VISVESVARAYA TECHNOLOGICAL UNIVERSITY

"Jnana Sangama", Belgaum - 590018

A project report on

"Social Media Censoring using Visual Sentiment Analyzer"

submitted in partial fulfillment for the award of the degree of

BACHELOR OF ENGINEERING

in

INFORMATION SCIENCE & ENGINEERING

by

1CR19IS091	Naga Vaishak S K
1CR19IS121	Ronald Ryan G
1CR19IS110	Puneeth S

Under the Guidance of

Dr. Swathi Y
Associate Professor
Department of ISE, CMRIT, Bengaluru

CMR INSTITUTE OF TECHNOLOGY

DEPARTMENT OF INFORMATION SCIENCE & ENGINEERING

#132, AECS Layout, IT Park Road, Bengaluru - 560037

May 2023

CMR INSTITUTE OF TECHNOLOGY

DEPARTMENT OF INFORMATION SCIENCE & ENGINEERING

#132, AECS Layout, IT Park Road, Bengaluru - 560037

Certificate

This is to Certified that the project work entitled **"Social Media Censoring using Visual Sentiment Analyzer" carried out by Naga Vaishak S K (1CR19IS185), Ronald Ryan G (1CR19IS121), and Puneeth S (1CR19IS110)** in partial fulfillment for the award of Bachelor of Engineering in **Information Science & Engineering** of the Visveswaraiah Technological University, Belgaum during the year **2022-23**. It is certified that all corrections/suggestions indicated for Internal Assessment have been incorporated in the Report deposited in the departmental library.
The project report has been approved as it satisfies the academic requirements in respect of Project work prescribed for the said Degree.

------------------------	------------------------	------------------------
Signature of Guide	Signature of HoD	Signature of Principal
Dr. Swathi Y	**Dr. Farida Begam**	**Dr. Sanjay Jain**
Associate Professor	Professor & Head	Principal
Department of ISE	Department of ISE	CMRIT,
CMRIT	CMRIT	Bengaluru - 37

External Viva

	Name of the Examiners	Institution	Signature with Date
1.	------------------------	------------------------	------------------------
2.	------------------------	------------------------	------------------------

CMR INSTITUTE OF TECHNOLOGY

DEPARTMENT OF INFORMATION SCIENCE & ENGINEERING

#132, AECS Layout, IT Park Road, Bengaluru - 560037

Declaration

We, Naga Vaishak S K (1CR19IS091), Ronald Ryan G (1CR19IS121), and **Puneeth S (1CR19IS110)** bonafide students of **CMR Institute of Technology**, Bangalore, hereby declare that the dissertation entitled, **"Social Media Censoring using Visual Sentiment Analyzer"** has been carried out by us under the guidance of **Dr. Swathi Y**, Associate Professor, CMRIT, Bangalore, in partial fulfillment of the requirements for the award of the degree of Bachelor of Engineering in Computer Science & Engineering, of the Visvesvaraya Technological University, Belgaum during the academic year 2022-23. The work done in this dissertation report is original and it has not been submitted for any other degree in any university.

Naga Vaishak S K
Ronald Ryan G
Puneeth S

Acknowledgement

The satisfaction and euphoria that accompany a successful completion of any task would be incomplete without the mention of people who made it possible. Success is the epitome of hard work and perseverance, but steadfast of all is encouraging guidance.

So, it is with gratitude that we acknowledge all those whose guidance and encouragement served as beacon of light and crowned our effort with success.

We would like to thank Dr. Sanjay Jain, Principal, CMRIT, Bangalore, for providing an excellent academic environment in the college and his never-ending support for the B.E program.

We would like to express our gratitude towards Dr. Farida Begam, Professor and HOD, Department of Information Science & Engineering CMRIT, Bangalore, who provided guidance and gave valuable suggestions regarding the project.

We consider it a privilege and honour to express our sincere gratitude to our internal guide Dr. Swathi Y, Associate Professor, Department of Information Science & Engineering, CMRIT, Bangalore, for their valuable guidance throughout the tenure of this project work.

We would like to thank all the faculty members who have always been very cooperative and generous. Conclusively, we also thank all the non-teaching staff and all others who have done immense help directly or indirectly during our project.

Naga Vaishak S K
Ronald Ryan G
Puneeth S

Abstract

Social Media Censoring using Visual Sentiment Analyzer is a project which aims at reducing the time taken by social media platforms to effectively classify, flag and remove a post which might seem particularly vulgar or offensive. This is done so by classifying an image in real-time, while the user is getting ready to upload an image itself. Hence, the system can be considered as truly dynamic. This dynamic nature of this real-time classification is possible due to an immensely powerful transfer learning model, InceptionV3. Other than that aspect, the app would be a simple social media app, similar to Instagram, Facebook or any other leading social media app. Users would be able to upload pictures along with captions. Users will be given the ability to browse through the images uploaded by those who they follow, like the pictures as well as comment on them. They would also be able to search for specific profiles. And many other such features from popular social media apps would be present in the app as well. The goal of the project is to identify and improve on the incompetence of existing image-classification systems by applying a novel model, and building an app considering the model as a base.

Keywords: *Visual Sentiment Analyzer, Transfer learning, InceptionV3, Image-classification*

Table of Contents

List of Figures

Chapter 1

Preamble

1.1 Introduction

It is becoming increasingly important in a number of applications to analyze the attitudes of individuals from social media information through text, speech, and photos. The idea of human-computer interaction is crucial nowadays and has a wide range of uses. Textual data is a key component of many of the existing research studies on sentiment analysis, and social media users contribute more photos and videos than they do text as well.

Making decisions and predicting certain things, like sentiments, entails the use of machines and computers. It applies sentiment analysis to photos using some artificial intelligence algorithms. Users upload a lot of information and images on Facebook and YouTube, which are both quite popular. Additionally, we have discovered that using social networking platforms like Instagram, Pinterest, and WhatsApp is very common among adults younger than 30 years of age.

Images are thought to convey feelings considerably more effectively than writing. Therefore, there is a desire to create a sentiment analysis model using social media image data. Image categorization utilizing deep learning techniques like CNNs or convolutional neural networks can be described as visual sentiment analysis.

The goal of visual sentiment analysis is to determine if an image evokes a pleasant or negative feeling. Most cutting-edge works utilize the content attached to a social media post that the user provides in order to achieve this goal. Due to the subjectivity of the user, such textual data is typically noisy since it frequently contains information that will help the social post spread as widely as possible.

1.2 Existing System

1. Most of the papers we referred to, primarily used only VGG-19, DenseNet-121 and ResNet50V2 for the image classification.
2. The existing censoring functionality on popular social media platforms works on the basis of a user first reporting the concerned post.
3. Very few applications run on both mobile and desktop
4. Lack of training data in the relevant field can be efficiently addressed through transfer learning, which entails adapting information or patterns learnt in one field or activity to different but related disciplines or challenges.

1.2.1 Drawbacks

1. The network's organizational structure must be changed to lessen overfitting. More and better features are needed in some cases in order to improve the results.
2. To further improve performance, additional social media sources like geo-location, user history, etc. must be looked at.

1.3 Proposed System

1. The goal of this project is to create a system that will effectively classify the images being uploaded at real-time and filter it out.
2. The application will be made on Flutter which is a leading open-source software which gives the application capability to run on Android, IOS and also Windows.
3. It uses an Artificial Intelligence model that will aid in the classification and subsequent screening of the inappropriate content.
4. The application uses Firebase API's for the back-end functionality in order to aid seamless usage of the application irrespective of the platforms and devices.

1.4 Plan of Implementation

Our work can be broken down into five major stages.

1. In Stage I, the image samples are to be extracted from multiple sources and made into a dataset, along with manual-labelling for every image.

2. In Stage II, the pre-processing of images will be done.
3. Stage III involves training of the model with the image samples from the dataset
4. Stage IV involves building a fully-functional clone of Instagram from scratch, using Flutter. The backend functionality will be based on Firebase.
5. The final stage is the integration of the model into the app

1.5 Problem Statement

Social media platforms are becoming an increasingly dangerous place for the average individual nowadays. Currently, there is a lot of obscene content on several social media sites that has the potential to affect certain people's mental health.

Regularly consuming such material could traumatize the individual and eventually cause them to become disoriented.

1.6 Objective of the Project

With this project, we aim to effectively filter the content present in social media platforms, making it a safe-space for all those who have access to it. Classifying and filtering vulgar, gore and offensive content in real-time helps in regulating the content that the users consume, and helps in protecting the users from any form of mental harm.

We want to contribute to enhancing the overall security and safety of online communities.

Chapter 2

Literature Survey

Image sentiment analysis is done by using any one of the following approaches involving the extraction of low-level features, semantic features, and with machine and deep learning models. The pixel-level features, along with some low-level features, such as color and texture are used.

Latent semantic analysis (LSA) and the Bag-of-Visual-Words (BoVW) technique were combined to classify the image emotions. They tackled the issue of accurately capturing the various emotion kinds that might be found throughout an image. Some extracted aesthetic features from the images that are related to beauty or art to perform visual sentiment analysis and achieved significant results compared to low-level features.

Some researchers of utilised the idea of deep coupled adjective and noun neural networks in their work to get around some of the difficulties in assessing the sentiment of images. By merging the convolutional neural network (CNN) with independent adjective and noun networks, they were able to achieve improved results. Convolutional neural networks (CNNs) were utilised by some to extract sentiment scores from some local regions and complete images that contain sentiment information.

The drawbacks found in existing works on visual sentiment analysis can be eliminated by employing fine-tuned transfer learning models that are trained on a large set of data. The transfer learning models are suitable for sentiment analysis and other natural language processing tasks and provide many advantages compared to other conventional models.

We understood that all of them primarily used three different transfer learning models—the VGG-19, DenseNet-121, and ResNet50V2 models—for visual sentiment prediction from the Crowdflower dataset. But we decided to go with a fairly unknown and unexplored transfer learning model known as InceptionV3. InceptionV3 seemed to be best-suited for our application as well as our restrictions.

The deep learning method's performance is dependent on huge data support. When obtaining a high number of labeled data samples is challenging, the efficacy of the learning algorithm suffers. When there is not enough data, the developed deep learning network structure is susceptible to overfitting.

Transfer learning, which involves applying knowledge or patterns learned in one field or activity to distinct but related disciplines or challenges, can effectively tackle the problem of limited training data in the appropriate field.

Chapter 3

Theoretical Background

Theoretical background highlighting some topics related to the project work is given below. The description contains several topics which are worth to discuss and also highlight some of their limitation that encourage going on finding solution as well as highlights some of their advantages for which reason these topics and their features are used in this project.

3.1 Inception V3

For our project, we have used the Inception V3 pre-trained image-recognition model. On the ImageNet dataset, it has been demonstrated that the image recognition model can achieve higher than 78.1 % accuracy. The model is the result of numerous concepts that have been established by various researchers over the years.

Convolutions, average pooling, max pooling, concatenations, dropouts, and fully linked layers are some of the symmetric and asymmetric building components that make up the model itself. The model makes considerable use of batch normalization, which is also applied to the activation inputs. To calculate loss, Softmax is used.

3.2 Flutter

Google developed the open-source Flutter framework for building mobile applications. With a single codebase, it enables developers to produce high-performance, aesthetically pleasing mobile applications for both the Android and iOS platforms. The programming language Dart, which was also created by Google, is used by Flutter. Flutter's ability to build extremely customised user interfaces using a wide variety of pre-built widgets is one of its key advantages. The flexibility of Flutter widgets makes it simple to alter how an app looks and feels. Developers may now design distinctive interfaces with eye-catching visuals to match the aesthetic of their brand or application.

3.3 Tensorflow

Google created the open-source machine learning framework known as TensorFlow. It covers a wide range of applications, including image and audio recognition, natural language processing, and predictive analytics, and offers a collection of tools and libraries for creating and training machine learning models, including deep neural networks.

TensorFlow is scalable for large-scale projects because it supports distributed computation across numerous devices. Developers may easily create and train machine learning models using its high-level APIs without having to be familiar with the underlying mathematics.

3.4 Tflite

A compact, open-source machine learning framework created for mobile and embedded devices is called Tflite TensorFlow Lite (TFLite). Developers may run machine learning models locally on mobile and IoT (Internet of Things) devices without relying on a cloud-based server thanks to this TensorFlow subset that is designed for these platforms. In order to lower the memory and storage requirements for running models on mobile devices, TFLite provides tools for model compression and optimisation. These tasks supported by TFLite include image classification, object identification, and natural language processing. Many cutting-edge mobile applications with machine learning capabilities have been made possible by the widespread use of TFLite in the development of mobile apps.

3.5 FireBase APIs

A group of software interfaces are made available by Firebase, a Google-owned platform for building mobile and online applications. These APIs give programmers the ability to create and maintain software that includes functions like real-time data synchronisation, user authentication, cloud storage, and messaging. Firebase offers a number of APIs that let programmers create and manage applications for a variety of use scenarios. For instance, the Cloud Firestore API offers a document-based NoSQL database for scalable storage and synchronisation, while the Realtime Database API enables developers to store and sync data in real-time across different clients.

Chapter 4

System Requirements Specification

A software requirement definition is an abstract description of the services which the system should provide, and the constraints under which the system must operate. It should only specify the external behavior of the system. The requirements are specified as below:

4.1 Functional Requirement

There are several functional requirements that would need to be considered in the implementation of a machine learning system for Visual Sentiment Analyzer for social Media Censoring. These requirements may include:

1. **Data Collection and Preprocessing:** The system should be able to collect image samples from the dataset pre-provided and preprocess the data to remove errors and noise.

2. **Model training:** The system should be able to train a machine learning model on a dataset of image samples to determine whether the image contains sensitive or vulgar content. If it doesn't contain any of those it will permitted to post on social media app. To use the InceptionV3 model for visual sentiment analysis, you would first need to train the model on a dataset of images that have been labeled with sentiments (positive or negative).

3. **Prediction:** You can then use the trained model to predict the sentiment of new images by feeding the images into the model and analyzing the output. The system should be able to input a new image sample as well as text in the images and output the prediction of the image as restricted if it contains any sensitive content else it will be permitted without any restrictions.

4. **Evaluation:** The system should be able to evaluate the performance of the model on the prediction task, using metrics such as accuracy, precision, and recall.

5. **Fine-tuning:** The system should allow for the fine-tuning of the model for a new task such as visual sentiment analysis by updating the model's parameters on a new dataset of images labeled with sentiments. Fine-tune the model by unfreezing some of the lower layers and training them on the new dataset.

6. **Deployment:** The system should be able to deploy the trained model for use in detecting the image as restricted or unrestricted for the social media app.

4.2 Non Functional Requirements

Non-functional requirements are those that have no direct bearing on the particular function that the system performs. Instead of describing particular behaviours, they define the standards that can be used to evaluate how a system works. Non-functional requirements can be caused by external reasons like:

4.2.1 Product Requirements

1. **Real-time Analysis:** Ensure the image classification process is performed quickly and efficiently, allowing the system to analyze and classify images while users are in the process of uploading them.

2. **User Interface:** Create a user-friendly and intuitive interface for the social media app, allowing users to easily upload images, add captions, browse through their feed, like pictures, comment on posts, and search for specific profiles.

3. **Profile Management:** Provide users with the ability to create and manage their profiles, including uploading profile pictures, adding personal information, and customizing privacy settings.

4. **Scalability and Performance:** Design the system to handle a large number of users and their interactions, ensuring that it can scale effectively and provide a smooth user experience even during peak usage times.

4.2.2 Organizational Requirements

1. **Data Acquisition and Annotation:** Establish a process for acquiring relevant image data for training the InceptionV3 model, ensuring the data is diverse,

representative, and properly annotated with sentiment and content labels. Consider ethical considerations and obtain necessary permissions for using the data.

2. **Feedback Mechanism:** Implement a mechanism for users to provide feedback and report issues with the app. Establish a process for collecting and analyzing user feedback, incorporating it into future iterations, and addressing any reported issues promptly.

4.2.3 User Requirements

1. User must sign up into the application by providing their email address, profile picture, username, bio and password.
2. Users must post on the app from time to time.
3. Users must interact with the posts of other users.
4. They must provide access to camera and gallery in order to be able to post.
5. They must be responsible enough to recognize content that is unacceptable on a social network and act accordingly while posting.

4.2.4 System Configuration

H/W System Configuration:
Processor - Core I3
Speed - 2.0 Ghz
RAM - 8GB(min)
Hard Disk - 40 GB
Keyboard - Standard Windows Keyboard
Mouse - Two or Three Button Mouse
Monitor - SVGA

S/W System Configuration:
Operating System : Windows 8/8.1/10
Coding Language : Python, Dart, JavaScript
Tools : Flutter, VSCode, Android Studio, Netron, Google Colab

Chapter 5

System Analysis

To solve a problem, one must first analyse the situation. By performing a system analysis, we can identify and characterise the challenges at hand, as well as define and assess potential solutions. It is a style of approaching the business and the issues it faces, together with a group of technologies that aid in problem solving. In system analysis, which establishes the goals for design and development, feasibility studies play a crucial role.

5.1 Feasibility Study

It seems possible to filter social media content using visual sentiment analysis. In order to analyse the sentiment expressed by images, image classification algorithms are employed. Using artificial intelligence (AI), the method entails scanning and analysing photographs and videos posted on social media sites to see if they contain anything that is against community standards or guidelines. This might include sexually explicit content, graphic violence, and hate speech.

Instead of depending solely on human censors, social media companies can more rapidly and reliably identify potentially dangerous or objectionable content by employing visual sentiment analysis. This might contribute to enhancing the overall security and safety of online communities. In this case three key considerations involved in the feasibility study are:

1. Economical Feasibility
2. Technical Feasibility
3. Social Feasibility

5.1.1 Economical Feasibility

The expense of deploying and maintaining the technology, the possible effects on user engagement and revenue, and the societal benefits all play a role in determining whether social media censorship using visual sentiment analysis is economically viable. Visual sentiment analysis would involve a substantial investment in infrastructure and AI technology, as well as continual upkeep and improvements. The potential effect on the right to free speech as well as the moral and legal ramifications of content censorship are additional factors to take into account. Social media sites would have to be open and honest about their censorship rules and practises to avoid unfairly singling out particular people or organisations. Overall, the economic viability of utilising visual sentiment analysis to filter social media is complicated and depends on a number of variables. Although there could be costs and risks, the advantages to society and user safety could make the investment and use of this technology acceptable.

5.1.2 Technical Feasibility

Due to recent developments in artificial intelligence (AI), social media censorship via visual sentiment analysis is highly technically feasible. Platforms can precisely identify and classify visual content that contravenes community standards or rules by training machine learning models with vast datasets of labelled photos and videos.

Natural language processing (NLP) developments have also made it possible to analyse text-based material, including captions and comments.

5.1.3 Social Feasibility

Social media filtering is a difficult topic that depends on a range of elements, such as user expectations, cultural norms, and concerns about censorship and free expression. On the one hand, a lot of users anticipate social media platforms to be in charge of removing unpleasant and dangerous content, and visual sentiment analysis can help increase the efficiency and precision of content moderation. This could contribute to fostering a safer and more encouraging online environment. There is a chance of bias and unfairness, and some people may see the use of visual sentiment analysis as a violation of their right to free expression.

5.2 Analysis

5.2.1 Performance Analysis

For the complete functionality of the project work, the project is run with the help of healthy networking environment. Performance analysis is done to find out whether the proposed system can perform in the best manner.

The accuracy, effectiveness, and scalability of visual sentiment analysis can be used to evaluate the efficacy of social media censorship. Although visual sentiment analysis algorithms are becoming more accurate, there is still a chance of false positives or false negatives, which can result in unfair censorship or permit harmful content to remain online.

5.2.2 Technical Analysis

The evaluation of the underlying technology and infrastructure utilised to construct the algorithm is a key component of the technical examination of social media censorship employing visual sentiment analysis. The training data is used by the algorithm to find patterns and features in the images that are linked to particular attitudes, like violence or hate speech.

Bias in the training data or algorithm itself poses a danger of unfair censorship or content misclassification. To reduce this risk, platforms must employ strategies like diversified training data and fairness metrics.

Overall, a thorough analysis of the underlying technology, infrastructure, and bias mitigation strategies is necessary for the technical analysis of social media censoring using visual sentiment analysis.

5.2.3 Economical Analysis

For running this system, we need not have any routers which are highly economical. So the system is economically feasible enough.

The algorithm must also be trained, optimised, and integrated with other components of the content moderation system. Social media sites must set aside resources for these tasks.Visual sentiment analysis may also come at a cost in terms of continuous upkeep and operational expenses.

On the plus side, putting visual sentiment analysis into practise can be quite profitable for social media sites.

Chapter 6

System Design

A design is a useful technical depiction of a future construction. It is the most important stage of a system's development. The process of translating requirements into a software representation is called software design.

In software engineering, design is encouraged at the design phase. The new system must be designed based on user requirements and a thorough analysis of the current system. The system design phase is currently underway. The best technique to faithfully transfer a customer's requirement into the final software solution is through design. Design develops a representation or model and offers information about the architecture, interfaces, and components required to implement a system.

6.1 System Development Methodology

System development method is a process through which a product will get completed or a product gets rid from any problem. Software development process is described as a number of phases, procedures and steps that gives the complete software. It follows series of steps which is used for product progress. The development method followed in this project is waterfall model.
The steps that we went through are:

1. Our work involves five stages— Stage I, Stage II, Stage III, Stage IV and Stage V. In Stage I, the image samples that were extracted from multiple sources and made into a dataset, loaded with their labels being determined by us manually and by using bounding boxes.

2. We extracted about 300 images on our own and we used Labelbox software for the labelling and assigning bounding boxes to all the images.

3. In Stage II, the pre-processing of images was done by converting the images into an RGB format, and the resizing of the images into different dimensions was

performed as required by the pre-trained model. The model requires an image dimension of 224 × 224 × 3, and the normalization of image pixels was done in this step using Roboflow.

4. In Stage III, the training and testing samples were created by dividing the input image samples (300) into training (270) and testing (30) images. Nearly 90% of the samples were utilized for training, with the remaining 10% being used for testing.

5. The training process was started by building the pre-trained model architectures with additional layers, and the prediction of unknown samples was done next, in the same stage.

6. The fine-tuning of the models was done by freezing some layers and unfreezing the remaining layers. The performance metrics of the models, such as accuracy, precision, recall, and F1 measures, were generated, and the comparative analysis of the different pre-trained models was done.

7. Stage IV was the longest among all the steps as we had to build a fully-functional clone of Instagram from scratch, using Flutter. The backend functionality will be based on Firebase.

8. In Stage V, we finally integrated the model into the app. And it will be called into action once the users attempt to post a picture.

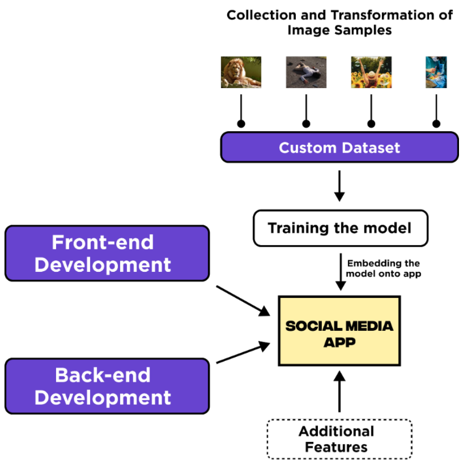


Figure 6.1: Various stages of project

6.2 Design using UML

Using both static and dynamic UML diagrams, the process of designing a UML diagram explains how the system's process interacts with its objects and how the process's objects communicate with one another. It has become increasingly challenging to create and maintain high-quality programmes in a timely manner in the ever-evolving world of object-oriented application development. The Unified Modelling Language (UML), which was created in response to this difficulty and the necessity for a universal object modelling language that anybody could use, is the information industries' equivalent of the blue print. It is a technique for thoroughly outlining the systems architecture. system that is simpler to construct or maintain and is more likely to withstand changes in necessity.

6.3 Data Flow Diagram

The DFD, also known as bubble chart is a straightforward graphical formalism that may be used to depict a system in terms of the data that is fed into it, the different operations that are performed on it, and the data that is generated as a result of those operations.

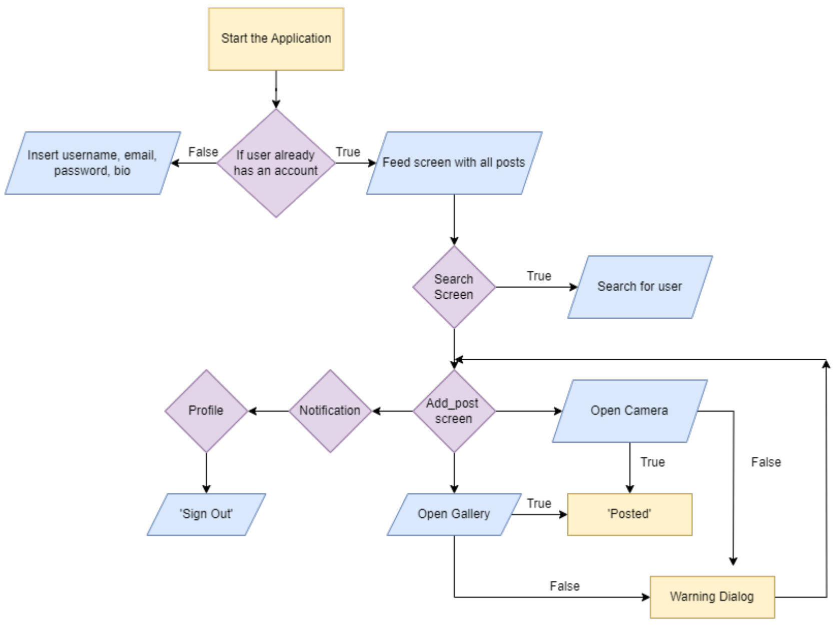


Figure 6.2: Dataflow diagram

6.4 Component Diagram

A component diagram in the Unified Modelling Language shows how smaller components are connected to create bigger components and/or software systems. They serve as examples of the structure of systems that can be arbitrary complicated.

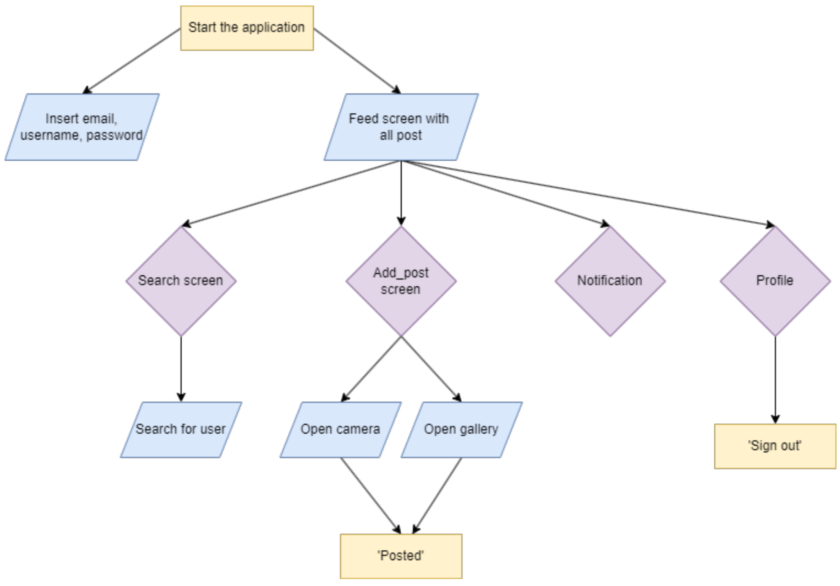


Figure 6.3: Component diagram

6.5 Use Case Diagram

An interaction between external entities and the system under study that is goal-oriented is defined by a use case. The actors in the system are the external entities that communicate with it. The use case diagram can be used to graphically represent a group of use cases that define all system features in great detail.

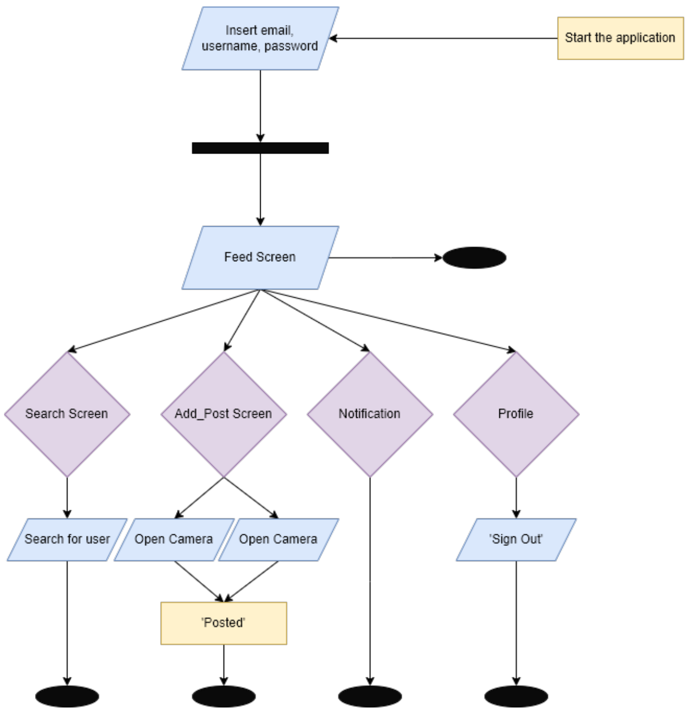


Figure 6.4: Use Case Diagram

Chapter 7

Implementation

7.1 Main

This is the Main dart file which decides whether the app is being opened in Windows or an Android device and elects which layout to display.

```
void main() async {
  WidgetsFlutterBinding.ensureInitialized();
  if (kIsWeb) {
    await Firebase.initializeApp(
      options: const FirebaseOptions(
          apiKey: "AIzaSyCBCE4vrQmudztbORmdKgnUpxqg-2oZgJk",
          authDomain: "social-media-censoring-2db25.firebaseapp.com",
          projectId: "social-media-censoring-2db25",
          storageBucket: "social-media-censoring-2db25.appspot.com",
          messagingSenderId: "545046566716",
          appId: "1:545046566716:web:ae1fae688f8f352e257039",
          measurementId: "G-BXZC1NY9ZC"
      ),
    );
  } else {
    await Firebase.initializeApp();
  }
  runApp(const MyApp());
}
class MyApp extends StatelessWidget {
  const MyApp({Key? key}) : super(key: key);

  @override
  Widget build(BuildContext context) {
    return MultiProvider(
      providers: [
        ChangeNotifierProvider(create: (_) => UserProvider(),),
      ],
      child: MaterialApp(
        debugShowCheckedModeBanner: false,
        title: 'Social Media App',
        theme: ThemeData.dark().copyWith(
          scaffoldBackgroundColor: mobileBackgroundColor,
        ),
        home: StreamBuilder(
          stream: FirebaseAuth.instance.authStateChanges(),
          builder: (context, snapshot) {
            if (snapshot.connectionState == ConnectionState.active) {
              if (snapshot.hasData) {
                return const ResponsiveLayout(
                  mobileScreenLayout: MobileScreenLayout(),
                  webScreenLayout: WebScreenLayout(),
                );
              } else if (snapshot.hasError) {
                return Center(
                  child: Text('${snapshot.error}'),
                );
              }
            }
            if (snapshot.connectionState == ConnectionState.waiting) {
              return const Center(
                child: CircularProgressIndicator(),
              );
            }
            return const LoginScreen();
          },
        ),
      ),
    );
  }
}
```

7.2 Login

The screen which loads up when the application is first opened.

```
    class LoginScreen extends StatefulWidget {
  const LoginScreen({Key? key}) : super(key: key);

  @override
  _LoginScreenState createState() => _LoginScreenState();
}

class _LoginScreenState extends State<LoginScreen> {
  final TextEditingController _emailController = TextEditingController();
  final TextEditingController _passwordController = TextEditingController();
  bool _isLoading = false;

  @override
  void dispose() {
    super.dispose();
    _emailController.dispose();
    _passwordController.dispose();
  }

  void loginUser() async {
    setState(() {
      _isLoading = true;
    });
    String res = await AuthMethods().loginUser(
        email: _emailController.text, password: _passwordController.text);
    if (res == 'success') {
      Navigator.of(context).pushAndRemoveUntil(
          MaterialPageRoute(
            builder: (context) => const ResponsiveLayout(
              mobileScreenLayout: MobileScreenLayout(),
              webScreenLayout: WebScreenLayout(),
            ),
          ),
          (route) => false);

      setState(() {
        _isLoading = false;
      });
    } else {
      setState(() {
        _isLoading = false;
      });
      showSnackBar(context, res);
    }
  }

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      resizeToAvoidBottomInset: false,
      body: SafeArea(
        child: Container(
          padding: MediaQuery.of(context).size.width > webScreenSize
              ? EdgeInsets.symmetric(
                  horizontal: MediaQuery.of(context).size.width / 3)
              : const EdgeInsets.symmetric(horizontal: 32),
          width: double.infinity,
          child: Column(
            crossAxisAlignment: CrossAxisAlignment.center,
            children: [
              Flexible(
                child: Container(),
                flex: 2,
              ),
              SvgPicture.asset(
                'assets/ic_instagram.svg',
                color: primaryColor,
                height: 64,
              ),
              const SizedBox(
                height: 64,
              ),
              TextFieldInput(
                hintText: 'Enter your email',
                textInputType: TextInputType.emailAddress,
                textEditingController: _emailController,
              ),
              const SizedBox(
                height: 24,
              ),
              TextFieldInput(
                hintText: 'Enter your password',
                textInputType: TextInputType.text,
                textEditingController: _passwordController,
                isPass: true,
              ),
              const SizedBox(
                height: 24,
              ),
              InkWell(
                child: Container(
                  child: !_isLoading
                      ? const Text(
                          'Log in',
                        )
                      : const CircularProgressIndicator(
                          color: primaryColor,
                        ),
                  width: double.infinity,
                  alignment: Alignment.center,
                  padding: const EdgeInsets.symmetric(vertical: 12),
                  decoration: const ShapeDecoration(
                    shape: RoundedRectangleBorder(
                      borderRadius: BorderRadius.all(Radius.circular(4)),
                    ),
                    color: blueColor,
                  ),
                ),
                onTap: loginUser,
              ),
              const SizedBox(
                height: 12,
              ),
              Flexible(
                child: Container(),
                flex: 2,
              ),
              Row(
                mainAxisAlignment: MainAxisAlignment.center,
                children: [
                  Container(
                    child: const Text(
                      'Don\'t have an account?',
                    ),
                    padding: const EdgeInsets.symmetric(vertical: 8),
                  ),
                  GestureDetector(
                    onTap: () => Navigator.of(context).push(
                      MaterialPageRoute(
                        builder: (context) => const SignupScreen(),
                      ),
                    ),
                    child: Container(
                      child: const Text(
                        ' Signup.',
                        style: TextStyle(
                          fontWeight: FontWeight.bold,
                        ),
                      ),
                      padding: const EdgeInsets.symmetric(vertical: 8),
                    ),
                  ),
                ],
              ),
            ],
          ),
        ),
      ),
    );
  }
}
```

7.3 Feed Screen

The screen which shows all the posts that have been posted by the various users logged onto app.

```
class FeedScreen extends StatefulWidget {
  const FeedScreen({Key? key}) : super(key: key);

  @override
  State<FeedScreen> createState() => _FeedScreenState();
}

class _FeedScreenState extends State<FeedScreen> {
  @override
  Widget build(BuildContext context) {
    final width = MediaQuery.of(context).size.width;

    return Scaffold(
      backgroundColor:
          width > webScreenSize ? webBackgroundColor : mobileBackgroundColor,
      appBar: width > webScreenSize
          ? null
          : AppBar(
              backgroundColor: mobileBackgroundColor,
              centerTitle: false,
              title: SvgPicture.asset(
                'assets/ic_instagram.svg',
                color: primaryColor,
                height: 32,
              ),
              actions: [
                IconButton(
                  icon: const Icon(
                    Icons.messenger_outline,
                    color: primaryColor,
                  ),
                  onPressed: () {},
                ),
              ],
            ),
      body: StreamBuilder(
        stream: FirebaseFirestore.instance.collection('posts').snapshots(),
        builder: (context,
            AsyncSnapshot<QuerySnapshot<Map<String, dynamic>>> snapshot) {
          if (snapshot.connectionState == ConnectionState.waiting) {
            return const Center(
              child: CircularProgressIndicator(),
            );
          }
          return ListView.builder(
            itemCount: snapshot.data!.docs.length,
            itemBuilder: (ctx, index) => Container(
              margin: EdgeInsets.symmetric(
                horizontal: width > webScreenSize ? width * 0.3 : 0,
                vertical: width > webScreenSize ? 15 : 0,
              ),
              child: PostCard(
                snap: snapshot.data!.docs[index].data(),
              ),
            ),
          );
        },
      ),
    );
  }
}
```

7.4 Add Post Page and ML model

This page allows users to upload a picture either from their gallery or from the camera. The ML model is integrated here to perform real-time classification of the pictures being uploaded.

```
    class AddPostScreen extends StatefulWidget {
  const AddPostScreen({Key? key}) : super(key: key);

  @override
  _AddPostScreenState createState() => _AddPostScreenState();
```

```
}


class _AddPostScreenState extends State<AddPostScreen> {
  Uint8List? _file;
  bool isLoading = false;
  List? _results;
  late PageController pageController;

  @override
  void initState() {
    super.initState();
    pageController = PageController();
  }

  @override
    void dispose() {
      super.dispose();
      _descriptionController.dispose();
      pageController.dispose();
    }

  final TextEditingController _descriptionController = TextEditingController();

_selectImage() async {
  return showDialog(
    context: context,
    builder: (BuildContext context) {
      return SimpleDialog(
        title: const Text('Create a Post'),
        children: <Widget>[
          SimpleDialogOption(
            padding: const EdgeInsets.all(20),
            child: const Text('Take a photo'),
            onPressed: () async {
              Navigator.pop(context);
              File img = await pickImage(ImageSource.camera);
              Uint8List file = img.readAsBytesSync();
              List results = await imageClassification(img);
              setState(() {
                _file = file;
                _results = results;
              });

              if (_results?[0]['label'] == 'Negative') {
                _showAlertDialog();
              } else {
                print(_results?[0]['label']);
              }
            },
          ),
          SimpleDialogOption(
            padding: const EdgeInsets.all(20),
            child: const Text('Choose from Gallery'),
            onPressed: () async {
              Navigator.of(context).pop();
              File img = await pickImage(ImageSource.gallery);
              Uint8List file = img.readAsBytesSync();
              List results = await imageClassification(img);
              setState(() {
                _file = file;
                _results = results;
              });
              if (_results?[0]['label'] == 'Negative') {
                _showAlertDialog();
              } else {
                print(_results?[0]['label']);
              }
            },
          ),
          SimpleDialogOption(
            padding: const EdgeInsets.all(20),
            child: const Text("Cancel"),
            onPressed: () {
              Navigator.pop(context);
              pageController.jumpToPage(0);
            },
          )
```

```
        ],
      );
    },
  );
}

_showAlertDialog() async {
  await showDialog(
    context: context,
    builder: (BuildContext context) {
      return AlertDialog(
        title: const Text('Negative Image Detected!',
        style: TextStyle(
          fontWeight: FontWeight.bold,
          fontSize: 18.0,
          color: Colors.redAccent
        ),),
        content: const Text("You can't proceed to post this picture. Please choose
        actions: [
          TextButton(
            child: const Text('Retry'),
            onPressed: () {
              Navigator.of(context).pop();
              clearImage();
              _selectImage();
            },
          ),
          TextButton(
            child: const Text('Cancel'),
            onPressed: () {
              Navigator.of(context).pop();
              clearImage();
            },
          ),
        ],
      );
    },
  );
}
```

```
void postImage(String uid, String username, String profImage) async {
    setState(() {
      isLoading = true;
    });
    try {
      // upload to storage and db
      String res = await FireStoreMethods().uploadPost(
        _descriptionController.text,
        _file!,
        uid,
        username,
        profImage,
      );
      if (res == "success") {
        setState(() {
          isLoading = false;
        });
        showSnackBar(
          context, 'Posted!',
        );
        clearImage();
      } else {
        showSnackBar(context, res);
      }
    } catch (err) {
      setState(() {
        isLoading = false;
      });
      showSnackBar(
        context,
        err.toString(),
      );
    }
  }

  void clearImage() {
    setState(() {
      _file = null;
    });
  }



  @override
  Widget build(BuildContext context) {
    final UserProvider userProvider = Provider.of<UserProvider>(context);

    return _file == null
        ? Center(
            child: IconButton(
              icon: const Icon(
                Icons.upload,
              ),
              onPressed: () => _selectImage(),
            ),
          )
        : Scaffold(
            appBar: AppBar(
              backgroundColor: mobileBackgroundColor,
              leading: IconButton(
                icon: const Icon(Icons.arrow_back),
                onPressed: clearImage,
              ),
              title: const Text(
                'Post to',
              ),
              centerTitle: false,
              actions: <Widget>[
                TextButton(
                  onPressed: () => postImage(
                    userProvider.getUser.uid,
                    userProvider.getUser.username,
                    userProvider.getUser.photoUrl,
                  ),
                  child: const Text(
                    "Post",
                    style: TextStyle(
                        color: Colors.blueAccent,
                        fontWeight: FontWeight.bold,
                        fontSize: 16.0),
                  ),
                )
              ],
            ),
            // POST FORM
            body: Column(
              children: <Widget>[
                isLoading
                    ? const LinearProgressIndicator()
                    : const Padding(padding: EdgeInsets.only(top: 0.0)),
                const Divider(),
                Row(
                  mainAxisAlignment: MainAxisAlignment.spaceAround,
                  crossAxisAlignment: CrossAxisAlignment.start,
                  children: <Widget>[
                    CircleAvatar(
                      backgroundImage: NetworkImage(
                        userProvider.getUser.photoUrl,
                      ),
                    ),
                    SizedBox(
                      width: MediaQuery.of(context).size.width * 0.3,
                      child: TextField(
                        controller: _descriptionController,
                        decoration: const InputDecoration(
                            hintText: "Write a caption...",
                            border: InputBorder.none),
                        maxLines: 8,
                      ),
                    ),
                    SizedBox(
                      height: 45.0,
                      width: 45.0,
                      child: AspectRatio(
                        aspectRatio: 487 / 451,
                        child: Container(
                          decoration: BoxDecoration(
                              image: DecorationImage(
                            fit: BoxFit.fill,
                            alignment: FractionalOffset.topCenter,
                            image: MemoryImage(_file!),
                          )),
                        ),
                      ),
                    ),
                  ],
                ),
                const Divider(),
              ],
            ),
          );
  }
}
```

7.5 InceptionV3 Model

The Python code for the image classification model is given below.

```
rows = open('train_labels_demo.csv').read().strip().split('\n')
X = []
Y = []
bboxes = []
img_paths = []
for row in rows:
  row = row.split(",")
  (filename, w, h, label, xmin, ymin, xmax, ymax) = row
  img_path = os.path.sep.join(['train', filename])
  image = cv2.imread(img_path)
  (h, w) = image.shape[:2]
  xmin = float(xmin) / w
  ymin = float(ymin) / h
  xmax = float(xmax) / w
  ymax = float(ymax) / h

  image = load_img(img_path, target_size = (224, 224))
  image = img_to_array(image)
  X.append(image)
  img_paths.append(img_path)
  Y.append(label)
  bboxes.append((xmin, ymin, xmax, ymax))

X = np.array(X, dtype = "float32") / 255.0
Y = np.array(Y)
bboxes = np.array(bboxes, dtype="float32")
img_paths = np.array(img_paths)
lb =  LabelBinarizer()
Y = lb.fit_transform(Y)
Y = to_categorical(Y)

split = train_test_split(X, Y, bboxes, img_paths, test_size = 0.1, random_state = 

(X_train, X_test) = split[:2]
(Y_train, Y_test) = split[2:4]
(BBX_train, BBX_test) = split[4:6]
(Paths_train, Paths_test) = split[6:]

img_rows, img_cols = 224, 224
inception = InceptionV3(weights = 'imagenet',
                        include_top = False,
                        input_shape = (img_rows, img_cols, 3))

for layer in inception.layers:
  layer.trainable = False

flatten = inception.output
flatten = Flatten()(flatten)

bboxHead = Dense(128, activation="relu")(flatten)
bboxHead = Dense(64, activation="relu")(bboxHead)
bboxHead = Dense(32, activation="relu")(bboxHead)
bboxHead = Dense(4, activation = 'sigmoid', name = "bounding_box")(bboxHead)

top_model = Dense(512, activation='relu')(flatten)
top_model = Dropout(0.5)(top_model)
top_model = Dense(512, activation='relu')(top_model)
top_model = Dropout(0.5)(top_model)
top_model = Dense(2, activation='softmax', name = 'class_label')(top_model)

model = Model(inputs = inception.input, outputs = (top_model, bboxHead))

losses = {"class_label": 'categorical_crossentropy', 'bounding_box': 'mean_squared
loss_wts = {"class_label": 1.0, 'bounding_box': 1.0}

opt = Adam(lr = 1e-4)
model.compile(optimizer=opt, loss=losses, metrics = ['accuracy'], loss_weights = l

trainTargets = {"class_label": Y_train, "bounding_box": BBX_train}
testTargets = {"class_label": Y_test, "bounding_box": BBX_test}

model.fit(X_train, trainTargets,
            validation_data = (X_test, testTargets),
            epochs = 30,
            batch_size = 32,
            verbose = 2,
            initial_epoch = 0)
```

Chapter 8

Testing

8.1 Testing Methodologies

8.2 Unit Testing

Developers often carry out unit testing, which entails creating and running test cases for each unit of code. The test cases are created to ensure that the unit of code carries out the required function, generates the anticipated output, and appropriately handles edge cases and fault situations. Since unit tests are frequently automated, testing during the development process can be done quickly and effectively. Every time the code is modified, automated unit tests can be run automatically to ensure that no new issues or regression.Unit testing has a number of advantages like helping to find defects early, improving code quality, etc.

8.3 System Testing

System testing is a kind of software testing that involves thoroughly examining an entire system or software application. It is done to make sure the programme complies with the requirements and runs properly in the environment it was designed for.After integration testing, in which distinct modules or components are tested before being combined to form an entire system, system testing is typically carried out. The integrated system is subjected to system testing to ensure that all modules or components function as intended and that the system satisfies the criteria outlined in the software requirements specification (SRS). Comparing the system's functionality against the SRS's requirements is known as functional testing. This includes evaluating specific features, functions, and user interfaces. To make sure that the system is simple to use and meets end users' needs, usability testing comprises evaluating the system's usability and user experience.

8.4 Quality Assurance

Software quality standards and criteria are ensured through the quality assurance (QA) process, which is a step in the software development process. By setting and upholding stringent quality standards throughout the development process, QA's major objective is to prevent errors and problems from appearing in the product. The following steps are often included in the QA process:

- Gathering and analysing the software application's requirements is the first step in the quality assurance process.
- Following the identification of the requirements, a test plan is created to make sure that each requirement is adequately tested.
- Any flaws or errors found during testing are noted, followed up on, and sent to the development team to be fixed.
- The reports give a general assessment of the software's quality and point out any shortcomings.

8.5 Functional Test

Functional testing is a type of software testing that focuses on ensuring that a system or software application performs as intended and satisfies the necessary functional criteria. It is a form of "black-box" testing where the tester is more interested in the output the system generates in response to particular inputs than the inner workings of the software. Functional testing is centered on the following items:

1. Valid Input : identified classes of valid input must be accepted
2. Invalid Input : identified classes of invalid input must be rejected
3. Functions : identified functions must be exercised.
4. Output : identified classes of application outputs must be exercised.
5. Systems/Procedures: Interfacing systems or procedures must be invoked

8.5.1 Accuracy of Features

1. The accuracy of images within the app is 81.2%
2. The accuracy of the transfer-learning model is 96.67%
3. The loss incurred is 0.95%

Chapter 9

Results and Performance Analysis

9.1 Application Screenshots

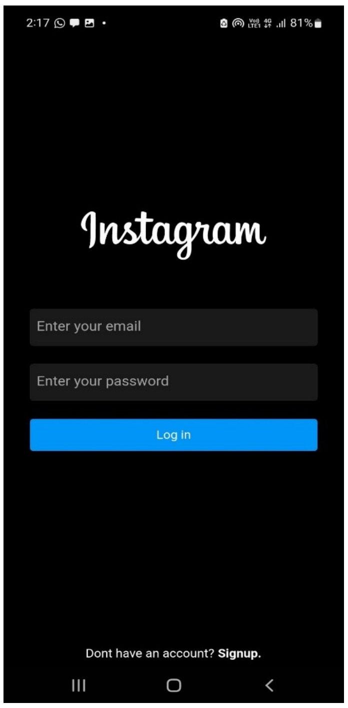


Figure 9.1: Login

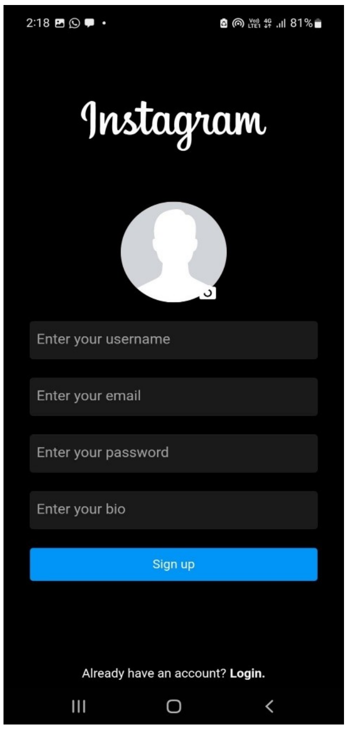


Figure 9.2: Sign Up

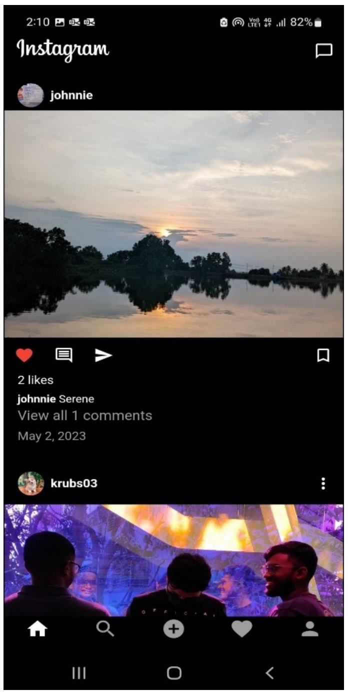


Figure 9.3: Feed Screen

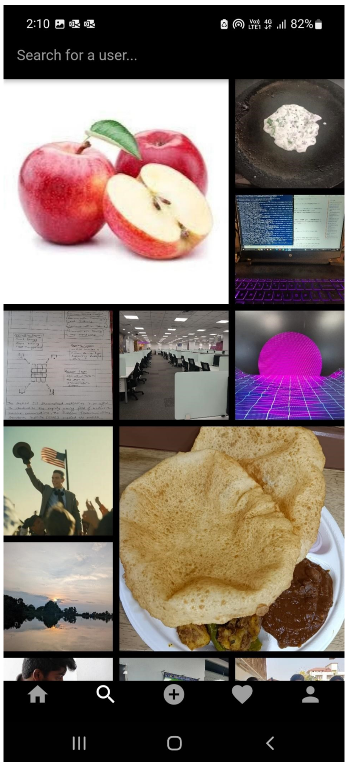
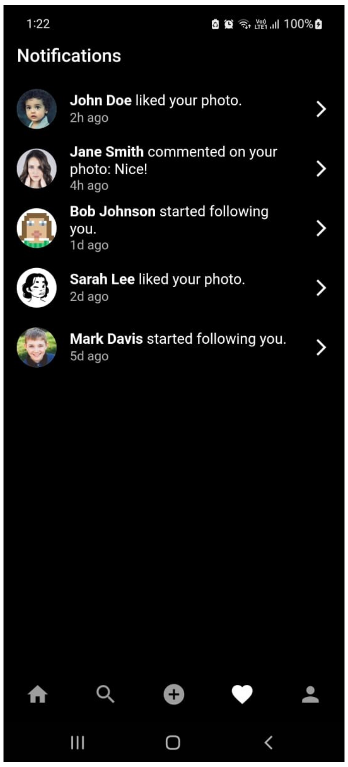


Figure 9.4: Search and Explore



Notifications

John Doe liked your photo.
2h ago

Jane Smith commented on your photo: Nice!
4h ago

Bob Johnson started following you.
1d ago

Sarah Lee liked your photo.
2d ago

Mark Davis started following you.
5d ago

Figure 9.5: Notifications

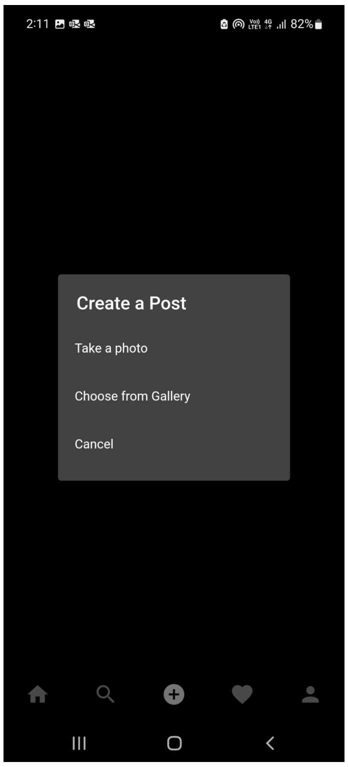
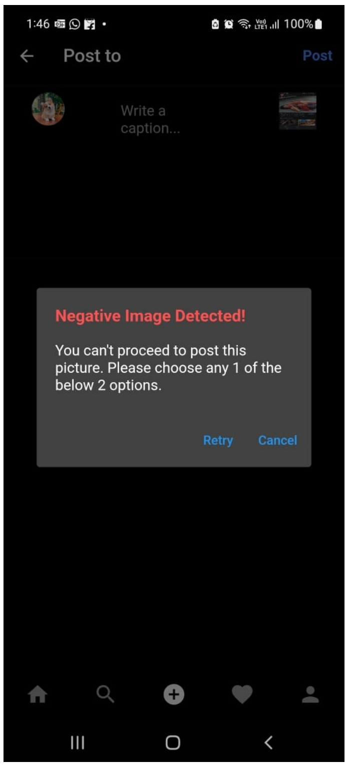


Figure 9.6: Posting



Negative Image Detected!

You can't proceed to post this picture. Please choose any 1 of the below 2 options.

Figure 9.7: Warning Dialog

Chapter 10

Conclusion

During the course of this project, we have used the transfer-learning model of Inception-V3 to predict the sentiments of the images unique to our use-case of social-media censoring. By unfreezing a portion of the model and freezing the first layers, we adjusted the models' performance. To lessen the impact of overfitting, some additional layers were added, including the dropout, batch normalization, and weight regularization layers.

The model, for now, has only been trained to perform screening of only one type of inappropriate content (violent/bloodshed). But it can also be extended to filter out additional types of inappropriate content like nudity, profane text, etc. This can be done by improving quality of the dataset. The best accuracy was observed on the 6th epoch (1.00) out of the 30 epochs we trained the model for. The model is still not fully accurate as it seems to wrongly classify a very few instances of the validation data.

Our future work would be dedicated towards refining and growing the dataset so that the model can be much more accurate, as well as adding some additional functionalities to the app to increase its usability while also removing certain bugs and making it entirely bug-free.

References

[1] Ganesh Chandrasekaran, Naaji Antoanela, Gabor Andrei, Ciobanu Monicaand Jude Hemanth: *Visual Sentiment Analysis Using Deep Learning Models with Social Media Data*, 2022

[2] Quoc-Tuan Truong, Hady W. Lauw: *Visual Sentiment Analysis for Review Images with Item-Oriented and User-Oriented CNN*, 2017

[3] Fireship on YouTube `https://www.youtube.com/@Fireship`

[4] NetNinja on YouTube `https://www.youtube.com/@NetNinja`

[5] Flutter Documentation `https://docs.flutter.dev/`

[6] About TFLite `https://www.tensorflow.org/lite`

[7] Google Firebase `https://firebase.google.com/docs?gclid=sAFp`

[8] InceptionV3 documentation `https://keras.io/api/applications/inceptionv3/`