



Final Report

VERITAS

Team Members:

- Hanzallah Azim Burney - 21701829 - CS
- Abdul Hamid Dabboussi - 21701076 - CS
- Mohamad Fakhouri - 21701546 - CS
- Sayed Abdullah Qutb - 21701024 - CS
- Hassan Raza - 21701811 - CS

Supervisor: Ercument Çiçek

Innovation Expert: Mustafa Sakalsız

Table of Contents

1.0 Introduction	3
2.0 Requirements Details	3
2.1 Functional Requirements	3
2.2 Non-Functional Requirements	4
2.2.1 Accessibility	4
2.2.2 Usability	4
2.2.3 Compatibility	4
2.2.4 Scalability	4
2.2.5 Performance	4
2.2.6 Extendability	5
2.2.7 Accuracy	5
2.2.8 Reliability	5
2.2.9 Testing and Error-Handling	5
2.2.10 Cost	5
3.0 Final Architecture and Design Details	6
3.1 Overview	6
3.2 Subsystem Decomposition	6
3.3 Hardware/Software Mapping	8
3.4 Persistent Data Management	8
4.0 Development/Implementation Details	9
4.1 Deep Learning Model	9
4.1.1 Data preparation	9
4.1.2 Data preprocessing	9
4.1.3 Model Description & Training	9
4.2 Backend and Frontend	12
4.2.1 Backend	12
	1

4.2.2 Frontend	13
5.0 Testing Details	14
6.0 Maintenance Plan and Details	16
6.1 Maintenance plan of the backend	16
6.2 Maintenance plan of the frontend	17
6.3 Maintenance plan of the machine learning model	18
7.0 Other Project Elements	18
7.1 Consideration of Various Factors in Engineering Design	18
7.2 Ethics and Professional Responsibilities	20
7.3 Judgements and Impacts to Various Contexts	21
7.4 Teamwork Details	21
7.4.1 Contributing and functioning effectively on the team	21
7.4.2 Helping creating a collaborative and inclusive environment	22
7.4.3 Taking lead role and sharing leadership on the team	22
7.4.4 Meeting objectives	22
7.5 New Knowledge Acquired and Applied	23
8.0 Conclusion and Future Work	24
9.0 Glossary	25
10.0 References	26
11.0 Appendix - User Manual	27
11.1 Introduction	27
11.2 Login	27
11.3 New User Registration	28
11.4 Veritas Home Page	29
11.5 Classification of Claims of Video	30
11.6 Saved Facts	31

1.0 Introduction

Social media platforms such as Twitter, Facebook, and YouTube are seeing a rise in the spread of fake news and misinformation on their platforms. According to research, Facebook users engage with misinformation 70 million times per month on average [1]. The rise of fake news is problematic for societies. It has the potential to sway public opinion, promote conspiracy theories, and instill fear, thereby eroding confidence in public institutions, and in democracy.

Therefore, our project aims to combat the spread of political lies and misinformation in order to better inform the public about what politicians are saying. To this end, we will build either a YouTube wrapper website that would perform fake news labelling and fact checking on live-streamed political speeches. The main targets of this project are political speeches and presidential debates from the United States of America. In order to accomplish this task, multiple ML/NLP papers on fake news will be consulted. A fake news/fact checking classification model will be built and trained on an appropriate dataset. Once a good enough classifier is achieved based on the metrics deemed ideal for such a problem, a platform would be built to use this classifier on political speeches streamed on YouTube. The model would then inform the viewer of any wrong claims being said by the speaker through various tags and captions.

2.0 Requirements Details

2.1 Functional Requirements

- The user can create an account using the authentication system provided by the application.
- The user would be able to copy the link of a political speech video from YouTube and paste it into our website to watch the video.
- Generate captions in real-time by transcribing the speaker's speech.
- Get feedback about possible misinformation in real-time.

- Have the ability to verify claims the website classifies as a lie with a click of a button that takes the user to a reliable site with the information.
- Save specific facts of their choice and have them available on the database.
- Access saved facts from the database and delete them any time they want.
- Post a review or feedback on a particular fact and get response from our team
- Get detailed information on accuracy of the fact check.

2.2 Non-Functional Requirements

2.2.1 Accessibility

- The website and its constituent services should be accessible by anyone on the internet. The website will be in the English language.

2.2.2 Usability

- The website should have a simple and intuitive interface with a minimalistic design.
- The website should be stable without interruptions and responsive enough to support *near* real-time feedback (max. 5 minutes).

2.2.3 Compatibility

- The website should be compatible with different browsers and should work on both mobile and computer.

2.2.4 Scalability

- The website should be able to scale and handle upwards of 1000 users at a given time.
- The backend should be scalable in the sense that it should be able to run prediction models on 50 videos at a given time.

2.2.5 Performance

- Performance should be good and results should be obtained within 5 minutes.

- Overall response time of a website must also be instantaneous (less than 1 second).

2.2.6 Extendability

- The website should be extendable so that in future it could accept videos from sources other than YouTube.

2.2.7 Accuracy

- The machine learning classifier's "accuracy" would be measured using Macro-F1, Micro-F1 and Accuracy metrics to ensure the reliability of the model.

2.2.8 Reliability

- The website should be reliable to users and by being available all the time especially during live-streamed presidential speeches where traffic would increase substantially.

2.2.9 Testing and Error-Handling

- It should be easily testable for any errors and bugs that might occur. After that writing of code should be such that any bugs found should be easily traceable and handled.

2.2.10 Cost

- The system should provide basic features such as basic video classification at no cost to the user.
- Advanced features such as extensive report generation, saving important facts, and increasing the range of videos available for classification to those that do not have already generated captions will be provided to users at a reasonable price point.
- The system costs should primarily include any TPU/GPU server cost required to train the deep learning classifier.

3.0 Final Architecture and Design Details

3.1 Overview

This section provides a detailed overview of the different aspects of the project starting with subsystem decomposition to describe how different software components are arranged and interact with one another. Hardware/software mapping subsection describes where the different software parts of the project live. Persistent data management subsection then describes how the project gets, handles and stores the data.

3.2 Subsystem Decomposition

The system is designed following the 3 Tier Architecture. The Interface Layer contains all the boundary objects that the user can interact with and send all user provided input to the Application Logic Layer. The Application Logic Layer is responsible for fetching the video and transcript, extracting relevant sentences we deem classification-worthy, running sentences through the classifier and generating results. Results with high confidence scores are sent back to the Interface while results with low confidence can be checked against a database query system which in turn sends back a result that is sent to the Interface. The Data Layer contains the Database Management component which checks low-confidence results against user saved facts to try and increase the confidence level and returns it's result to the Application Logic Layer. The database can also provide users with a list of saved facts.

Below is the diagram representing this architecture,

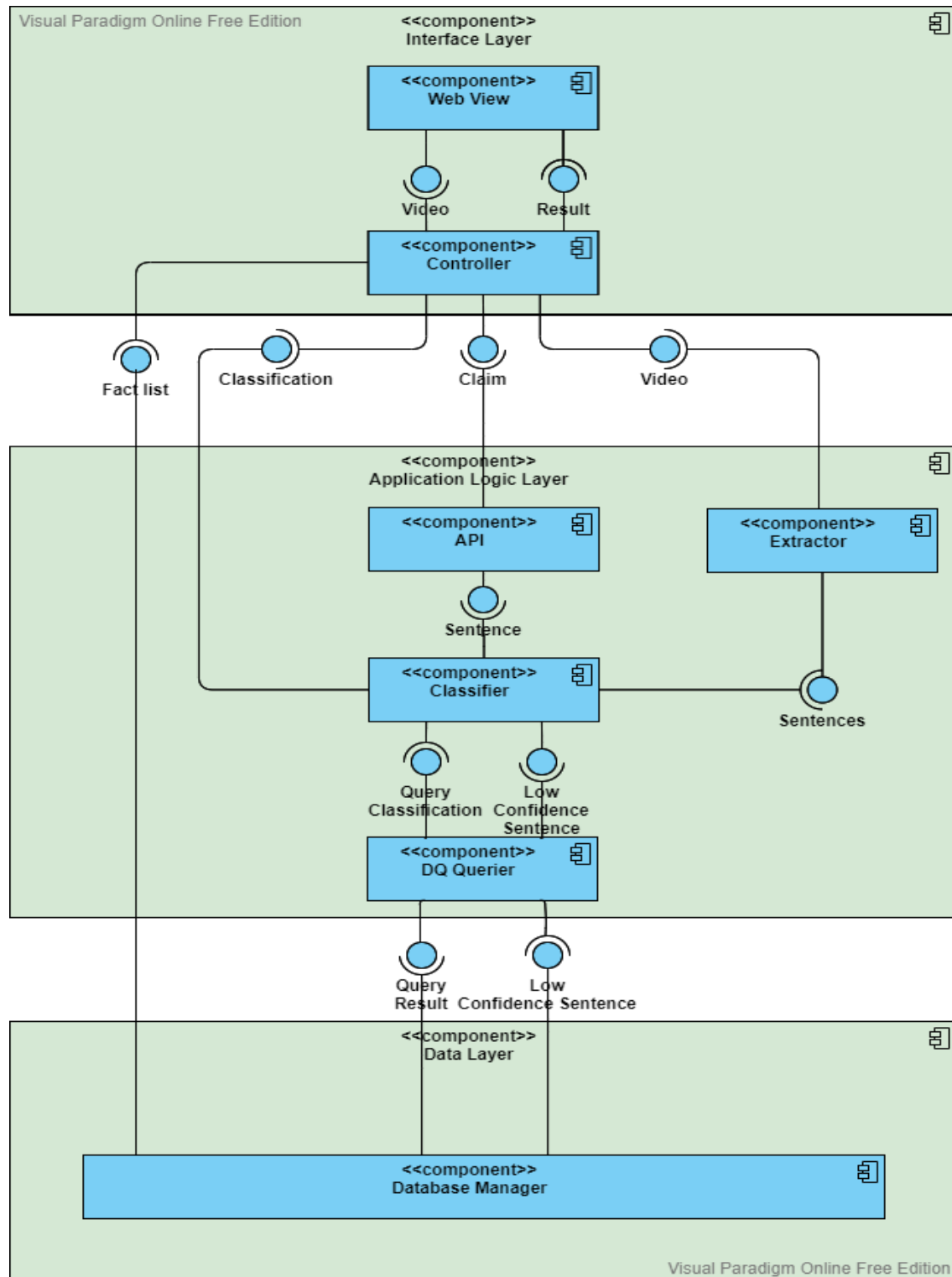


Fig. 1: 3-Tier Architecture

3.3 Hardware/Software Mapping

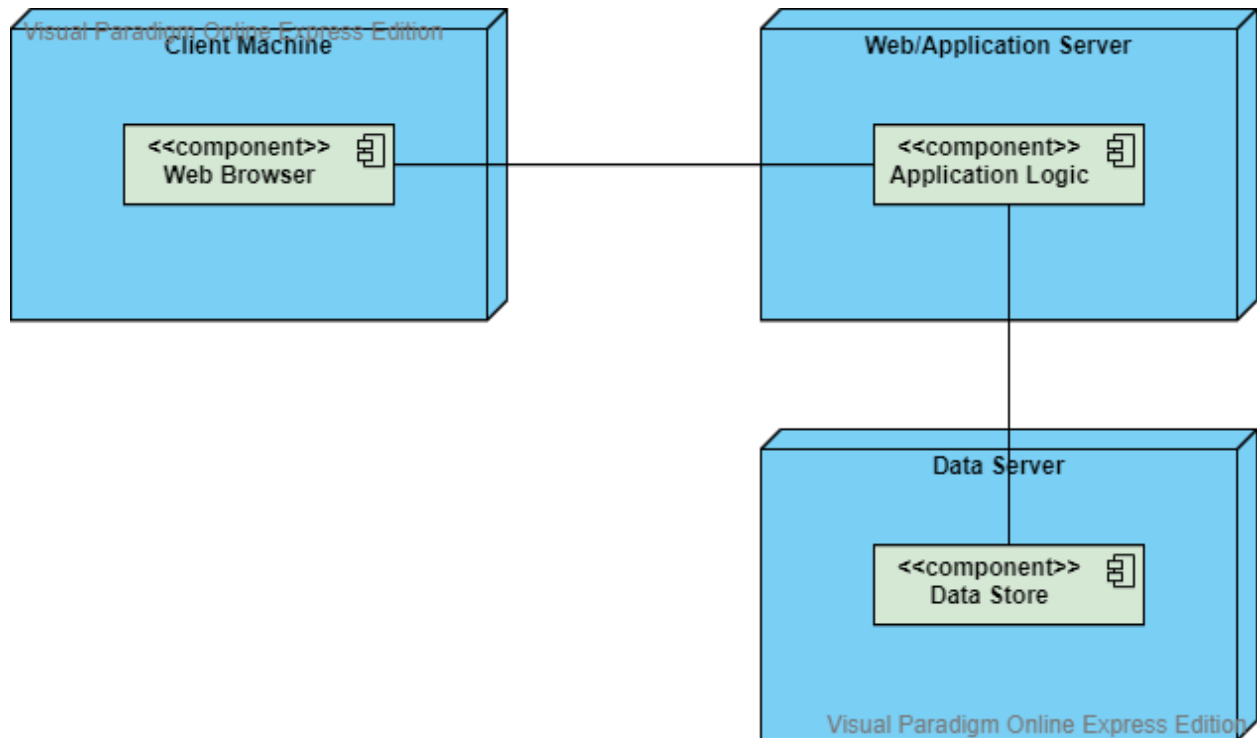


Fig. 2: Hardware/Software Mapping diagram

The different system components discussed in the Subsystem Decomposition section live in different places and on different types of hardware. All user interface components which are associated with a web browser will be on the client's machine while the application logic layer and the data layer both live in the cloud on different servers. The cloud part is managed by the cloud provider and all communication between components are managed using the providers fully managed services to insure scalability, elasticity, high availability and resilience.

3.4 Persistent Data Management

The Veritas system requires some data to be persisted over more than one single execution. The data that will be persisted includes saved facts, user account preferences, and fact reports generated for a video. The persistent data is saved in a secure cloud database. The model provides a fast, efficient, and robust way of inserting,

manipulating, and retrieving data to and from the database. It also ensures that data is backed up in case of corruption or loss of any data on the database.

4.0 Development/Implementation Details

4.1 Deep Learning Model

4.1.1 Data preparation

We used the MultiFC [2] dataset provided by Augenstein et. al. The MultiFC dataset contains many various classes (labels), so we preprocessed the data to reduce it to only two classes. We combined the classes ['mostly true', 'true', 'true!', 'true messages', 'truth!'] into the “True” class, the classes ['mostly false', 'pants on fire!', 'fiction', 'false'] into the “False” class, and everything else was dropped. After this step, the false class contained 9272 claims and the true class contained 5760, so the dataset was balanced. We saved 10% of the dataset (with equal True-False class ratios) to use it as the final test set.

4.1.2 Data preprocessing

We fit a Tokenizer (from `keras.preprocessing.text`) on our training dataset and chose a maximum sequence length of 1500. We had 70221 unique words in the tokenizer’s dictionary. The final input shape to the model was (13529, 1500) and the output was a 13529 binary vector.

4.1.3 Model Description & Training

We used a fairly complex model with many layers. We chained LSTM layers with convolutional and dense layers, while using tanh activations and regularizations between the layers.

The full model description is given below:

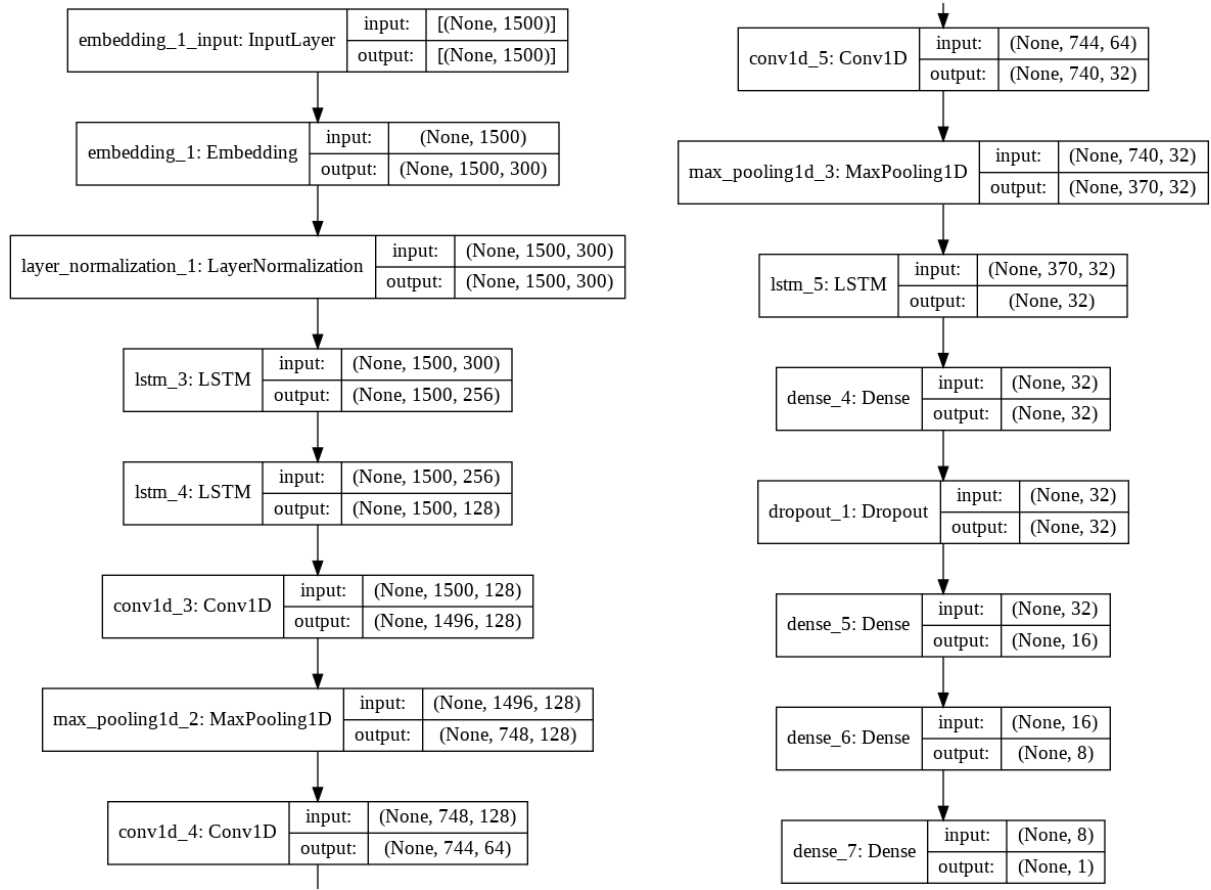


Fig. 3: Deep Learning Model description

For the last layer, we used a sigmoid activation function.

We use the Adam optimizer with a starting learning rate of 1e-4. For our loss function, since we processed the dataset to have two classes, we used binary cross entropy. We also used accuracy as a metric. We computed and used class weights and used a reducing learning rate and early stop (with 10 epoch patience) for our callbacks. We used a batch size of 256 and a 20% validation split. We used Google's TPU's to optimize the training time. On the TPU, each epoch took about 10 seconds to finish. We plotted the model losses and accuracies on the train sets and validation sets in Figures 2 and 3 respectively.

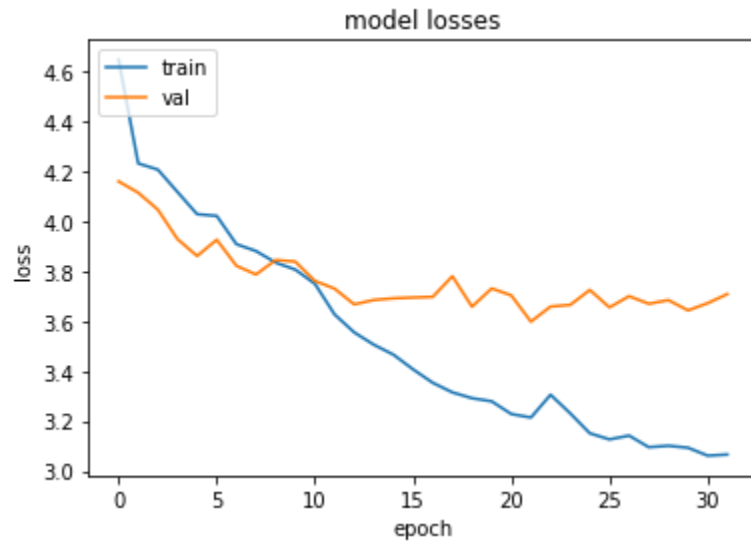


Fig. 4: Validation and training losses

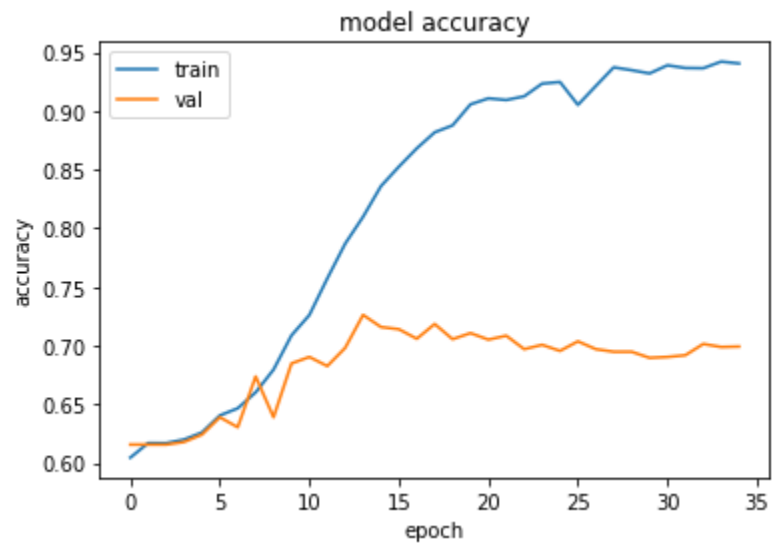


Fig. 5: Validation and training accuracies

Afterwards, we tested our model on the test set that we created earlier and found the following results (after using 0.5 as a cutoff):

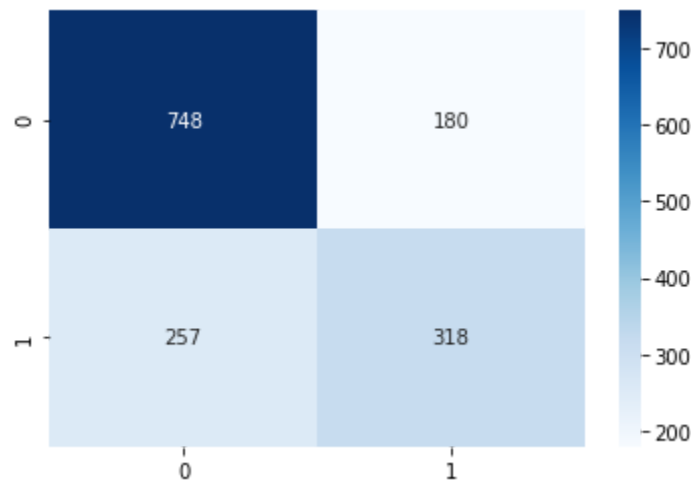


Fig. 6: Confusion matrix on test set

Where 0 is the False class, and 1 is the True class. We also compute the following metrics to check the performance:

	Precision	Recall	F1
False	0.74	0.81	0.77
True	0.64	0.55	0.59

We then saved the model and the tokenizer to deploy them on Veritas.

4.2 Backend and Frontend

The frontend and backend of the project are both implemented using Python's Django framework.

4.2.1 Backend

The backend of the project is implemented by making an API through which the claims are being classified and presented to the frontend. The backend of the project is written using Python's Django framework and is accompanied with some other libraries written

in Python. One of the libraries used in the backend is the Youtube Transcript API [3]. Youtube Transcript API transcribes the video and returns the result as a JSON object and each line of the transcription is being sent to the Machine Learning classifier and is classified as either True, False or Maybe. The other libraries used in the backend are the ones related to the ML Classifier, such as nltk, pandas, dill, tensorflow, gensim all which are written in Python and assists with the Natural Language Processing, Word Embedding, unsupervised learning and matrix related operations of the ML model. For instance, Gensim is used to remove stopwords, strip punctuation, tags, numeric, white spaces and more [4]. The backend is also capable of checking which claims are most worthy of being classified; this is done using ClaimBuster API [5]. After the claims are ranked, classified and then sent back to the frontend to be displayed for the user.

In addition, backend is in charge of the database and connects the frontend with the database. For the database, we chose a Relational Database Management System because it is easier to work with compared to Non-Relational DBMS and we already had studied RDBMS in CS353. PostgreSQL was our main database system, since it was an RDBMS and is provided freely as an add-on in Heroku, the cloud platform we hosted our web application in. The connection between the Database and Frontend is provided by Django as well. The user can create an account, this account is saved in the database and the user can login next time in the system. For the claims, they are classified first and can be saved by the user later in the database. The user can login to the system and access the saved claims, all which are provided through the backend.

4.2.2 Frontend

The frontend of the project is implemented using Django framework templates. The pages are loaded dynamically and are in correspondence with the backend. Each URL request call is responded through the backend and sent to the frontend and the related template is loaded. Apart from the Django framework for the frontend, other web developing languages are used as well. HTML, CSS, Javascript are the core of the mentioned web development languages and are used extensively in the project. There

are many libraries used in the frontend to enhance the services used. To name some of these libraries, Twitter's bootstrap 4.0 framework for the design of the web pages. The small icons used throughout the web pages are provided by font-awesome. DOM manipulation is necessary for working with the HTML elements and dynamically loading them when requests are received from the backend, so jQuery is used there. Javascript's main alerts are replaced by the Sweetalert library which are better in design and shape. Some free themes are used from the internet and the necessary credits are given in order to be able to use them in our project. Youtube's Iframe API is used to work with the youtube video and present it in the frontend while the claims are being shown and classified as a video overlay, similar to a subtitle [6].

5.0 Testing Details

To ensure proper behaviour and satisfaction of functional requirements, we decided to adopt the test-driven development (TDD) process. To accomplish our task of developing a responsive web page to act as a platform for our product, the main functional requirements that represented features the user can directly interact with were identified. Selenium was chosen as the platform of choice because of its versatility and support of Python.

The main features that highlight the flow of using our software are:

- Logging in with a valid username and password
- Entering a Youtube video url and viewing the video, list of claims, and statistics
- Saving claims the user verifies
- Checking database of saved claims

From there, Selenium based test cases were written for each of the features. As per TDD standards, the test cases had red lines and green lines. By the time the test cases were ready, the testing script would run and all cases would fail. After having the tests ready, we started coding the platform itself using the test cases as a foundation. As common with TDD, having the test cases ready and having already thought about

design, the development process of the platform was considerably faster than what it would have been otherwise.

By the time the coding was done and bugs were fixed to the satisfaction of our test cases, all red lines were passed and our platform reached the green line on all tests, we knew that the code was functional and satisfied our testing. All that still needed to be done was to clean up the code, refactor it and make it production-grade ready. As our project neared the finish line, the benefit of test driven development became clear to us. TDD ensured that all the code written was tested with no margin for error. It also ensured that, due to the refactoring step, the code was cleaner and safer. Development velocity was also improved.

Below is the output of our test script once all green lines were successfully reached:

```
-----  
Test Name:  
- Test login  
Messages:  
- Message - Successful login  
Errors: None  
-----  
Test Name:  
- Test video link  
Messages:  
- Message - Successful video input  
Errors: None  
-----  
Test Name:  
- Test fact saving  
Messages:  
- Message - Successful fact save  
Errors: None  
-----  
Test Name:  
- Check saved facts  
Messages:  
- Message - Successfully load saved facts  
Errors: None  
-----  
Test Name:
```



```
- Delete Fact
Messages:
- Message - Successfully deleted fact
Errors: None
```

6.0 Maintenance Plan and Details

Our maintenance plan includes different parts such as the maintenance of the classifier, cloud usage, database usage, API usage, libraries and frameworks involved in the application. Metrics of the mentioned services can be generated through the providers of those services, such as youtube, or Heroku. The maintenance of all services can be categorized into 3 types:

6.1 Maintenance plan of the backend

The maintenance plan for the backend includes the maintenance of the cloud service provider, the database service provider, API calls, Youtube API calls, Transcriber, which are Heroku, Heroku's PostgreSQL, Django, Youtube, and Youtube Transcript API respectively. Metrics for CPU Usage, Memory Usage, Disk Usage, and API Usage will be analyzed and the type of maintenance and frequency of maintenance will be planned according to these metrics. Based on the metrics provided by the service provider, we will decide to scale up or down some of the services, such as upgrading the current Heroku cloud plan to a better one with more disk usage or upgrading the database service provider, Heroku's PostgreSQL and adding extra disk capacity to the database. Currently we are using a free dyno plan on Heroku and the standard plan is the next level which our app can be upgraded to. Similarly the database, it is running on a hobby dev plan and a paid plan provides more disk usage and more space to be used.

The frequency of such maintenance can be seen in the table below.

Service	Maintenance Frequency
Check health of backend(Django)	Daily
Check memory usage of cloud service	Daily
Check disk usage of cloud service	Daily
Check disk usage of database	Daily
Check CPU usage of cloud service	Weekly
Check Youtube API calls	Hourly
Check transcriber API calls	Hourly

Table. 1: Maintenance frequency of backend services

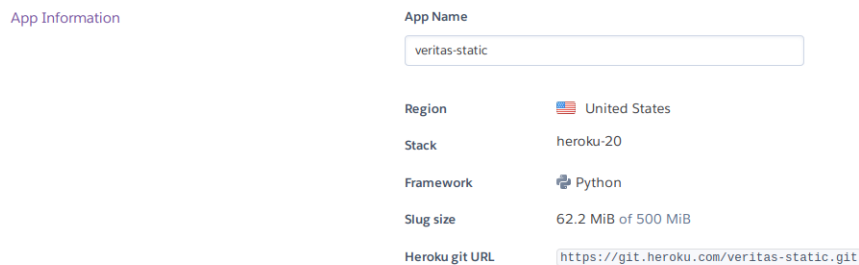


Fig. 7: Disk usage of Veritas web app

6.2 Maintenance plan of the frontend

Maintenance of the frontend will be short and quicker, since the frontend does not use many services simultaneously and possesses lower priority compared to backend and the machine learning model classifier. The maintenance plan of the frontend includes maintaining the website, the templates used and usage of Django's frontend framework. Maintenance of the frontend includes making changes on the frontend, which is why we plan to have maintenance checks on the frontend on a daily basis. Responsive check is

one of the main reasons for daily maintenance check, so that the web app is accessible from any type of device. The table below describes the frequency of maintenance on the mentioned services.

Service	Maintenance Frequency
Check health of frontend(Django)	Daily
Check health of website	Daily
Check templates used in website	Daily
Check responsiveness of the website	Daily

Table. 2: Maintenance frequency of frontend services

6.3 Maintenance plan of the machine learning model

Maintenance plan for the Machine Learning model involves updating the datasets acquired from the currently used fact-checking websites. The model can be retrained based on the newly updated data. Hence the accuracy of fact-checking will be improved and the range of political speeches being checked will be expanded and more videos can be fact-checked. Another maintenance plan for the Machine Learning Model is looking for new websites which work on fact-checking political speeches. The frequency of maintenance depends on how often the current fact-checking websites are being updated, and we plan to check this weekly, and use these to further retrain the model periodically in order to get performance improvements on the classifier.

7.0 Other Project Elements

7.1 Consideration of Various Factors in Engineering Design

Several crucial factors guided the engineering design of Veritas since the goal of labelling fake news is a sensitive task that can have major repercussions on the political landscape of a society.

Data privacy was a core factor that was considered in the engineering design. The primary source of personal data in Veritas comes from the Youtube videos that are uploaded for fake news classification. To comply with data privacy guidelines none of the metadata or any other potentially sensitive information related to the video is processed or stored by the application.

Progression in the field of machine learning is another factor that was considered in the engineering design. One of the most crucial aspects of Veritas is the deep learning model that is used for classification. It is expected, however, that the model will evolve over time due to improvements in natural language processing and classification models. Therefore, the system is built in an extremely modular pipeline where the model can be easily replaced without the need for any major refactoring or functional changes.

Economic factors were also important in the engineering design since we expect the platform to be used by a considerable amount of potential users. The major point in this regard was the correct monetization of features at reasonable price points. It was decided that since in the current version the web app has a lot of advertisement space available, we will pursue advertisement as a primary source of revenue. Features such as more extensive report generation, saving important facts, and increasing the range of videos available for classification to those that do not have already generated captions will be provided at an appropriate price point for users who opt for them in the future. There were also economic factors that constraint our development processes. For example, it was not economically feasible for us at this point to integrate a video transiber from services such as Amazon Cloud Services (AWS). This feature would have allowed us to expand our video reach to those videos whose captions are not available and potentially provide more accurate data to the model for prediction.

Culturally we needed to consider the nature of the American political landscape and how users will perceive the application in the context of their society. The application design especially on the front-end makes the limitations and usage benefits clear to the

user. The classification model also needs to understand cultural and social nuances in textual data in order to provide a more accurate classification for the videos.

The systems global reach also played a part in deliberations regarding the engineering design. The system is designed to be flexible and scale well as the global reach of the classification system increases. This means providing services in various languages and political contexts by having flexible deep learning models representing the service in a specific region.

7.2 Ethics and Professional Responsibilities

Veritas deals with the sensitive task of labelling statements as either being factual or not. This creates certain ethical concerns and dilemmas that require appropriate resolution in order to satisfy legal, regulatory, and moral requirements.

A major ethical issue that stems from the nature of the application is that there may be statements that are incorrectly classified, which may reflect unfavourably on the person who made the statement, and can affect the mind-set of voters in a negative manner. In order to minimize this users will be reminded that the predictions on the veracity of statements are based on a certain confidence level that the deep learning model considers, and in no way it can be viewed as being unequivocally correct.

Another ethical responsibility that the project faces is the proper handling of YouTube video data. These video content will not be stored locally and will solely be used to verify the veracity of the political statements made even though these videos are available publicly.

In terms of adhering to professional standards, the team will ensure that the project conforms to the IEEE code of ethics. Any external libraries, algorithms, and techniques used will be given proper credit and cited according to IEEE standards in order to avoid copyright infringements. The team will also ensure that the product is properly documented and all legal and licensing requirements are met before any official public release of the product.

7.3 Judgements and Impacts to Various Contexts

The impact level (on a scale of 10) and the kind of impact our project makes in different contexts in the world is as follows:

	Impact Level	Impact Description
Impact in Global Context	6	It can be used by anyone as it is accessible by everyone.
Impact in Societal Context	9	A leader may be telling the public some lies and leading them astray. But due to this the public might become aware.
Impact in Economic Context	7	Our product is free, therefore it does not have any economic impact.
Impact in Environmental Context	1	For the usage of computers and running our model electricity is consumed which has an adverse effect on the environment.

Table. 3: Impact level in various contexts

7.4 Teamwork Details

7.4.1 Contributing and functioning effectively on the team

We planned weekly meetings from the beginning of the year in order to collaborate, discuss different ideas and be up to date about what each of our team members was doing. In these meetings we discussed if any of us faced any problems in their tasks as well. We divided the work so that Hanzallah, Abdul Hamid, and Mohamad Fakhouri mainly worked on back-end while Hassan and Abdullah mainly worked on front-end. For the report writing stuff everyone contributed almost equally writing some part of the reports.

Everyone helped each other where it was needed however, Hanzallah, Mohamad Fakhouri, and Abdul Hamid primarily worked on the backend including the ML model for

making the predictions, on extraction of classifiable claims from videos, developing the database modules while the front-end was worked on to an extent by Hassan and Abdullah.

7.4.2 Helping creating a collaborative and inclusive environment

Creation of an inclusive and collaborative environment was crucial to ensuring that all team members stay on the same page when it came to project goals, and to allow members to take greater responsibility in order to increase development velocity. To put this into action the team maintained open communication channels on platforms such as Whatsapp where everyone was periodically asked to provide relevant updates on their tasks. Team members were encouraged to be inquisitive, solve problems together, and to give feedback on modules developed by others. In weekly meetings all group members were asked to provide an update on their sections similar to how stand up scrum meetings are conducted.

7.4.3 Taking lead role and sharing leadership on the team

Leadership experience was an important consideration for all team members since the senior project offers a strong platform in order to develop these skills at the undergraduate level. To ensure that all team members get an opportunity to lead various aspects of the project it was decided that the team be reorganized into a frontend and backend section. Within each section the relevant team members took turns in leading the meetings and agenda, and representing their sections in the overall group meeting. All the group members also turned over the responsibility of leading weekly meetings every other meeting. This allowed everyone to gain valuable insight into leading teams, organizing project tasks, and ensuring that project schedule and costs remain within the specified limits.

7.4.4 Meeting objectives

In order to meet individual and collective objectives on-time and with the highest standards of quality, the team decided to hold weekly meetings (similar to Scrum

sprints) and to maintain continuous communication on channels such as Whatsapp. Increasing accountability measures in such a manner ensured that everyone was kept updated on the progress of different modules and there was no deviation from the set design and architecture. The Gantt Chart was adhered to in terms of task priorities and dependencies which increased the development velocity since less time was spent planning the next task and figuring out which tasks can be done in parallel. In order to reduce blockages that can hinder the fulfillment of objectives, the team increased resources to remove that hindrance as soon as it occurred. Human resources were primarily shifted from low priority tasks to the task that was causing the blockage.

7.5 New Knowledge Acquired and Applied

For this project our previous knowledge was inadequate and we had to learn a lot of new knowledge regarding Backend, Frontend, Natural Language Processing (NLP) and Machine Learning (ML) algorithms.

First we had to read some new and old research papers in order to understand how our model will work and how we will classify sentences. This also included finding a way to understand the sentence in some sentence in order to classify it. However, for the implementation of models, all of us have taken Machine Learning courses at the University which helped us in the implementation process.

After this, the next big step was to extract meaningful sentences from videos. This required a knowledge of NLP as well, we needed to extract sentences from videos and after that send only meaningful sentences to the classifier. This was because every sentence is not a claim or a fact, there are some sentences said formally like for introduction, we needed to filter those sentences. For this we needed to read some papers as well.

We also gained knowledge on the transcribing video data and the challenges associated with it. The transcriber is something that is in the works for coming iterations and

requires either more financing to use third party software such as AWS Transcribe or needs a highly technical in-house approach to develop.

Then on the front-end part, we had to include things that are not taught in Bilkent. We used HTML/CSS and Javascript codes, and we learned a lot about styles, designs, interaction between pages, page elements etc. For the new things we read some blogs and used StackOverflow in order to understand how different things are done like displaying of subtitles or claims one by one after they are said.

It could be said that we learned many new things in every domain, either back-end or front-end. Moreover this helped us develop good teamwork between us.

8.0 Conclusion and Future Work

During our testing and implementation, the classifier proved to be the most challenging aspect. This is due to the inherent difficulty of checking whether the claims are true or not. The claims themselves need not contain enough information; the context is of extreme importance. Another challenge is that these facts are updated as new events happen. To counter these issues, we opted to crowdsource labels and to retrain the model every once in a while to stay up to date. In the current implementation, we have to re-train the model manually. Future work would focus on finding better ways to implement this retraining procedure (either using online learning, or batching with the new observations). Another possible approach that we would like to explore in the future is to analyze the sentiment of what is being said about these claims on the web; this ensures that the model bases its predictions on the latest developments available online.

Another perspective that we might wish to pursue in the future is to not only use the claims, but also the tone of the speaker in the video. The way the speaker talks, waits, and stresses syllables can give insights into the truthness of the claims being made. This approach would require more involved signal processing (for the audio) and a more

sophisticated model that uses both the audio data and the textual data from the claims. This is certainly an interesting approach that we could take in the future.

From a software developmental perspective, Veritas can be ported into many different mobile operating systems, such as Android and iOS, for better compatibility. This would ensure that Veritas reaches an entire audience as mobile users tend to prefer to use native mobile apps over web apps. We also acknowledge that many small features could also be implemented; Veritas was a proof of concept and nowhere near a finished product. If we decide to continue developing it into a finished product, we would need to implement many other small, handy features after we take in feedback from our beta tests.

9.0 Glossary

ML - Machine Learning

NLP - Natural Language Processing

API - Application Program Interface

f1 score - Harmonic mean of precision and recall

Macro f1 - Simple average of the f1 scores of each label

Micro f1 - Calculating precision and recall by summing all the True positives and Type errors instead of calculating for each label, and then calculating f1 score using these values.

10.0 References

- [1] "Fake News, Misinformation, & Fact-Checking | Ohio University MPA | Ohio University", Ohio University, 2020. [Online]. Available: <https://onlinemasters.ohio.edu/masters-public-administration/guide-to-misinformation-and-fact-checking/>. [Accessed: 03- Feb- 2020].
- [2] "MultiFC: A Real-World Multi-Domain Dataset for Evidence-Based Fact Checking of Claims", Augenstein et. al. Available: <https://arxiv.org/abs/1909.03242> [Accessed: 03-Feb-2020]
- [3] "youtube-transcript-api," *PyPI*. [Online]. Available: <https://pypi.org/project/youtube-transcript-api/>. [Accessed: 30-Apr-2021].
- [4] "gensim," *PyPI*. [Online]. Available: <https://pypi.org/project/gensim/>. [Accessed: 30-Apr-2021].
- [5] "ClaimBuster: Automated Live Fact-checking", IDIR Lab. Available: <https://idir.uta.edu/claimbuster/> [Accessed: 29-April-2021]
- [6] "YouTube Player API Reference for iframe Embeds," *Google*. [Online]. Available: https://developers.google.com/youtube/iframe_api_reference. [Accessed: 30-Apr-2021].

11.0 Appendix - User Manual

11.1 Introduction

Veritas is a web application that takes a youtube video, assesses the claims made in that video and classifies those claims if they are true or false. This document explains how a user of the application can do different things offered by our application.

11.2 Login

Fig. 11.1: Veritas login

At the main landing screen, if you are not logged in click on the login button, it will take you to the login page. If you have an account, enter your email and password and press the login button, to login. If you do not have an account click on create an account, this will take you to the registration form.

11.3 New User Registration

Register

Username*

Required: 150 characters or fewer. Letters, digits and @/./+/-/_ only.

Email*

Password*

Your password can't be too similar to your other personal information.
Your password must contain at least 8 characters.
Your password can't be a commonly used password.
Your password can't be entirely numeric.

Password confirmation*

Enter the same password as before, for verification.

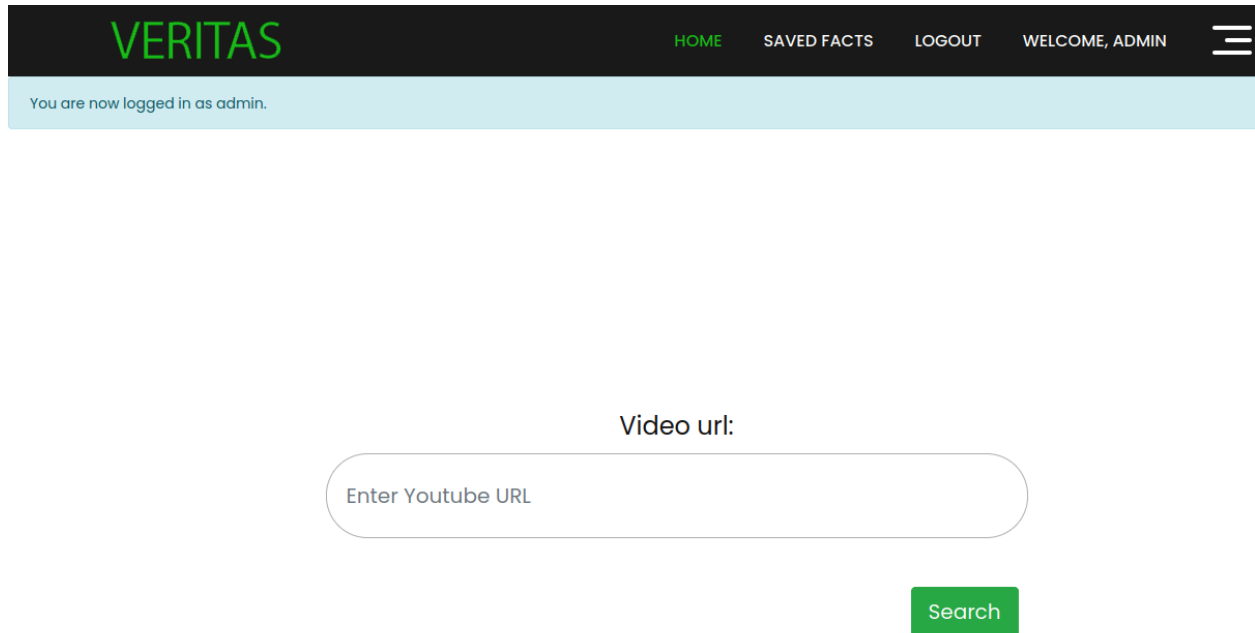
Register

If you already have an account, login instead.

Fig. 11.2: User registration

Enter your username, email and password you want to have and press the Register button. For the password it should contain at least 8 characters and it cannot be entirely numeric. Your new account will be created and you will be logged in. If you already have an account click on the login button.

11.4 Veritas Home Page



The screenshot shows the Veritas Home Page. At the top, there is a dark navigation bar with the 'VERITAS' logo in green on the left. On the right, there are links for 'HOME' (in green), 'SAVED FACTS', 'LOGOUT', and 'WELCOME, ADMIN', followed by a hamburger menu icon. Below the navigation bar, a light blue banner displays the message 'You are now logged in as admin.' The main content area is white and features a 'Video url:' label above a large, rounded rectangular input field. Inside the input field, the placeholder text 'Enter Youtube URL' is visible. To the right of the input field is a green 'Search' button.

Fig. 11.3: Veritas home

After logging in you will be taken to the home screen. Here you can enter the URL of a youtube video in the area given and press Search. Moreover you have the option to logout at top, and the option to view any facts you have saved.

11.5 Classification of Claims of Video

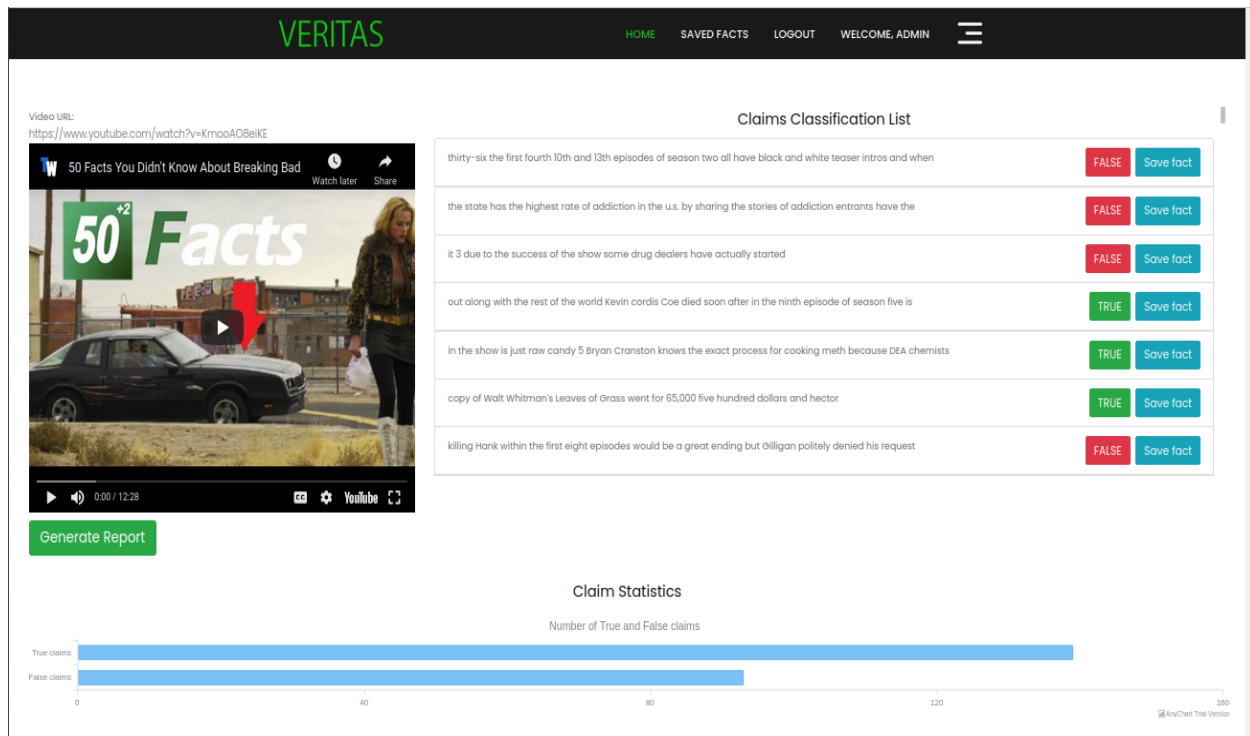
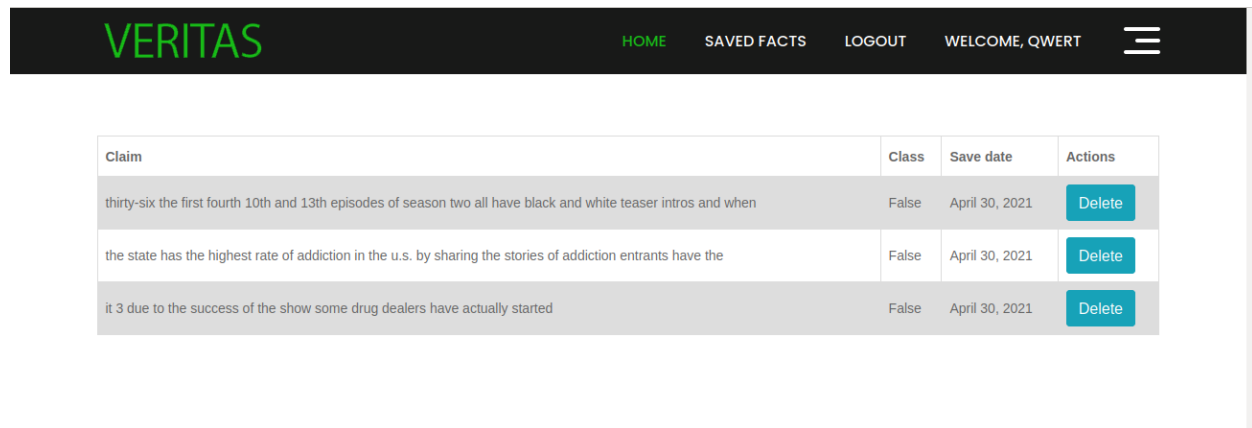


Fig. 11.4: Classifications page

After pressing the search button from the home screen, your video will be played and a list of all the claims made in that video and if they are authentic or not will be displayed alongside the video. Now you can download the report for your video containing different stats about the claims of the video, using the button of Generate Report. The facts shown alongside the video have the option to save them, by pressing the save fact button, this will save them in your account and you can view them afterwards. Moreover at the bottom, there is a graph representing the number of true and false claims made in the video so that the user can visualize the ratio of correct claims made.

11.6 Saved Facts



Claim	Class	Save date	Actions
thirty-six the first fourth 10th and 13th episodes of season two all have black and white teaser intros and when	False	April 30, 2021	Delete
the state has the highest rate of addiction in the u.s. by sharing the stories of addiction entrants have the	False	April 30, 2021	Delete
it 3 due to the success of the show some drug dealers have actually started	False	April 30, 2021	Delete

Fig. 11.5: *Saved facts*

By pressing the saved facts button at top you can view all the facts you have saved till now. You can delete any of the saved facts as well by pressing the delete button.

There is a home button at the top of the screen, pressing it will take you back to the home screen where you can enter the URL of a new video. In Order to logout of your account press the logout button at top-left of the screen.