



Department of Computer Engineering

Senior Design Project

Final Report

Supervisor:

Asst. Prof. Mustafa Özdal

Jury Members:

Asst. Prof. Ercüment Çiçek

Asst. Prof. Hamdi Dibekliolu

Innovation Expert:

Ahmet Kocamaz

Presented by:

Erkam Berker Şenol

Mert Aytöre

Gökhan Şimşek

Dias Alymbekov

Figali Taho

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.

Contents

1	Introduction	3
1.1	Design Trade-Offs	4
1.1.1	Security vs. Cost	4
1.1.2	Space vs. Speed	4
1.2	Engineering Standards	4
1.3	Interface Documentation Guidelines	5
1.4	Definitions, Acronyms and Abbreviations	5
2	Final Architecture and Design	7
2.1	Subsystem Decomposition	7
2.1.1	Client	8
2.1.2	Server	10
2.2	Hardware Software Mapping	12
2.3	App and Web-app Design	12
2.3.1	Live Broadcasting and Video Streaming	12
2.3.2	Live Map	13
3	Algorithms	14
3.1	Data Filtering	14
3.1.1	Text Filtering	14
3.1.2	Image Filtering	15
3.2	User TrustRank	16

4	Impact of Engineering Solutions	17
4.1	Global Impact	17
4.2	Social Impact	17
5	Tools and Technologies Used	19
5.1	OpenCV	19
5.2	Pandas	19
5.3	Numpy	19
5.4	Scikit-Learn	20
5.5	NLTK	20
5.6	Flask	20
5.7	Django	20
5.8	Git	21
5.9	Docker	21
5.10	GoogleCloud	21
5.11	DigitalOcean	22
5.12	Redis	22
5.13	MapBox	23
5.14	Firebase	23
5.15	Android	23
5.16	Wowza Streaming Engine	24
5.17	Wowza GoCoder SDK	24
6	Class Diagram	25
	References	34

1. Introduction

With the rise of social media platforms mainly in the last decade, the face of journalism and reporting has experienced a dramatic change from the traditional media sources and traditional reporting methods, to a more digital landscape, with the proliferation of Twitter and Facebook usage. According to a research by Pew Research Center, about four in ten Americans get their news online, and this number is expected to rise with the younger audience shifting to web sources [1]. This digitalization has not only made it more possible for media companies to find and reach a wider audience, but has also enabled non-professional civilians to partake in the news creation cycle. This has led to the rise of concepts such as citizen journalism and collaborative journalism.

Collaborative journalism is where multiple sources act together in order to create a news story, and not only professionals but also citizens can collaborate in this form of journalism. What we consume as news is usually a result of collaborative journalism, and the part citizen journalism plays in this collaboration is rising.

While there are numerous news outlets that provide collaborative journalism, and social media platforms that showcase these news pieces, there does not exist a popular platform focused exclusively on news that displays information on events not only via professional news outlets but also from the citizens' point of view. Furthermore, it often happens that we have to search extensively to be able to see beyond what algorithms choose as "appropriate/personalized" material, or to be able to escape filter bubbles we often find ourselves in nowadays on the web [2]. With Diafano, we propose to fill this void and create a platform that combines citizen stories with professional news, and provides multi-perspective, transparent journalism.

In this report, we aim to provide an overview of the low-level architecture and design of our system. First of all, the design trade-offs and the engineering standards are described. Also

the documentation guidelines are listed. Afterwards, the packages in our system and their functionalities are described along with detailed class diagram views. Furthermore, interfaces of all classes in all packages are included. We provide the clarifications of the descriptions of functionalities of each software component in the system.

1.1 Design Trade-Offs

1.1.1 Security vs. Cost

Diafano collects a lot of data from users, which is mainly multimedia format data, mainly in the form of videos or images. Thus, it is crucial for the system to ensure security and keep the information of the users secure. For security, we rely on encrypted databases. Relatedly, the security introduces monetary, time, and labor cost.

1.1.2 Space vs. Speed

The large scale of data also introduces a lot of possible lag which can be as a result of the requirement of saving the data to the servers. The photos and videos should be constantly stored upon user request, as well as details about geolocation or relates descriptions about them should be safely stored in the servers continuously and should be updated correctly. This naturally increases the processing time and slows down the system. In order to ensure fast response time, we will be saving all the data in the server time. In this way we will ensure that the system is highly responsive in all times. All connections and data exchanges with the server will be handled in background threads. This way, we will be able to keep the system fast while managing lots of data at the same time.

1.2 Engineering Standards

In the reports, UML [3] design principles are used in the description of class interfaces, diagrams, scenarios and use cases, subsystem compositions, and hardware-software components depiction. UML is a commonly used standard that allows simpler description of the compon-

ents of a software project. With standard UML models we were able to represent the system structure, software components, and functionalities.

1.3 Interface Documentation Guidelines

In the documentation, all class names are singular and named with the standard 'Camel-Case' format. The variable and method names follow the same convention 'variableName' and 'methodName()'. The class descriptions follow the hierarchy where the class name comes first, the attributes follow, and finally the methods are listed. After the class names, the description and function of the class is provided. The detailed outline is provided below:

ClassName

- Description of class

Attributes

- Attribute name
- Type of attribute

Methods

- Method name
- Parameters
- Return value

1.4 Definitions, Acronyms and Abbreviations

Some abbreviations and commonly used phrases are provided below:

Explore local: Browse section for the local news.

Explore global: Browse section for the global news.

Stream: Streaming button for video streaming live. Upon user's request, the video will be

stored/posted by user on the system to be later accessed by anyone whom the user has permitted via settings.

UI: User Interface

API: Application Programming Interface

HTTP: Hypertext Transfer Protocol

TCP: Transmission Control Protocol

UDP: User Datagram Protocol

Server: The part of the system responsible from logical operations, scheduling, and data management

Client: The part of the system the users interact with

2. Final Architecture and Design

2.1 Subsystem Decomposition

In the subsystem decomposition, the subsystem structure of our system is described in detail. First of all, the partitioning of the system is shown with diagrams and the classes within each layer are presented. Afterwards, hardware/software mapping of the system is provided which shows the allocation of resources and how various parts of our system will work in different hardware components. The persistent data management is also depicted; our database system and persistent objects are described. Access control and security defines the access boundaries of the users and how the security is managed in our system. In global software control, we illustrate how the software is controlled and how is the general flow in our system. Finally, in the boundary conditions, we specify the initialization, termination, and failure conditions.

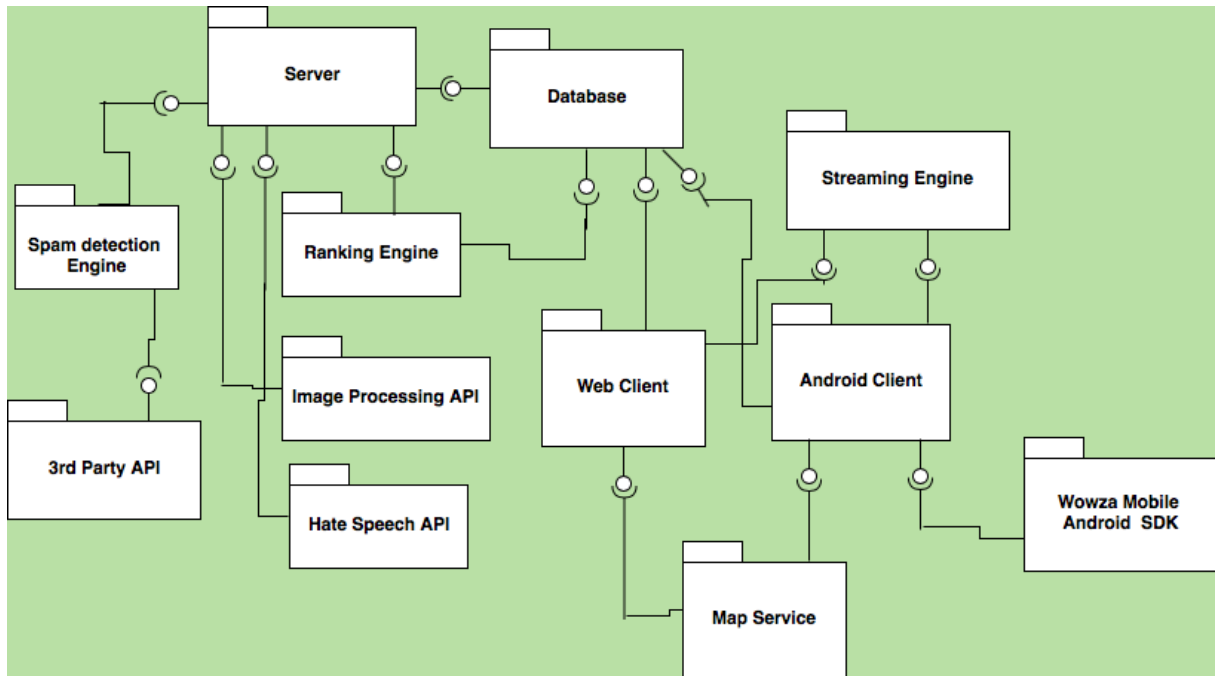


Figure 2.1: Subsystem Decomposition

2.1.1 Client

2.1.1.1 Presentation

Presentation package manages the client operations and that are related to the user interface. The package handles the views that the user should see on different scenarios.

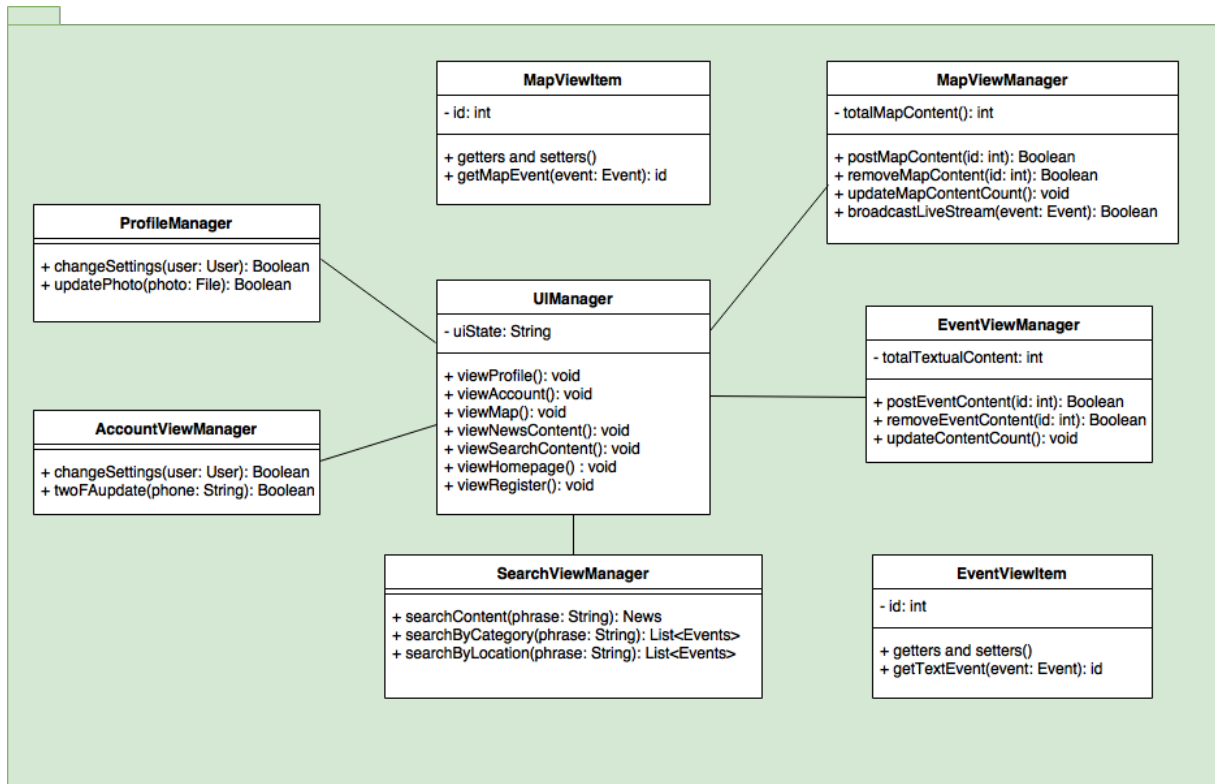


Figure 2.2: Presentation Package in Client

2.1.1.2 Controller

Controller package manages the client operations and the interaction between the client and the server.

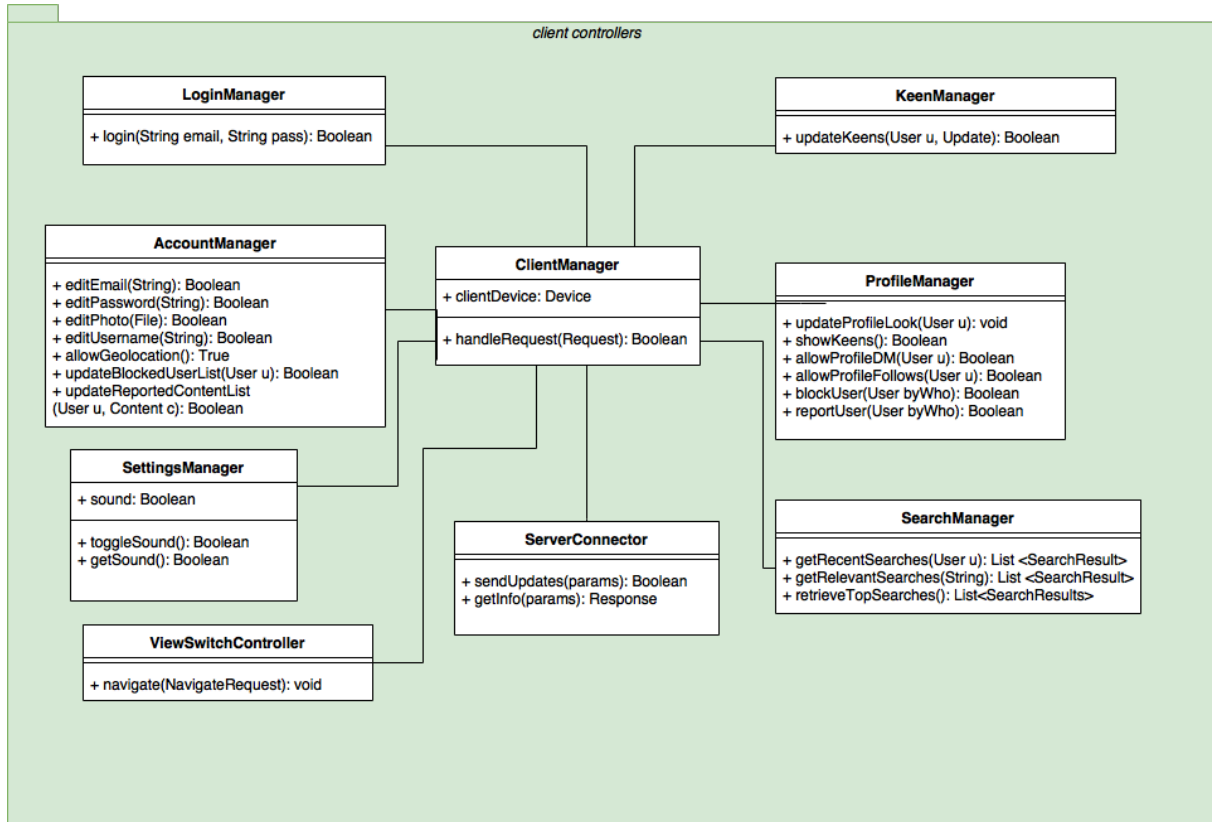


Figure 2.3: Controller Package in Client

2.1.2 Server

Server is an integral part of our system. The live video is recorded on the client side and delivered to the server. The server receives the data and binds it to event. The users requesting the video from an event will receive live video from the server. The server is responsible for consistent representation of the data and proper binding.

Furthermore server handles filtering operations. The filtering will allow user choose if they want to be exposed to a sensitive information. All the controllers are located on the server. It controls the persistency of the data and performs operation on top of it.

The server accommodates logic and data layers. In this way entities presented within this layer are allocated on the server. Logic tier is the back end application part responsible for handling requests and the data layer is the database responsible for permanent storage of entities in the application.

2.1.2.1 Logic

The Logic tier is application layer that controls the flow of information between presentation layer and data layer. This tier accommodates all the heavy operations the application needs to handle.

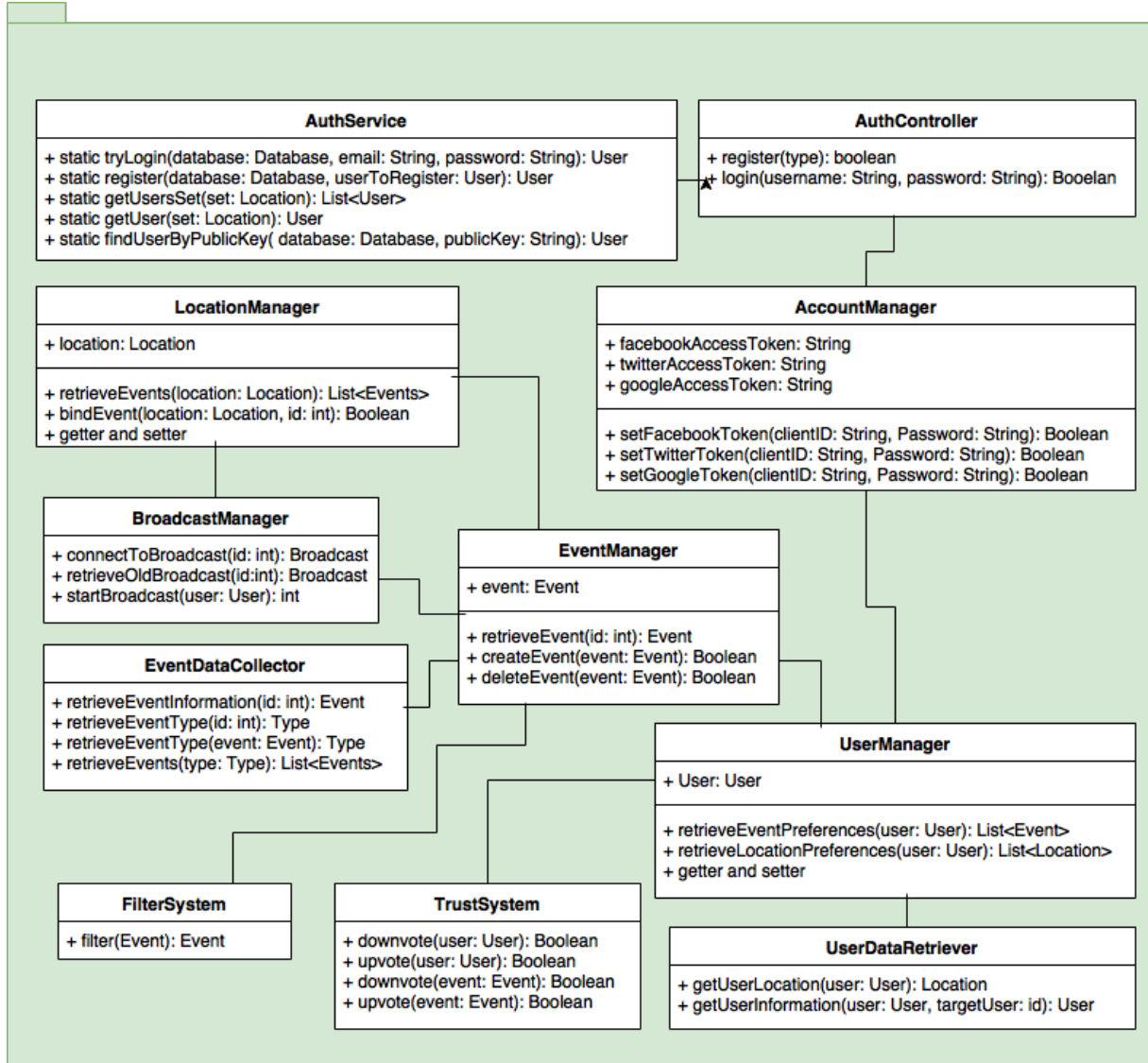


Figure 2.4: Logic Package in Server

2.1.2.2 Data

Data Tier is the lowest layer that provides permanent storage of the data. Basic retrieval mechanisms such as search are implemented in this layer. This layer stores the data in a

structured way and provides that data to the logic tier on demand.

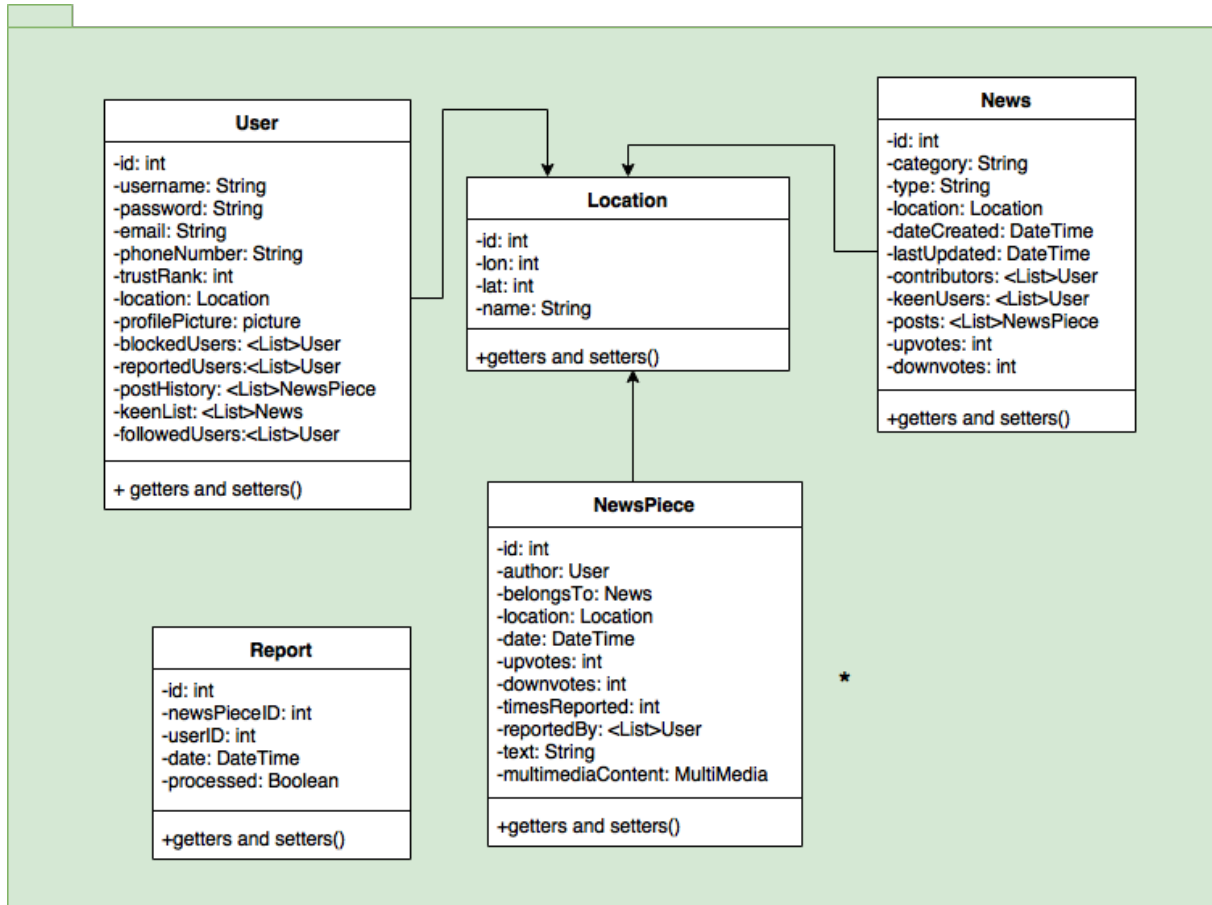


Figure 2.5: Data Subsystem in Server

2.2 Hardware Software Mapping

2.3 App and Web-app Design

2.3.1 Live Broadcasting and Video Streaming

Diafano is a platform for people to share their stories, and a nice and efficient way to share a story is through images, and videos. We have decided to utilize videos streaming and live broadcasting to make our platform more powerful and useful.

For this purpose we are using the Wowza Streaming Engine and Wowza GocoderSDK for

Android mobile devices. The Streaming Engine provides integration with the SDK for broadcasting videos in real time, as well as saving videos for playback or Video On Demand applications.

Users can see news or posts in their RSS feed, and chose to see a live broadcast or stream that was previously recorded.

2.3.2 Live Map

We offer a Map functionality, which is updated in real-time. Users can choose to navigate through the RSS feed or through the map. The benefits of the Map are that the users can see what is going on in the locations where its currently happening. We expect this to be the main screen of our app.

3. Algorithms

3.1 Data Filtering

Considering the initial aim of the system to create a safe unbiased community it is very important to control sensitive content. Platforms such as Facebook and Twitter suffer from the problem of fake news. One of the reasons is presence of a biased content. Filtering is a way to avoid biased content.

3.1.1 Text Filtering

In order to provide a smooth experience for users, we may consider problem of Fake News. One step towards solving this problem might be differentiating among hate speech, offensive language and regular text. For this purpose we use model offered by a collaboration between Carnegie Mellon University and Hamad Bin Khalifa University. In their work, researchers use multi-class classifier with 5-fold cross-validation to predict each of the three classes. They first use logistic regression with an L1 penalty to select the best features then use L2 regularization to predict the class (Linear SVC led to similar results). Overall the model gets an F1 score of 0.91, although performance is worst for the hate class.[4] The following features are used for the classification:

- Bigram, unigram, and trigram features, each weighted by its TF-IDF.
- Syntactic structure: sequences of one, two, and three adjacent Penn Part-of-Speech (POS) tags.
- Tweet quality: Flesch-Kincaid Reading Ease and Fleisch Readability scores.
- Tweet sentiment: amount of positive, negative, neutral, and overall sentiment of each tweet

using VADER sentiment.

- Binary and count indicators for hashtags, mentions, and URLs.
- Number of characters, words, and syllables.

Another part is spam classification for this purpose, we use a third-party api with a model trained to detect spam.

3.1.2 Image Filtering

In addition to text filtering we try to incorporate in our platform privacy values from the very beginning. In order to facilitate spreading of important news, we blur the faces of the people present on the images. For that purpose we apply face detection using Haar Cascade. The choice of the model is dictated by production constraints, where the speed of detection is critical, another advantage of the following algorithm is a low number of facial features used for face identification.

In order to make the service scalable, we use in-memory data structure store Redis. When new is stored the service is triggered. It retrieves the picture and puts it into the queue stored in Redis. In such a way prediction and queuing are run in parallel and the whole service becomes scalable to a larger cluster.

Haar-Cascade algorithm It is a machine learning based approach where a cascade function is trained from a lot of positive and negative images. It is then used to detect objects in other images.

Here we will work with face detection. Initially, the algorithm needs a lot of positive images (images of faces) and negative images (images without faces) to train the classifier. Then we need to extract features from it. For this, Haar features shown in the below image are used. They are just like our convolutional kernel. Each feature is a single value obtained by subtracting sum of pixels under the white rectangle from sum of pixels under the black rectangle. [5]

3.2 User TrustRank

We have adapted a version of the page-rank algorithm. We wanted a way to make our user trustable and somehow rank them according to what their content is about, as well as what other users "think" about them, in a continuous manner. Therefore we designed an algorithm that will consider both the previous rank, the current understanding of other users over this user through its content, number of followers/followings, and the content of the user and whether it is:

violent

sensitive

misleading

There is no analysis made on top of links between users, in order to detect spam in networks, but this is expected to be in the next step of this algorithm.

4. Impact of Engineering Solutions

4.1 Global Impact

Diafano brings a much needed collaborative social news platform to the market. By allowing its users to browse news from anywhere in the world as soon as they happen, all in one place, and from a variety of sources including citizen journalists, Diafano will have a lasting impact on every party involved.

Diafano provides transparency, and gives a voice to everyone.

4.2 Social Impact

Diafano is not positioning itself as a social network, instead it is a news platform aimed to reduce bias and eliminate information bubbles.

Currently we can identify two type of news outlets. Conventional news outlets and social networks. News outlets support information with an authority of their brands. However it is harder to distinguish less biased information on social networks. The situation becomes even more complicated if we think about the way the information is presented in the social networks. Recommendation engines integrated in people's news feeds do not allow those people step outside of the information bubbles created by this recommendation engines.

Diafano is a news platform, that incorporates bias reduction and elimination of information bubbles as its core priorities. We do this by constantly classifying data as hate-speech or spam. In addition to this we measure and classify engagement of users with a particular news. All the collected data contributes to the final trust rank score. Further we identify the level of bias of a particular user.

Another important value is privacy by design that we incorporate by anonymizing faces on uploaded images. As users upload news we blur faces and identify if provided text body can be classified as hate speech.

5. Tools and Technologies Used

5.1 OpenCV

OpenCV is an open source computer vision and machine learning software library. OpenCV was built to provide a common infrastructure for computer vision applications and to accelerate the use of machine perception in the commercial products. Being a BSD-licensed product, OpenCV makes it easy for businesses to utilize and modify the code. [5]



5.2 Pandas

Pandas is an open source, BSD-licensed library providing high-performance, easy-to-use data structures and data analysis tools for the Python programming language.[6]



5.3 Numpy

the fundamental package for scientific computing with Python.[7]



5.4 Scikit-Learn

Scikit-Learn is a free software machine learning library for the Python programming language. It features various classification, regression and clustering algorithms including support vector machines, random forests, gradient boosting, k-means and DBSCAN, and is designed to interoperate with the Python numerical and scientific libraries NumPy and SciPy.[8]



5.5 NLTK

Natural Language Toolkit is a leading platform for building Python programs to work with human language data. It provides easy-to-use interfaces to over 50 corpora and lexical resources such as WordNet, along with a suite of text processing libraries for classification, tokenization, stemming, tagging, parsing, and semantic reasoning, wrappers for industrial-strength NLP libraries, and an active discussion forum.[9]

5.6 Flask

Flask is a web microframework for Python based on Werkzeug, Jinja 2.[10]



5.7 Django

Django a high-level Python Web framework that encourages rapid development and clean, pragmatic design. Built by experienced developers, it takes care of much of the hassle of Web development, so you can focus on writing your app without needing to reinvent the wheel. It's free and open source. [11]



5.8 Git

Git is a free and open source distributed version control system designed to handle everything from small to very large projects with speed and efficiency. [12]



5.9 Docker

the company driving the container movement and the only container platform provider to address every application across the hybrid cloud. Today's businesses are under pressure to digitally transform but are constrained by existing applications and infrastructure while rationalizing an increasingly diverse portfolio of clouds, datacenters and application architectures. Docker enables true independence between applications and infrastructure and developers and IT ops to unlock their potential and creates a model for better collaboration and innovation. [13]



5.10 GoogleCloud

Platform, offered by Google, is a suite of cloud computing services that runs on the same infrastructure that Google uses internally for its end-user products, such as Google Search and YouTube. Alongside a set of management tools, it provides a series of modular cloud services including computing, data storage, data analytics and machine learning. Registration requires a credit card or bank account details.[14]



5.11 DigitalOcean

DigitalOcean is an American cloud infrastructure provider headquartered in New York City with data centers worldwide. DigitalOcean provides developers cloud services that help to deploy and scale applications that run simultaneously on multiple computers. As of December 2015, DigitalOcean was the second-largest hosting company in the world in terms of web-facing computers.[15]



5.12 Redis

an open source (BSD licensed), in-memory data structure store, used as a database, cache and message broker. It supports data structures such as strings, hashes, lists, sets, sorted sets with range queries, bitmaps, hyperloglogs and geospatial indexes with radius queries. Redis has built-in replication, Lua scripting, LRU eviction, transactions and different levels of on-disk persistence, and provides high availability via Redis Sentinel and automatic partitioning with Redis Cluster.[16]



5.13 MapBox

MapBox is a large provider of custom online maps for websites and applications such as Foursquare, Lonely Planet, Evernote, the Financial Times, The Weather Channel and Snapchat.[2] Since 2010, it has rapidly expanded the niche of custom maps, as a response to the limited choice offered by map providers such as Google Maps. Mapbox is the creator of, or a significant contributor to some open source mapping libraries and applications, including the MBTiles specification, the TileMill cartography IDE, the Leaflet JavaScript library, and the CartoCSS map styling language and parser.[17]



5.14 Firebase

a mobile and web application development platform developed by Firebase. [18]



5.15 Android

a mobile operating system developed by Google, based on a modified version of the Linux kernel and other open source software and designed primarily for touchscreen mobile devices such as smartphones and tablets. In addition, Google has further developed Android TV for televisions, Android Auto for cars, and Wear OS for wrist watches, each with a specialized user interface. Variants of Android are also used on game consoles, digital cameras, PCs and other electronics.[19]



5.16 Wowza Streaming Engine

a unified streaming media server software developed by Wowza Media Systems. The server is used for streaming of live and on-demand video, audio, and rich Internet applications over IP networks to desktop, laptop, and tablet computers, mobile devices, IPTV set-top boxes, internet-connected TV sets, game consoles, and other network-connected devices. The server is a Java application deployable on most operating systems.[20]

5.17 Wowza GoCoder SDK

is a cross-platform, extensible API for capturing and streaming live audio and video in iOS and Android mobile apps. The SDK supports a wide range of iOS and Android devices as well as 4K video resolution and other customizable encoder settings.[21]



6. Class Diagram

We will be omitting including an entire class diagram, as we have separately implemented a web and Android client for our system. Instead we will include the main classes in each of the Client and Server tiers. We can provide a full view diagram upon request.

Class Definitions in Client

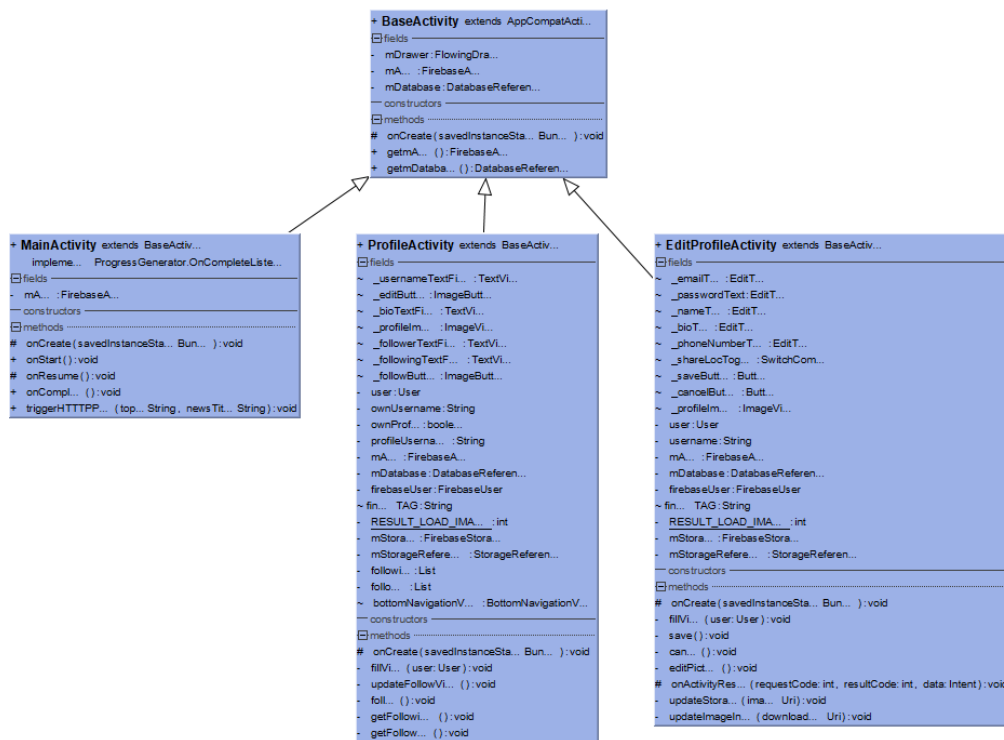


Figure 6.1: Presentation Package in Client, edit profile and settings

```

+ MapActivity extends AppCompatActivity
    implements MapboxMap.OnMapClickListener
               GoogleApiClient.OnConnectionFailedListener
               GoogleApiClient.ConnectionCallbacks

[-] fields
- mapView : MapView
- mapboxMap : MapboxMap
- marker : Marker
- final TAG : String
~ counter : int
- mPlaceDetectionClient : PlaceDetectionClient
~ mLikelyPlaceNames : String[]
~ mLikelyPlaceAddresses : String[]
~ mLikelyPlaceAttributions : String[]
~ mLikelyPlaceLatLngs : LatLng[]
- mGoogleApiClient : GoogleApiClient
- mLastKnownLocation : Location
- mCurrentLocation : Location
- mCameraPosition : CameraPosition
- mAuth : FirebaseAuth
- mDatabase : DatabaseReference
- firebaseUser : FirebaseUser
- clusterManagerPlugin : ClusterManagerPlugin<MyMarker>
- final KEY_CAMERA_POSITION : String
- final KEY_LOCATION : String
~ bottomNavigationView : BottomNavigationView
- clusterListView : ListView
~ llBottomSheet : LinearLayout
~ cluster : Cluster
~ bottomSheetBehavior : UserLockBottomSheetBehavior

[-] constructors

[-] methods
# onCreate (savedInstanceState: Bundle): void
- refresh(): void
+ onBackPressed(): void
- showClusterDetails (cluster: ArrayList<MyMarker> ): void
+ dispatchTouchEvent(event: MotionEvent): boolean
+ onMapClick (point: LatLng): void
+ onStart(): void
+ onResume(): void
+ onPause(): void
+ onStop(): void
+ onLowMemory(): void
# onDestroy(): void
# onSaveInstanceState (outState: Bundle): void
+ onConnectionFailed (connectionResult: ConnectionResult): void
+ onConnected (bundle: Bundle): void

```

```

+ CardFragment extends Fragment
  fields
  + listitems : ArrayList<News>
  ~ currentitems : ArrayList<News>
  ~ userLikes : HashMap<String, Integer>
  ~ userReports : HashMap<String, Integer>
  ~ MyRecyclerView : RecyclerView
  ~ mAdapter : MyAdapter
  ~ search : SearchView
  ~ _sortByText : TextView
  ~ _datesortText : TextView
  ~ toolbar : Toolbar
  ~ dateSort : DateSort
  ~ sort : Sort
  ~ searchQuery : String
  ~ mDatabase : DatabaseReference
  ~ username : String
  ~ bottomNavigationView : BottomNavigationView
  constructors
  methods
  + onCreate (savedInstanceState: Bundle):void
  + onCreateView (inflater: LayoutInflater , container: ViewGroup , savedInstanceState: Bundle):View
  + onCreateOptionsMenu (menu:Menu , inflater: MenuInflater ):void
  + onOptionsItemSelected (item: MenuItem ):boolean
  + onActivityCreated (savedInstanceState: Bundle):void
  + makeIntent (type:String , id:String , broadcastName:String):void
  + initializeList ():void
  + refreshNewsList (newitems: ArrayList<News> ):void
  + clear():void
  + sortAlert():void
  + datesortAlert():void
  + getTime (elapsedTime: long):String

```

Figure 6.3: Newsfeed Manager/View

```

+ CommentCardFragment extends Fragment
├── fields
~  commentsList:ArrayList<Comment>
~  commentKeys:ArrayList<String>
~  order:ArrayList<Integer>
~  userComments:HashMap<String, Integer>
~  userReports:HashMap<String, Integer>
~  MyRecyclerView :RecyclerView
~  mAdapter:MyAdapter
~  idData:String
~  mDatabase :DatabaseReference
~  username:String
~  sortText:TextView
~  sorttype:Sort
├── constructors
├── methods
+  onCreate (savedInstanceState: Bundle):void
+  onCreateView (inflater: LayoutInflater , container:ViewGroup, savedInstanceState: Bundle):View
+  onActivityCreated (savedInstanceState: Bundle):void
+  initializeCommentList (newsId:String):void
+  refreshCommentList (orderedList:ArrayList<Comment> , keysOrder:ArrayList<String>):void
+  sortAlert ():void
+  update (c:Comment, key:String):void

```

Figure 6.4: Comment View/Manager

Class Definitions in Server

+ **BroadcastActivity** extends GoCoderSDKBootstrapper
implements WZStatusCallback
View.OnClickListener
ActivityCompat.OnRequestPermissionsResultCallback

fields

- final hostAddress:String
- final portNumber:int
- final applicationName:String
- final TYPE:String
- final PERMISSIONS_REQUEST_LOCATION:int
- goCoder:WowzaGoCoder
- goCoderCameraView:WZCameraView
- goCoderAudioDevice:WZAudioDevice
- goCoderBroadcaster:WZBroadcast
- goCoderBroadcastConfig:WZBroadcastConfig
- controllerThumbnail:boolean
- final PERMISSIONS_REQUEST_CODE:int
- mPermissionsGranted:boolean
- mRequiredPermissions:String[]
- mFusedLocationClient:FusedLocationProviderClient
- mAuth:FirebaseAuth
- mDatabase:DatabaseReference
- firebaseUser:FirebaseUser
- mStorage:FirebaseStorage
- mStorageReference:StorageReference
- imageStorageReference:StorageReference
- profileUsername:String
- broadcastName:String
- broadcastLat:double
- broadcastLong:double
- title:String
- text:String
- currentTimestamp:long
- thumbnailPath:String
- images:List<String>
- urlThumbnail:String
- myAsyncClass:MyAsyncClass
- # mTimerView:TimerView
- # mStatusView:StatusView
- broadcastButton:ImageButton
- ~ runningNow:boolean
- ~ stoppingNow:boolean

constructors

methods

- runMyAsyncTask(ctx:Context, url:String):void
- # onCreate(savedInstanceState: Bundle):void
- # onResume():void
- + onRequestPermissionsResult(requestCode:int, permissions:String[], grantResults:int[]):void
- getLocation():void
- hasPermissions(context:Context, permissions:String[]):boolean
- + onClick(view:View):void
- setLatLng(lat:double, longitude:double):void
- + onWZStatus(goCoderStatus:WZStatus):void²⁹
- + saveThumbnailToDb():void
- + onWZError(goCoderStatus:WZStatus):void
- + onWindowFocusChanged(hasFocus:boolean):void

```

+ ClusterAdapter extends ArrayAdapter
[-] fields -----
- final TAG:String
- mContext:Context
- mResource:int
- lastPosition :int
+[-] constructors -----
+[-] methods -----
+ getView (position:int, convertView: View , parent:ViewGroup):View
+ moveToSingleNews(key:String):void
+ getTime(elapsedTime: long):String

```

```

+ Filter
[-] fields -----
~ comments:ArrayList<Comment>
~ commentKeys:ArrayList<String>
~ news:ArrayList<News>
~ query:String
~ date :DateSort
~ sorttype:Sort
+[-] constructors -----
+[-] methods -----
+ filterNews ():ArrayList<News>
+ filterComment ():sortReturn

```

Figure 6.6: Helpers modules

+ Hashtags
 fields
 ~ indices :int[]
 ~ text:String
 constructors
 methods

+ NewsComment
 fields
 + commentList:List<Comment>
 constructors
 methods

+ MediaPath
 fields
 + broadcast_name:String
 + image_path:List<String>
 constructors
 methods

+ Location
 fields
 + latitude:double
 + longitude:double
 constructors
 methods

+ Comment
 fields
 + author:String
 + downvotes:int
 + date_created:long
 + upvotes:int
 + txt:String
 + misleading:int
 + violent:int
 + possibly_sensitive:int
 constructors
 methods

+ News
 fields
 + author:String
 - author_verified:boolean
 + date_created:long
 + downvotes:int
 + followers_count:int
 + last_modified:String
 + location:Location
 + media_path:MediaPath
 + misleading:int
 + possibly_sensitive:double
 + type:String
 + title:String
 + txt:String
 + upvotes:int
 + violent:int
 constructors
 methods

+ User
 fields
 + bio:String
 + email:String
 + location:Location
 + location_share:boolean
 + name:String
 + phone_number:String
 + profile_picture:String
 + trust_rank:double
 constructors
 methods

+ MapPost
 fields
 + obj:HashMap<String, HashMap>
 + type:String
 constructors
 methods


```

+ MyJobService extends JobService
  fields
  - final TAG:String
  constructors
  methods
  + onStartJob(jobParameters: JobParameters ):boolean
  + onStopJob(jobParameters: JobParameters ):boolean

```

```

+ MyFirebaseMessagingService extends FirebaseMessagingService
  fields
  - final TAG:String
  constructors
  methods
  + onMessageReceived (remoteMessage: RemoteMessage ):void
  - scheduleJob():void
  - handleNow():void
  - sendNotification (messageBody:String):void

```

```

+ MyFirebaseInstanceIdService extends FirebaseInstanceIdService
  fields
  - final TAG:String
  constructors
  methods
  + onTokenRefresh():void
  - sendRegistrationToServer(token:String):void

```

Figure 6.8: Notifications' module

Bibliography

- [1] Pew Research Center: How Americans Get Their News, 2016. <http://www.journalism.org/2016/07/07/pathways-to-news/>. Accessed: 2017-10-8.
- [2] Beware Online Filter Bubbles, ted.com, 2011. https://www.ted.com/talks/eli_pariser_beware_online_filter_bubbles. Accessed: 2017-10-8.
- [3] UML - Basics , ibm. <http://www.ibm.com/developerworks/rational/library/769.html>. Accessed: 2018-01-04.
- [4] Automated hate speech detection and the problem of offensive language. <https://arxiv.org/pdf/1703.04009.pdf>. Accessed: 2018-02-25.
- [5] Opencv. <https://opencv.org/>. Accessed: 2018-01-05.
- [6] Pandas. <https://pandas.pydata.org/>. Accessed: 2018-01-05.
- [7] Numpy. <http://www.numpy.org/>. Accessed: 2018-01-05.
- [8] Scikit-learn. <http://scikit-learn.org/stable/>. Accessed: 2018-01-05.
- [9] Nltk. <https://www.nltk.org/>. Accessed: 2018-01-05.
- [10] Flask. <http://flask.pocoo.org/>. Accessed: 2018-01-05.
- [11] Django. <https://www.djangoproject.com/>. Accessed: 2018-01-05.
- [12] Git. <https://git-scm.com/>. Accessed: 2018-01-05.
- [13] Docker. <https://www.docker.com/>. Accessed: 2018-01-05.

- [14] Google cloud. <https://cloud.google.com/>. Accessed: 2018-01-05.
- [15] Digital ocean. <https://www.digitalocean.com/>. Accessed: 2018-01-05.
- [16] Redis. <https://redis.io/>. Accessed: 2018-01-05.
- [17] Mapbox. <https://https://www.mapbox.com/>. Accessed: 2018-01-05.
- [18] Firebase. <https://firebase.google.com/>. Accessed: 2018-01-05.
- [19] Android. <https://www.android.com/>. Accessed: 2018-01-05.
- [20] Wowza streaming engine. <https://www.wowza.com/>. Accessed: 2018-01-05.
- [21] Wowza gocoder sdk. <https://www.wowza.com/>. Accessed: 2018-01-05.