Department of Computer Engineering

Senior Design Project

Low Level Design Report

Supervisor:
Asst. Prof. Mustafa Özdal
Jury Members:
Asst. Prof. Ercüment Çiçek
Asst. Prof. Hamdi Dibeklioğlu
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

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 components 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 he 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. Packages

Our system is composed of 4 packages: Presentation, Controller, Logic, and Data packages. Presentation and Controller packages are in Client subsystem while the Logic and Data packages are in Server subsystem.

2.1 Client

2.1.1 Presentation Package

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.

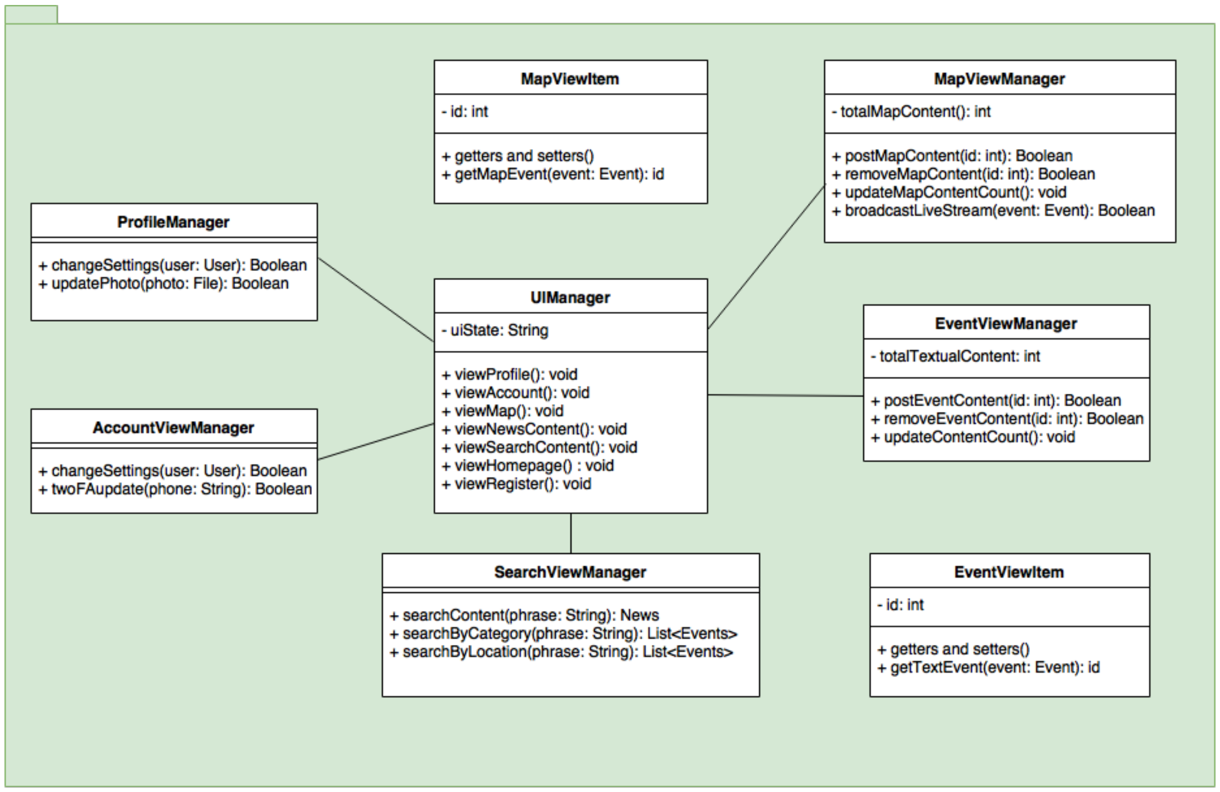
MapViewItem
- id: int
+ getters and setters()
+ getMapEvent(event: Event): id
MapViewManager
- totalMapContent(): int
+ postMapContent(id: int): Boolean
+ removeMapContent(id: int): Boolean
+ updateMapContentCount(): void
+ broadcastLiveStream(event: Event): Boolean
ProfileManager
+ changeSettings(user: User): Boolean
+ updatePhoto(photo: File): Boolean
UIManager
- uiState: String
+ viewProfile(): void
+ viewAccount(): void
+ viewMap(): void
+ viewNewsContent(): void
+ viewSearchContent(): void
+ viewHomepage() : void
+ viewRegister(): void
EventViewManager
- totalTextualContent: int
+ postEventContent(id: int): Boolean
+ removeEventContent(id: int): Boolean
+ updateContentCount(): void
AccountViewManager
+ changeSettings(user: User): Boolean
+ twoFAupdate(phone: String): Boolean
SearchViewManager
+ searchContent(phrase: String): News
+ searchByCategory(phrase: String): List<Events>
+ searchByLocation(phrase: String): List<Events>
EventViewItem
- id: int
+ getters and setters()
+ getTextEvent(event: Event): id

Figure 2.1: Presentation Package in Client

- **UIManager:** The class where every other view manager classes are connected to. The user interface is managed via this class' methods that are executed from the controller's decisions.
- **ProfileManager:** This class is responsible from displaying the profile of a user and permits to interact with the interface while changing their settings.
- **AccountViewManager:** Apart from the properties in the ProfileManager class, other settings like two-factor authentication and password settings is done via this class' methods.
- **SearchViewManager:** This class handles how the search view of news will be displayed on the user interface.
- **MapViewItem:** This class is an instance of the view that a live video broadcast news will form on the user interface.

- **EventViewItem:** This class is an instance of the view that a textual news will form on the user interface.
- **MapViewManager:** Handles the main controls over the map of the live videos with utilizing MapViewItem instances.
- **EventViewManager:** Handles how the instances of EventViewItem is expected to appear on the user interface.

2.1.2 Controller Package

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

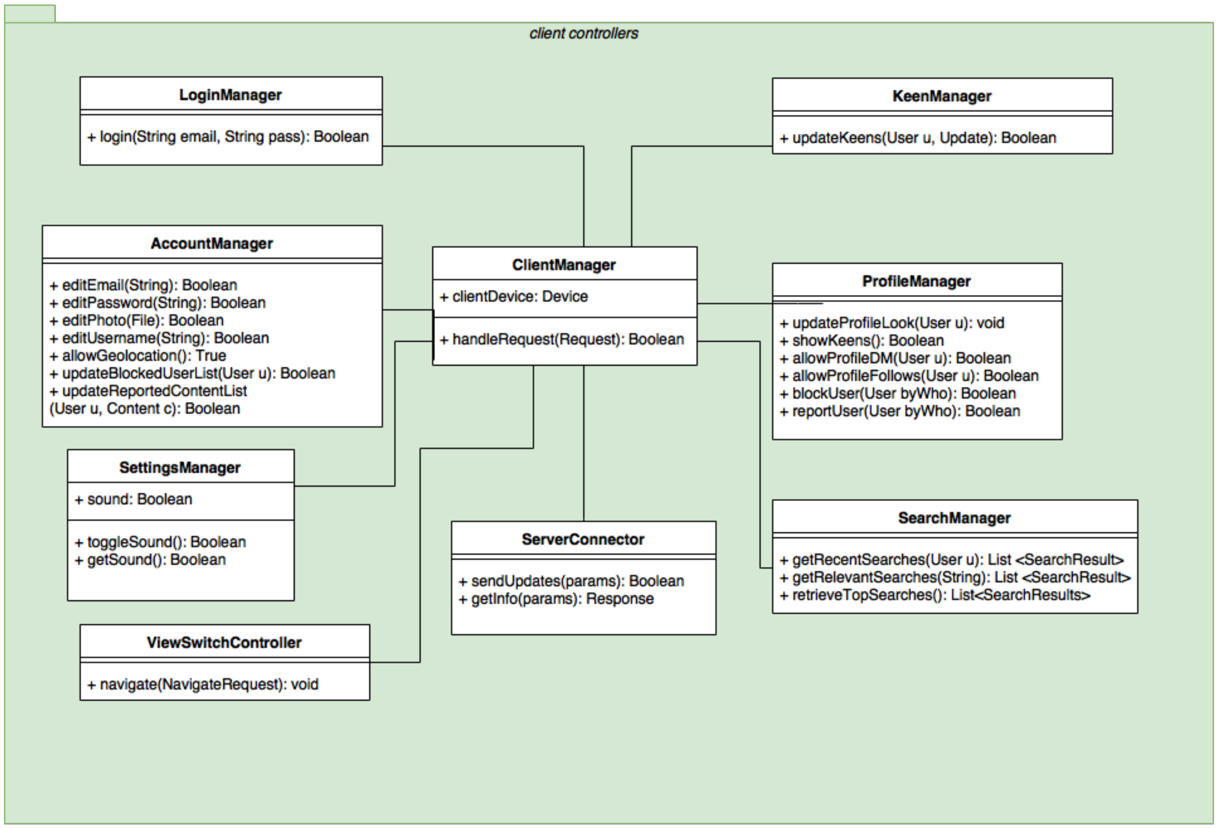
client controllers
LoginManager
+ login(String email, String pass): Boolean
KeenManager
+ updateKeens(User u, Update): Boolean
AccountManager
+ editEmail(String): Boolean
+ editPassword(String): Boolean
+ editPhoto(File): Boolean
+ editUsername(String): Boolean
+ allowGeolocation(): True
+ updateBlockedUserList(User u): Boolean
+ updateReportedContentList
(User u, Content c): Boolean
ClientManager
+ clientDevice: Device
+ handleRequest(Request): Boolean
ProfileManager
+ updateProfileLook(User u): void
+ showKeens(): Boolean
+ allowProfileDM(User u): Boolean
+ allowProfileFollows(User u): Boolean
+ blockUser(User byWho): Boolean
+ reportUser(User byWho): Boolean
SettingsManager
+ sound: Boolean
+ toggleSound(): Boolean
+ getSound(): Boolean
ServerConnector
+ sendUpdates(params): Boolean
+ getInfo(params): Response
SearchManager
+ getRecentSearches(User u): List <SearchResult>
+ getRelevantSearches(String): List <SearchResult>
+ retrieveTopSearches(): List<SearchResults>
ViewSwitchController
+ navigate(NavigateRequest): void

Figure 2.2: Controller Package in Client

- **LoginManager:** This class will allow for the user to login to the system by communicating further down with the Logic Tier.

- **AccountManager:** This class will allow for the user to edit information it has previously provided to the system regarding email, password, username, avatar, geolocation related permissions, as well as lists of users it blocks or content it reports.
- **KeenManager:** This class will handle keens of a certain user, upon user request to client. Keens include certain news that user wants to follow.
- **ClientManager:** This class handles requests of different managers in the tier, and communicates with ServerConnector to reach the Server tier for updating/retrieving or storing information.
- **SettingsManager:** This class handles device sound of the broadcasts. The current settings allows for toggling of sound in the device on/off.
- **SearchManager:** This class serves as the controller for the searching capability of our system.
- **ProfileManager:** Handles general controls of the capabilities of random users on another profile, like recent broadcasts or data user has shared/upvoted or users keens, Direct Messaging capability, follows, reports or blocks of a user.
- **ServerConnector:** Handles requests to server.
- **ViewSwitchController:** Handles requests to switch between different tabs.

2.2 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.

2.2.1 Logic Package

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.

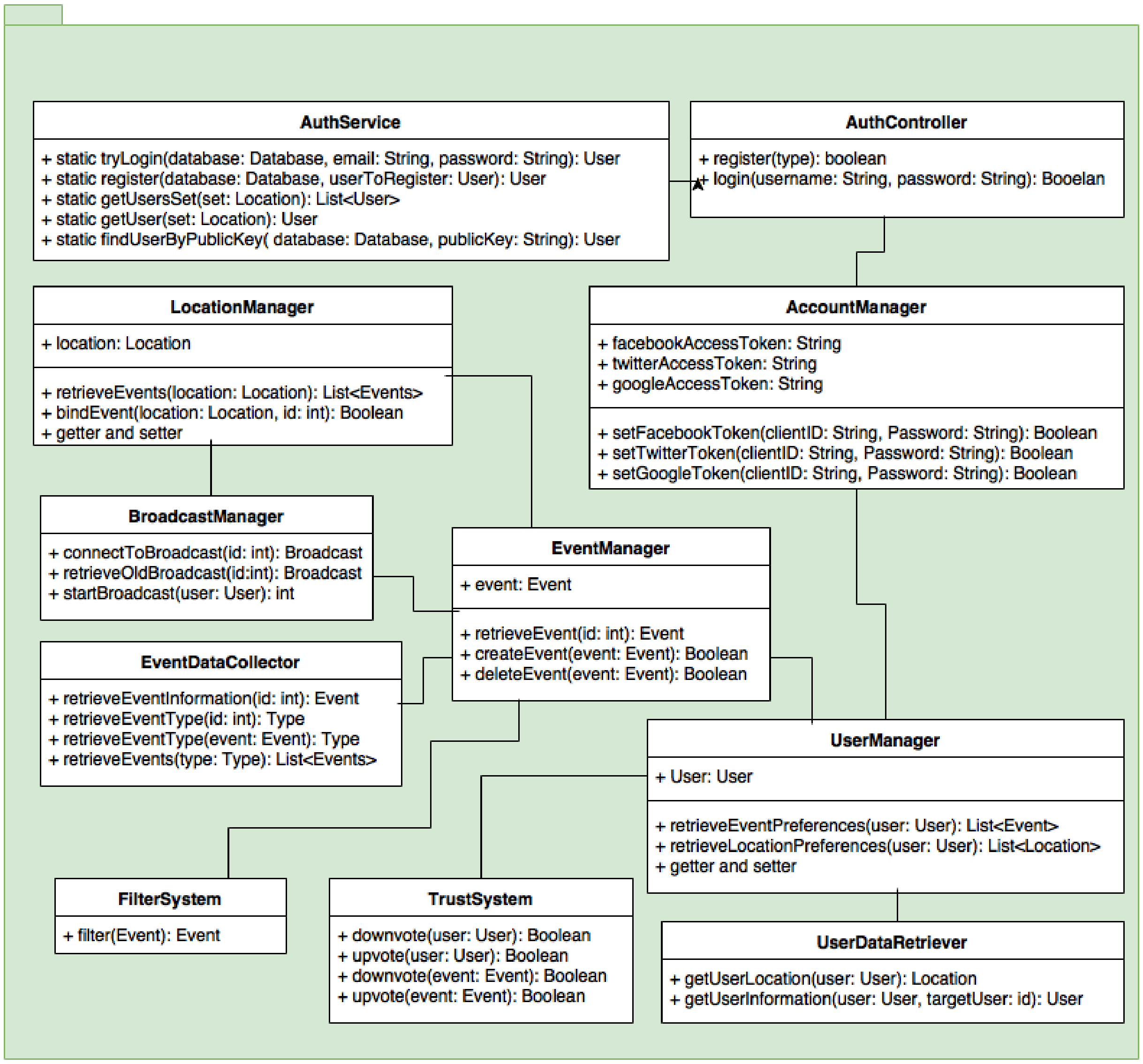
AuthService
+ static tryLogin(database: Database, email: String, password: String): User
+ static register(database: Database, userToRegister: User): User
+ static getUsersSet(set: Location): List<User>
+ static getUser(set: Location): User
+ static findUserByPublicKey(database: Database, publicKey: String): User
AuthController
+ register(type): boolean
+ login(username: String, password: String): Booelan
LocationManager
+ location: Location
+ retrieveEvents(location: Location): List<Events>
+ bindEvent(location: Location, id: int): Boolean
+ getter and setter
AccountManager
+ facebookAccessToken: String
+ twitterAccessToken: String
+ googleAccessToken: String
+ setFacebookToken(clientID: String, Password: String): Boolean
+ setTwitterToken(clientID: String, Password: String): Boolean
+ setGoogleToken(clientID: String, Password: String): Boolean
BroadcastManager
+ connectToBroadcast(id: int): Broadcast
+ retrieveOldBroadcast(id:int): Broadcast
+ startBroadcast(user: User): int
EventManager
+ event: Event
+ retrieveEvent(id: int): Event
+ createEvent(event: Event): Boolean
+ deleteEvent(event: Event): Boolean
EventDataCollector
+ retrieveEventInformation(id: int): Event
+ retrieveEventType(id: int): Type
+ retrieveEventType(event: Event): Type
+ retrieveEvents(type: Type): List<Events>
UserManager
+ User: User
+ retrieveEventPreferences(user: User): List<Event>
+ retrieveLocationPreferences(user: User): List<Location>
+ getter and setter
FilterSystem
+ filter(Event): Event
TrustSystem
+ downvote(user: User): Boolean
+ upvote(user: User): Boolean
+ downvote(event: Event): Boolean
+ upvote(event: Event): Boolean
UserDataRetriever
+ getUserLocation(user: User): Location
+ getUserInformation(user: User, targetUser: id): User

Figure 2.3: Logic Package in Server

- **EventManager:** Fundamental class that will be responsible for events. It will gather all the information about the event.
- **EventDataColector:** This class is used to retrieve data related to an event. The data to be retrieved is incorporated in a single entity, location might be retrieved separately.
- **BroadcastManager:** Class that is responsible for broadcasting. It will manage the interaction between the client that is broadcasting with the server as well as clients that will watch the broadcast through the server.

- **FilterSystem:** This class is used to filter the events and their content post by the users.
- **LocationManager:** Class that is responsible for dealing with news and users' location data.
- **UserManager:** This class handles all the preferences, notifications about the user.
- **UserDataRetriever:** This class retrieves a user related data. The data is given according to restrictions set by an owner of the data.
- **TrustSystem:** Class that is responsible for assigning a trust score to each user using the data of their news posts, votes, reports and more.
- **AccountManager:** This class is used to maintain accounts of users.
- **AuthController:** This class is used to register users and sign them in.
- **AuthService:** This is class is used for saving and retrieving information about users from the database.

2.2.2 Data Package

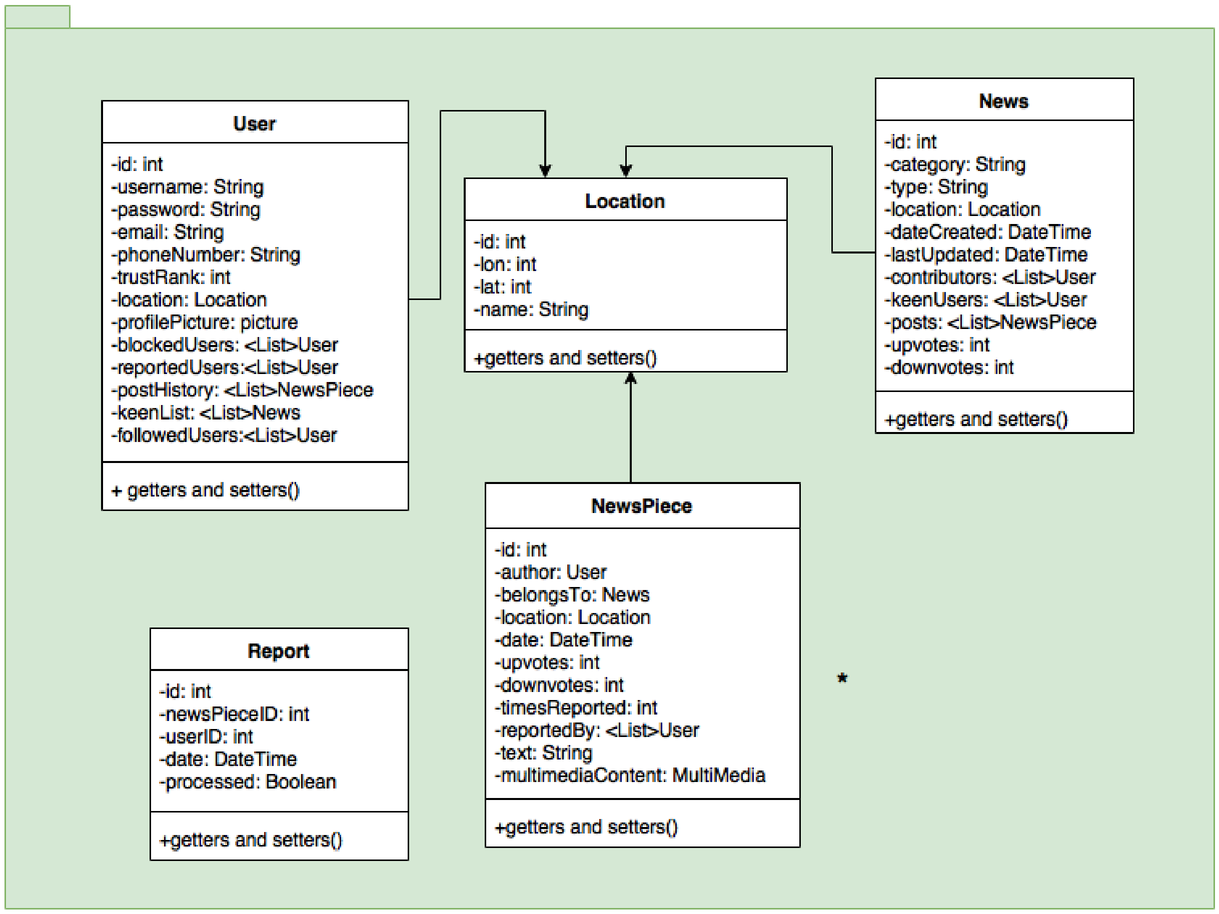

Figure 2.4: Data Subsystem in Server

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.

- **User:** Data class representing users of Diafano. User class includes data such as user email, nickname, phone number, trustrank, and other information.
- **Location:** Data class that keeps information on locations, such as their name, people close to that location, latest events that have happened there.
- **News:** Data class that keeps information about a news topic, including its starting date, sources, multiple contents, location, category.

- **NewsPiece:** Data class representing a single news piece written by a certain user. NewsPiece class includes data such as its author, news topic it belongs to, location, time, upvotes and downvotes.
- **Report:** Data class that keeps information about reports by users, such as user id, news piece id, and whether the report has been processed or not.

3. Class Interfaces

3.1 Client

3.1.1 Presentation

class MapViewItem	
The class that is responsible from displaying map items as news.	
Attributes	
private int id	
Methods	
public id getMapEvent(Event event)	This method retrieves an event id entity given an Event object on the map.
getters and setters	

class EventViewItem	
The class that is responsible from displaying the textual news in the feed.	
Attributes	
private int id	
Methods	
public id getTextEvent(Event event)	This method retrieves an event id entity given an Event object that lies on the news feed as text.
getters and setters	

class MapViewManager	
The class that is responsible from managing the live broadcasts and pins on the map.	
Attributes	
private int totalMapContent	
Methods	
public Boolean postMapContent(int id)	This method posts the event to the map given an event id and creates a MapViewItem.
public Boolean removeMapContent (int id)	This method removes the event from the map given an event id.
public void updateMapContentCount()	This method updates the attribute totalMap-Content according to the changes happen through addition and deletion.
public Boolean broadcastLiveStream (Event event)	When a live broadcast is started, this method is called and the live stream is shown both in the map and the news feed.

class EventViewManager	
The class that is responsible from displaying and managing the textual news in the feed.	
Attributes	
private int totalTextualContent	
Methods	
public Boolean postEventContent(int id)	This method posts the textual event to the news feed given an event id and creates a EventVie-wItem.
public Boolean removeEventContent (int id)	This method removes the textual event from the news feed given an event id.
public void updateContentCount()	This method updates the attribute totalTextu-alContent according to the changes happen to the feed through addition and deletion.

class SearchViewManager	
The class that is responsible from displaying searching news in the application.	
Methods	
public News searchContent(String phrase)	This method searches the News given a specific phrase as news. Returns the most appropriate news related.
public List<Event>searchByCategory (String phrase)	This method searches the news/events given a category available in the category list.
public List<Event>searchByLocation (String phrase)	This method searches the news/events given a location available in the location list

class ProfileManager	
The class that is responsible from managing the contents specific to user profiles.	
Methods	
public Boolean changeSettings(User user)	This method retrieves the changes from the text fields in the settings page and updates the user with the changes.
public Boolean updatePhoto(File file)	Given a photo as a file, user's photo is updated.

class AccountViewManager	
The class that is responsible from managing the contents of the account specific data.	
Methods	
public Boolean changeSettings (User user)	This method retrieves the changes from the text fields in the settings page and updates the user with the changes.
public Boolean twoFAupdate (String phone)	Given a phone number as a string starting with two zeros '00', user's account is connected with a phone number of their choice.

class UIManager	
The class that is responsible from displaying all the contents.	
Attributes	
public String uiState	
Methods	
public void viewProfile()	This method is executed when the user wants to see a profile.
public void viewAccount()	This method is executed when the user wants to their own account settings.
public void viewMap()	This method is executed when the user wants to open the map for live broadcasts.
public void viewNewsContent()	This method is executed when details are expanded for textual news . Other operations related to the news can be performed after this method.
public void viewSearchContent()	This method is executed after a user performs a search and is displayed with the results.
public void viewHomepage()	This method is executed initially as a landing page.
public void viewRegister()	This method is executed when a new user is registering to the application.

3.1.2 Controller

class LoginManager	
-	
Methods	
public boolean login(String email, String password)	This method allows user to login into the system.

class KeenManager	
-	
Methods	
public boolean updateKeens(User u, Update)	This method allows user to follow some news he keens, and will update the keens of particular user accordingly.

class AccountManager	
-	
Methods	
public boolean editEmail(String newE-mail)	This method allows users to edit their provided emails.
public boolean editPassword(String new-Pass)	This method allows users to edit their provided passwords.
public boolean editPhoto(File photo)	This method allows users to add or edit their provided avatars.
public boolean editUsername(String newEmail)	This method allows users to edit their provided usernames.
public boolean allowGeolocation()	This method allows users to allow their geo-location for the content they share.
public boolean updateBlockedUserL-ist(User u, User blocked)	This method helps update the list of users User u has blocked, namely when they chose to block another user.
public boolean updateReportedContent-List(User u, Content c)	This method helps update users report on some content.

class ProfileManager	
-	
Methods	
public void updateProfileLook(User u)	Here the user can change the main color of the application (basically how application looks).
public boolean showKeens(String newPass)	This method allows users to see their keens.
public boolean allowProfileDM(File photo)	This method allows users to allow other users to send a direct message to them.
public boolean allowProfileFollows(User u)	This method allows users to allow other users to follow them.
public boolean blockUser(User whom, User byWho)	This method helps a user block another user.
public boolean reportUser(User byWho, User whom)	This method helps reporting of a user from another user.

class ViewSwitchController	
-	
Methods	
public void navigate(NavigateRequest)	This class handles the movement changes in the tabs.

class ServerConnector	
-	
Methods	
public void sendUpdates(params)	Parameters change according to requests - generic method.
public boolean getInfo(params)	Retrieves information from the servers.

class SearchManager	
-	
Methods	
public List⟨SearchResult⟩ getRecentSearches(User u)	Returns a list of results for recent searches the user has done.
public List⟨SearchResult⟩ getRelevantSearches(String)	This method returns a list of search results related to keyword of search.
public List⟨SearchResult⟩ retrieveTopSearches()	This method returns a list of top searches overall - this is relevant in cases of popular news.

3.2 Server

3.2.1 Logic

class EventManager	
Fundamental class that will be responsible for events. It will gather all the information about the event.	
Attributes	
private Event event	
Methods	
public Event retrieveEvent(int id)	This method retrieves an Event entity given an id for an Event.
public Boolean createEvent(Event event)	This method creates an Event from the system entity given an Event.
public Boolean deleteEvent(Event event)	This method deletes an Event from the system entity given an Event.

class EventDataColector	
This class is used to retrieve data related to an event. The data to be retrieved is incorporated in a single entity, location might be retrieved separately.	
Methods	
public Event retrieveEventInformation(int id)	This method retrieves an Event entity given an id for an Event.
public Type retrieveEventType(int id)	This method returns a type of an event given an event id.
public Type retrieveEventType(Event event)	This method returns a type of an event given an Event entity.
public List ⟨ Events ⟩ retrieveEvents(Type type)	This method returns a list of events given a type for events.

class BroadcastManager	
Class that is responsible for broadcasting. It will manage the interaction between the client that is broadcasting with the server as well as clients that will watch the broadcast through the server.	
Methods	
public Broadcast connectToBroadCast(int id)	This method connects a User to a broadcast given an id of a broadcast.
public Broadcast retrieveOldBroadcast(int id)	This method retrieves and old broadcast from an archive given an id of a broadcast.
public int startBroadcast(User user)	This method initiates a new broadcast session for a given User and returns an id of an initiated broadcast.

class FilterSystem	
This class is used to filter the events and their content post by the users.	
Methods	
public Event filterEvent(Event event)	This method filters sensitive data from an event and returns the processed Event.

class LocationManager	
Class that is responsible for dealing with news and users' location data.	
Attributes	
private Location location	
Methods	
public List ⟨Events⟩ retrieveEvents(Location location)	This method retrieves a list of Events for a particular location.
public Boolean bindEvent(Location location, int id)	This method binds an Event to a given location.
getters and setters	

class UserManager	
This class handles all the preferences, notifications about the user.	
Attributes	
private User user	
Methods	
public List⟨Event⟩ retrieveEventpreferences(User user)	This method retrieves a list of events created according to the preferences of a user.
public List⟨Location⟩ retrieveEventpreferences(User user)	This method retrieves a list of locations created according to the preferences of a user.
getters and setters	

class UserDataRetriever	
This class retrieves a user related data. The data is given according to restrictions set by an owner of the data.	
Methods	
public Location getUserLocation(User user)	This method retrieves a lcoation of a given user.
public User getUserInformation(User user, int targetUser)	This method retrieves an information about a target user given an id of a target user. Information is provided according to permission set by a target user.

class TrustSystem	
Class that is responsible for assigning a trust score to each user using the data of their news posts, votes, reports and more.	
Methods	
public Boolean downvote(User user)	This method downvotes the user, in this way the rank of a user is decreased.
public Boolean upvote(User user)	his method upvotes the user, in this way the rank of a user is increased.
public Boolean downvote(Event event)	his method downvotes the Location, in this way the rank of a Locaiton is decreased.
public Boolean upvote(Event event)	This method upvotes the Location, in this way the rank of a Location is increased.

class AccountManager	
This class is used to maintain accounts of users.	
Attributes	
private String facebookAcessToken	This attribute represents the OAuth 2.0 token required to access the user's Facebook account (if allowed by the user).
private String twitterAcessToken	This attribute represents the OAuth 2.0 token required to access the user's Twitter account (if allowed by the user).
private String googleAcessToken	This attribute represents the OAuth 2.0 token required to access the user's Google account (if allowed by the user).
Methods	
getters	
public Boolean setFacebookToken(String clientID, String password‘)	This method uses the OAuth 2.0 to get a token for the user's Facebook account.
public Boolean setTwitterToken(String clientID, String password‘)	This method uses the OAuth 2.0 to get a token for the user's Twitter account.
public Boolean setGoogleToken(String clientID, String password‘)	This method uses the OAuth 2.0 to get a token for the user's Google account.

class AuthController	
This class is used to register users and sign them in.	
Methods	
public Boolean register(Event event)	This method registers a user given a required information. With a given information it hashes the password and stores all the information in the database
public Boolean login(String username, String password)	This method signs in a user given a right combination of a username and a password.

class AuthService	
This is class is used for saving and retrieving information about users from the database. events. It will gather all the information about the event.	
Methods	
public static User tryLogin(Database database, String email, String password)	Given a user name and a password this method checks if a user is not null and a hashed password exists in the database.
public static User register(Database database, User userToRegister)	This method creates a new user in the database
public static List ⟨User⟩ getUsersSet(Location set)	This method returns a list of users in a given location given as a set.
public static User getUser(Location set)	This method returns a random user from a location.
public static User getUserByPublicKey(Database database, String publicKey)	Given a public key this method returns a user that satisfies given information.

3.2.2 Data

class User	
Data class representing users of Diafano.	
Attributes	
private int id	
private String username	
private String password	
private String email	
private String phoneNumber	
private int trustRank	
private Location location	
private Picture profilePicture	
private List⟨User⟩ blockedUsers	
private List⟨User⟩ reportedUsers	
private List⟨NewsPiece⟩ postHistory	
private List⟨News⟩ keenList	
private List⟨User⟩ followedUsers	
Methods	
getters and setters	

class News	
Data class that keeps information about a news topic.	
Attributes	
private int id	
private String category	
private String type	
private String name	
private Location location	
private DateTime dateCreated	
private DateTime lastUpdated	
private List⟨User⟩ contributors	
private List⟨User⟩ keenUsers	
private List⟨NewsPiece⟩ posts	
private int upvotes	
private int downvotes	
Methods	
getters and setters	

class NewsPiece	
Data class that keeps information about a single news piece.	
Attributes	
private int id	
private User author	
private News belongsTo	
private Location location	
private DateTime date	
private String text	
private MultiMedia multimediaContent	
private int timesReported	
private List⟨User⟩ reportedBy	
private int upvotes	
private int downvotes	
Methods	
getters and setters	

class Location	
Data class that keeps information about a location.	
Attributes	
private int id	
private int lon	
private int lat	
private String name	
Methods	
getters and setters	

class Report	
Data class that keeps information about reports.	
Attributes	
private int id	
private int newsPieceID	
private int userID	
private DateTime date	
private Boolean processed	
private String explanation	
Methods	
getters and setters	

4. Glossary

A list of phrases of ambiguity is provided below:

Keen: It is a very similar concept to subscriptions, namely a user ”keens” some news and receives notifications or updates whenever anything new happens related to them.

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.