Low-Level Design 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

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.

1.1 Object Design Trade-offs

1.1.1 Usability vs Functionality

Functionality refers to the sort of operations that Veritas supports, whereas usability describes the ease of operating the User Interface (UI). The project's UI requirements allow greater flexibility and ease with which an intuitive interface on the front-end can be built. The core functionalities of the project are of more interest, and require considerable intellectual attention for proper and correct implementation. Therefore, the project design emphasizes the functional aspects of the system more than the UI aspects.

1.1.2 Loosely Coupled vs Tightly Coupled

Veritas is designed to have a very loosely coupled three-tier architecture that is highly modular in nature. This will allow easily modifying and extending functionality by, for example, integrating new APIs. The loosely coupled nature of the system will allow us to implement different design patterns that enable quick refactoring of components. This makes testing the system easier as well since components can be distinguished and tested individually more easily.

1.1.3 Extensibility vs Leverage

Leverage means that the components of a software system are modified on an as needed basis to meet the requirements of a specific software. We considered whether to increase leverage in our application components or to increase their reusability and modularity. We decided that with a complex, predictive, machine learning based system it was better to allow for ease of development for future updates. This was also because the fields of misinformation detection through NLP based machine learning, and live video transcription are highly evolving and would require frequent updates to the system.

1.1.4 Scalability vs Performance

Scalability can negatively affect performance as the system grows such as if there is a need for data consistency but the network has a high latency. Through the Veritas design we wanted to prioritize performance over scalability since Veritas first needs to correctly address the issue of real time fact checking with high performance metrics such as generation of a transcript from the YouTube video within a minute. The design is innately inclined towards scalability but that can be dealt with in future versions of the system.

1.1.5 Efficiency vs Portability

Veritas is designed to be more efficient than portable by being built for web browsers only in order to focus on maximizing full use of the underlying web infrastructure. This will enable us to deliver results quickly to the end-user and not spend time resources working on replicating the functionalities on other platforms such as those that are mobile based.

1.1.6 Ease of Use vs Performance

We decided to create a database-backed login and registration system, which would securely save the credentials using PHP's latest php_hash function but this would require extra time and effort to code and integrate. In addition, we would need extra tables in our database to save and manage user's data, which leads to using extra space as well. However we preferred to keep the user registration system as simple as it could be by allowing Google registration system. The database management will be done by Google's OAuth 2.0 authentication system, but Google Sign-In system is comparatively slower in performance but easier to use, so we went with the easy option.

1.2 Interface Documentation Guidelines

Class: ClassName
Class Description
Properties
● public string propertyName1 ● public int propertyName2
Methods
● public string methodName(parameters): method description ● public string methodName2(parameters): method two description

1.3 Engineering Standards

The report follows the Unified Modelling Language (UML) [2] standard for structural and behavioral modelling of the system. Structural modelling is used to construct the object model, package, and deployment diagrams, whereas the behavioral modelling is used to construct operation, and sequence diagrams. The report uses the Institute of Electrical and Electronics Engineers (IEEE) [3] standard for referencing.

1.4 Definitions, Acronyms, and Abbreviations

Term	**Definitions, acronyms, and abbreviations**
3-Tier Architecture	● Interface Layer (User interactions) ● Application Layer (ML Models) ● Data Layer (Database of facts)
DB	Databases
NLP	Natural Language Processing
UML	Unified Modelling Language
IEEE	Institute of Electrical and Electronics Engineers
API	Application Programming Interface. The term is used to define interfaces that allow application components to interact with each other.
ML Model	A black box that contains a mathematical representation of the real world with which we can train real-world data to obtain predictions.

2.0 Packages

Inspired by the 3-Tier architecture, we package Veritas in three main packages: Interface package, Application Logic package, and Data package.

2.1 Interface Package

This package is responsible of the interaction with the user and of the communication with the Application Logic Package’s classes

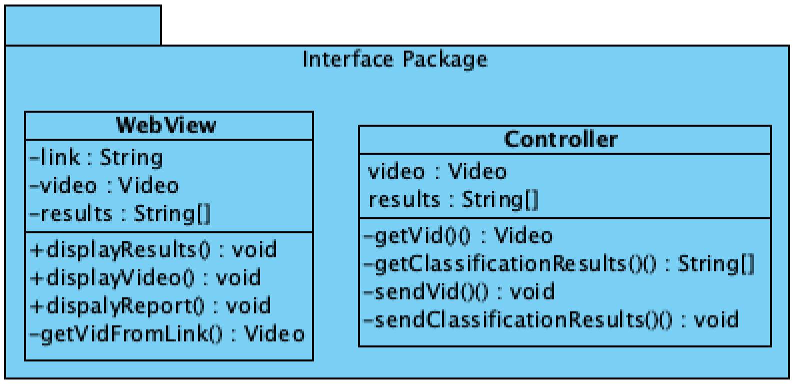

Figure 1: Interface Package

2.2 Application Logic Package

This package is responsible for transcribing the video, extracting relevant sentences, running sentences through the classifier and generating results. It also sends the sentences with low confidence score classifications to the DB package for further checks.

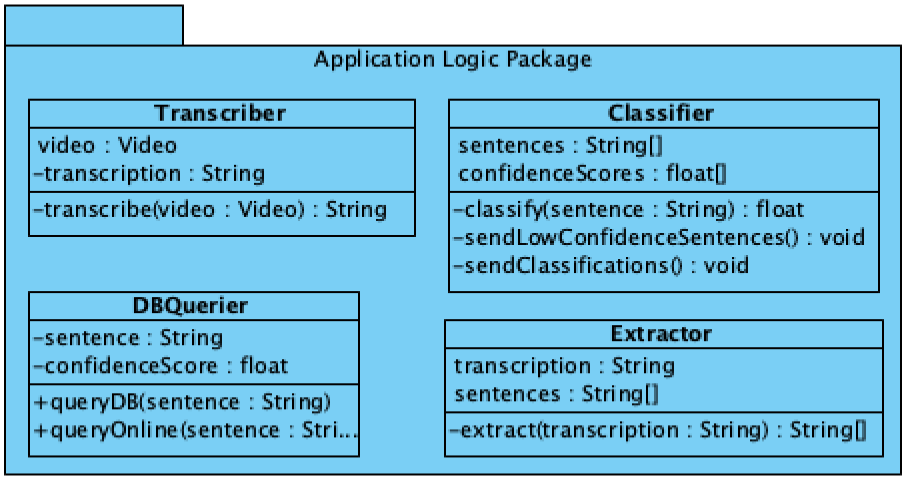
Application Logic Package
Transcriber
video : Video
-transcription : String
-transcribe(video : Video) : String
Classifier
sentences : String[]
confidenceScores : float[]
-classify(sentence : String) : float
-sendLowConfidenceSentences() : void
-sendClassifications() : void
DBQuerier
-sentence : String
-confidenceScore : float
+queryDB(sentence : String)
+queryOnline(sentence : Stri...
Extractor
transcription : String
sentences : String[]
-extract(transcription : String) : String[]

Figure 2: Application Logic Package

2.3 Data Package

This package checks the received sentences from the Application Logic Package using its own database and, if necessary, some sources from the internet. It is responsible for storing re-labeled data and for managing the search-sources.

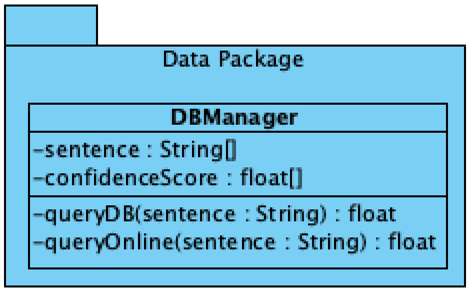
Data Package
DBManager
-sentence : String[]
-confidenceScore : float[]
-queryDB(sentence : String) : float
-queryOnline(sentence : String) : float

Figure 3: Data Package

3.0 Class Interfaces

3.1 Interface Layer

Class: WebView
Class responsible for all the visual elements that the user interacts with. It takes all user input and displays all visual details displayed on the screen.
Properties
● string link ● Video video ● string[] results
Methods
● public displayResults(): method to render results on screen ● public displayVideo(): method to render video on screen ● public displayReport(): method to render report on screen ● private getVidFromLink(): method to retrieve video using link

Class: Controller
Class responsible for connecting the visual layer (WebView) with the logic layer.
Properties
● Video video ● string[] results
Methods
● private getVid(): method to retrieve video from WebView ● private getClassificationResults(): method to retrieve classification results from Classifier ● private sendVid(): method to send video to Transcriber ● private sendClassificationResults(): method to send classification results to WebView

3.2 Application Logic Layer

Class: Transcriber
Class responsible for transcribing the video chosen by the user.
Properties
● Video video ● String[] transcription
Methods
● private String[] transcribe(video): method to transcribe the video and generate text

Class: Extractor
Class responsible for extracting relevant sentences from the transcription.
Properties
● string transcription

• string[] sentences
Methods
• private string[] extract(string transcription): method to extract sentences from transcription

Class: Classifier
Class responsible for classifying the extracted sentences
Properties
• String[] sentences • float[] confidenceScores
Methods
• private float classify(string sentence): method to classify the sentence and return confidenceScore • private sendLowConfidenceSentence(): method to send low confidence sentences to DBQuerrier • private sendClassification(): method to send classification to Controller

Class: DBQuerier
Class responsible for checking low confidence sentences again database and internet
Properties
• string sentence • float confidenceScore
Methods
• private float queryDB(string sentence): method to check sentence against DB • private float queryOnline(string sentence): method to check sentence online

3.3 Data Layer

Class: DBManager

Class responsible for acting as an interface between the system and the cloud database used
Properties
• string DBPath
Methods
• private add(string sentence): method to add sentence to cloud DB • private search(string sentence): method search for sentence in DB

4.0 Glossary

ML - Machine Learning - 2,5

NLP - Natural Language Processing - 2,3,5

API - Application Program Interface - 3,5

UI - User Interface - 2

5.0 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] Uml.org. 2021. *Welcome To UML Web Site!*. [online] Available at: <https://www.uml.org/> [Accessed 3 February 2021].

[3] “The world's largest technical professional organization dedicated to advancing technology for the benefit of humanity.,” *IEEE*. [Online]. Available: https://www.ieee.org/. [Accessed: 03-Feb-2021].