DocuSign Envelope ID: 72C90A03-080C-4BBA-B66D-D054BE574A31

Video Perception Analyzer

A Project Report
Presented to
The Faculty of the College of
Engineering

San Jose State University
In Partial Fulfillment
Of the Requirements for the Degree
Master of Science in Software Engineering

By
Swathi Anandram, Aryan Jadon, Harika Nalam, Shreya Nimbhorkar
May 2023

DocuSign Envelope ID: 72C90A03-080C-4BBA-B66D-D054BE574A31

Copyright © 2023

Swathi Anandram, Aryan Jadon, Harika Nalam, Shreya Nimbhorkar

ALL RIGHTS RESERVED

DocuSign Envelope ID: 72C90A03-080C-4BBA-B66D-D054BE574A31

APPROVED

DocuSigned by:
Vijay Eranti
3AF6B24FF2B8417...

Vijay Eranti, Project Advisor

Dan Harkey, Director, MS Software Engineering

Rod Fatoohi, Department Chair

DocuSign Envelope ID: 72C90A03-080C-4BBA-B66D-D054BE574A31

ABSTRACT

Project Report
By Swathi Anandram, Aryan Jadon, Harika Nalam, Shreya Nimbhorkar

Videos constitute a significant portion of all data in the world and have a vast potential to store essential information. Video understanding is crucial in solving real-world problems such as identifying workplace hazards, security, and monitoring. Computer Vision (CV), a subdomain of Artificial Intelligence (AI), aims to derive and understand meaningful information from videos for recommendation and classification. A model built using CV techniques builds a relationship and interaction knowledge graph to answer relevant questions.

According to recent news articles, California legislators have approved a bill under which social media platforms would require to publicly disclose their policies on how they screen and recommend content, a requirement to combat the spread of hate and violence online. Many social media platforms violated this rule mandated by the government to filter the restricted content according to the user's age. We propose a state-of-art solution to abide by these government regulations.

This project proposes a perception analyzer to automatically analyze video content using a multimodel approach that applies multiple Deep Learning (DL) and Machine Learning (ML) models. The system will learn and extract information from video data, including images, audio, and video metadata. The proposed solution allows users to view videos, upload videos, search for their favorite videos. Video Perception Analyzer behind the scenes will filter disinformation, and violent or inappropriate content, and recommend videos relevant to their choice.

DocuSign Envelope ID: 72C90A03-080C-4BBA-B66D-D054BE574A31

Acknowledgments

The authors are deeply indebted to Professor Vijay Eranti for his constant support and guidance throughout the project. The authors deeply thank Professor Dan Harkey for his feedback and support in completing this project.

DocuSign Envelope ID: 72C90A03-080C-4BBA-B66D-D054BE574A31

Table of Contents

DocuSign Envelope ID: 72C90A03-080C-4BBA-B66D-D054BE574A31

DocuSign Envelope ID: 72C90A03-080C-4BBA-B66D-D054BE574A31

DocuSign Envelope ID: 72C90A03-080C-4BBA-B66D-D054BE574A31

List of Figures

List of Tables

DocuSign Envelope ID: 72C90A03-080C-4BBA-B66D-D054BE574A31

Chapter 1. Project Overview

1.1 Introduction

Video footage accounts for a sizable fraction of all data worldwide. Video data, in addition to being abundant, has a huge potential to hold essential information. Understanding video is a critical activity in the field of computer vision due to its breadth, depth, and usefulness.

Some of the task associated with understanding video includes video classification, action detection, action forecasting, and dense captioning. Additionally, video audio transcription can also help us understand the information in the video. Using computer vision models along with natural language processing is one of the effective ways to tackle the problem of video understanding.

One of the most fundamental video comprehension tasks is video classification, which involves identifying an activity in a video. A model accomplishes this by assigning a series of scores to a video, each corresponding to an action class. The score reflects how likely it is that the action will be executed at any point in the video.

Action detection is a far more difficult task than video classification. It combines the subtasks of object detection (where the agents are situated), object tracking (where the agents are traveling), and video classification (what activities the agents are conducting) into a single concept. Video classification and action detection, both generate scores for predefined categories of actions: video classification generates a single set of scores for an entire video clip, whereas action detection generates a set of scores for each detected agent on a per-frame basis. Dense

DocuSign Envelope ID: 72C90A03-080C-4BBA-B66D-D054BE574A31

captioning extends beyond categories by assigning natural language descriptions to subsets of video frames.

Given the current and previous frames, action forecasting predicts activities performed in future frames of a movie. In its most basic version, a single, future, global action performed by a single agent in a video is anticipated. This is similar to video categorization but forecasts the likelihood of future, unseen behaviors (rather than past, observed actions).

The video perception analyzer is an application that lets users view videos of their interests. The application provides users with a platform to select from various categories. The main focus of this application is to give users a trustable system to view videos of their interests. Users can check why a particular video has been shown to them, which internally uses explainable AI to provide recommendations. The application can upload videos for further analysis by Machine Learning (ML) models such as OpenMM and Detectron2. Once ML models analyze the video, the video is checked for any hatred/violent/misinformation content. Similar videos and categories are shown to the user based on liked videos and categories selected by ML models. The user's age is used for recommendation, as some content might not be suitable for people below a certain age. The main application is developed for mobile devices using React Native, but users can view the application as a web application. As the user can scroll through videos, using the application on mobile is easier. This project implements machine learning models APIs to understand a video layer-wise, we use results and predictions from all machine learning models for recommendation purposes.

DocuSign Envelope ID: 72C90A03-080C-4BBA-B66D-D054BE574A31

1.2 Proposed Area of Study and Academic Contribution

The critical aspects of Computer Vision (CV) include visual recognition tasks, image classification, localization, and detection. In 2015, Convolutional Neural Network (CNN) [1], an Artificial Neural Network (ANN), became well-received in various computer vision-related tasks. Using backpropagation, learning a feature's spacial hierarchy was carried out by CNN automatically and adaptively. It used several building blocks, including pooling, convolution, and fully connected layers. Pixels in an image are used for building feature recognition by constantly applying filters by CNN. Upon comparison with different classification methods, a substantially low amount of pre-processing is required by CNN. CNN can learn these filters and attributes with sufficient training, unlike basic methods where filters need to be hand-engineered.

The structure of a CNN is similar to the neuron's connectivity pattern in the human brain. It was modeled after the organization of the visual cortex. Individual neurons react to inputs only in the Receptive Field, a constrained visual field area. The entire visual field is covered by a series of such fields that overlap. CNN is thought to be essential for vision tasks.

The Vision Transformer (ViT) [2] emerged in 2022 as an alternative to CNN's current state-of-the-art computer vision and is thus widely adapted in various image recognition tasks. Regarding computational accuracy and efficiency, ViT models outperform the current state-of-the-art (CNN) by nearly x4.

The transformer first appeared in a paper [3] in 2017 with the cryptic title "Attention Is All You Need." In many other techniques of AI, the system would initially focus on local patches of input data before expanding to the entire dataset. For example, nearby words would be grouped in a language model. On the other hand, the transformer runs processes that connect or

DocuSign Envelope ID: 72C90A03-080C-4BBA-B66D-D054BE574A31

pay attention to every element in the input data. Researchers refer to the process as "self-attention." It means the transformer can see the complete data set traces as soon as it begins training.

After a few months following the release of "Attention Is All You Need," one of the most noticeable steps towards expanding the breadth of transformers was taken. A scientist at Google Brain Berlin named Alexey Dosovitskiy worked on computer vision. It is a branch of AI that teaches computer image processing and classification. Like others in the field, he worked with CNNs, which had driven all significant advances in Deep Learning (DL), particularly in CV, for years.

The researchers eventually developed a network known as the Vision Transformer, or ViT, showcased in May 2021. The model's architecture was almost similar to the transformer presented in 2017, with only negligible differences allowing it to study images rather than words. Vision Transformers (ViT) have recently achieved competitive performance in benchmarks for many CV applications, including image classification, object detection, and semantic image segmentation very highly.

1.3 Current State of Art

Most modern image classification systems are based on the ImageNet dataset, which contains about 1.2 million high-resolution training images. No initial segmentation or labels will be applied to test images. Thus algorithms will need to create labeling that identifies the things in the images.

DocuSign Envelope ID: 72C90A03-080C-4BBA-B66D-D054BE574A31

Many Leading CV groups from INRIA, Oxford, and XRCE tested this dataset's latest computer vision techniques. The following new models have excelled in ImageNet: ZFNet (2013)[4], GoogLeNet (2014)[5], VGGNet (2014) [6], ResNet (2015) [7], DenseNet (2016) [8], etc. These models use CNN as their backbone architecture.

Creating bounding boxes and labels for each object is required within an image to detect the object. Using classification and localization on numerous objects instead of simply one dominant object varies from the classification and localization job. Object bounding non-object bounding boxes are the only two object classification classes you have.

Recent years have seen a drastic change in the direction of faster, more effective detection systems. It can be seen as a shift towards sharing processing on a whole image in techniques like You Only Look Once (YOLO) [9], Single Shot MultiBox Detector (SSD) [10], and Region-Based Fully Convolutional Networks (R-FCN)[11]. These methods differentiate themselves from the pricey subnetworks linked to the three R-CNN techniques. The primary justification for these trends is to avoid having different algorithms concentrate on their subproblems in isolation, as doing so often lengthens training time and can reduce network accuracy.

The practice of pursuing a single object of interest, or several objects, in a particular scene, is known as object tracking. It frequently finds use in video and in-person encounters where observations are performed after initial object detection. Deep learning researchers have experimented with several methods in recent years to adapt to the properties of the visual tracking challenge.

DocuSign Envelope ID: 72C90A03-080C-4BBA-B66D-D054BE574A31

Numerous avenues have been investigated, including using other network models like the Recurrent Neural Net [12] and Deep Belief Net [13], and designing the network structure to accommodate video processing and end-to-end learning. It also involves optimizing the procedure, structure, and parameters or even fusing deep learning with conventional computer vision techniques or strategies from other disciplines, like language processing and speech recognition.

Segmentation, a key component of computer vision, separates entire images into groups of pixels that may be categorized and labeled. Semantic segmentation, in particular, aims to comprehend the semantic function of each pixel in the image. DeepLab [14], RefineNet [15], and Dilated Convolutions [16] are just a few examples of fully convolutional networks that have been frequently used in recent semantic segmentation research.

1.4 Project Motivation

The question of social media's long-term trustworthiness and utility has come into sharper light as they have become popular sources for everyday news consumption. We know from a prior study that one way to increase social media sustainability is by being upfront about algorithms. It helps users receive a fair content curation service and clarifies social media's obligations.

Gov. Gavin Newsom (D) of California has signed a law requiring social media transparency that he claims will shield citizens from hateful and misleading online content. According to A.B. 587, "Social media businesses must openly display their guidelines for handling hate speech, misinformation, abuse, and extremism on their platforms and provide information on how well the guidelines are being followed." According to the recently approved

DocuSign Envelope ID: 72C90A03-080C-4BBA-B66D-D054BE574A31

legislation, platforms must also submit semi-annual reports to the attorney general's office of the state outlining their rules on hate speech, extremism, and misinformation.

We propose a state-of-art solution to abide by these government regulations. This project proposes a perception analyzer to automatically analyze video content using a multimodel approach that applies multiple Deep Learning (DL) and Machine Learning (ML) models. The system will learn and extract information from video data, including images, audio, and video metadata. Once developed, the system will act as an explainable AI that allows us to meet the needs of stakeholders, including regulators, customers, and their consumers. Our approach answers how to be transparent about algorithms and how it promotes social media sustainability. The project examines user behavior and algorithm transparency to determine ways to make social media more socially responsible.

By using our solution, content platforms can explain how content recommendations are produced and why people are viewing the content that has been recommended to them by being honest about the algorithms. Accordingly, consumers could modify their feed to avoid or choose particular material or apply superior judgment to control the flow of information. Through our project, we want to demonstrate the theoretical benefits of open, accountable, and sustainable social media.

DocuSign Envelope ID: 72C90A03-080C-4BBA-B66D-D054BE574A31

Chapter 2. Project Architecture

2.1 Introduction

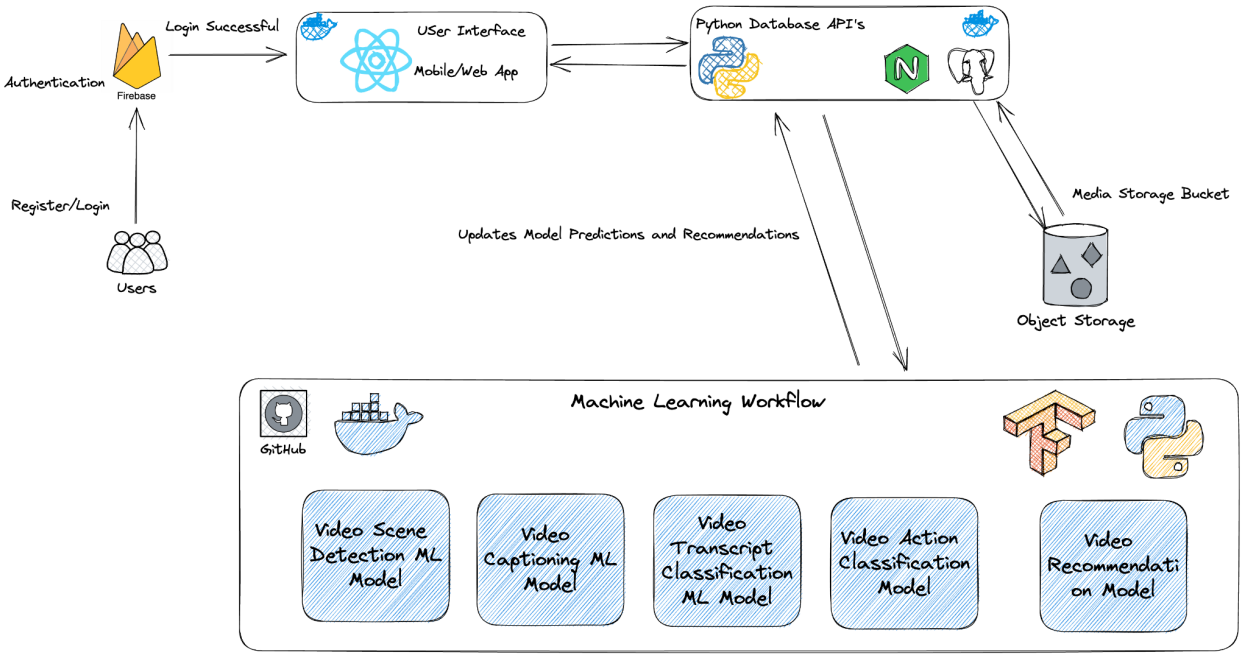


Fig 1. System Architecture

The project consists of Machine Learning (ML) and web application modules that allow the system to have a user interface for users while performing ML tasks. For the Software engineering modules, we make use of three-layered architecture. The first layer is the presentation layer, or the user interface layer, which provides the user with an interface to interact with the application. The second layer is the application layer responsible for processing the API calls. The database layer is the third layer which stores and manages the application-specific data. The application consists of web and mobile interfaces with which users can interact to perform desired tasks, such as viewing videos of their interests. The database serves as the connection between software engineering flow and ML flow. The following section consists of a detailed description of the architecture subsystems.

DocuSign Envelope ID: 72C90A03-080C-4BBA-B66D-D054BE574A31

2.2 Architecture Subsystems

2.2.1 Frontend

We propose a multipurpose application that allows users to have a seamless experience as a mobile and web application. The application uses React framework to build web components and React Native for the mobile application. This interactive application interface uses HTML5, CSS3, and Bootstrap5. The user must log in or register to access the application, as the content recommended to all users will be particular to them based on their interests. The application automatically eliminates any misinformation that the users cannot view. Any violent videos will have age restrictions, and a user of the permitted age to view such videos will still be allowed to either view or ignore such videos. Users can also search for any particular category of videos from the user interface and mark them as "like/dislike" to view similar videos.

2.2.2 Backend

The user interfaces call the APIs to perform various actions, such as getting data from the database or returning a search result for the corresponding user input. The frontend sends the HTTP requests to the backend, which processes the request along with any request payload data to return the necessary response, used by the frontend to show the outputs accordingly to the user. We use Python to handle the API calls. The backend serves as the middleman between the database and the user interface.

2.2.3 Database

The project uses a Postgres Relational database to store the data specific to the application. The database is used across the application to store and manage raw data and

DocuSign Envelope ID: 72C90A03-080C-4BBA-B66D-D054BE574A31

prediction or recommendation results obtained from the ML models. Postgres Relational databases provide more flexibility than other databases while handling extensive data. The APIs can fetch the necessary data stored in the database as per user input back to the user interface. Once the data is provided as input to the ML models, storing their results in the dataset will make the application faster.

2.2.4 Machine Learning

1) *Data Acquisition:* This step collects the data from various sources for model analysis with all the necessary characteristics.
2) *Data Cleaning:* The collected data moves onto the data cleaning stage. This stage consists of data cleaning, removing unnecessary data, imputing missing values, and removing columns with little data that are not useful are performed.
3) *ML Models:* The cleaned data is analyzed using a multimodal approach with multiple Machine Learning and Deep Learning models to gain insights and analyze which model provides better recommendation results and accuracy, which makes the application more reliable and robust.
4) *Recommendation results:* The results and predictions obtained from ML models are further stored in the database with all the necessary columns to organize the results. Since we use a multimodel approach, we store the results from all the models and compare them to provide the user with the most relevant, accurate, and reliable results. Storing recommendation results will ensure that the user experience viewing the videos of their interest is proper.

DocuSign Envelope ID: 72C90A03-080C-4BBA-B66D-D054BE574A31

Chapter 3. Technology Descriptions

3.1 Client Technologies

React Native

Applications operating on iOS and Android devices can be developed using React Native. Therefore it can be used for cross-platform applications using a unique codebase. Building software that is adaptable to various hardware platforms is known as cross-platform programming. Microsoft Windows, Linux, macOS, or any two of these operating systems can all be used by a cross-platform program. For the creation of interactive user interfaces, React utilizes HTML, CSS, and JavaScript. In contrast, React Native lets developers build mobile applications using native UI elements and APIs. A combination of JavaScript and JXL, a unique scripting language resembling XML, creates React Native. The system has the capacity to interact with both existing native app threads and JavaScript-based threads.

Because React Native doesn't render WebViews in its code, it differs from other cross-platform programming tools. It uses genuine native views and components to function. Platform-specific user interfaces give iOS and Android applications a native feel and a fluid UX. Our project creates an application using Expo and React Native. The same native view components produced when writing native code are rendered using the same declarative UI approach.

The application can be created once and run on numerous operating systems, including Android, iOS, and even the web. We built mobile applications with native user interfaces and device capabilities by reusing prior web development knowledge. Because most of the same code is reused, production is also sped up. Similar to React, React Native enables programmers to create declarative user interfaces in JavaScript, for which it internally builds a hierarchy tree

DocuSign Envelope ID: 72C90A03-080C-4BBA-B66D-D054BE574A31

of UI components, or what React refers to as a virtual DOM. React Native converts the virtual DOM into native views for mobile devices using platform native bindings that communicate with JavaScript application functionality, as opposed to ReactJS, whose output is intended for a browser.

3.2 Middle-Tier Technologies

Python with Django

A standardized method of giving data to other apps is through a REST API. The information is then available for use by those applications. APIs occasionally provide a means for additional applications to modify the data. A REST API request has a few important choices, including

- *GET*: The most popular option, which depends on the endpoint you access and any parameters you supply to return some data from the API
- *POST*: Adds a new entry to the database by creating it.
- *PUT:* Searches for a document at the specified URI that you supply. Update the current record if it already exists. Otherwise, make a new mark.
- *DELETE*: Deletes the item at the specified URI
- *PATCH*: Modify a record's specific attributes

An API typically serves as a portal to a database. The formatting of the answer and database queries are handled by the API backend. What you get is a static response of the resource you sought, typically in JSON format. Applications interact with one another and even with themselves using APIs. Serialization is the process of querying and converting numbers from tabular databases into JSON or another format.

DocuSign Envelope ID: 72C90A03-080C-4BBA-B66D-D054BE574A31

The main benefit of using the Django REST Framework is simple encoding. Python is used in Django to describe database models. Although you can write raw SQL, the Django ORM primarily manages all database migrations and searches. Even for a challenging undertaking, the Django community has many resources available to you. Routing, forms, authentication, and administration tool features are all included by default. There is no need to look for extra tools or be concerned about interoperability problems brought on by third-party tools. It is easier to concentrate on creating code by using basic user constructs, loops, and conditions. It is a sophisticated, optimized structure that is incredibly quick and dependable.

3.3 Data-Tier Technologies

Postgres Database

PostgreSQL, often simply known as Postgres, is a powerful, open-source relational database management system (RDBMS) that uses and extends the SQL language. Known for its robustness, advanced features, and strong standards compliance, Postgres supports both transactional (ACID-compliant) and non-transactional processing systems, and it has built-in support for full-text search and other modern database features.

It is capable of handling a wide range of workloads, from single-machine applications to large internet-facing applications with many concurrent users. Additionally, it offers various features like complex queries, foreign keys, triggers, updatable views, transactional integrity, and multiversion concurrency control.

The extensibility of PostgreSQL allows it to be used for much more than just a traditional relational database, as it can also function as a Postgres Database database or an object-oriented database.

DocuSign Envelope ID: 72C90A03-080C-4BBA-B66D-D054BE574A31

3.4 Machine Learning Technologies

PyTorch

A Python-based library called PyTorch offers flexibility as a deep-learning programming platform. PyTorch's workflow is similar to Python's numpy scientific computing library. PyTorch has additional benefits such as multi-GPU support, customized data drivers, and streamlined preprocessors. Arrays with multiple dimensions make up tensors. PyTorch supports different Tensor classes. It has the following advantages:

- *Simple API*: It is as straightforward as Python can be.
- *Support for Python*: As was already stated, PyTorch integrates seamlessly with the Python data science stack. You might be unable to tell it apart from numpy because it is that close.
- *Dynamic computation graphs:* Rather than predefined graphs with predetermined functionalities, PyTorch offers a structure that allows us to construct computational graphs as we go and even modify them in the middle of a program. This is useful when it is unknown how much memory will be needed to build a neural network.

DocuSign Envelope ID: 72C90A03-080C-4BBA-B66D-D054BE574A31

Chapter 4. Project Design

4.1 Client Design

4.1.1 Use Case Diagram

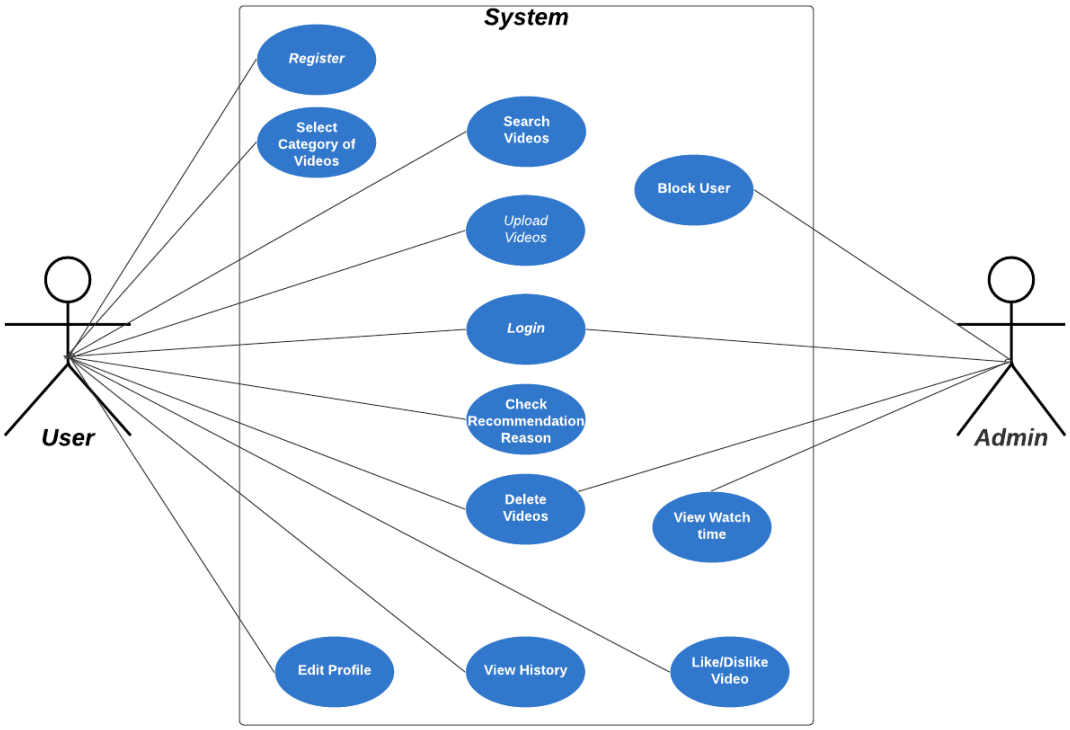


Fig 2. Use Case Diagram
Diagram Link - *Use case diagram.png*

A use case diagram is a tool for condensing information about a system and the people within it. The project consists of two actors, namely, the user and the administrator. Separate user interfaces are created for ease of operation for both users. The System shows the various use cases that correspond to the actors and the actions that they can perform.

DocuSign Envelope ID: 72C90A03-080C-4BBA-B66D-D054BE574A31

4.1.2 Frontend Architecture

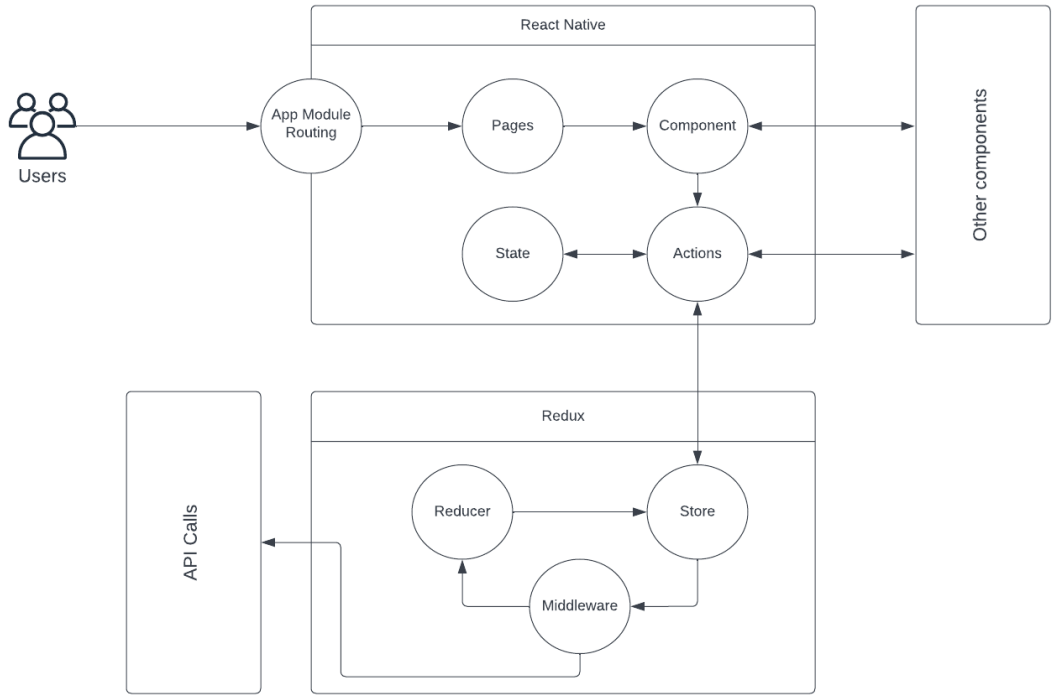


Fig 3. Frontend flow Architecture

The flow starts with the user accessing a particular page. The app module routing redirects the user to the corresponding page. The robust architecture provides a mechanism for easy debugging, making the application more maintainable. Let us understand the subcomponents of the architecture.

1) ***Component:*** A component splits the UI into reusable and independent chunks where each component can be isolated from other components. When the user is redirected to a component based on the routing module, the particular component is rendered on the DOM. Components can have references to other components as shown in the architecture. The other components can be related or unrelated to one another. This reference to other components ensures the reusability of components.

DocuSign Envelope ID: 72C90A03-080C-4BBA-B66D-D054BE574A31

2) ***State:*** It contains data about the components. The state of a component keeps changing, and during the change, the component is re-rendered. The change in the state can be a result of a user action or actions triggered by the system, which determines the behavior of the component and the way it renders.
3) ***Actions***: These are JavaScript objects sent using the dispatch() method. They must have a type property that indicates the corresponding action that needs to be carried out. They serve as a source of information for the store. If any change is required to the state, the corresponding change is only possible by actions.
4) ***Store:*** It holds the state of the whole application as a JavaScript object. It is created by using reducers. For our project, there is a single store for the entire application.
5) ***Reducers:*** It specifies the change in the state upon execution of any action. They execute action by taking payload as an argument and return a new state based on action. They are functions that take both state and action passed as arguments and return the new state.
6) ***Middleware:*** It can add a layer of functionality by monitoring all the actions sent to the reducers. Middleware can be used in modifying a particular action or removing the action. Asynchronous API calls can be made using the middleware.

DocuSign Envelope ID: 72C90A03-080C-4BBA-B66D-D054BE574A31

4.1.3 Firebase Authentication Architecture

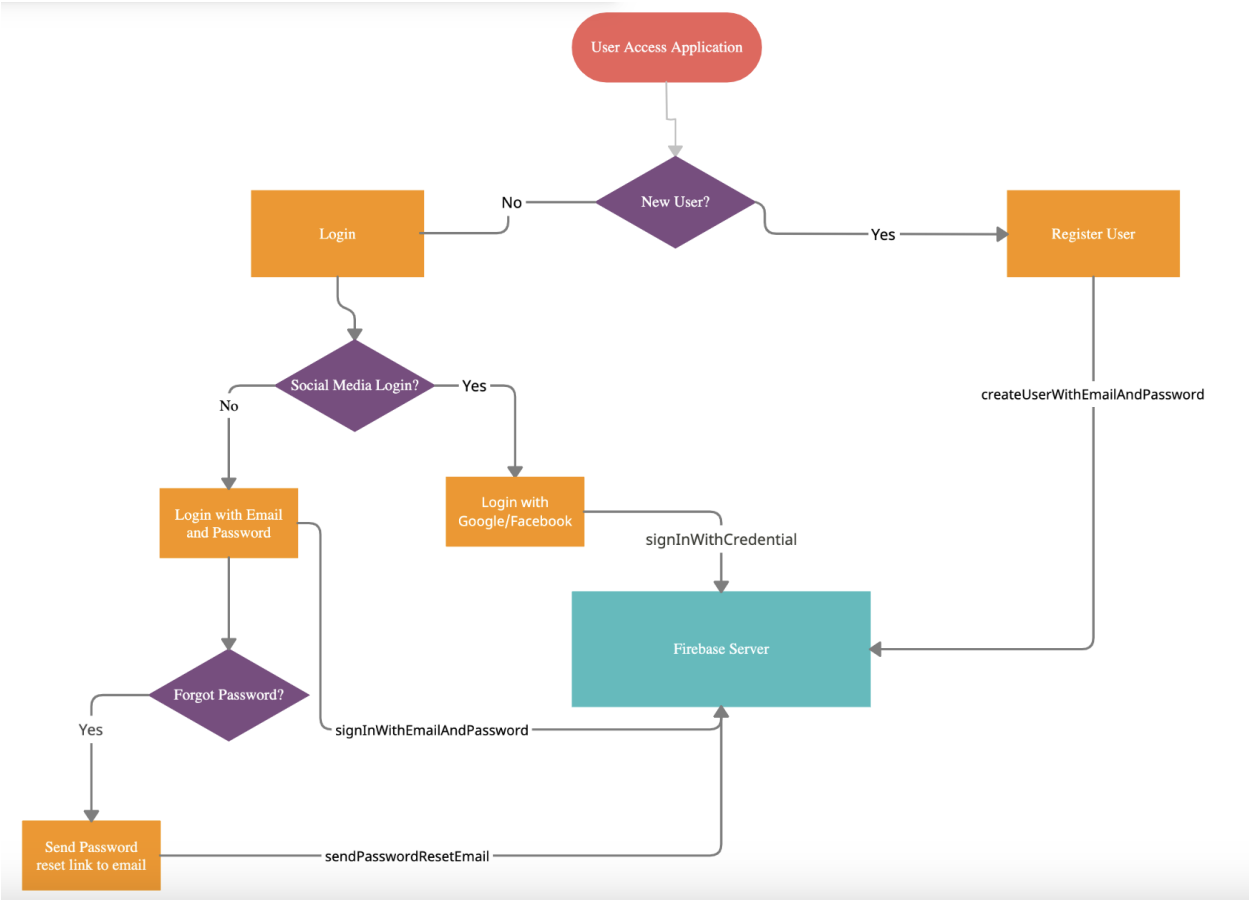


Fig 4. Authentication Architecture

When the user accesses the application, in case of a new user, the user must be registered using the Firebase Server. The method createUserWithEmailAndPassword() is called from the UI, passing the email id and the password. The user receives an email validation link from the Firebase server; upon successful validation, the user can now access the application. If the user is returning, they can log in using social media accounts such as Google or Facebook. The action calls the signInWithCredential() method while passing the social media account credentials. The user can log in with email and password using the signInWithEmailAndPassword() method, which takes email and password as inputs. If the user forgets the password, it can be reset using the sendPasswordResetEmail() method, which sends a password reset link to the email address entered, which can be used to reset and reaccess the application.

DocuSign Envelope ID: 72C90A03-080C-4BBA-B66D-D054BE574A31

4.1.4 Wireframe Design

1) Authentication

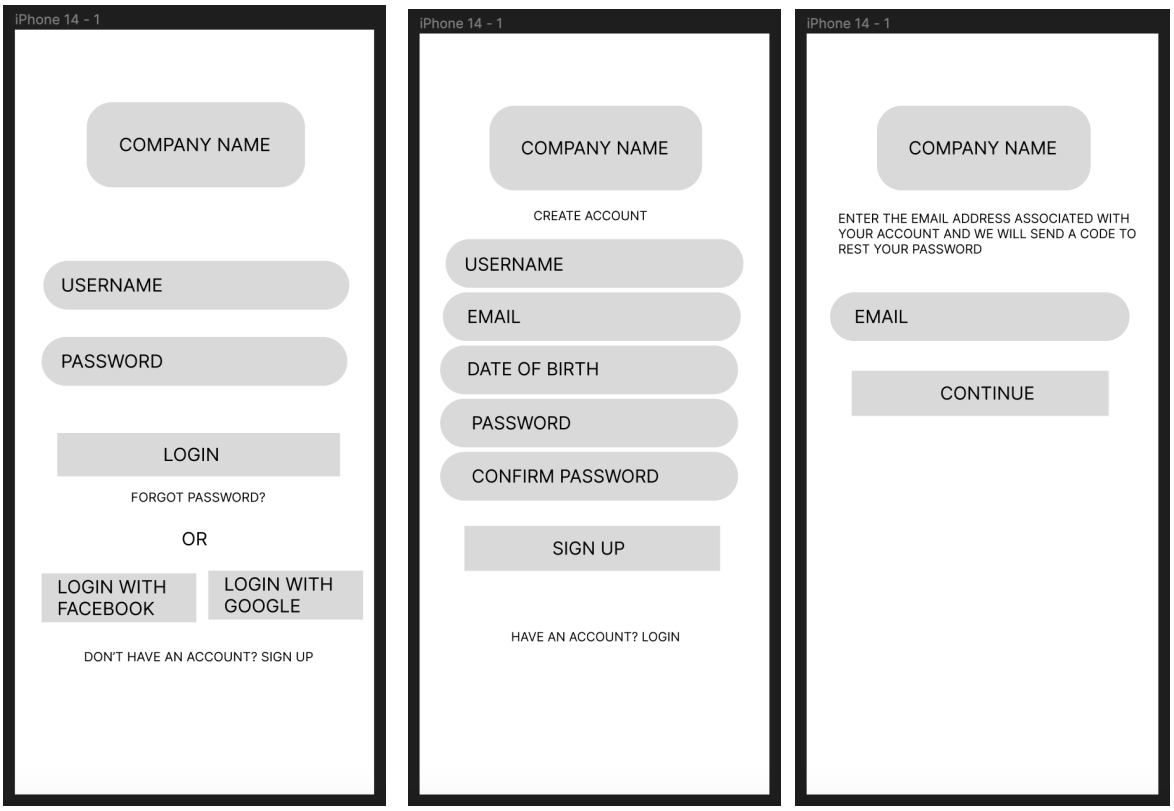


Fig 5. Authentication Wireframes

When the user accesses the application, the first page is the login page. Users can log in with Facebook/Google or provide the username and password set during registration. The screen also offers the option to sign up if the user has no account. Forgot Password option is present to reset the password. The second screen shows the Sign-Up form, which includes username, email, date of birth, password, and password confirmation. The user's date of birth determines the user's age for filtering out age-inappropriate content. The Sign-up page also has a button to navigate to the Login page.

DocuSign Envelope ID: 72C90A03-080C-4BBA-B66D-D054BE574A31

2) Viewing videos of interest

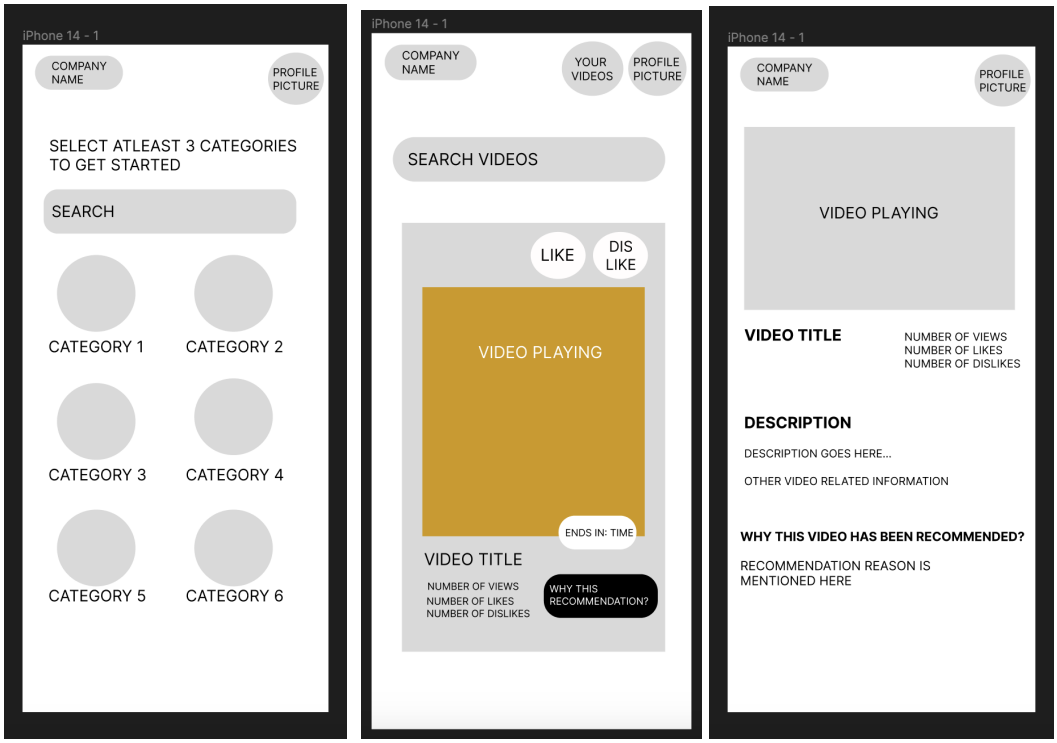


Fig 6. System Wireframes

After successfully logging in to the application, users are given multiple category options to select according to their preferences. A search bar is displayed to make the categories selection easier. After selecting the categories, the user can view the videos in those categories, and the following videos are recommended based on recommendation results stored in the database that ML models generated. The user can like a video that increases the recommendation for similar videos or dislike a video that does the vice versa. The user can also view the number of likes, dislikes, and views present for that particular video. The video auto-plays when it is present on the screen. There are options to navigate to the profile section, personal video upload section, or a section to view why the video has been recommended to the user. Upon clicking the "Why this Recommendation?" button, the user is navigated to the recommendation reason. This screen contains basic information about the video and why it has been recommended.

DocuSign Envelope ID: 72C90A03-080C-4BBA-B66D-D054BE574A31

3) Edit Profile

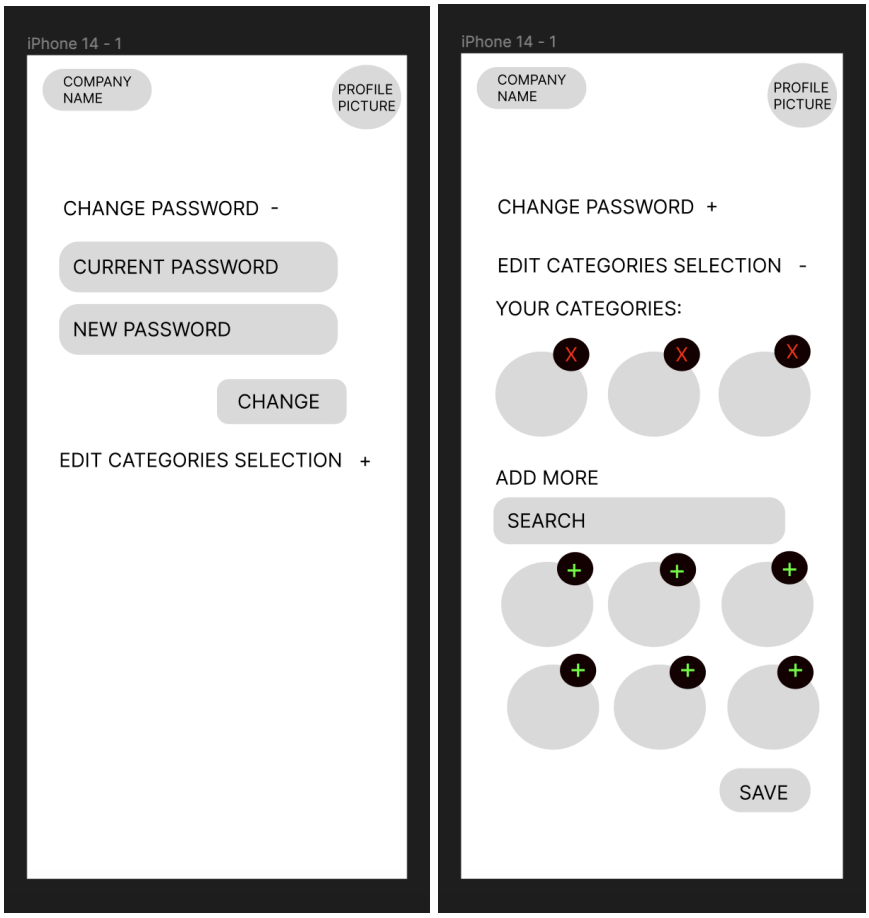


Fig 7. Edit Profile Wireframes

The user can click the profile picture icon to navigate to the edit profile page. The page is divided into two subsections which can be minimized or maximized. The first section is to change the password of the application. The current and new password is entered, and the user can change the password by clicking the change button. The second subsection is to edit the categories that are selected. The previously chosen categories are shown along with an option to remove those, and new categories can be added using the search bar and add buttons on categories. The save button will trigger the changes, updating the recommendation results.

DocuSign Envelope ID: 72C90A03-080C-4BBA-B66D-D054BE574A31

4) Upload Video

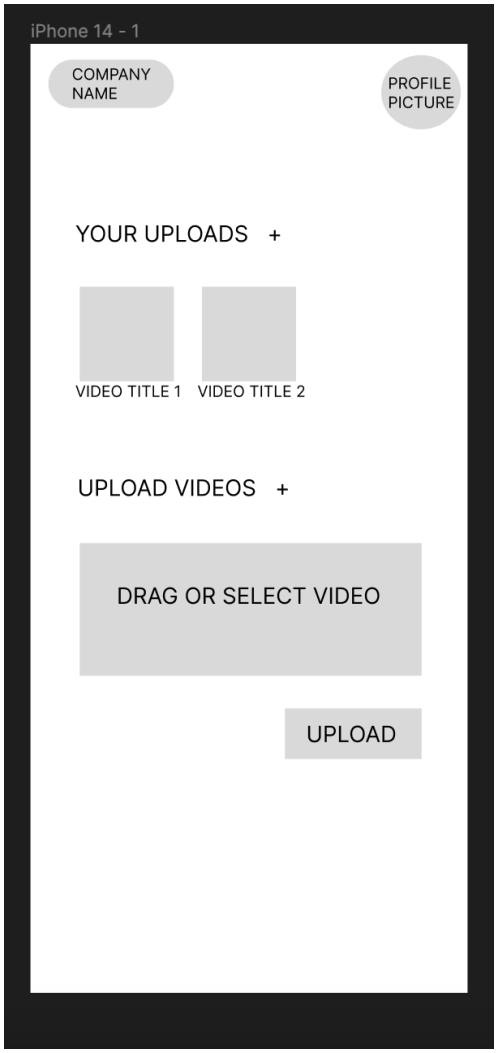


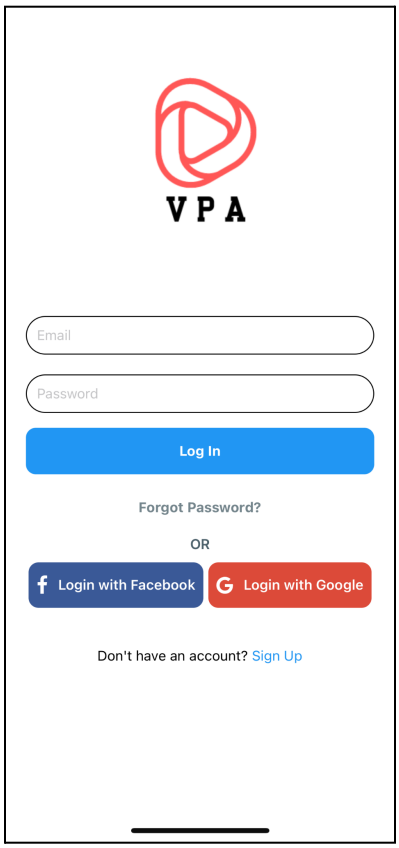
Fig 8. Upload videos wireframe

The user can upload a video by clicking on your videos button on the homepage. It is further divided into two subsections. The first section shows the videos that have been uploaded and approved. The second section shows a box to upload a video which will trigger a call to store the video in an S3 bucket.

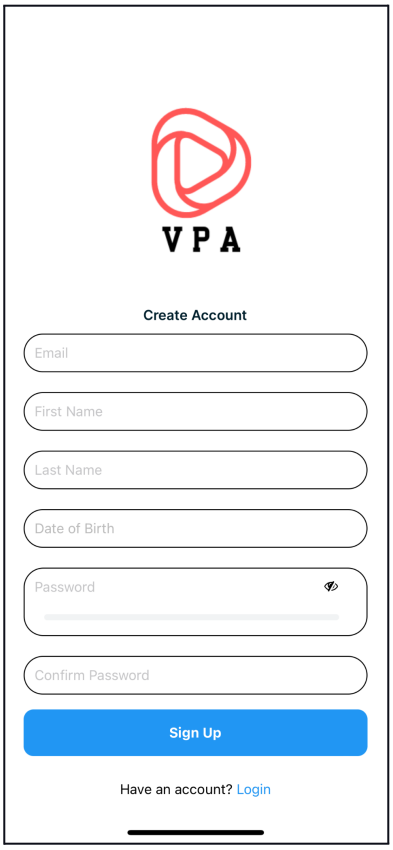
DocuSign Envelope ID: 72C90A03-080C-4BBA-B66D-D054BE574A31

4.1.5 UI Screenshots

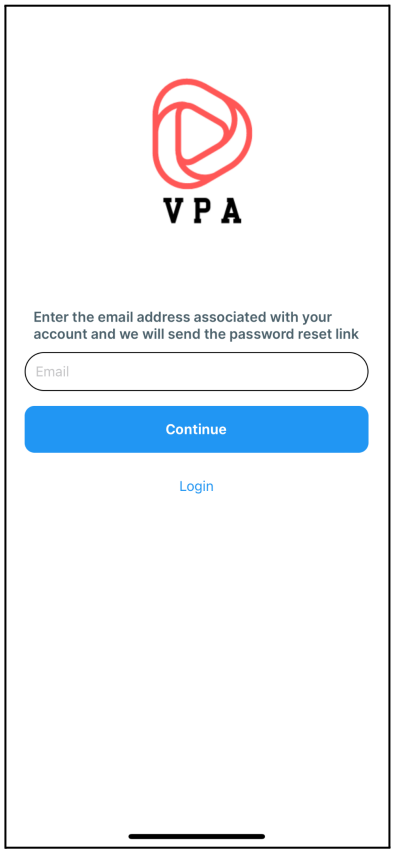
1) Authentication



Login Screen



Sign Up Screen

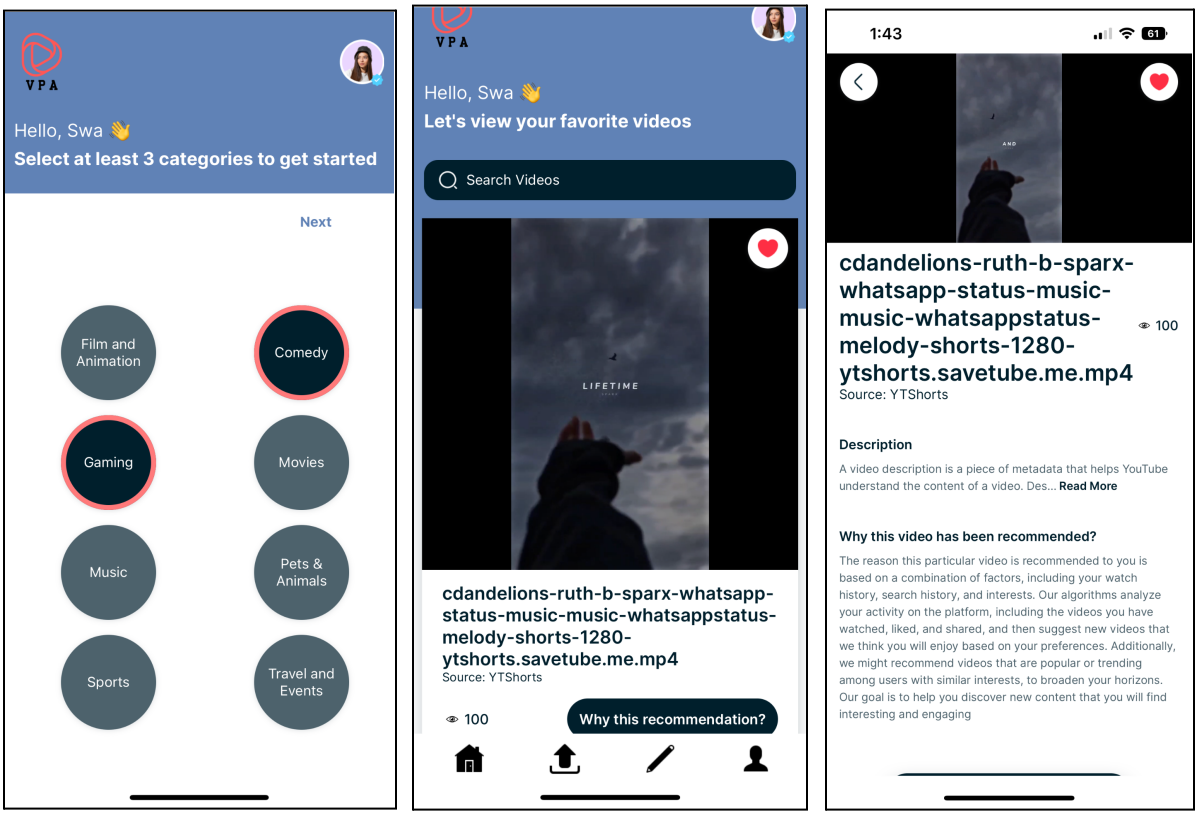


Reset Password Screen

Upon accessing the application, the initial screen that appears is the login page. To log in, users have the option to either use their Facebook/Google credentials or enter their previously set username and password during registration. Additionally, for those without an account, the login page also provides the option to sign up. In case a user forgets their password, there is a "Forgot Password" feature available to reset it. The following screen displays the Sign-Up form which includes fields for username, email, date of birth, password, and password confirmation. The user's date of birth is used to determine their age, which is then used to filter out age-inappropriate content. On the Sign-Up page, there is also a button that allows users to switch back to the Login page if they already have an account.

DocuSign Envelope ID: 72C90A03-080C-4BBA-B66D-D054BE574A31

2) Viewing videos of interest

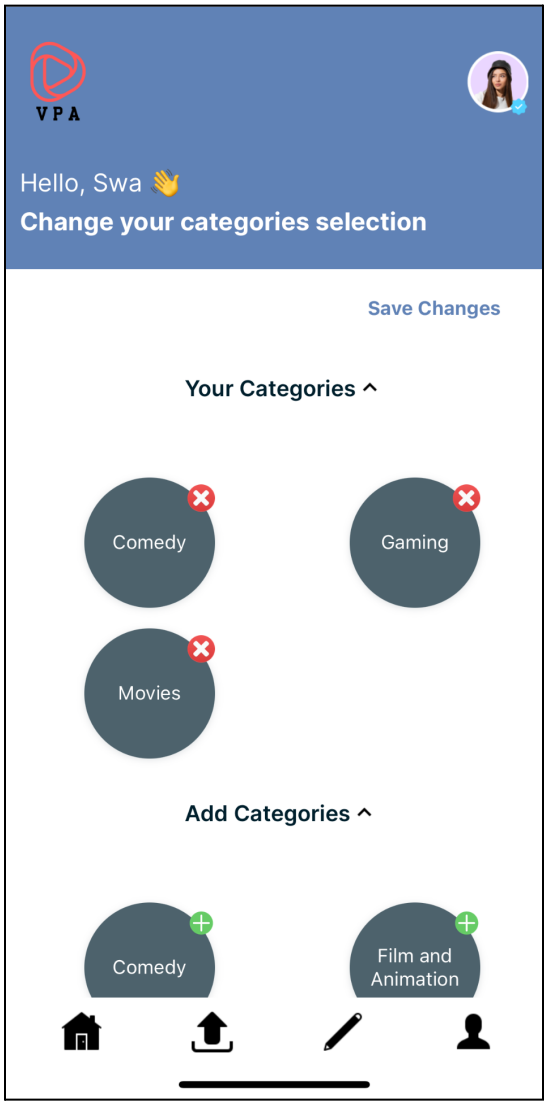


Categories Selection ***Home Page*** ***Recommendation Reason***

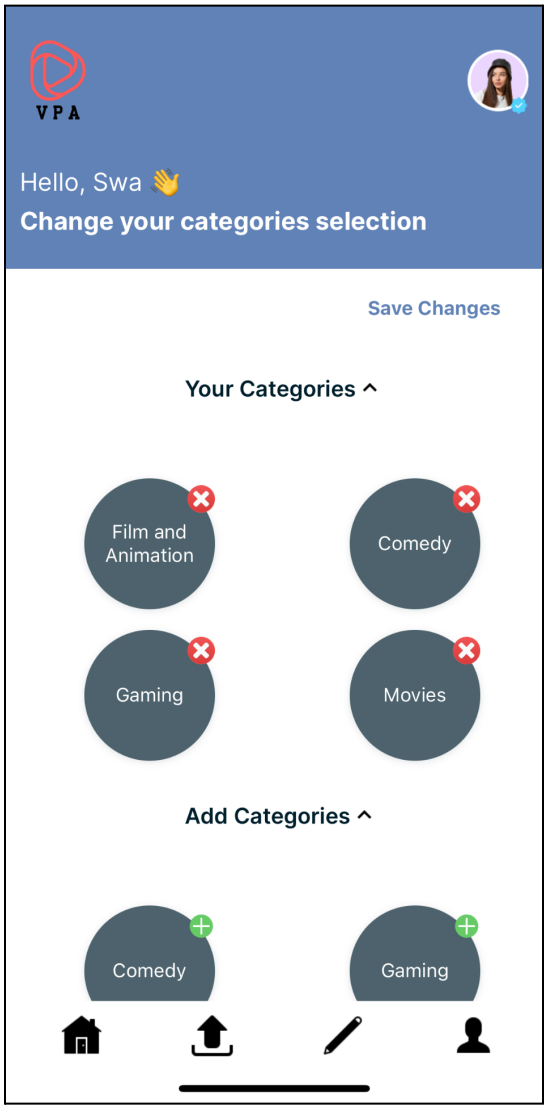
Once users have successfully logged in to the application, they are presented with multiple category options to choose from based on their preferences. To facilitate category selection, a search bar is displayed. Upon selecting the categories, users can view the videos available in those categories, and the application recommends additional videos based on the results stored in the database generated by machine learning models. Users have the ability to like or dislike a video, which influences the recommendation of similar videos. When a video is on the screen, it automatically plays. Users can navigate to different sections such as the profile, personal video upload, or a section that displays why a particular video has been recommended to them. By clicking the "Why this Recommendation?" button, users are taken to a screen that provides basic information about the video and why it has been recommended.

DocuSign Envelope ID: 72C90A03-080C-4BBA-B66D-D054BE574A31

3) Edit Categories



Edit Categories Selection

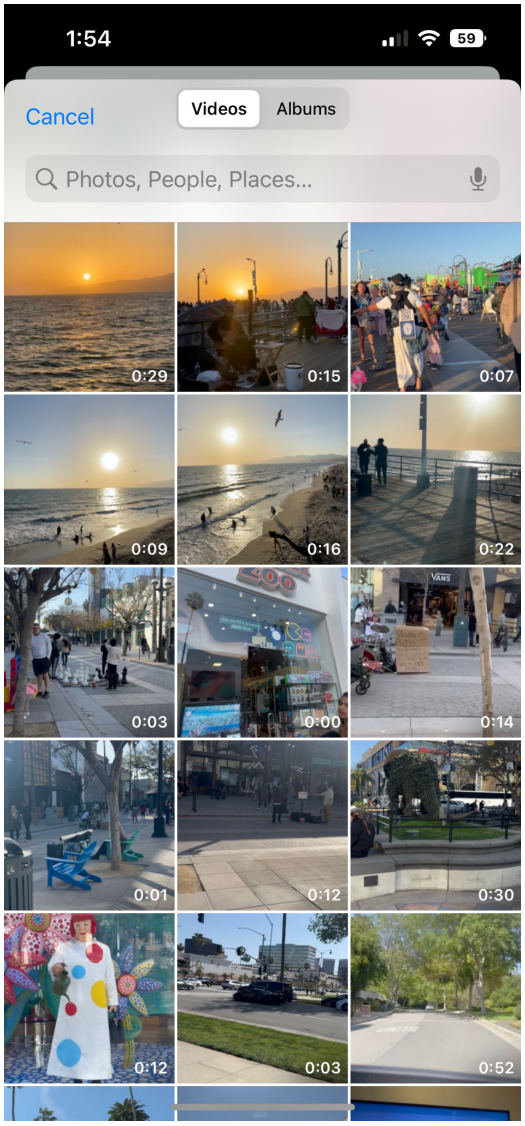


Added Film and Animation to Selected Categories

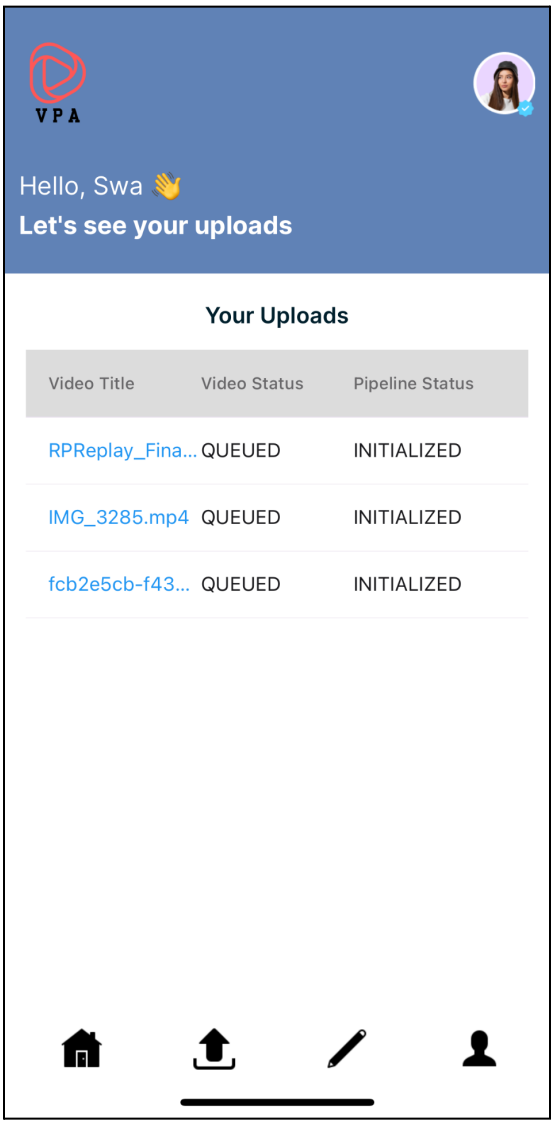
This section is designed to allow users to modify their category preferences. The application displays the previously selected categories and provides an option to remove them. Users can also add new categories using the Add button associated with each category. Once the desired changes have been made, users can save their modifications by clicking the "Save Changes" button. This will trigger an update to the recommendation results based on the new category preferences.

DocuSign Envelope ID: 72C90A03-080C-4BBA-B66D-D054BE574A31

4) Upload Video and Uploaded Video Information



Video Picker



Uploaded Video Status

To upload a video, users can click on the "Upload" button located on the homepage. The first image displayed presents the video picker, which enables users to select the desired video for upload. Once a video is selected, the upload process begins.

The second image displays the user's uploaded videos, along with their upload status and their position within the machine learning pipeline.

DocuSign Envelope ID: 72C90A03-080C-4BBA-B66D-D054BE574A31

5) UI Wireframe flow

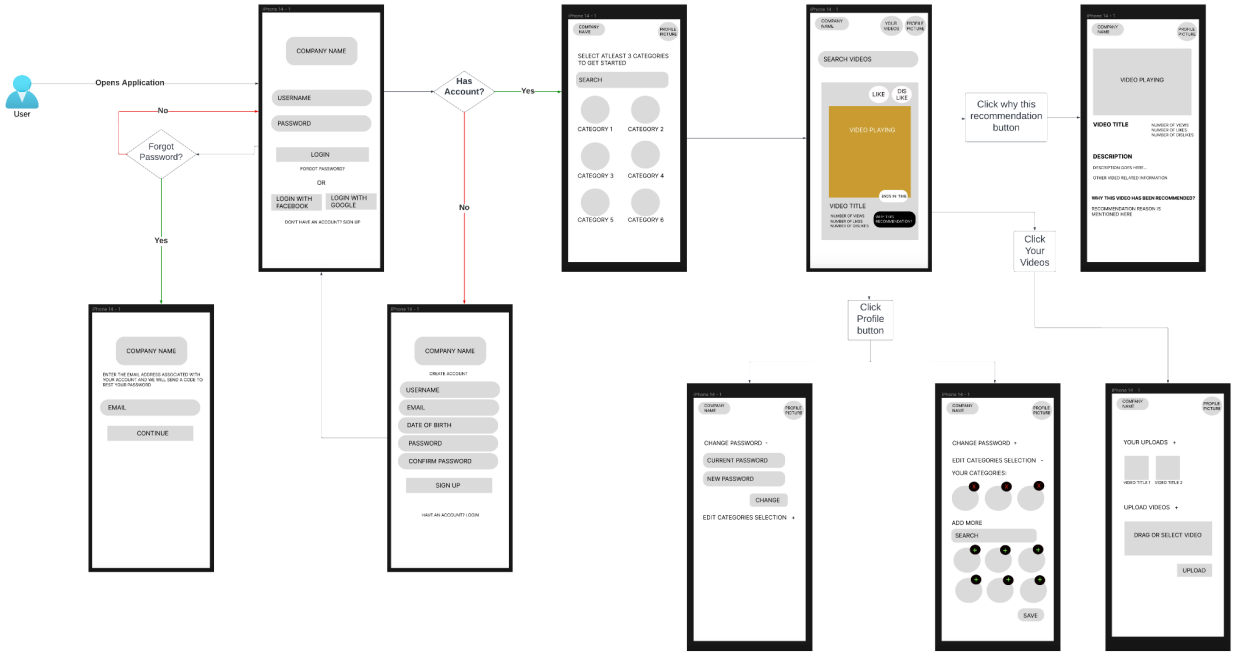


Fig 9. User Interface Flow
Diagram Link - [UI Flow]

Fig 6. shows the user flow throughout the application. It offers various screens with the application's buttons, content, and actions a user can perform. It shows the necessary steps for a user to complete a task they want to succeed in or to achieve a goal. It also shows the guidance provided to the user on each screen in the text that helps navigate the application seamlessly. Having a layout of UI wireframe flow also helps the user to understand the application better.

DocuSign Envelope ID: 72C90A03-080C-4BBA-B66D-D054BE574A31

4.2 Middle-Tier Design

4.2.1 Sequence Diagram

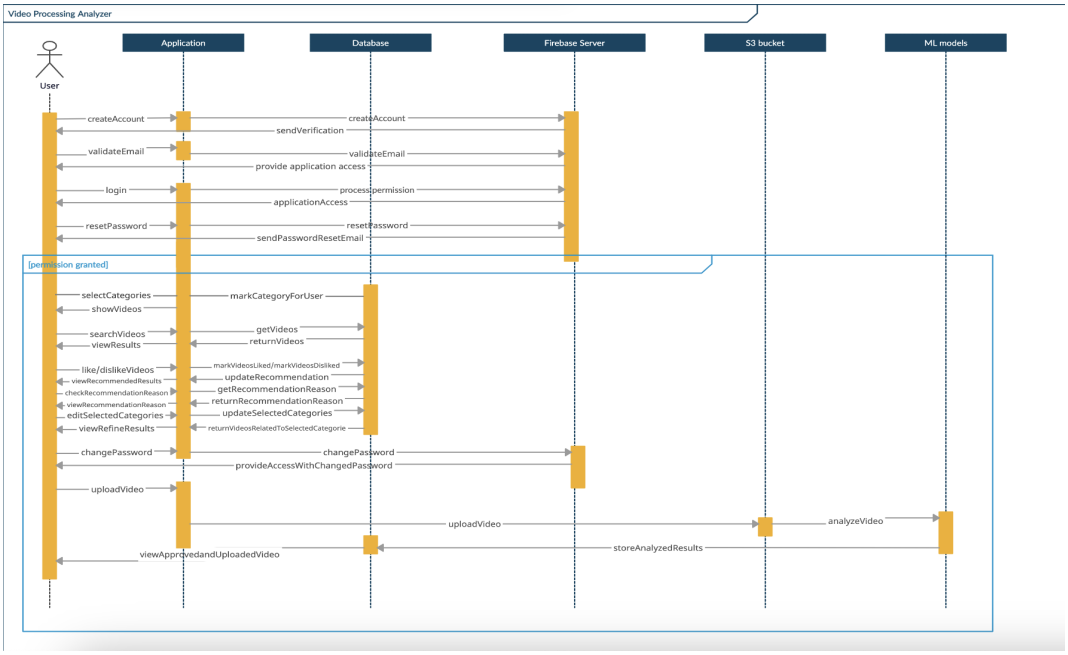


Fig 10. Sequence diagram
Diagram Link - Sequence Diagram

The user only interacts with the application. For account creation, login, and password reset functionalities, the application uses Firebase Server. Once the user submits the account creation request, an email is sent to the user for account verification. If the user forgets the password set for the application, the password reset option is sent to the user for validation. After the account is validated, the user can access the application's home page. Once they have permission to access the application, they can select their interest categories to start viewing the videos. They can also search the videos if they want to view something specific. There are options to like/dislike and view why the ML algorithms recommend a particular user to a specific category or video. The user also has the opportunity to change the password or edit the categories selected initially. The user can upload videos stored in the S3 buckets, which are further fed to the ML models for analysis. After successful analysis, the recommendation results are stored in the database, allowing users to view their uploaded videos.

DocuSign Envelope ID: 72C90A03-080C-4BBA-B66D-D054BE574A31

4.2.2 Activity Diagram

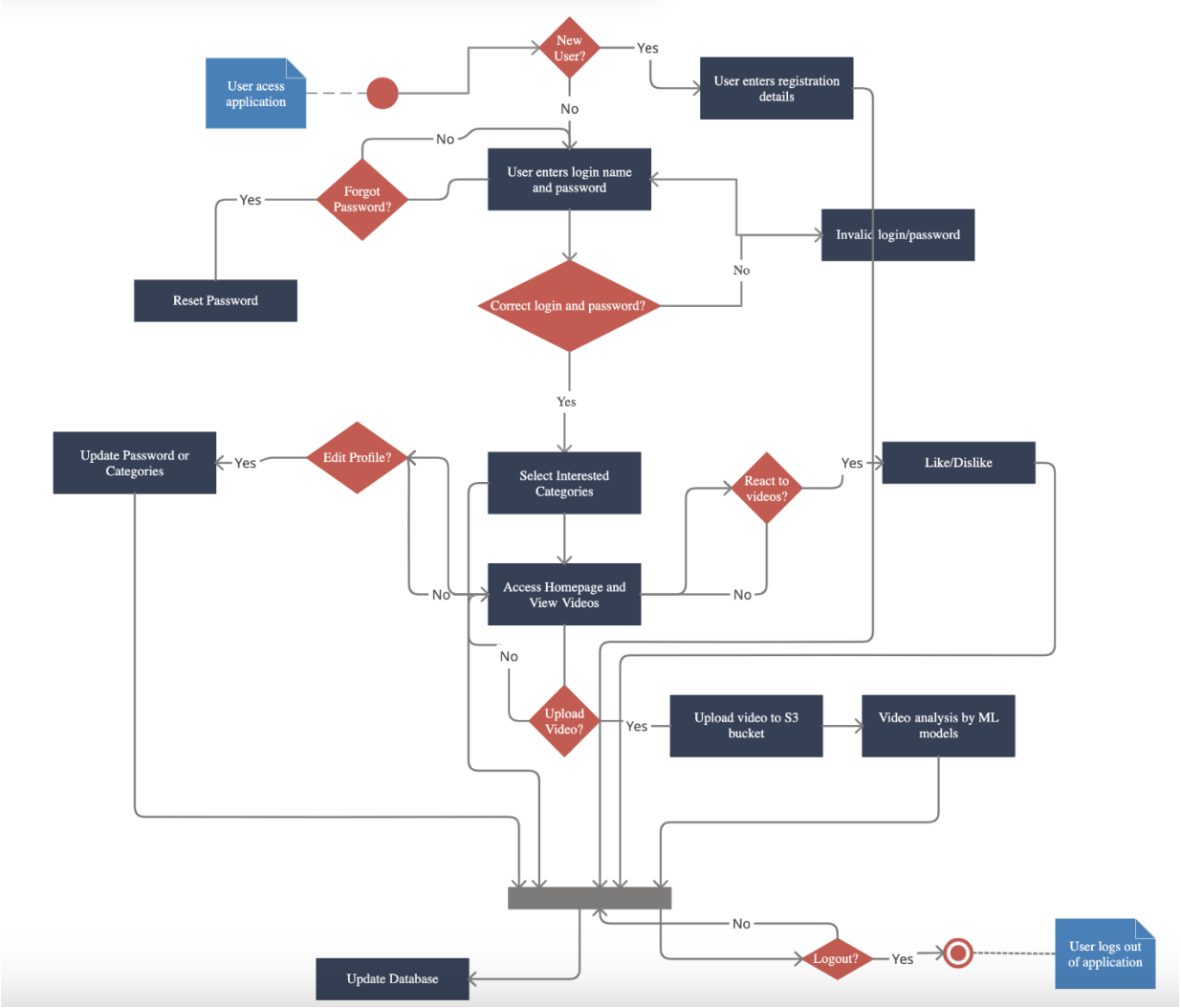


Fig 11. Activity diagram
Diagram Link - *ActivityDiagram.png*

This diagram explains the flow/activities a user can perform on the application. It shows the effective workflows of the application. It also shows the interaction between various application components such as frontend, backend, and database. The application has multiple systems, and the diagram aims to describe the flow from one system to another. Users can also use the activity diagram for reference to understand the project without having technical knowledge.

DocuSign Envelope ID: 72C90A03-080C-4BBA-B66D-D054BE574A31

4.3 Data-Tier Design

4.3.1 Class Diagram

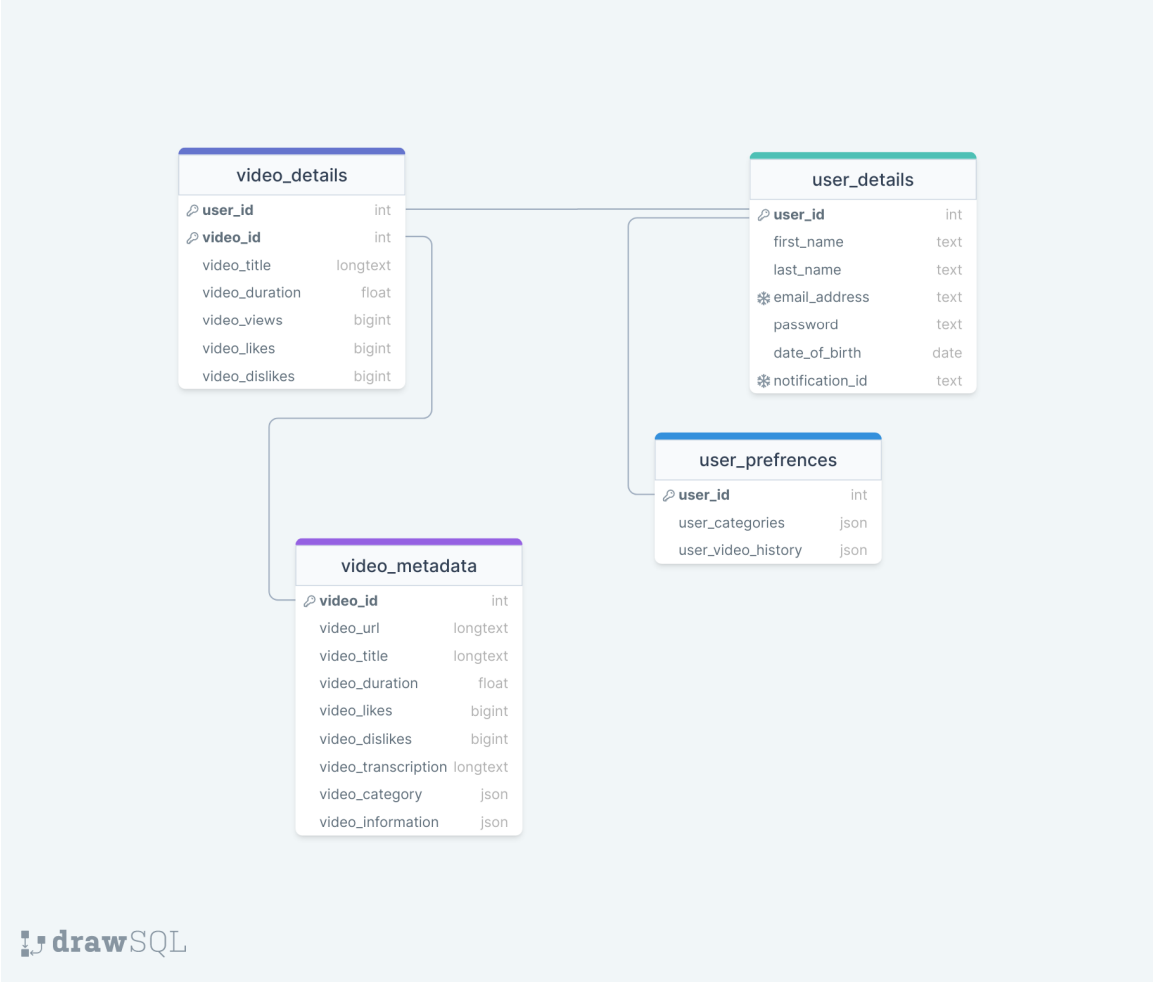


Fig 12. Class Diagram
Diagram Link - [Class Diagram]

This Diagram explains the database schema for the project, Currently there are mainly different tables for the project - video_details, video_metadata, user_details and user_preferences. video_id in the video_details is the foreign key to the video_metadata with 1:1 mapping, user_id in the user_preferences is the foreign key to video_details and user_details.

DocuSign Envelope ID: 72C90A03-080C-4BBA-B66D-D054BE574A31

4.4 Deployment Design

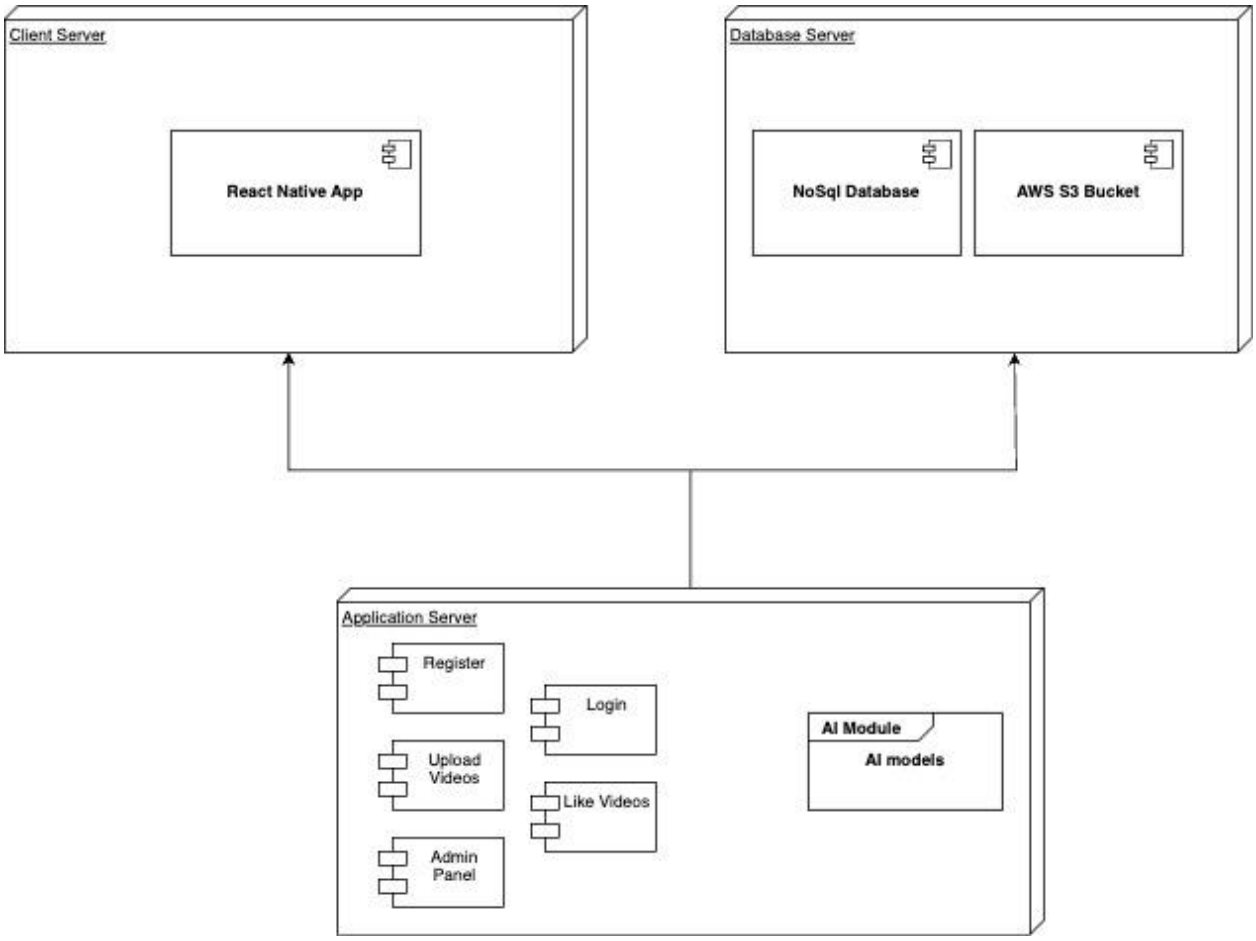


Fig 13. Deployment Diagram
Diagram Link - [Deployment Diagram]

This diagram explains the entire application, with its components, the nodes they run on, and their connections. The client server, database server, and application server are the deployment units. The client server is the unit whose component is the React Native app. The database unit has two components - Postgres database and S3 Bucket. The application server unit has various modules, such as register, login, upload videos, like videos, and the admin panel. It also consists of a module of AI models that assist in prediction for recommendation purposes.

DocuSign Envelope ID: 72C90A03-080C-4BBA-B66D-D054BE574A31

Chapter 5. Project Implementation

We managed the project using agile development approach. We have used GitHub for source code management, revision control, and team collaboration. In addition to offering performance and other helpful metrics to monitor and handle projects, GitHub is a free version control system.

5.1 Project Objectives

5.1.1 Functional Objectives

1) ***User Registration***

- *Input*: The user enters a unique username, password, email, and date of birth or can also log in through Facebook/Google
- *Output*: Receives an OTP on the entered email address, which can be used for verification or will be redirected to the home page using details obtained from the profile.
- *Processing*: Firebase will be used for user registration

2) ***User Login***

- *Input*: The user enters username and password, which were entered during the registration or can also log in through Facebook/Google
- *Output*: Gets access to the application
- *Processing*: Firebase verifies the user login

3) ***Categories Selection***

- *Input*: The user is given various video categories to choose from.

DocuSign Envelope ID: 72C90A03-080C-4BBA-B66D-D054BE574A31

- *Output*: The user can view recommended videos based on selected categories.
- *Processing*: Mark the selected category as interested for the particular user in the database for recommendation purposes.

4) View videos

- *Input*: The user can swipe up or down and watch or rewatch a video.
- *Output*: Video is played
- *Processing*: Video is buffered and rendered on the front end, fetched from the backend.

5) Search videos

- *Input*: The user can type the name of the video or category in a textbox.
- *Output*: Videos related to the entered name or category are displayed, or the "No results found" message is displayed.
- *Processing*: Search the video name or category in the database.

6) Like/Dislike Video

- *Input*: Video can be marked as liked/disliked using a button
- *Output*: The video will still finish playing, but the recommendation decreases for similar videos in case of dislike. The recommendation increases for similar videos if a video is liked.
- *Processing*: After liking/disliking a video, the video in the database is updated as liked/disliked for this particular user.

7) Upload Video

- *Input*: The user can upload a video from their profile page.
- *Output*: Video is uploaded and can be viewed

DocuSign Envelope ID: 72C90A03-080C-4BBA-B66D-D054BE574A31

- *Processing*: Video is stored in S3 buckets for analysis by the ML models for video understanding and shows “Flagged” with appropriate reasons for flagging the video in case of any misinformation/hatred/violence present in the video. Following the video analysis, the results are stored in the database.

8) Report Video

- *Input*: The user clicks a button to report a video and selects the reason for reporting
- *Output*: The video will be flagged with the appropriate comments
- *Processing*: Database record against the video is updated as reported which will stop similar video recommendations for the particular user

9) Edit Profile

- *Input*: The user can view the profile information by clicking the profile icon
- *Output*: Edit profile page opens
- *Processing*: The user can change the categories selection or the password for the application.

10) Recommendation Reason

- *Input*: The user can click on the “Why this recommendation” button to view the reason behind the recommendation of a video
- *Output*: Opens a tooltip showing the reason for the recommendation
- *Processing*: The recommendation reason is fetched which is determined by the explainable AI

DocuSign Envelope ID: 72C90A03-080C-4BBA-B66D-D054BE574A31

5.1.2 Non-functional Objectives

1) Security

Providing a dedicated barrier for authentication using Firebase to ensure only registered users can use the application.

2) Reliability

Each recommended video feed loads within 5 seconds.

3) Compatibility

The application must be compatible with a screen of any size and should adjust its layout according to the device used.

4) Availability

The application is monitored and maintained regularly to ensure that it is working as expected in the deployed environment at least 95% of the time.

5) Scalability

The load time across different browsers remains consistent.

6) Maintainability

Mean Time To Restore the System should be less than an hour in case of failure.

7) Localization

Any date or time displayed on the application should be localized to the country where the application is used.

8) Usability

Make the application self-explanatory and straightforward for the users.

DocuSign Envelope ID: 72C90A03-080C-4BBA-B66D-D054BE574A31

5.2 Client Implementation

There are various client implementation components that are broken down below for better understanding:

- *User Interface*: The first step in creating our application is to design the user interface. The app's simple and intuitive design makes it easy for users to navigate and create and watch videos. The user interface is designed to allow users to browse and view videos easily. The interface is intuitive and easy to navigate, with clear buttons and menus. The app's home screen displays a continuous stream of short videos, with each taking up almost the entire screen. Users can swipe up to move to the next video or swipe down to return to the previous video. The top of the screen displays a dropdown menu that allows users to switch between the home screen, search, and user profile screens.
- *User authentication*: Users must create an account or sign in to access the app's features. We achieve it using Firebase. The app uses social media login options such as Facebook and Google.
- *Video Uploading and Storage:* The user should be able to upload videos to the server and store it in a database. When a user uploads a video, the app should send the video file to the server and any metadata about the video (such as the title, description, and tags). The server then stores the video file in a database and generates a unique identifier for the video. The app can then use this identifier to display the video in the user's feed.
- *Video Player*: A video player that supports high-quality video streaming, an inbuilt library of React Native.
- *Video feed and recommendation system:* The app displays a continuous stream of short videos personalized to each user's interests and viewing history. When a user opens the

DocuSign Envelope ID: 72C90A03-080C-4BBA-B66D-D054BE574A31

app, the app retrieves a list of recommended videos from the server based on the user's interests and viewing history. As the user watches videos, the app sends feedback to the server about which videos the user likes and dislikes. The server uses this feedback to refine the list of recommended videos and provide a more personalized experience for each user.

The stages of execution on the client side are as follows:

- Create a Django REST API to handle video data. Create a Django app to define video models and use the Django REST framework to create API endpoints for fetching videos and recommendations.
- Create a React Native app to display the video recommendations. We use the react-native-fetch-blob library to fetch the video data from the Django API.
- Implement a video recommendation algorithm. A machine learning algorithm generates personalized video recommendations based on the user's viewing history.
- We use the Django API to fetch the video recommendations. We create a Django endpoint that takes a user ID as a parameter and returns a list of recommended videos based on the user's viewing history.
- We use the react-native-fetch-blob library to fetch the video recommendations from the Django API. We use the fetch API to request HTTP to the Django API and the react-native-fetch-blob library to download the videos and cache them locally on the device.
- Display the video recommendations in the React Native app. We use the react-native-cacheable-image component to display thumbnail images for the videos.

DocuSign Envelope ID: 72C90A03-080C-4BBA-B66D-D054BE574A31

- Personalize the recommendations based on the user's viewing history. To do this, we must keep track of the user's viewing history on the server side using the Django app and Postgres Database. We use this information to filter the video recommendations only to show videos that the user is being recommended.
- Implement caching to improve performance. Since fetching video recommendations from the server can be slow, you can implement caching on the client side to store the recommendations locally and avoid making unnecessary requests to the server. We use the react-native-cache-store library to cache the video recommendations on the device.

5.3 Middle-Tier Implementation

The implementation of the described functionality follows a structured approach, consisting of multiple steps to ensure its successful integration. These steps can be further divided into sub-tasks to fulfill all the requirements of the application. The essential steps for implementing the functionality are as follows:

1. Environment Setup and Dependency Installation: The initial step involves setting up the environment and installing the necessary packages and dependencies. This ensures a smooth development process and allows for testing the end-to-end flow with basic CRUD (Create, Read, Update, Delete) operations on the database.
2. User Profile Creation and Editing API: The backend service provides a POST API for users to create and edit their profiles. This API allows newly registered users to enter their performative information and preferences for video categories. The information is stored securely in the database.
3. User Details and Preferences Retrieval API: A GET API is implemented to fetch user details and video category preferences from the database. This API enables users to view

DocuSign Envelope ID: 72C90A03-080C-4BBA-B66D-D054BE574A31

and modify their stored information, ensuring a personalized and customizable user experience. Appropriate validation, such as date of birth and email verification, is applied to maintain data integrity and security.

4. Admin Dashboard and User Videos API: The backend service provides an admin dashboard interface accessible to the admin user. The dashboard allows the admin to view all the videos uploaded by active users of the video platform. Additionally, an API is implemented to enable the storage of user-uploaded videos. These videos are stored in a Digital Ocean Spaces bucket, while their unique IDs are stored in a Postgres database for efficient retrieval.
5. Video Recommendation API and ML Agent Interaction: To enhance user engagement, the backend service offers an API for the user interface to recommend videos based on user preferences and activity. The backend service acts as a communication bridge between the recommender agent and the user interface, facilitating seamless recommendations. The recommended videos are stored in the database, mapped to the user's unique ID, ensuring personalized video recommendations.

By following these implementation steps, the project ensures the seamless integration of the described functionality. The APIs are categorized based on their interaction with the user interface, admin dashboard, database, and ML agent. The backend service enables users to create and manage their profiles, provides admin access to user and video information, interacts with the ML agent for video understanding, and facilitates personalized video recommendations. This structured approach ensures a cohesive and user-centric application experience.

DocuSign Envelope ID: 72C90A03-080C-4BBA-B66D-D054BE574A31

5.4 Data-Tier Implementation

To ensure accurate and efficient functionality, the backend service of the project relies on various cloud services that facilitate seamless interaction between the user interface and the database. The specific cloud services utilized are as follows:

1. Digital Ocean Spaces Buckets: Digital Ocean Spaces buckets are utilized for storing videos uploaded by users. This cloud-based storage solution ensures secure and scalable storage of user-generated video content, providing reliable access and efficient management of video files.
2. Digital Ocean Droplets: Digital Ocean Droplets are employed to invoke the backend service's APIs. These Droplets serve as virtual machines, enabling the execution of API calls and facilitating seamless communication between the frontend user interface and the backend service. The usage of Droplets enhances the overall performance and responsiveness of the application.
3. Postgres Relational Database: For efficient data storage and management, a Postgres Relational Database is employed. This database solution is leveraged to store user preferences and details securely. The relational nature of Postgres allows for structured data storage, ensuring data integrity and facilitating complex querying operations for user-related information.

By integrating these cloud services into the project's backend infrastructure, the application benefits from reliable storage, efficient API invocation, and secure data management. This cloud-based approach optimizes system performance, enhances scalability, and ensures the effective handling of user-generated content and user-related data.

DocuSign Envelope ID: 72C90A03-080C-4BBA-B66D-D054BE574A31

5.5 Machine Learning Implementation

The implementation of the machine learning component in this project follows a structured approach, encompassing five key steps. Each step consists of sub-tasks aimed at achieving the successful deployment of the machine learning functionality.

1. Data Pipeline: The first step involves creating a robust data pipeline to facilitate the evaluation of machine learning models. This includes developing pipelines to efficiently read and extract relevant data from datasets and user uploaded videos..
2. REST API Development for Inference: In this step, REST APIs are developed to leverage the trained machine learning model weights for inference purposes. Specifically, a REST API template is created to accommodate several separate machine learning models, each designed to address distinct functionalities related to video understanding.
3. Inference using API's POST Requests: API's POST requests are utilized for video inference tasks, allowing the application to process and analyze stored videos. These requests are crucial for generating recommendations tailored to individual users' preferences. Videos stored in digital ocean spaces are fetched and utilized in the inference process to provide accurate recommendations.
4. Status Request using API's GET Requests: API's GET requests enable users to retrieve the status of their inference job on videos. This functionality allows users to monitor and track the progress of their video analysis, ensuring a transparent and informative experience.
5. Storage of Inference Results in Postgres Database: To ensure effective management and accessibility of machine learning model inference results, a pipeline is developed to store

DocuSign Envelope ID: 72C90A03-080C-4BBA-B66D-D054BE574A31

the outcomes in a Postgres Database. This database acts as a central repository for storing and managing the valuable insights obtained from the machine learning models.

By following these implementation steps, the project establishes a comprehensive framework for leveraging machine learning in video understanding. The data pipeline ensures efficient data processing, while REST APIs facilitate seamless inference and status tracking. Additionally, the integration of a Postgres Database enables efficient storage and retrieval of inference results, fostering enhanced data management practices.

DocuSign Envelope ID: 72C90A03-080C-4BBA-B66D-D054BE574A31

Chapter 6. Testing and Verification

6.1 Test Strategy

The project incorporates a range of testing methodologies to ensure the reliability and quality of the mobile application. Various types of testing, including unit testing, integration testing, and functional testing, are employed to thoroughly assess different aspects of the application's functionality. Unit testing focuses on testing individual components to validate their correctness and adherence to specifications. Integration testing verifies the seamless interaction between different modules and components. Functional testing evaluates the application's behavior and user interactions to ensure its intended functionality is achieved.

In addition to these testing approaches, performance testing is conducted specifically for the ML models used within the application. This testing assesses the models' performance, ensuring they meet the required speed, accuracy, and resource utilization criteria.

To effectively track and manage any identified bugs or issues, the Jira tool is utilized. Jira enables the project team to log, monitor, and prioritize bugs, providing a centralized platform for effective bug tracking and resolution. By utilizing Jira, the project ensures efficient bug management, enabling timely bug fixes and enhancing the overall application quality.

By employing a combination of testing methodologies and utilizing bug tracking tools like Jira, the project team ensures comprehensive evaluation of the mobile application's functionality and performance. This approach enables the identification and resolution of any issues or bugs, ultimately delivering a reliable and high-quality mobile application to the end-users.

DocuSign Envelope ID: 72C90A03-080C-4BBA-B66D-D054BE574A31

6.2 Unit and Functional Testing

The project employs a comprehensive testing approach to ensure the quality and reliability of both the frontend and backend components. For frontend unit testing, the Jest framework is utilized, enabling the testing of individual components to verify their functionality in isolation. This facilitates the identification of any potential issues or errors within the frontend codebase. On the backend, Django's built-in test class, known as "TestCase," is employed. Each method within this class tests a specific functionality, allowing for systematic and targeted backend testing.

In addition to unit testing, functional testing is conducted to evaluate the application's overall behavior and functionality. This includes testing essential user actions such as signup, login, video uploading, and liking videos. Functional tests in this project are performed manually, ensuring that the application's features and user interactions are thoroughly examined. To detect any bugs or inconsistencies, these functional tests are conducted during the early stages of the development lifecycle. This proactive approach helps identify potential issues at an early stage, allowing for timely resolutions and improved software quality.

By incorporating both unit and functional testing methodologies, the project aims to deliver a robust and reliable application. Unit testing ensures the correctness of individual components, while functional testing provides an understanding of the application's behavior and user experience. This comprehensive testing strategy enables the identification of bugs, ensures the application functions as intended, and enhances the overall quality and user satisfaction.

DocuSign Envelope ID: 72C90A03-080C-4BBA-B66D-D054BE574A31

6.3 Testing of ML Models

The testing phase of ML models is a crucial aspect of the project, ensuring the accuracy, reliability, and effectiveness of the developed models. The objective of testing is to evaluate the performance and generalization capabilities of the ML models on unseen data.

The testing process involves using carefully curated datasets that represent a diverse range of scenarios and cover different aspects of the problem domain. Performance metrics such as accuracy, precision, recall, F1-score, or mean Average Precision (mAP) are computed to assess the models' predictive abilities. Additionally, techniques such as cross-validation or holdout validation are employed to validate the models' performance and identify any potential overfitting or underfitting issues.

The testing phase also involves rigorous error analysis to understand the models' limitations and identify areas for improvement. Throughout the testing process, the ML models are fine-tuned and iteratively refined to achieve optimal performance. By conducting comprehensive testing, the project ensures the reliability and effectiveness of the ML models in providing accurate insights and supporting the application's video understanding capabilities.

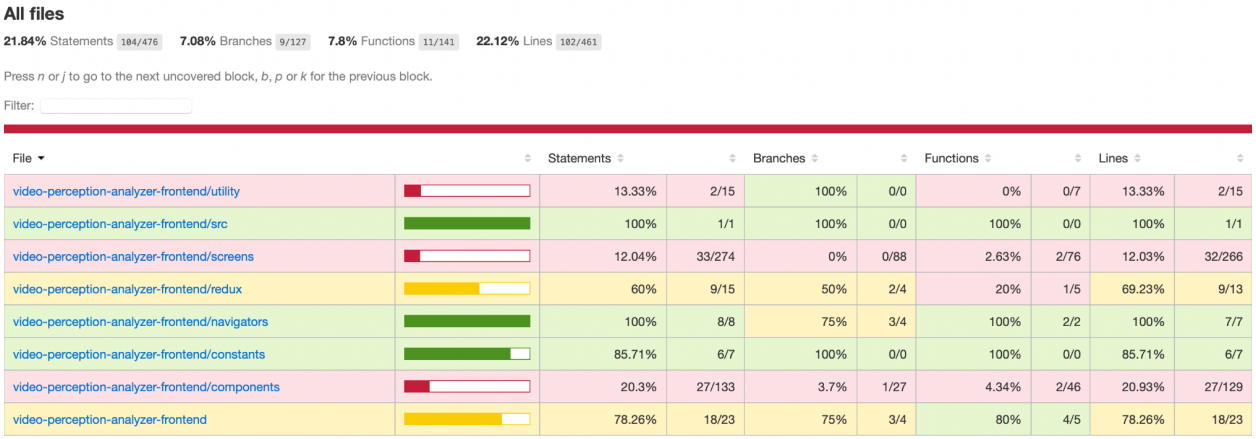
6.4 Integration Testing

Integration testing plays a crucial role in validating the routes and services between the React Native application and Django APIs within the project. This testing phase focuses on ensuring seamless communication and functionality between these components. To achieve this, integration tests are incorporated into the continuous integration Jenkins pipeline, enabling automatic execution as soon as changes are pushed to the project's GitHub repository. By integrating testing into the CI pipeline, the project ensures that any potential issues or

DocuSign Envelope ID: 72C90A03-080C-4BBA-B66D-D054BE574A31

inconsistencies in the routes and services are identified early on, allowing for prompt resolution and maintaining the overall reliability and stability of the application.

6.5 Test Results



All files

21.84% Statements 104/476 **7.08%** Branches 9/127 **7.8%** Functions 11/141 **22.12%** Lines 102/461

Press *n* or *j* to go to the next uncovered block, *b*, *p* or *k* for the previous block.

Filter:

File	Statements		Branches		Functions		Lines	
video-perception-analyzer-frontend/utility	13.33%	2/15	100%	0/0	0%	0/7	13.33%	2/15
video-perception-analyzer-frontend/src	100%	1/1	100%	0/0	100%	0/0	100%	1/1
video-perception-analyzer-frontend/screens	12.04%	33/274	0%	0/88	2.63%	2/76	12.03%	32/266
video-perception-analyzer-frontend/redux	60%	9/15	50%	2/4	20%	1/5	69.23%	9/13
video-perception-analyzer-frontend/navigators	100%	8/8	75%	3/4	100%	2/2	100%	7/7
video-perception-analyzer-frontend/constants	85.71%	6/7	100%	0/0	100%	0/0	85.71%	6/7
video-perception-analyzer-frontend/components	20.3%	27/133	3.7%	1/27	4.34%	2/46	20.93%	27/129
video-perception-analyzer-frontend	78.26%	18/23	75%	3/4	80%	4/5	78.26%	18/23

Fig 14. Code Coverage Report

This report shows how much of the code in each folder of our project is covered by the test file of the project. The table shows that almost 78% of the code is covered by tests, which leaves less room for errors in the project.

Following are the tests performed and their results:

Table 1. UI Test Results

Test Case	Test Description	Expected Result	Test Result
1.	Has 1 child	The component has 1 child component.	Pass
2.	Renders email and password input fields	The component renders the email and password input fields.	Pass
3.	Navigates to "Home" screen on button press	The component navigates to the "Home" screen when the "Sign up" button is pressed.	Pass
4.	Renders correctly	The component renders with the expected test ID.	Pass
5.	Login renders the correct	The login component renders consistently over time.	Pass

DocuSign Envelope ID: 72C90A03-080C-4BBA-B66D-D054BE574A31

Chapter 7. Performance and Benchmarks

7.1 Performance Plan

Performance testing is a non-functional testing technique that helps to understand the application's speed, scalability, and performance for a given load. This testing is one of the critical steps to finding applications' computational bottlenecks. It is one of the best ways to understand peak traffic events and the application performance on variable loads. This testing typically involves testing a large number of users simultaneously and responding instantly to user requests. Some of the key performance testing scenarios are.

1. Load Testing: This involves simulating heavy traffic to the application to see how it handles a large number of users simultaneously. The test should measure the application's response time, throughput, and resource utilization under different levels of load.
2. Stress Testing: This involves pushing the application to its limits to see how it behaves under extreme conditions. This test aims to identify the breaking point of the application and measure how it recovers from failures.
3. Scalability Testing: This involves testing the application's ability to scale horizontally or vertically to handle the increased load. The test should measure how the application responds to changes in load and how it scales to meet the demand.

DocuSign Envelope ID: 72C90A03-080C-4BBA-B66D-D054BE574A31

7.2 Benchmarking Strategies

Comparing an application's functionality and performance to those of related software or industry standards constitutes one of the benchmarking methodologies for that application. Some of the key benchmarking strategies are

1. Functionality benchmarking: This involves comparing the features and functionality of our application with similar existing applications and industrial standards. This can be done by testing the application against a set of predefined use cases and comparing the results with other applications.
2. Performance benchmarking: This involves comparing the performance of our application with similar existing applications and industrial standards. This can be done by measuring key performance metrics such as response time, and throughput, and comparing them with other applications.
3. User experience benchmarking: This involves comparing the user experience of our application with similar existing applications and industrial standards. This can be done by conducting user surveys, and usability tests, and identifying areas where the application can improve its user experience.

DocuSign Envelope ID: 72C90A03-080C-4BBA-B66D-D054BE574A31

Chapter 8. Deployment, Operations, Maintenance

8.1 Deployment Strategies

Deployment of the application is done via Jenkins' continuous deployment pipeline. The backend of the application is deployed on a DigitalOcean droplet. As soon as the code is pushed to the source code manager, which is GitHub in this application, the Jenkins pipeline triggers a build to start stage 1, which runs the test cases. If all the test cases get passed, the second stage in the pipeline, which is deployment, starts. The front end runs on the Expo Go App, which provides a platform to test the application without requiring to run builds locally.

8.2 Operational Dependencies

Following are the operation dependencies of our application :

1. Storage capacity of videos and AI models for recommendation
2. Device operating system: The mobile application relies on the device's operating system to function correctly. It requires specific versions of the operating system or particular device configurations to work properly.
3. Hardware components: The mobile application requires specific hardware specifications, such as screen resolution or RAM, to operate effectively.
4. Application programming interfaces (APIs): The mobile application relies on APIs provided by third-party services to access data or perform specific functions.
5. Cloud-based services: The mobile application may rely on cloud-based services for storage, hosting, and processing. The app that offers video streaming may use cloud-based hosting to deliver the videos to users.

DocuSign Envelope ID: 72C90A03-080C-4BBA-B66D-D054BE574A31

6. Software frameworks: The mobile application may rely on software frameworks to provide functionality or simplify development. The app offers authentication may use a software framework like Firebase.
7. Network connectivity: The mobile application relies on network connectivity to function correctly. It requires a reliable internet connection to access data or connect with other users.

8.3 Maintenance Strategies

Maintaining the application requires a range of strategies to ensure that it stays functional and effective over time. One critical aspect is to release regular updates that fix bugs, enhance performance, and add new features. This keeps the app fresh and engaging for users, and shows that the developers are committed to ongoing improvement.

Another important strategy is to gather feedback from users, both through direct communication and through analytics. This can help identify pain points in the user experience and identify areas for improvement. Regular testing is also essential to identify and fix issues before they become major problems. This includes testing for compatibility with new devices and operating systems, as well as testing for bugs and crashes.

Security is another key consideration, as mobile applications are vulnerable to a range of threats, from data breaches to malware attacks. Implementing measures such as data encryption, secure login, and regular security audits can help protect user data and increase user trust in the app.

DocuSign Envelope ID: 72C90A03-080C-4BBA-B66D-D054BE574A31

Cloud-based infrastructure can also be a valuable strategy for maintaining a mobile app. By using cloud-based hosting, storage, and processing services, developers can ensure that the app can scale to meet increasing demand and that it remains available to users.

Finally, documentation is critical for maintaining a mobile application. Detailed documentation that outlines the app's features, functionality, and technical specifications can help ensure that developers can easily maintain and update the app over time, even as the team changes and evolves. By implementing these strategies, developers can keep their mobile application running smoothly and providing value to users over the long term.

DocuSign Envelope ID: 72C90A03-080C-4BBA-B66D-D054BE574A31

Chapter 9. Summary, Conclusions, and Recommendations

9.1 Summary

The project focuses on developing a versatile application comprising ML and web application modules, built upon a robust three-layered architecture. The presentation layer empowers users with a user-friendly interface for seamless interaction, while the application layer efficiently processes API calls. The database layer serves as the bridge between software engineering and ML workflows, facilitating storage and management of application-specific data. The application supports both web and mobile interfaces, enhancing accessibility and user convenience.

The Postgres Database is utilized for storing and managing raw data and recommendation results obtained from the ML models. Python is employed for handling API calls, ensuring smooth communication between components, while React Native is leveraged for mobile application development, enabling the creation of a native-like experience.

The project adopts a multimodal approach, incorporating multiple ML and DL models to gain insights, perform thorough analysis, and compare results, ultimately providing users with relevant, accurate, and reliable recommendations. As a video streaming application, it offers a comprehensive set of features, including user registration and login, category selection, video search, video playback, like/dislike functionality, video upload capability, video reporting, profile editing, recommendation explanations, and a video feed with a sophisticated recommendation system.

DocuSign Envelope ID: 72C90A03-080C-4BBA-B66D-D054BE574A31

Designed with simplicity, intuitiveness, and user-friendliness in mind, the application ensures an enjoyable and straightforward experience for users. User authentication is handled through Firebase, guaranteeing secure access. Video storage and analysis are facilitated using Spaces buckets, ensuring efficient handling of multimedia content. The recommendation system enhances user engagement by delivering personalized video feeds based on users' viewing history and interests.

The project's objectives encompass both functional and non-functional aspects, ensuring the application's overall quality. Functional objectives encompass core functionalities, while non-functional objectives address critical aspects such as security, reliability, compatibility, availability, scalability, maintainability, localization, and usability. The client implementation focuses on designing an intuitive user interface, implementing user authentication, enabling seamless video uploading and storage, ensuring smooth video playback, and integrating an effective video feed with a robust recommendation system.

By combining innovative ML models, a user-friendly interface, and a range of essential features, the project aims to create an engaging and reliable video streaming application. The adherence to non-functional objectives ensures the application's security, reliability, compatibility, availability, scalability, maintainability, and user-friendliness. With a focus on client implementation, the project prioritizes intuitive user interfaces, efficient video management, and a personalized recommendation system to deliver a seamless and captivating user experience.

DocuSign Envelope ID: 72C90A03-080C-4BBA-B66D-D054BE574A31

9.2 Conclusions

The Video Perception Analyzer application represents a successful integration of various technologies to create a robust video analysis and recommendation system. By utilizing React-Native, Python, Django, and other cutting-edge tools, the application enables seamless video browsing, searching, and uploading. The integration of machine learning models for video classification, action detection, and content verification ensures accurate analysis and enhanced user safety.

With personalized recommendations based on user preferences, the application further enhances user engagement and satisfaction. The Video Perception Analyzer application sets the stage for a sophisticated and enjoyable video browsing experience while leveraging the power of machine learning for accurate analysis and recommendation.

9.3 Recommendations for Further Research

Video understanding has emerged as a critical research area with applications in various domains such as surveillance, autonomous driving, and video content analysis. While significant progress has been made in this field, there are still several avenues for further exploration and improvement. we present recommendations for future research in the domain of video understanding, focusing on key areas that can contribute to advancing the state-of-the-art techniques.

Temporal Modeling and Long-Term Dependencies: One important direction for future research is to enhance the modeling of temporal information in videos. Current methods primarily focus on short-term dependencies and local motion patterns. However, capturing long-term dependencies and complex temporal dynamics can significantly improve video

DocuSign Envelope ID: 72C90A03-080C-4BBA-B66D-D054BE574A31

understanding tasks, such as action recognition and video captioning. Exploring novel architectures, such as temporal convolutional networks or recurrent neural networks with attention mechanisms, can effectively capture and leverage long-term temporal information.

Multimodal Fusion for Improved Understanding: Another promising avenue for further research is the integration of multimodal information to enhance video understanding. Videos consist of not only visual content but also audio and textual cues. Leveraging the complementary nature of these modalities can provide richer context and improve performance in tasks like video captioning, emotion recognition, and video summarization. Developing effective fusion strategies, such as attention mechanisms and graph-based models, to effectively combine visual, audio, and textual modalities will be crucial.

Weakly Supervised and Self-Supervised Learning: Acquiring large-scale annotated video datasets is a challenging and time-consuming task. Therefore, exploring weakly supervised and self-supervised learning approaches can alleviate the need for extensive manual annotations. Weakly supervised methods can leverage weak labels or partial annotations, while self-supervised learning can exploit the inherent structure or temporal relationships within videos for training. Investigating innovative techniques, such as co-training with auxiliary tasks, unsupervised representation learning, or generative models, can enable more efficient training and better generalization in video understanding tasks.

Robustness to Variations and Adversarial Attacks: Videos often exhibit variations in lighting conditions, viewpoints, occlusions, and complex background clutter, which can

DocuSign Envelope ID: 72C90A03-080C-4BBA-B66D-D054BE574A31

adversely affect video understanding performance. Additionally, adversarial attacks aimed at fooling video understanding models pose a significant challenge. Therefore, research efforts should focus on developing robust and resilient video understanding models that can handle such variations and mitigate the impact of adversarial attacks. Techniques like domain adaptation, generative adversarial networks, and robust optimization can be explored to enhance model robustness.

Advancing the field of video understanding requires continuous exploration and innovation. By focusing on the recommendations outlined in this report - temporal modeling, multimodal fusion, weakly supervised and self-supervised learning, and robustness to variations and adversarial attacks - researchers can make significant contributions towards improving the accuracy, efficiency, and generalization capabilities of video understanding systems. These efforts will pave the way for the development of more intelligent and reliable video analysis applications in various domains.

DocuSign Envelope ID: 72C90A03-080C-4BBA-B66D-D054BE574A31

Glossary

Term	Definition
Artificial Intelligence (AI)	The integration of computer science and comprehensive datasets is what drives the field of artificial intelligence, allowing for effective problem-solving.
Machine Learning (ML)	Machine learning, a field within computer science and artificial intelligence, involves the utilization of algorithms and data to simulate human learning, with the aim of enhancing its precision over time.
Deep Learning (DL)	Deep learning is a technique within machine learning that enables computers to learn through examples, replicating how humans naturally acquire knowledge.
Convolutional Neural Network (CNN)	A Convolutional Neural Network, also known as a ConvNet or CNN, is a type of Deep Learning algorithm capable of receiving an input image and assigning significance, through learnable weights and biases, to different features/objects within the image, allowing it to distinguish between them.
Artificial Neural Network (ANN)	Algorithms modeled after the functioning of the human brain, known as Artificial Neural Networks (ANN), are utilized to predict complex patterns and forecast problems.
Vision Transformer (ViT)	ViT, short for Vision Transformer, is an image classification model that utilizes a Transformer-style structure to analyze image patches.
You Only Look Once (YOLO)	One of the most widely used object detection algorithms and model architectures is known as You Only Look Once (YOLO). Its popularity is largely due to implementation of a highly accurate neural network architecture that enables fast processing speeds.
Single Shot MultiBox Detector (SSD)	The bounding box regression technique SSD utilizes is based on MultiBox, a rapid method for proposing class-agnostic bounding box coordinates.
Region-Based Fully Convolutional Networks (R-FCN)	R-FCN employs score maps that are sensitive to position to resolve the conflict between translation-invariance in image classification and translation-variance in object detection.
Application Programming Interface (API)	An API, or application programming interface, is a method through which two or more computer programs can interact and exchange information with

DocuSign Envelope ID: 72C90A03-080C-4BBA-B66D-D054BE574A31

	one another.
HyperText Markup Language (HTML)	HTML, or HyperText Markup Language, is the established markup language utilized for creating documents that are intended for display in a web browser.
Cascading Style Sheets (CSS)	Cascading Style Sheets, also known as CSS, is a type of style sheet language that is employed to define the layout and presentation of a document written in a markup language like HTML or XML.
Extensible Markup Language (XML)	XML, or Extensible Markup Language, is both a file format and a markup language utilized for the storage, transmission, and reconstruction of various types of data.
Hypertext Transfer Protocol (HTTP)	HTTP, or Hypertext Transfer Protocol, is a protocol at the application layer that facilitates data transfer between networked devices and operates on higher layers of the network protocol stack.
Postgres Database (not only SQL)	Postgres Database offer advantages for storing and retrieving data that is modeled using structures other than the tabular relationships employed by relational databases.
User Interface (UI)	UI, or User Interface, pertains to the visual components such as screens, buttons, toggles, and icons, among others, that you engage with when using an electronic device, website, or application.
User Experience (UX)	UX, or User Experience, encompasses your complete interaction with the product, including your emotional response to the interaction.
Document Object Model (DOM)	The Document Object Model, or DOM, is an interface that is platform-agnostic and language-independent, which interprets an XML or HTML document as a hierarchical structure where every node represents an object denoting a particular segment of the document. This logical hierarchy of nodes in the DOM corresponds to the actual tree-like structure of the document.
REpresentational State Transfer (REST)	REST, which stands for REpresentational State Transfer, is an architectural design that establishes a set of conventions for intercommunication between computer systems on the web. These standards simplify the process of exchanging data between different systems.
JavaScript Object Notation (JSON)	JSON is a data interchange and open standard file format that leverages human-readable text to store and exchange data objects containing arrays and attribute-value pairs.

DocuSign Envelope ID: 72C90A03-080C-4BBA-B66D-D054BE574A31

Structured Query Language (SQL)	Structured Query Language (SQL) is a programming language that adheres to a standardized set of rules and is utilized to manage relational databases and conduct various operations on their data.
Object Relational Mapping (ORM)	Object-Relational Mapping (ORM) is a method that enables querying and manipulation of data in a database using an object-oriented approach.
Binary Javascript Object Notation (BSON)	BSON is a binary-encoded serialization of JSON documents that supports additional non-JSON-native data types, such as binary data and dates, as optional extensions.
Graphics Processing Units (GPU)	A graphics processing unit (GPU) is a dedicated electronic circuit that is designed to rapidly manipulate and modify memory in order to accelerate the generation of images in a frame buffer for output to a display device.
One-time password (OTP)	A single-use password, also referred to as a one-time PIN, dynamic password, or one-time authorization code, is a password that can only be used for a single login session or transaction on a digital device or computer system.

DocuSign Envelope ID: 72C90A03-080C-4BBA-B66D-D054BE574A31

References

1. K. O'Shea and R. Nash, "An introduction to convolutional neural networks," 2015. [Online]. Available: https://arxiv.org/abs/1511.08458

2. A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly, J. Uszkoreit, and N. Houlsby, "An image is worth 16x16 words: Transformers for image recognition at scale," 2020. [Online]. Available: https://arxiv.org/abs/2010.11929

3. A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention is all you need," 2017. [Online]. Available: https://arxiv.org/abs/1706.03762

4. M. D. Zeiler and R. Fergus, "Visualizing and understanding convolutional networks," 2013. [Online]. Available: https://arxiv.org/abs/1311.2901

5. C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," 2014.

6. K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," 2015.

7. K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," 2015.

8. G. Huang, Z. Liu, L. van der Maaten, and K. Q. Weinberger, "Densely connected convolutional networks," 2018.

9. J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," 2015. [Online]. Available: https://arxiv.org/abs/1506.02640

10. W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. C. Berg, "SSD: Single shot MultiBox detector," in Computer Vision – ECCV 2016. Springer International Publishing, 2016, pp. 21–37. [Online]. Available: https://doi.org/10.1007%2F978-3-319-46448-0 2

11. J. Dai, Y. Li, K. He, and J. Sun, "R-fcn: Object detection via region-based fully convolutional networks," 2016. [Online]. Available: https://arxiv.org/abs/1605.06409

12. R. M. Schmidt, "Recurrent neural networks (rnns): A gentle introduction and overview," 2019. [Online]. Available: https://arxiv.org/abs/1912.05911

13. Yuming Hua, Junhai Guo and Hua Zhao, "Deep Belief Networks and deep learning," Proceedings of 2015 International Conference on Intelligent Computing and Internet of

DocuSign Envelope ID: 72C90A03-080C-4BBA-B66D-D054BE574A31

Things, 2015, pp. 1-4, doi: 10.1109/ICAIOT.2015.7111524.

14. L.-C. Chen, G. Papandreou, I. Kokkinos, K. Murphy, and A. L. Yuille, “Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs,” 2016. [Online]. Available: https://arxiv.org/abs/1606.00915

15. G. Lin, A. Milan, C. Shen, and I. Reid, “Refinenet: Multi-path refinement networks for high-resolution semantic segmentation,” 2016. [Online]. Available: https://arxiv.org/abs/1611.06612

16. F. Yu and V. Koltun, “Multi-scale context aggregation by dilated convolutions,” 2015. [Online]. Available: https://arxiv.org/abs/1511.07122

17. Feichtenhofer, C., Pinz, A., & Zisserman, A. (2019). Detect to track and track to detect. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (pp. 5388-5397).

18. Carreira, J., & Zisserman, A. (2017). Quo vadis, action recognition? A new model and the kinetics dataset. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (pp. 4724-4733).

19. Tran, D., Bourdev, L., Fergus, R., Torresani, L., & Paluri, M. (2015). Learning spatiotemporal features with 3D convolutional networks. In Proceedings of the IEEE International Conference on Computer Vision (ICCV) (pp. 4489-4497).

20. Wang, L., Xiong, Y., Wang, Z., Qiao, Y., Lin, D., Tang, X., & Van Gool, L. (2016). Temporal segment networks: Towards good practices for deep action recognition. In European Conference on Computer Vision (ECCV) (pp. 20-36).

21. Simonyan, K., & Zisserman, A. (2014). Two-stream convolutional networks for action recognition in videos. In Advances in neural information processing systems (pp. 568-576).

22. Wang, X., Girshick, R., Gupta, A., & He, K. (2018). Non-local neural networks. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (pp. 7794-7803).

23. Feichtenhofer, C., Fan, H., Malik, J., & He, K. (2019). Slowfast networks for video recognition. In Proceedings of the IEEE International Conference on Computer Vision (ICCV) (pp. 6202-6211).

24. Girdhar, R., Carreira, J., Doersch, C., & Zisserman, A. (2019). Video action transformer network. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (pp. 244-253).

DocuSign Envelope ID: 72C90A03-080C-4BBA-B66D-D054BE574A31

25. Jiang, Y.-G., Liu, J., Roshan Zamir, A., Toderici, G., Laptev, I., Shah, M., & Sukthankar, R. (2019). Thumos challenge: Action recognition with a large number of classes. http://crcv.ucf.edu/THUMOS14/

26. Kay, W., Carreira, J., Simonyan, K., Zhang, B., Hillier, C., Vijayanarasimhan, S., ... & Zisserman, A. (2017). The kinetics human action video dataset. arXiv preprint arXiv:1705.06950.