Bilkent University

Department of Computer Engineering

Senior Design Project

Newspector: Detection of Unreliable News

High-Level Design Report

Ahmet Ayrancıoğlu 21601206
Deniz Dalkılıç 21601896
Kaan Gönç 21602670

Supervisor: Varol Akman

Jury Members: Halil Altay Güvenir and Özcan Öztürk

High-Level Report
December 31, 2019

This report is submitted to the Department of Computer Engineering of Bilkent University in partial fulfillment of the requirements of the Senior Design Project course CS491/2.

1. Introduction

The news is one of the most important parts of everyone's life as it is a medium for them to learn about the developments in the world. The news directly affects people and society as a whole. No matter where you are in the world either traveling or working in the office you can get all the latest updates from the world. Without the circulation of the news, it is impossible for people to know what is happening around them. The latest news covers all the news from various fields such as sports, entertainment, fashion, charity events and much more. The world is changing every moment and, we can keep an eye on the changes through the news. Therefore, the importance of the news in our daily lives cannot be measured in words. This important role of news in the world leads various news companies to be founded and thus creating a great rivalry between such sources on being the fastest and the most accurate one among all to deliver news to society. An important question that should be asked here is how much we should believe such news companies, and which one to follow if we want to access the latest news first.

When we consider these issues, the concept of accessing reliable breaking news comes out to be a real concern for many people. Anything may happen at any point in time, so it is very important for the media to stay alert to cover the incidents when it happens. The breaking news gets a lot of attention when it comes to the news portal. The breaking news may be related to any incident or the latest discovery around the world. An important point is that there are too many news sources on the web and only a limited of them seem to be producing reliable news. We are planning to define worldwide known and trusted news companies' websites and twitter accounts as our sources of data in our project. And our interpretations about the companies will be based on the speed of publishing the breaking news and the reliability of the content of the news.

1.1 Purpose Of The System

Our mission is composed of two subgoals. The first goal is to identify which news companies provide the breaking news with the same content on their platforms and comparing their publishing times. The information that we will obtain from this goal will help us identify the quickness of different news companies in delivering the breaking news. Our second goal is a lot harder to accomplish when it is compared to the first one. We are aiming to compare the claims given in these breaking news (having the same content) from different news sources and try to obtain information on the reliability of these sources.

We are going to combine these two subgoals in a system having a website and a mobile application, as well as supporting a web browser plugin to display our results and inform people. The website will be based on displaying the latest breaking news and the specific sources which published that news so that people can change between different news sources to examine how differently multiple sources published the news having the same content. The mobile application will be based on sending notifications to users when breaking news is published and keeping users tuned in by informing the user when a different source publishes the same news or a news source changes their content. Finally, the plugin application will be serving people who do not want to interact with our website or

mobile application but still want to be informed about the analysis of the news that we provide while reading it from the source website.

1.2 Design Goals

Supportability

- All classes and scripts should have a description explaining the functionality of the class and the reasoning behind its creation. Additionally, all functions should also have brief descriptions that give insight into why the developer chose to write that function.

- The project should be divided into well-defined and independent subprojects and layers in order for multiple people to work on the overall application together efficiently.

- The project should be structured in a way that would support extending the project even though some of the decisions for achieving said extendibility could be considered over-engineering by some for the initial state of the project.

- The Newspector website and mobile application should use the same codebase in order to avoid maintaining two codebases for displaying the same information on different devices.

Usability

- Since Newspector will have a browser plugin and will show warnings and extra information over the news, the information shown should be seen with ease but should not disturb the user from reading the news article.

- The warning popup should be able to clearly show the user the possibly unreliable news and should provide a button for the user to go the Newspector website for further analysis of the news.

- The Newspector plugin should be available to download in the Chrome Web Store for ease of access.

- The plugins should have an option to show users a page with instructions and a rough explanation of how the Newspector works behind the scenes.

Reliability

- Gathering the news pieces from selected sources is crucial for the Newspector to function. So, Newspector should be able to gather news from at least 90% of the selected sources at a given time.

- To increase the trust of users towards the Newspector, false reporting of solid, well-supported information that can be validated by other sources should be kept at a minimum of 10%.

- The sources should be selected carefully by a group of knowledgeable people in order to reduce bias.

- Newspector should update its sources so that no deadlinks will ever be presented to a user such as removed news articles, etc.

- The data storage solution for the project should be selected in a way that will allow the Newspector to be functional when a crash happens on one of the servers that store the news data.

- The data storage solution for the project should support scalability and should not require the reconstruction of the database when the stored data gets too big.

Performance

- The gathering of the news from selected sources should be fast and effective to include the most up to date news for cross-validation.

- Examination of the news article being read by the user should be fast and be done in real-time as the user reads.

- The Newspector plugin should not cause any slowdown on the website the user is on.

- The servers running the cross-validation algorithms should be fast and can be auto-scaled as the population of the userbase grows to provide the same performance for every user.

Extendibility

- The addition of further news sources should be easy. Other parts of the application should not be dependent on where the news is coming from.

- The addition of further categories should not affect the rest of the cross-validation algorithms.

Portability

- The process of porting the Newspector to different mobile platforms should be easy and should be connected to the same backend server.

- The user experience of the Newspector should be the same and provide exactly the same information on various platforms.

Privacy

- The use of cookies and other means of gathering information should be done with the permission of the user.

- Data gathered by users should not be associated with users directly unless they give their consent.

- Data gathered should be securely stored in our databases as it can easily be used to target news or advertisement by other platforms.

1.3 Definitions, acronyms, and abbreviations

UI: User Interface

API: Application Programming Interface

Server: Houses the database and is the backend of the system. All logical operations on the data are done here.

DB: Database

NLP: Natural Language Processing

NoSQL: Not Only SQL

1.4 Overview

The Newspector aims to help readers become more aware of the world around them by gathering breaking news from various different sources from the web and cross-checking them for any contradictions and reporting them to the user. To achieve this goal, the Newspector will use different approaches and tools. Our main interfaces to the users will be a website, a mobile application and a browser plugin that will display relevant information on the latest breaking news using Natural Language Processing. There will also be a Newspector website where users can directly go to and browse news there and check any news they want there. Finally, there will be a Newspector backend that will handle the cross-checking and the NLP part. The requirements below will focus on these 3 parts and how the Newspector should work.

2. Current Software Architecture

In today's market, there are many breaking news reporting apps and many tools about fact-checking on the web. However, these are more specialized in their own areas of interest which are to display breaking news and fact check separately. The system which shows the highest similarity in terms of functionality to our Newspector is the "Google News" as it categorizes the news according to their content and displays them to the user as a group. Google News provides a section for breaking news and a full coverage option for particular news. By looking a the full coverage, a user can see other articles published about the same topic. [1] However, Google News can include articles from sources that may be categorized as contempt and sometimes not all sources that published an article on the same topic are shown. Another lacking feature is that google does not show any comparisons between the content of these articles. Finally, even though google sends notifications about the news that may interest the user, there is no option to track any particular news. We believe our application will be the first in its kind that will provide the user's the opportunity to track breaking news from various sources in real-time and also provide information on the possible contradictions between news from different sources.

3. Proposed Software Architecture

3.1 Overview

The Newspector will use Natural Language Processing, Text Analysis, and Sentiment Analysis algorithms to examine and make comparisons of news headlines.

The platform will consist of two main layers. The first one will be our user ınterface applications which include a web browser plugin that the user installs as an extension, a web page and a mobile application where the user is able to access chunks of breaking news having the same content. The second layer is the server application that executes all the examination algorithms, analysis and comparison processes mentioned before. After extracting headlines of breaking news, the server will make its computations and will respond back to our user interface applications with the related information. Regarding the result coming from the server, the plugin will provide information to the user about the news currently being read and will show a popup that asks the user to follow a path to see the contradicted or supported articles on the Newspector web page. When directed to our webpage, the users will be able to access various breaking news grouped by their topics and they will be able to choose between various news sources if they want to examine the details of the news. Additionally, information about the discrepancies between the headlines belonging to the same group will be provided to the user so that the user may have a bigger interest in reading these conflicting news. Finally, our most important feature is the real-time reporting system in our mobile application. Whenever breaking news is added to our servers, we will send a notification about the news to all users. If a user wants to follow specific news, he will be able to press a single icon and notifications about the other news sources' breaking news on the same topic will be sent to the user when available. In this way, we believe that people won't spend their time constantly check for updates on the web

but rather wait for our application to send notifications about breaking news from various sources.

In order to achieve a great user experience, we needed a software architecture design pattern that could help us in the implementation process. As we mentioned previously, The Newspector will consist of the three User Interface Applications and a Server application. We decided to use Model-View-Controller (MVC) software architecture design for our user interface applications so that our application is going to be reusable, easy to update and safe. We created subsystems so that the implementation part would be easier for a team. In MVC software architecture, Model parts are also entities for us to keep information in the database. We are planning to develop our user interface applications in Flutter and Dart and use Google's Firestore as our database, which is a NoSQL database, for keeping our data. Information about news sources and breaking news headlines will be kept in the Cloud Firestore database. These will be the main sources of data for our entities in the user interface applications. Putting news under categories and finding information about conflicting news will be done by the server application using the information provided by the entities and will also be kept in the Firestore database. The server application will be totally separate from the UI applications so that it will only update the database and the UI applications will be serving for displaying the data kept in the database. This will help us in creating a more robust system. We are planning to use dart language for both updating and obtaining data from the Firestore database in all of our applications.

3.2 Subsystem Decomposition

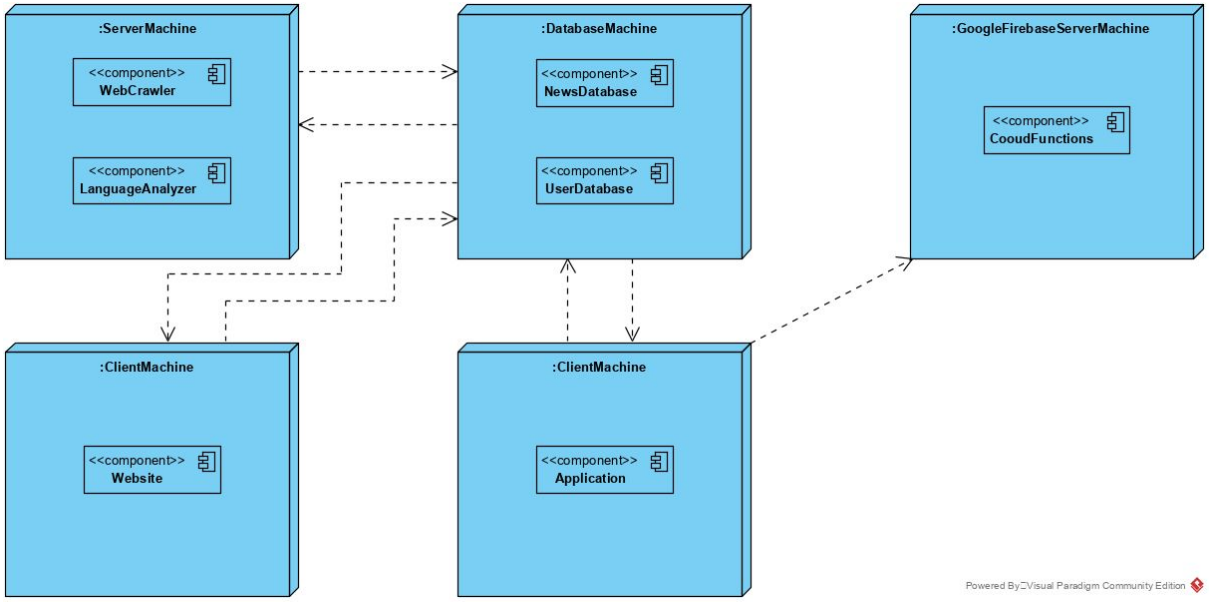

Figure 1 - Deployment Diagram

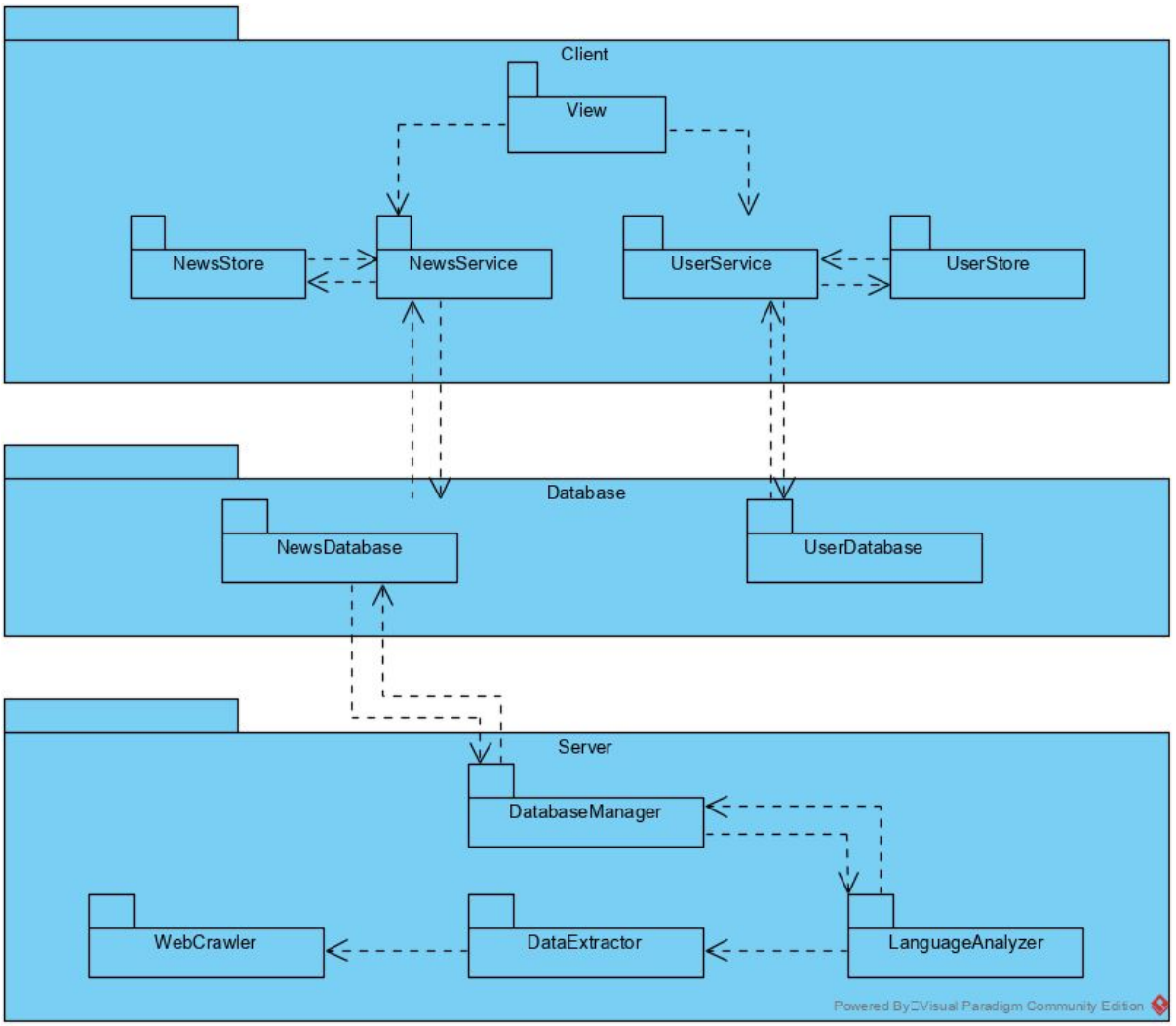

Figure 2 - Subsystem Composition Diagram

3.3 Hardware/Software Mapping

The architecture of the Newspector is mainly divided into three levels in terms of Hardware and Software relationships. The first level is directly related to the client-side. Client machines such as mobile phones and computers are taking the role of delivering data to the user, which is to display breaking news information provided by our database. The second level consists of Google's servers which will help us store our data. As the number of the target audience can be high, we decided that the easiest and fastest way to handle the storage issue is to use servers of Google. As mentioned in detail later, Google's Firebase and Firestore databases provide high flexibility to us on our stored data, with different features such as cloud functions, authentication management, notification services and so on. The third and the final level is our own servers which will always be active running our fetch and analysis algorithms regularly on various news sources and providing information to our database.

3.4. Persistent Data Management

The Newspector application needs to have a database to store information related to various news sources, breaking news headlines (maybe the articles too), the relations between matching or conflicting breaking news and more. We believe that a NoSQL database is much more beneficial for our purpose and also much easier to interact with. Additionally, in terms of the pricing issues, Google firebase databases adjust pricing according to the amount of information you fetch instead of the amount of information you keep in the database. Therefore, we are planning to use Google's Cloud Firestore as a database that is fast, reliable and easy to interact with. Cloud Firestore is a fast, fully managed, serverless, cloud-native NoSQL document database that simplifies storing, syncing, and querying data for your mobile, web, and IoT apps on a global scale. Its client libraries provide live synchronization and offline support, while its security features and integrations with Firebase and Google Cloud Platform (GCP) accelerate building truly serverless apps. [2]

3.5. Access Control and Security

Access control is a security technique that regulates who or what can view or use resources in a computing environment. The Newspector has one type of user (a newsreader) and it does not have admin users who can edit what is being displayed on the UI applications. At this stage of our design, we are not sure whether we should implement a user based system where we will keep information about the users and apply modifications for each user separately by interpreting their past preferences using the applications. We believe that such a system will affect the purpose of The Newspector negatively as we want it to be simple and serve for only a specific purpose. However, if we decide to add a membership to our application and keeping information about user experience and preferences, we are confident that we can implement such a system using Google's Cloud Firestore features easily. It handles access control and security by itself and we will only need to integrate a login system to our user interface applications. [3]

3.6. Global Software Control

In the Newspector system, an event-driven control mechanism will be used. In our project, the system is designed to respond to user actions within the system. Each event that the user performs corresponds to a request creation and the requests run code segments in the application. As the main aim of the application is to provide various information about the breaking news each user request will end up receiving the latest data from our database. Users will have no option to edit any information except their preferences for tracking the latest news topics.

3.7 Boundary Conditions

3.7.1 Initialization of the System

In order to use The Newspector, users need to have an internet connection for using the mobile application, accessing the website and using the plugin services. The only action that the user should do in order to use our system is to download the mobile application, open the website or to activate the plugin. If the user used our application before, our client system will use the cached data from the previous session ın order to provide a faster and better user experience.

3.7.2 Termination of the System

The Newspector consists of multiple subsystems and the way each subsystem handles the termination is different. Our servers which are used for fetching, analyzing news and updating our database will always be active so that our system will be able to fetch the latest news and deliver them to our database regularly. Additionally, as we use Google servers, our database will always be active as long as Google provides us a trouble-free service. Finally, in the case of a termination, our client applications will cache the latest information to the local storage so that the users will be able to access their previous session faster and provided a smoother experience.

3.7.3 Failure of the System

There are three main scenarios where our application may fail to deliver the latest breaking news to the user. The first one is that the client has no internet connection, and fails to connect to our database. The second one is that there is a problem in Google's Firestore and the database fails to respond to queries made by the client. The final one is that the client is connected to the internet and the database is online but there is a problem with fetching the latest news. In any of these cases, the user will fail to fetch any new information from the system and therefore will not be able to display the latest breaking news. Additionally, the user won't be able to receive notifications from the mobile application.

In the case of a failure, in order to provide a good experience, the application will display the cached data so that the user can access their previous session until the failure is dealt with.

4. Subsystem Services

4.1. Client

4.1.1. View Subsystem

View Subsystem handles interaction between the user and our application. This subsystem consists of various small and modular UI Components that can be modified and combined to create more complex and desired pages, menus, pop-ups, lists, etc. that the user can interact with ease. This subsystem does not include any logic inside and will only transfer the user requests and actions to the Controller Subsystem and show the latest information to the user.

4.1.2. Controller Subsystem

Controller Subsystem handles the user requests and actions transferred from the View Subsystem. It is responsible for keeping the information up to date, changing user preferences, fetching new data from the database, and saving and loading cached data from the device. There are two main controllers in this subsystem.

News Service: It handles every request about news such as fetching new data from the database, saving the current state of the session to the local storage, synchronization of local data and database.

User Service: It handles every request about the user such as changing the user's preferences, settings, and the news they are currently following.

4.1.3. Model Subsystem

Model subsystem stores information about the current session and acts as a local database. It does not include any logic inside only data structures. There are two main stores inside this subsystem.

News Store: It holds every news related entities and objects inside.

User Service: It holds every user-related information inside.

4.2. Database

We have two main databases that are loosely connected to each other. Our first database is the news database and it holds information about the analyzed news and their connection to other news. Our second database is the user database and it holds information about our users and their preferences, the news they are following, etc.

News Entity Fields
Id: Unique identifier for this news.
Headline: Headline of the news article.
Source: Source of the news.
Date: Last modified date of the news.
History: The history of the news, like a git history of a file.
Analysis Report: Our analysis report of the news.
Status: Status of the news, if it was changed or not and more.
CluesterID: The id of the cluster that this news is a part of.

User Entity Fields
Id: Unique identifier for this user.
FollowingNews: A list of cluster Ids this user follows.
DeviceId: The unique identifier for the user's device.
NotificationToken: The unique identifier needed to send the user a notification.

4.3. Newspector Server

Our server handles fetching the latest news, analyzing them and updating our database accordingly.

Web Crawler: It is responsible for searching our sources and looking for news articles that are recently released, or changed. It constantly searches our sources and updates our database with the latest news.

Data Extractor: After the web crawler gets the news articles source code, the data extractor extracts the important information that we need in order to apply our analysis algorithms and updates the database accordingly.

Language Analyser: After we extract the needed information in a compact form, the language analyzer analysis the news to try to categorize them and find disparities between related news.

Database Manager: It is responsible for updating our news database with the latest analysis and the information. Other controllers and programs in this subsystem use it to update the database.

5. Design and Architectural Changes From Previous Reports

After receiving feedback from our supervisor Varol Akman and innovative expert Mustafa Sakalsız, we have decided on changing our design and architecture. Previously, it was stated in our analysis report that The Newspector will be analyzing news articles having the same content and finding the discrepancies between claims in the sentence level. However, according to our research in the open-source NLP tools on the market, we have decided that we won't be able to accomplish our goal as these tools were not sufficient enough to analyze and compare advance vocabulary that is used in news articles for regularly obtaining correct results, and therefore may affect our user experience negatively. In order to avoid such advanced vocabulary and to increase the precision in the discrepancy detection of our system, we have decided to focus only on the news headlines without including any extra content from the article itself. The reason for this is that the news headlines generally consist of approximately 7 words, explicitly defining the content by using real names for people, locations, events and movements that help us understand what is inside the news at the first glance. Additionally, news headlines do not contain advance vocabulary and punctuation so that any person having an average understanding of the English language could understand what is being told easily. Overall, we will be providing news headlines as data to our NLP tools and will be using the analysis result for our matching and comparison algorithms.

This design change led to some significant changes in our architecture too. In our analysis report, it is stated that our algorithms will be based on finding claims in articles, tokenizing each of them and separating each token into its simple form and applying coreference resolution techniques to match objects between different sentences. However, focusing only on the headlines will make our job a lot easier as it will make us analyze and compare two short sentences. Another thing is that our machine learning algorithms will not be provided insignificant claims but rather will be provided significant headlines which could help us obtain better results in our learning algorithms.

6. Potential Risks at the Development Cycle

In this section, we will state the potential risks which we can encounter while we are in the development process. We consider this chapter would be helpful in order to address the stated risks.

Debugging and Real-Time Testing:

Since the success of our application will be based on real-time experiences using mobile and web applications, the testing process will most probably be challenging. The reason for this is that our system is capturing the latest news published on several sources and providing instant notifications to users after the analysis is made which makes it a lot harder in the development process to provide reliable test data to simulate real-time behavior.

API Security:

If we are to implement a membership system to The Newspector, APIs that we use will have potential risks in terms of security. To safeguard the security of our APIs, validation of SSL certificates is always necessary. Nefarious API traffic intervention and inadequate validation will certainly deliver us right into a hacker's hands.

Stable Tool Functionality:

Open source tools are always questionable in terms of providing continuous functionality. As NLP is a complex field and we will be using open source tools to achieve the goals we need to make sure that we are using stable versions of these tools so that at some time they would not stop functioning which would make our system fail and ruin the user experience. Therefore, we need multiple alternatives in case we encounter problems in any step during our project life cycle.

7. References

[1] “And now, News,” *Official Google Blog*, 23-Jan-2006. [Online]. Available: https://googleblog.blogspot.com/2006/01/and-now-news.html. [Accessed: 31-Dec-2019].

[2] “Cloud Firestore | Firebase,” *Google*. [Online]. Available: https://firebase.google.com/docs/firestore. [Accessed: 31-Dec-2019].

[3] “Firebase Authentication | Firebase,” *Google*. [Online]. Available: https://firebase.google.com/docs/auth. [Accessed: 31-Dec-2019].