn, sql)
    ibm_db.bind_param(stmt, 1, password)
```

```
    ibm_db.bind_param(stmt, 2, email)

    res = ibm_db.execute(stmt)

    return res



def check_email_exists(email):

    conn = connect_db()

    sql = "SELECT * FROM USERS where \"email\" = ?"

    stmt = ibm_db.prepare(conn, sql)

    ibm_db.bind_param(stmt, 1, email)

    ibm_db.execute(stmt)

    if ibm_db.fetch_row(stmt) == False:

        return False

    else:

        return True
```

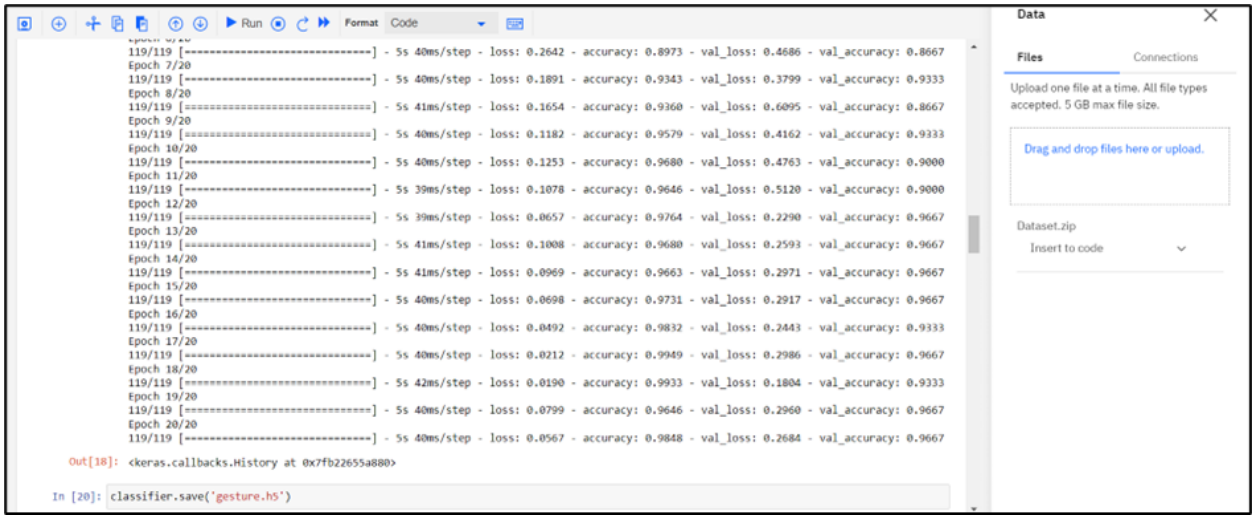
Model/Hand-Gesture-Classification.ipynb:



```
In [10]: train_datagen = ImageDataGenerator(rescale=1./255,shear_range=0.2,zoom_range=0.2,horizontal_flip=True)
         test_datagen = ImageDataGenerator(rescale=1./255)

In [11]: x_train = train_datagen.flow_from_directory('/home/wsuser/work/Dataset/train/', target_size=(64, 64), batch_size=5, color_mode='grayscale'
         x_test = test_datagen.flow_from_directory('/home/wsuser/work/Dataset/test/', target_size=(64, 64), batch_size=5, color_mode='grayscale'

         Found 594 images belonging to 6 classes.
         Found 30 images belonging to 6 classes.

In [12]: x_train.class_indices

Out[12]: {'0': 0, '1': 1, '2': 2, '3': 3, '4': 4, '5': 5}

In [13]: classifier = Sequential()

In [14]: classifier.add(Conv2D(32, (3, 3), input_shape=(64, 64, 1) ,activation='relu'))
         classifier.add(MaxPooling2D(pool_size=(2, 2)))
         classifier.add(Conv2D(32, (3, 3), activation='relu'))
         classifier.add(MaxPooling2D(pool_size=(2, 2)))
         classifier.add(Flatten())

In [15]: classifier.add(Dense(units=128, activation='relu'))
         classifier.add(Dense(units=6, activation='softmax'))

In [16]: classifier.summary()

         Model: "sequential"
```

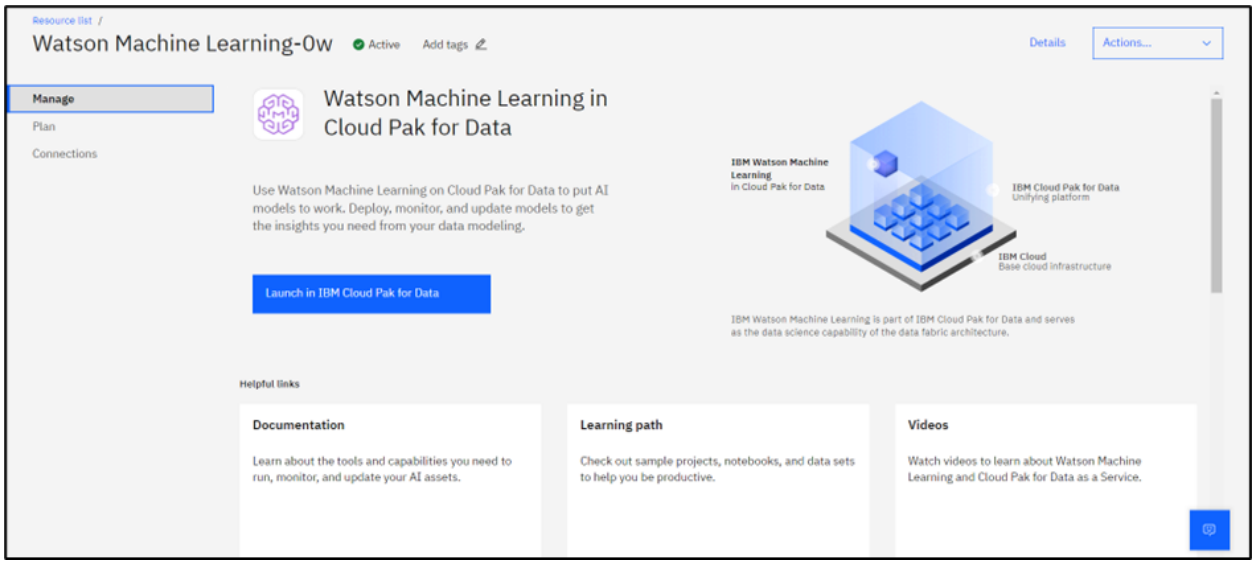


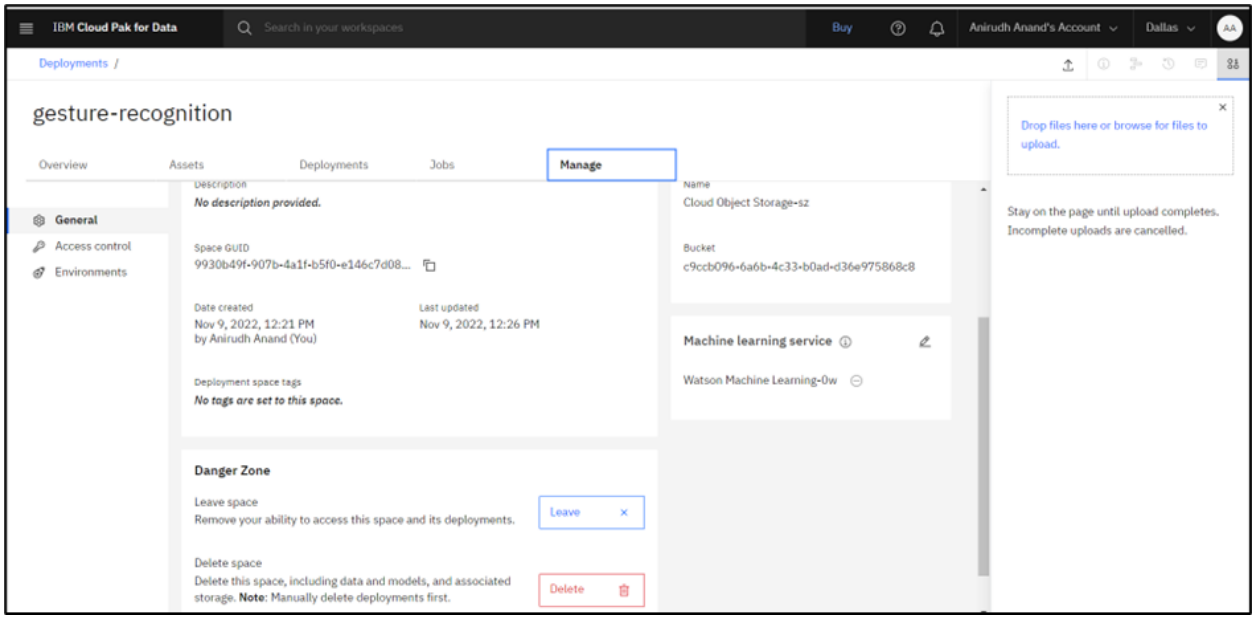
```
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d (Conv2D)             (None, 62, 62, 32)        320

 max_pooling2d (MaxPooling2D  (None, 31, 31, 32)       0
 )

 conv2d_1 (Conv2D)           (None, 29, 29, 32)        9248

 max_pooling2d_1 (MaxPooling  (None, 14, 14, 32)       0
 2D)

 flatten (Flatten)           (None, 6272)              0

 dense (Dense)               (None, 128)               802944

 dense_1 (Dense)             (None, 6)                 774

=================================================================
Total params: 813,286
Trainable params: 813,286
Non-trainable params: 0
_________________________________________________________________

In [17]: classifier.compile(optimizer='adam', loss='categorical_crossentropy', metrics=['accuracy'])

In [18]: classifier.fit_generator(
             generator=x_train, steps_per_epoch=len(x_train),
             epochs=20, validation_data=x_test, validation_steps=len(x_test)
         )

         /tmp/wsuser/ipykernel_4012/2617134232.py:1: UserWarning: `Model.fit_generator` is deprecated and will be removed in a future versio
```



```
119/119 [==============================] - 5s 40ms/step - loss: 0.2642 - accuracy: 0.8973 - val_loss: 0.4686 - val_accuracy: 0.8667
Epoch 7/20
119/119 [==============================] - 5s 40ms/step - loss: 0.1891 - accuracy: 0.9343 - val_loss: 0.3799 - val_accuracy: 0.9333
Epoch 8/20
119/119 [==============================] - 5s 41ms/step - loss: 0.1654 - accuracy: 0.9360 - val_loss: 0.6095 - val_accuracy: 0.8667
Epoch 9/20
119/119 [==============================] - 5s 40ms/step - loss: 0.1182 - accuracy: 0.9579 - val_loss: 0.4162 - val_accuracy: 0.9333
Epoch 10/20
119/119 [==============================] - 5s 40ms/step - loss: 0.1253 - accuracy: 0.9680 - val_loss: 0.4763 - val_accuracy: 0.9000
Epoch 11/20
119/119 [==============================] - 5s 39ms/step - loss: 0.1078 - accuracy: 0.9646 - val_loss: 0.5120 - val_accuracy: 0.9000
Epoch 12/20
119/119 [==============================] - 5s 39ms/step - loss: 0.0657 - accuracy: 0.9764 - val_loss: 0.2290 - val_accuracy: 0.9667
Epoch 13/20
119/119 [==============================] - 5s 41ms/step - loss: 0.1008 - accuracy: 0.9680 - val_loss: 0.2593 - val_accuracy: 0.9667
Epoch 14/20
119/119 [==============================] - 5s 41ms/step - loss: 0.0969 - accuracy: 0.9663 - val_loss: 0.2971 - val_accuracy: 0.9667
Epoch 15/20
119/119 [==============================] - 5s 40ms/step - loss: 0.0698 - accuracy: 0.9731 - val_loss: 0.2917 - val_accuracy: 0.9667
Epoch 16/20
119/119 [==============================] - 5s 40ms/step - loss: 0.0492 - accuracy: 0.9832 - val_loss: 0.2443 - val_accuracy: 0.9333
Epoch 17/20
119/119 [==============================] - 5s 40ms/step - loss: 0.0212 - accuracy: 0.9949 - val_loss: 0.2986 - val_accuracy: 0.9667
Epoch 18/20
119/119 [==============================] - 5s 42ms/step - loss: 0.0190 - accuracy: 0.9933 - val_loss: 0.1804 - val_accuracy: 0.9333
Epoch 19/20
119/119 [==============================] - 5s 40ms/step - loss: 0.0799 - accuracy: 0.9646 - val_loss: 0.2960 - val_accuracy: 0.9667
Epoch 20/20
119/119 [==============================] - 5s 40ms/step - loss: 0.0567 - accuracy: 0.9848 - val_loss: 0.2684 - val_accuracy: 0.9667

Out[18]: <keras.callbacks.History at 0x7fb22655a880>

In [20]: classifier.save('gesture.h5')
```

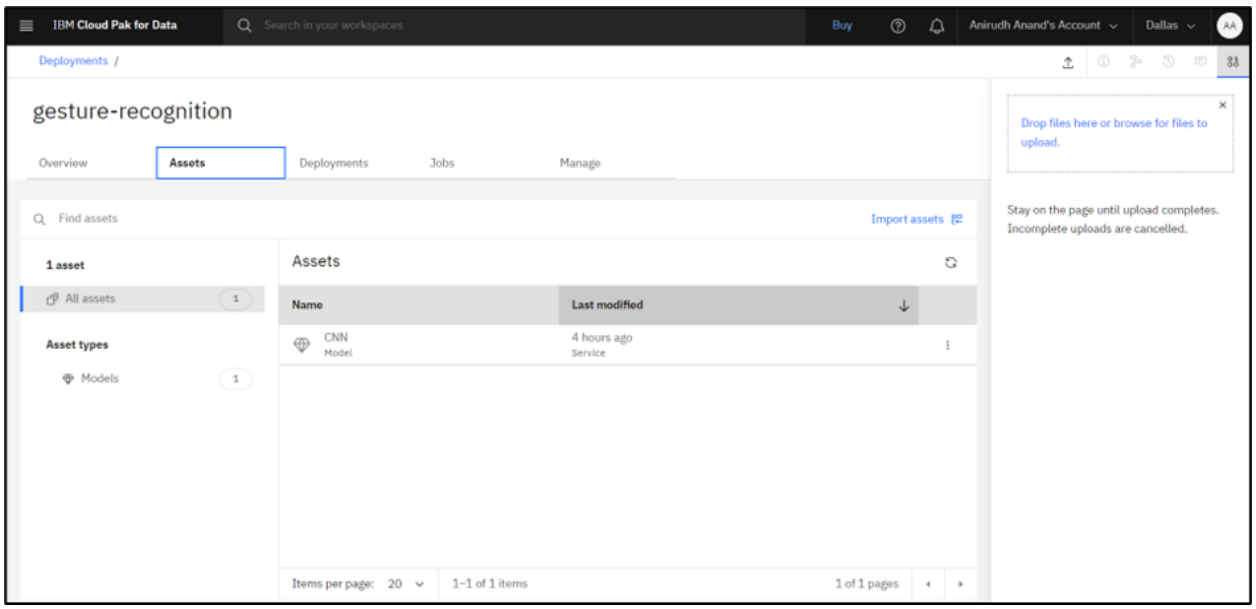
Model Training on the Cloud



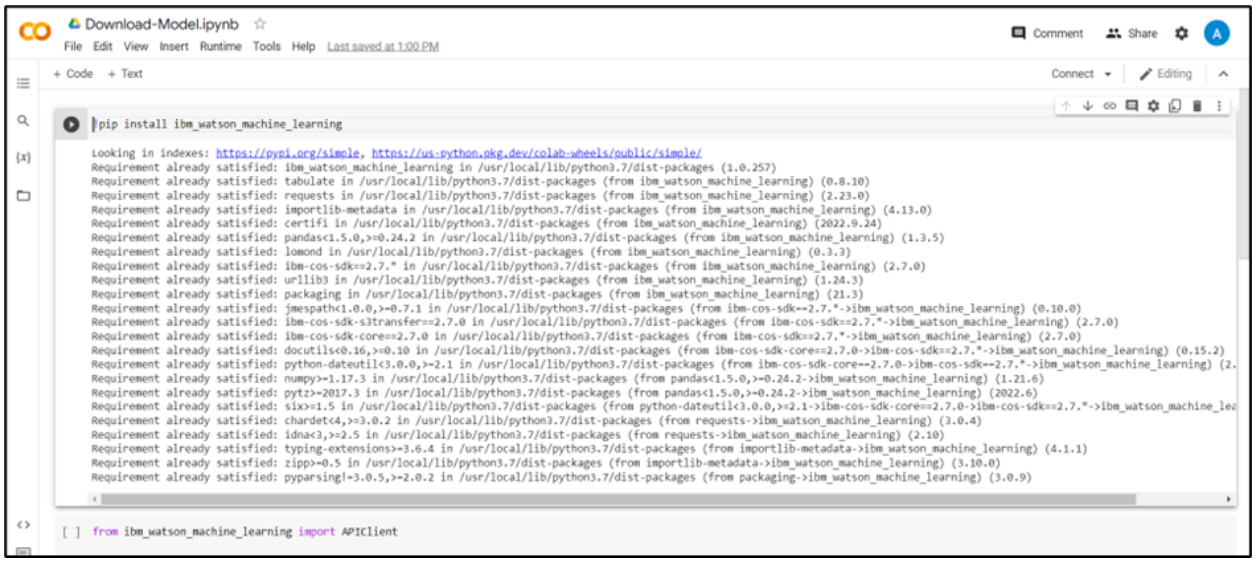



Run
Format
Code
In [25]:
from ibm_watson_machine_learning import APIClient
wml_credentials={
"url":"https://us-south.ml.cloud.ibm.com",
"apikey":"xhYfr2sw7BoEM8unaHSrXivomwlh4Cqyk0F7MzxcloSa"
}
In [26]:
client = APIClient(wml_credentials)
In [27]:
client
Out[27]:
<ibm_watson_machine_learning.client.APIClient at 0x7fb1eec10760>
In [28]:
def guid_space_name(client,Gesture):
space=client.spaces.get_details()
return(next(item for item in space['resources'] if item['entity']['name']==Gesture)['metadata']['id'])
In [29]:
space_uid=guid_space_name(client,'gesture-recognition')
print("Space UID = " + space_uid)
Space UID = 9930b49f-907b-4a1f-b5f0-e146c7d081b1
In [30]:
client.set.default_space(space_uid)
Out[30]:
'SUCCESS'
In [31]:
client.software_specifications.list(100)
Data
Files
Connections
Upload one file at a time. All file types accepted. 5 GB max file size.
Drag and drop files here or upload.
Dataset.zip
Insert to code


Run
Format
Code
In [32]:
software_spec_uid = client.software_specifications.get_uid_by_name('tensorflow_rt22.1-py3.9')
software_spec_uid
Out[32]:
'acd9c798-6974-5d2f-a657-ce06e986df4d'
In [34]:
model_details = client.repository.store_model(model='gesture-classifier.tgz',meta_props={
client.repository.ModelMetaNames.NAME:"CNN",
client.repository.ModelMetaNames.TYPE:'tensorflow_rt22.1',
client.repository.ModelMetaNames.SOFTWARE_SPEC_UID:software_spec_uid
})
model_id = client.repository.get_model_uid(model_details)
This method is deprecated, please use get_model_id()
/opt/conda/envs/Python-3.9/lib/python3.9/site-packages/ibm_watson_machine_learning/repository.py:1453: UserWarning: This method is deprecated, please use get_model_id()
warn("This method is deprecated, please use get_model_id()")
In [35]:
model_id
Out[35]:
'b551a542-6d4f-43d1-a1df-815fe17e7e8c'
Data
Files
Connections
Upload one file at a time. All file types accepted. 5 GB max file size.
Drag and drop files here or upload.
Dataset.zip
Insert to code


IBM Cloud Pak for Data
Search in your workspaces
Buy
Anirudh Anand's Account
Dallas
Deployments /
gesture-recognition
Overview
Assets
Deployments
Jobs
Manage
Find assets
Import assets
1 asset
All assets 1
Asset types
Models 1
Assets
Name
Last modified
CNN
Model
4 hours ago
Service
Items per page: 20
1–1 of 1 items
1 of 1 pages
Drop files here or browse for files to upload.
Stay on the page until upload completes. Incomplete uploads are cancelled.

Downloading Model Weights



GitHub & Project Demo Link

- Project Repository (GitHub):

  https://github.com/IBM-EPBL/IBM-Project-21974-1659800381

- Project Demo Link (Google Drive):

  https://drive.google.com/file/d/1ZVg7bqdMY6afFfRHukROtoy74wmqAJZp/view