Oceanus Documentation

수묵화 트위터

Entertainment Technology Center

TABLE OF CONTENTS

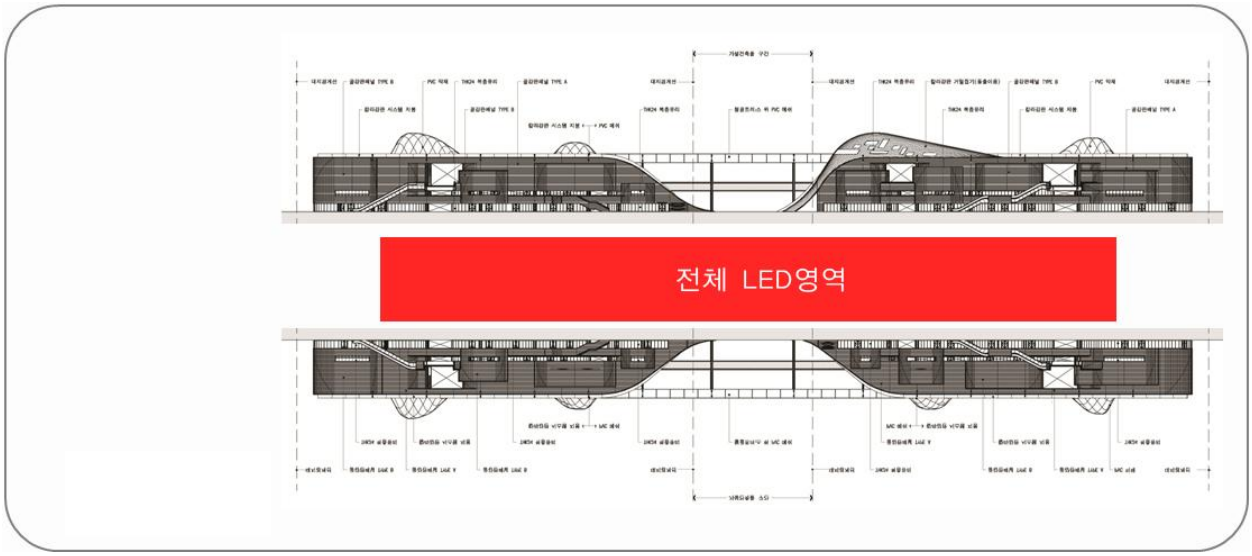
TWITTER INK PAINTINGS

PRODUCTION OVERVIEW

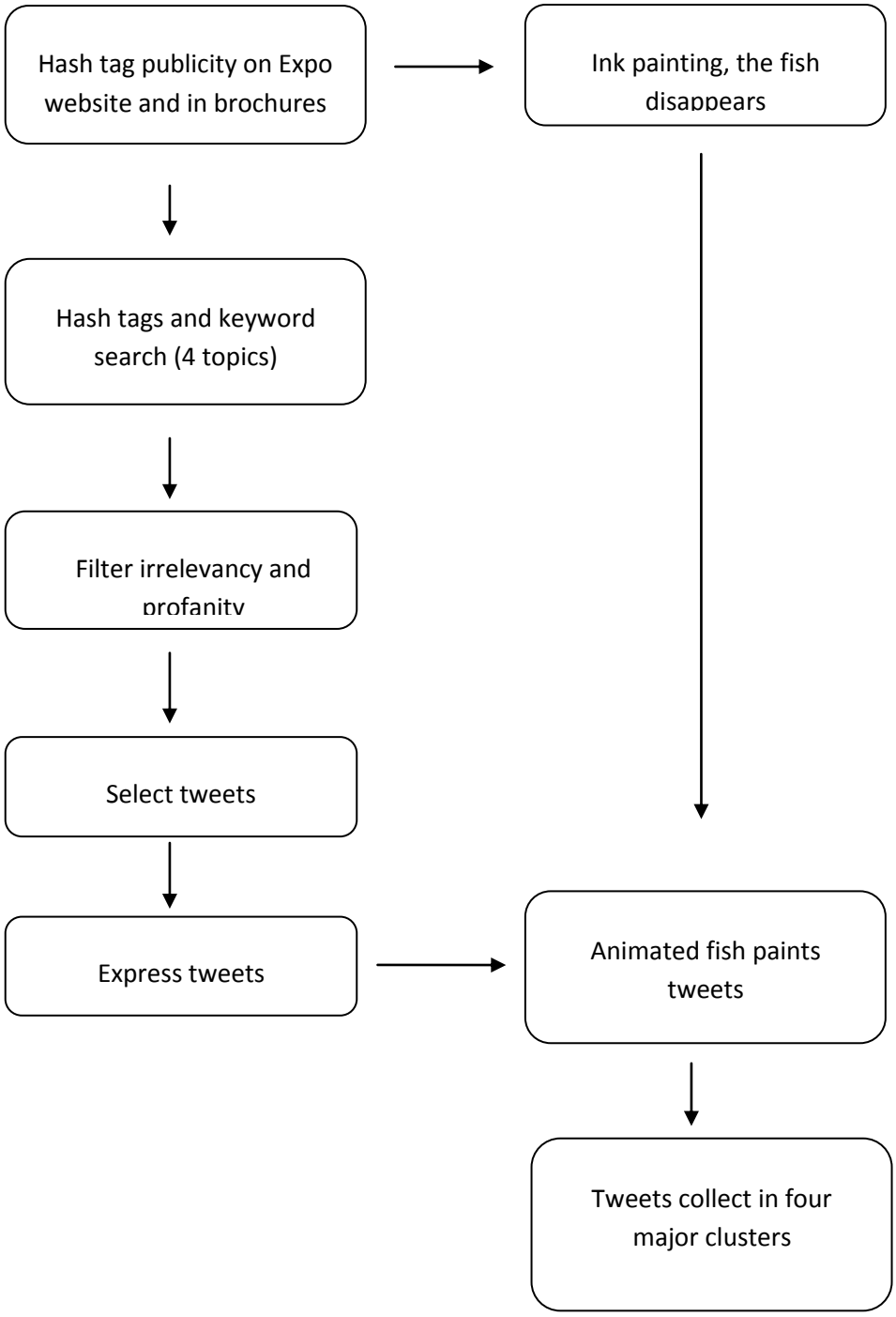
- Visitors to the EDG as well as those around the world can use smart phones and the internet to post Twitter messages through a “future-oriented” social network
- A certain amount of time-related tweets will be expressed through hash tags and keyword searches to EXPO
- There will be four subjects for people to open their minds through tweets to what is happening at and beyond the Expo experience from all over the world
- When expressing tweets onscreen as a graphic representation of ink painting style, the audience can have fun not only reading what they and others have to say but also marvel at the beauty of Korea’s media art

IMPLEMENTATION AREA

Entire LED ceiling area of EDG



CONFIGURATION



ADDITIONAL INFORMATION

Steps	Servers	LED
Hash tags and keyword search (4 topics)	Audience is registered through the internet and smart phones and send tweets through the Yeosu Expo hash tags and keywords associated with the search	Fish swim around the screen in the appearance of the ink painting style. The fish's tail acts a paint brush full of ink and draws a smooth curve for each movement. The four categories become whirlpools of hash tags from Twitter followers.
Filter profanity	Search for inappropriate expressions primarily by using a program, then use human volunteers as a secondary filter	
Select tweets	Once tweets pass the filter, those approved will be chosen at random to be displayed	
Express tweets	Tweets are expressed in the selected LED area	Many fish swim and write tweets in the ink painting style in the four areas. Tweets build up on four themes and show in beautiful curves and slowly fade out after a certain period of time as new tweets are accumulated.

PRODUCTION EXAMPLES

Figure 1, the entire LED ceiling

Figure 2, the first part of LED ceiling

Figure 3 the entire LED ceiling concept art2

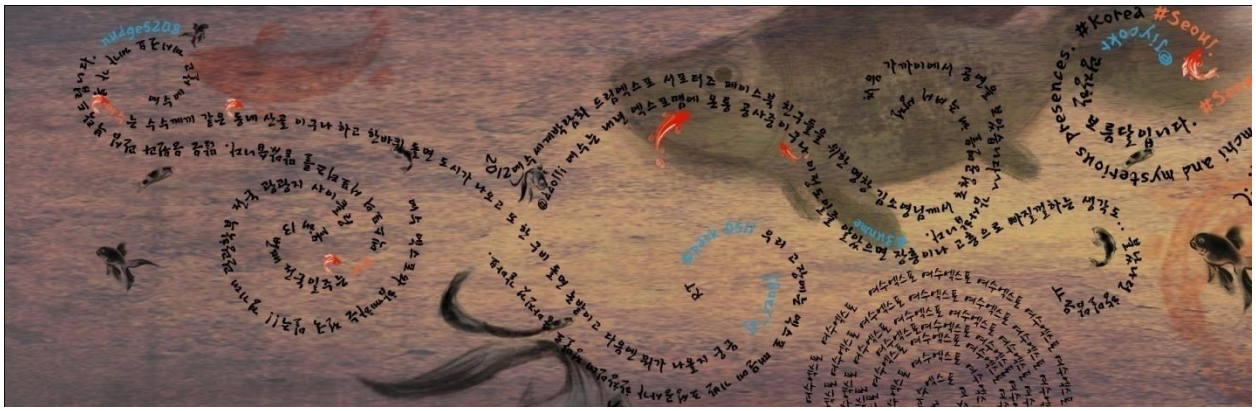


Figure 4 the first part of LED ceiling

STORYBOARD

Figure 1 School of fish come into the EDG ceiling

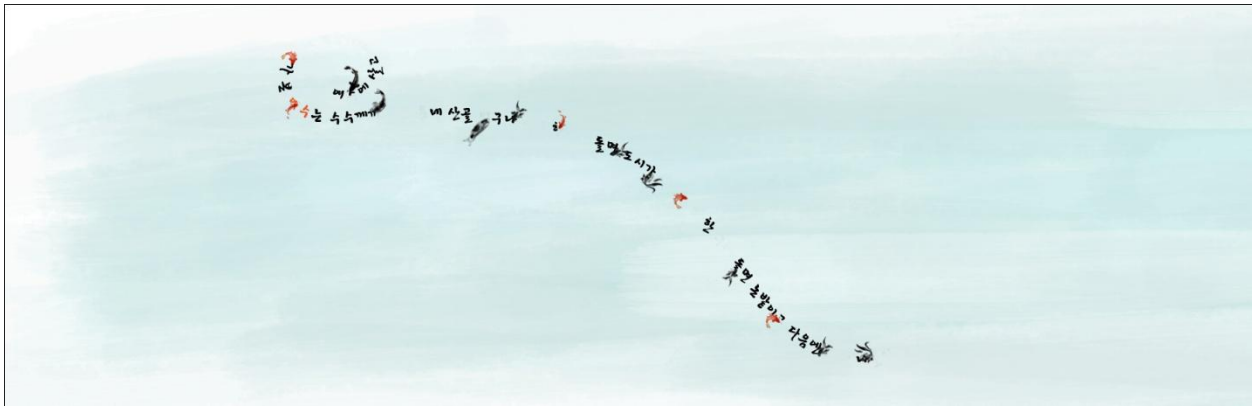


Figure 2 When they move, it create twitter feeds on ceiling. The movement looks like dance

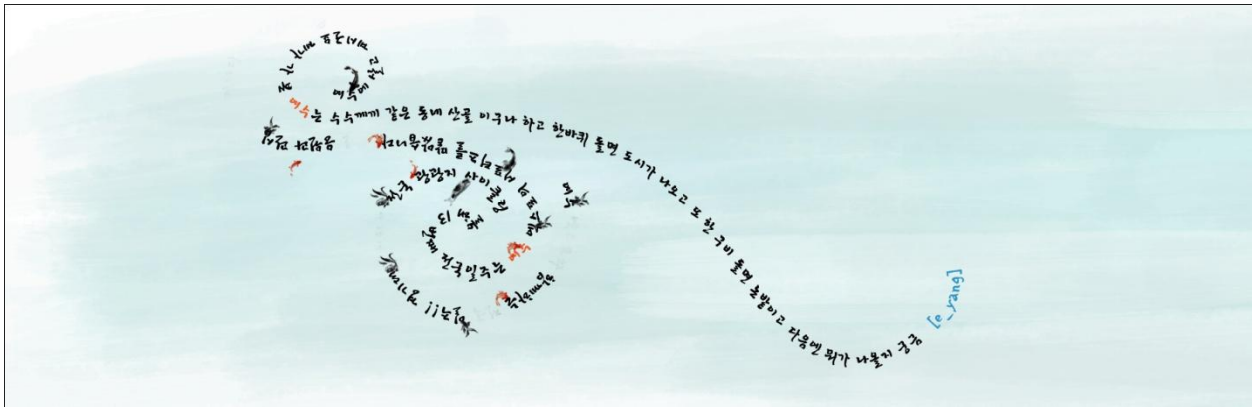


Figure 3 Twitter feeds will pile up on screen

Left Side Entrance of Concourse

Right Side Entrance of Concourse

Oceanus Twitter Archive
Software Architecture Document

Version 0.6

REVISION HISTORY

Date	Version	Description	Author
06/15/2011	0.5	Initial Draft	Shawn Wall
06/29/2011	0.6	Added Censor Details	Shawn Wall

CONTRIBUTORS

Name	Department
Shawn Wall	Technology

REVIEW HISTORY

Date	Version	Comment	Reviewer

INTRODUCTION

This document provides a comprehensive architectural overview of the Oceanus Twitter Archive web application. It outlines the technologies used in implementing the project, and is intended to capture and convey the significant architectural decisions that have been made during the design of the system. It serves as a communication medium between the software architects, developers, deployment specialists, and other team members regarding those decisions.

The Twitter Archive web application is being created for the purpose of searching, filtering, and archiving tweets for later retrieval by a consumer application for display. An administrative user will be able to log into the system, set up “feeds” consisting of twitter searches, and then monitor results for those feeds. The user selects what tweets are approved and those tweets are archived to a SQL Server database for later retrieval by the consumer application.

PURPOSE

This document will serve two major purposes:

- Guide the concrete software implementation of the Oceanus Twitter Archive web application.
- Assist enterprise architect, operation, platform team, and security auditor conducting an enterprise level best practice and standard validation.

SCOPE

This document describes the architecture of the data structures, web front-end, communication layers, and server configuration of the web application. It explains the interactions between the end users of the application, the application with the external service API, and administration users managing the system.

WHAT THIS DOCUMENT IS NOT

This document does not attempt to describe the existing functionality in detail available within Twitter or Unity, as they are integration points in the architecture.

ASSUMPTIONS

- The Twitter API version available at the time of implementation will be used for development. Future API versions may introduce incompatibilities that will need to be remedied by future updates to the source code.

RELATED DOCUMENTS

Twitter API Documentation
http://dev.twitter.com/doc
Twitterizer .NET Twitter Library
http://www.twitterizer.net
OAuth
http://oauth.net

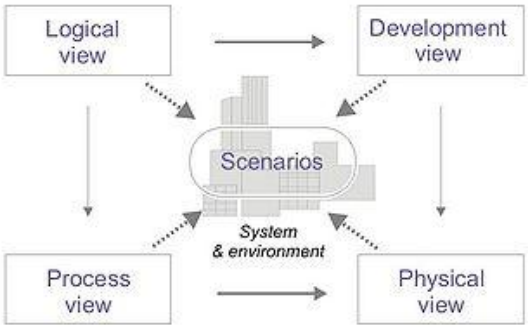
GLOSSARY

Term	Definition
4+1 Architectural Model	4+1 is a view model designed by Philippe Kruchten for describing the architecture of software-intensive systems, based on the use of multiple, concurrent views.

UML	Unified Modeling Language is a standardized general-purpose modeling language in the field of software engineering. UML includes a set of graphical notation techniques to create abstract models of specific systems.
Twitter	Social networking and microblogging service utilising instant messaging, SMS or a web interface.

ARCHITECTURAL REPRESENTATION

This document use Kruchten's 4+1 view model. This model breaks down the system into a set of views, each capturing a specific aspect of the system, address concerns for end users, developers, system integrators and system engineers:



View	Description
Use case view	Describes the end-user view of the system functionalities.
Logical view	An abstract description of the system's parts, contexts, and how they interact with each other to accomplish system functionalities.
Development view	Focuses on the actual software module organization, and implementation. It also addresses architecture goals and constraints and their solutions.
Process view	The process architecture takes into account non-functional requirements, such as scalability and availability. It addresses issues of concurrency and distribution.
Physical view	Describes deployment structures, configurations and communication protocols.

USE CASE VIEW

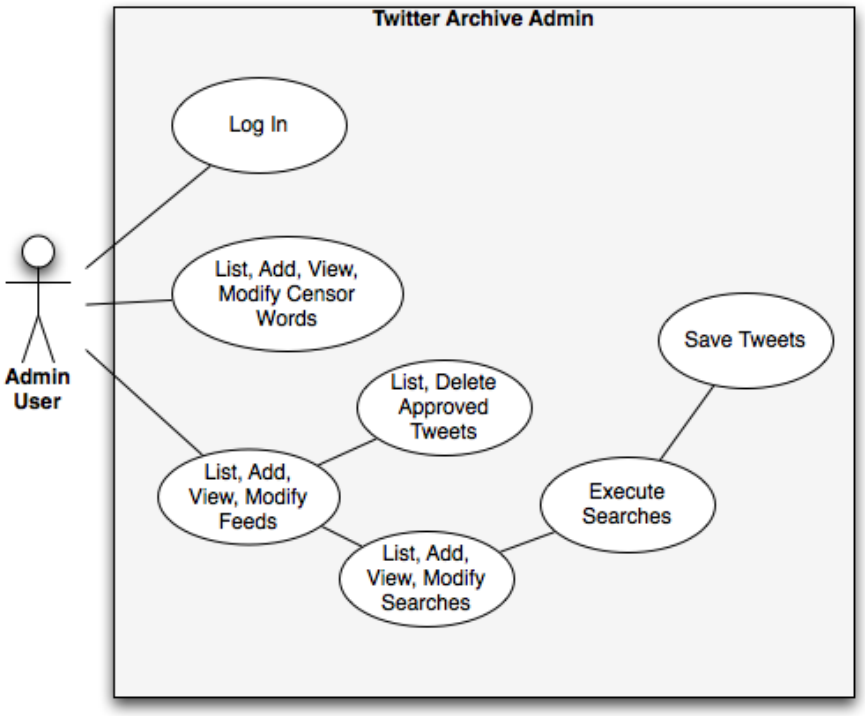
Use-Case diagrams show each of the Actor(s) goals or tasks in using the application. An Actor, as defined by the Rational Unified Process, is any person or system that is external to the system being implemented that the system interacts with. The Use Cases do not define implementation but rather outline the tasks the Actors want to accomplish.

ACTORS

Name	Description
Admin User	An admin user is one of a group of individuals who will use the web-based administration system to create feeds, set up searches for those feeds, view results, and save tweets to the feeds.
EDG System	The EDG System will communicate with the Archive system via a web service to consume a stream of tweets that have been filtered by Admin Users.

USER USE CASES

ADMIN USER USE CASES – DIAGRAM



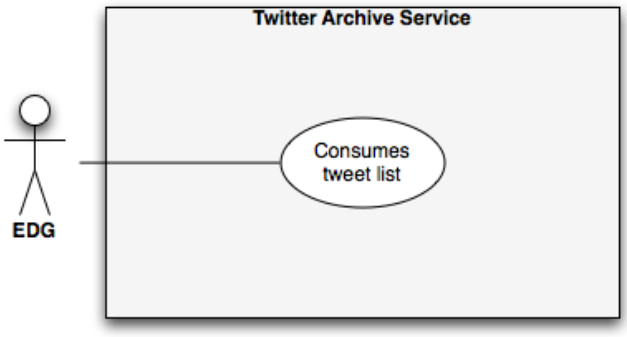
ADMIN USER USE CASES - DESCRIPTIONS

Name	Description
Log in	Log in with an existing username and password combination to obtain access to the application.
List, Add, View, Modify Feeds	View a tabular list of feeds, add new feeds, view details for existing feeds, and modify details for existing feeds. Up to 4 feeds can be designated as Active feeds at a

	given time.
List, Add, View, Modify Searches	Feeds are related to one or more searches which consist of search criteria. The list of searches associated with a given feed can be viewed from the Feed details. These searches can be viewed and modified, along with creation of new searches.
Execute Searches	The defined searches can be executed against twitter and the results are shown in tabular paged format to the Admin User.
Save Tweets	From the list of twitter results, the admin user can select which tweets are approved by selecting all or de-selecting particular tweets.
List, Delete Approved Tweets	The list of approved tweets can be viewed in tabular paged format for a given feed, and previously approved tweets can be deleted if desired.
List, Add, View, Modify Censor Words	A list of words to be censored is accessible via the admin user. The user can add & modify words to be used by the censoring filter in the application.

SYSTEM USE CASES

EDG SYSTEM USE CASES – DIAGRAM



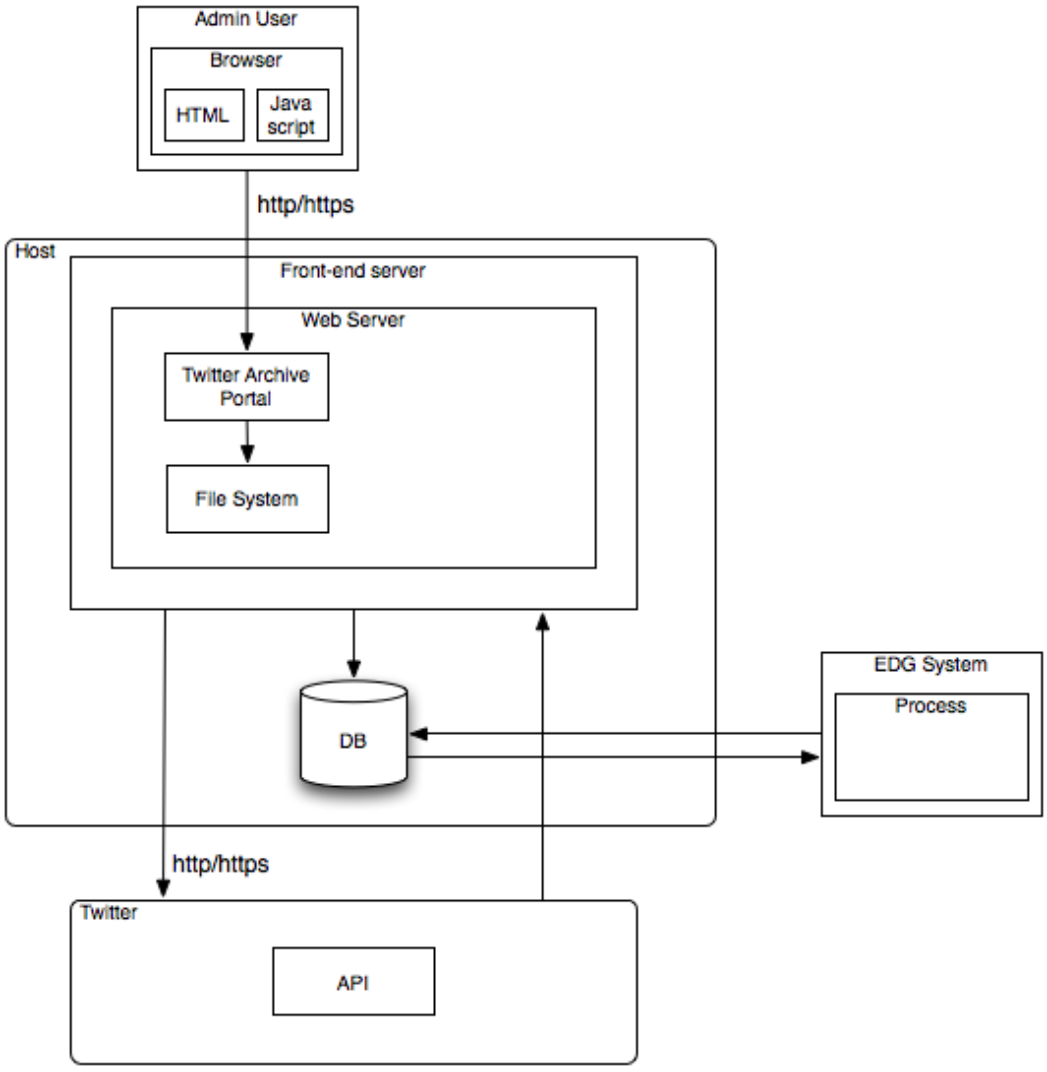
EDG SYSTEM USE CASES - DESCRIPTIONS

Name	Description
Consumes Tweet List	The EDG System will consume a direct output of twitter messages from the data store that it will then process for display.

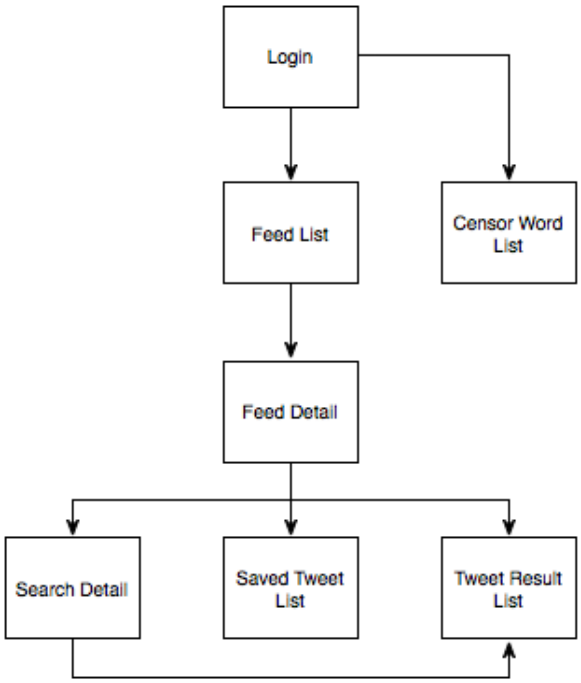
LOGICAL VIEW

The following view is the Logical View of the solution being implemented.

LOGICAL SYSTEM DIAGRAM



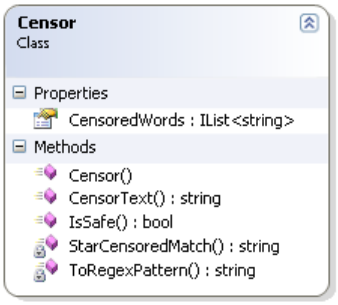
SITEMAP



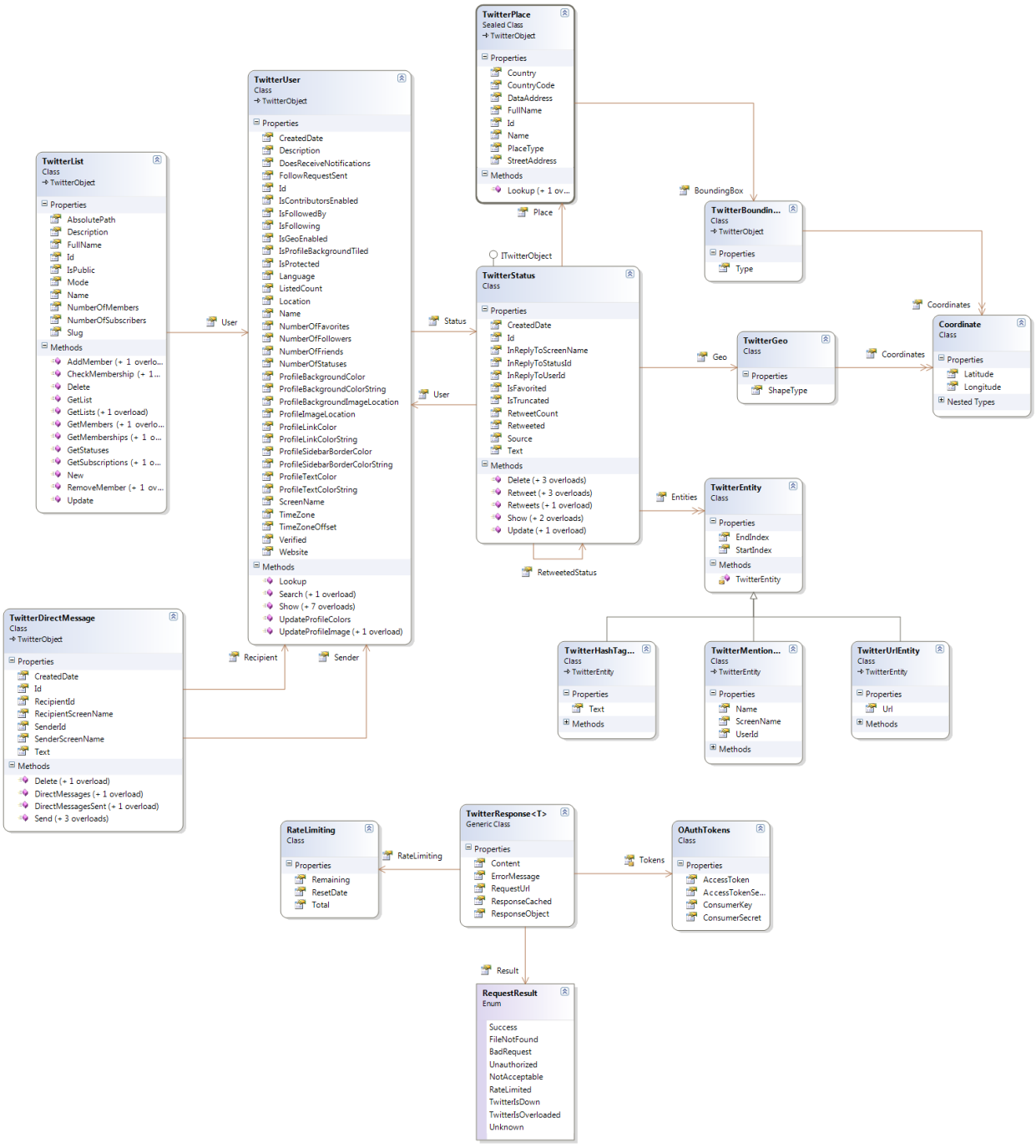
CLASS DIAGRAMS

CMU.TWITTER.ARCHIVE.MODEL


Feed
Class
EntityObject
Fields
Properties
Id : int
IsActive : bool
Name : string
Searches : EntityCollection<Search>
Tweets : EntityCollection<Tweet>
Methods
CreateFeed() : Feed
OnIdChanged() : void
OnIdChanging() : void
OnIsActiveChanged() : void
OnIsActiveChanging() : void
OnNameChanged() : void
OnNameChanging() : void
Search
Class
EntityObject
Fields
Properties
Feed : Feed
FeedReference : EntityReference<Feed>
Geocode : string
Id : int
IsoLanguageCode : string
MaxId : string
Name : string
Query : string
TwitterSearchOptions : SearchOptions
Methods
CreateSearch() : Search
OnGeocodeChanged() : void
OnGeocodeChanging() : void
OnIdChanged() : void
OnIdChanging() : void
OnIsoLanguageCodeChanged() : void
OnIsoLanguageCodeChanging() : void
OnMaxIdChanged() : void
OnMaxIdChanging() : void
OnNameChanged() : void
OnNameChanging() : void
OnQueryChanged() : void
OnQueryChanging() : void
TwitterSearch() : TwitterResponse<TwitterSearchResultCollection>
Tweet
Class
EntityObject
Fields
Properties
CreatedAt : DateTime
Feed : Feed
FeedReference : EntityReference<Feed>
FromUser : string
FromUserId : string
Id : int
IsoLanguageCode : string
ProfileImageUrl : string
Text : string
TwitterId : string
Methods
CreateTweet() : Tweet
CreateWithTwitterSearchResult() : Tweet
OnCreatedAtChanged() : void
OnCreatedAtChanging() : void
OnFromUserChanged() : void
OnFromUserChanging() : void
OnFromUserIdChanged() : void
OnFromUserIdChanging() : void
OnIdChanged() : void
OnIdChanging() : void
OnIsoLanguageCodeChanged() : void
OnIsoLanguageCodeChanging() : void
OnProfileImageUrlChanged() : void
OnProfileImageUrlChanging() : void
OnTextChanged() : void
OnTextChanging() : void
OnTwitterIdChanged() : void
OnTwitterIdChanging() : void
TwitterEntities
Class
ObjectContext
Fields
_FeedSet : ObjectQuery<Feed>
_SearchSet : ObjectQuery<Search>
_TweetSet : ObjectQuery<Tweet>
Properties
FeedSet : ObjectQuery<Feed>
SearchSet : ObjectQuery<Search>
TweetSet : ObjectQuery<Tweet>
Methods
AddToFeedSet() : void
AddToSearchSet() : void
AddToTweetSet() : void
OnContextCreated() : void
TwitterEntities() (+ 2 overloads)
TwitterContextFactory
Class
Methods
Create() : TwitterEntities
IRepository
IRepository
Interface
Methods
Create<T>() : void
Delete<T>() : void
Edit<T>() : void
Get<T>() : T
List<T>() : ObjectQuery<T>
Query<T>() : ObjectQuery<T>
Save() : int
RepositoryBase
Abstract Class
Fields
keyPropertyName : string
Methods
Create<T>() : void
CreateGetExpression<T>() : Expression<Func<T, bool>>
Delete<T>() : void
Edit<T>() : void
Get<T>() : T
GetKeyPropertyValue<T>() : int
List<T>() : ObjectQuery<T>
Query<T>() : ObjectQuery<T>
Save() : int
IDisposable
EFRepository
Class
RepositoryBase
Fields
_context : ObjectContext
Methods
Create<T>() : void
Delete<T>() : void
Dispose() : void
Edit<T>() : void
EFRepository() (+ 1 overload)
Get<T>() : T
GetEntitySetName<T>() : string
List<T>() : ObjectQuery<T>
Query<T>() : ObjectQuery<T>
Save() : int



TWITTERIZER


TwitterList
Class
TwitterObject
Properties
AbsolutePath
Description
FullName
Id
IsPublic
Mode
Name
NumberOfMembers
NumberOfSubscribers
Slug
Methods
AddMember (+ 1 overlo...
CheckMembership (+ 1...
Delete
GetList
GetLists (+ 1 overload)
GetMembers (+ 1 overlo...
GetMemberships (+ 1 o...
GetStatuses
GetSubscriptions (+ 1 o...
New
RemoveMember (+ 1 ov...
Update
User
TwitterUser
Class
TwitterObject
Properties
CreatedDate
Description
DoesReceiveNotifications
FollowRequestSent
Id
IsContributorsEnabled
IsFollowedBy
IsFollowing
IsGeoEnabled
IsProfileBackgroundTiled
IsProtected
Language
ListedCount
Location
Name
NumberOfFavorites
NumberOfFollowers
NumberOfFriends
NumberOfStatuses
ProfileBackgroundColor
ProfileBackgroundColorString
ProfileBackgroundImageLocation
ProfileImageLocation
ProfileLinkColor
ProfileLinkColorString
ProfileSidebarBorderColor
ProfileSidebarBorderColorString
ProfileTextColor
ProfileTextColorString
ScreenName
TimeZone
TimeZoneOffset
Verified
Website
Methods
Lookup
Search (+ 1 overload)
Show (+ 7 overloads)
UpdateProfileColors
UpdateProfileImage (+ 1 overload)
Status
User
TwitterPlace
Sealed Class
TwitterObject
Properties
Country
CountryCode
DataAddress
FullName
Id
Name
PlaceType
StreetAddress
Methods
Lookup (+ 1 ov...
Place
BoundingBox
TwitterBoundin...
Class
TwitterObject
Properties
Type
ITwitterObject
TwitterStatus
Class
Properties
CreatedDate
Id
InReplyToScreenName
InReplyToStatusId
InReplyToUserId
IsFavorited
IsTruncated
RetweetCount
Retweeted
Source
Text
Methods
Delete (+ 3 overloads)
Retweet (+ 3 overloads)
Retweets (+ 1 overload)
Show (+ 2 overloads)
Update (+ 1 overload)
RetweetedStatus
Geo
TwitterGeo
Class
Properties
ShapeType
Coordinates
Coordinates
Coordinate
Class
Properties
Latitude
Longitude
Nested Types
Entities
TwitterEntity
Class
Properties
EndIndex
StartIndex
Methods
TwitterEntity
TwitterHashTag...
Class
TwitterEntity
Properties
Text
Methods
TwitterMention...
Class
TwitterEntity
Properties
Name
ScreenName
UserId
Methods
TwitterUrlEntity
Class
TwitterEntity
Properties
Url
Methods
TwitterDirectMessage
Class
TwitterObject
Properties
CreatedDate
Id
RecipientId
RecipientScreenName
SenderId
SenderScreenName
Text
Methods
Delete (+ 1 overload)
DirectMessages (+ 1 overload)
DirectMessagesSent (+ 1 overload)
Send (+ 3 overloads)
Recipient
Sender
RateLimiting
Class
Properties
Remaining
ResetDate
Total
RateLimiting
TwitterResponse<T>
Generic Class
Properties
Content
ErrorMessage
RequestUrl
ResponseCached
ResponseObject
Tokens
OAuthTokens
Class
Properties
AccessToken
AccessTokenSe...
ConsumerKey
ConsumerSecret
Result
RequestResult
Enum
Success
FileNotFound
BadRequest
Unauthorized
NotAcceptable
RateLimited
TwitterIsDown
TwitterIsOverloaded
Unknown

DEVELOPMENT VIEW

TECHNICAL STACK

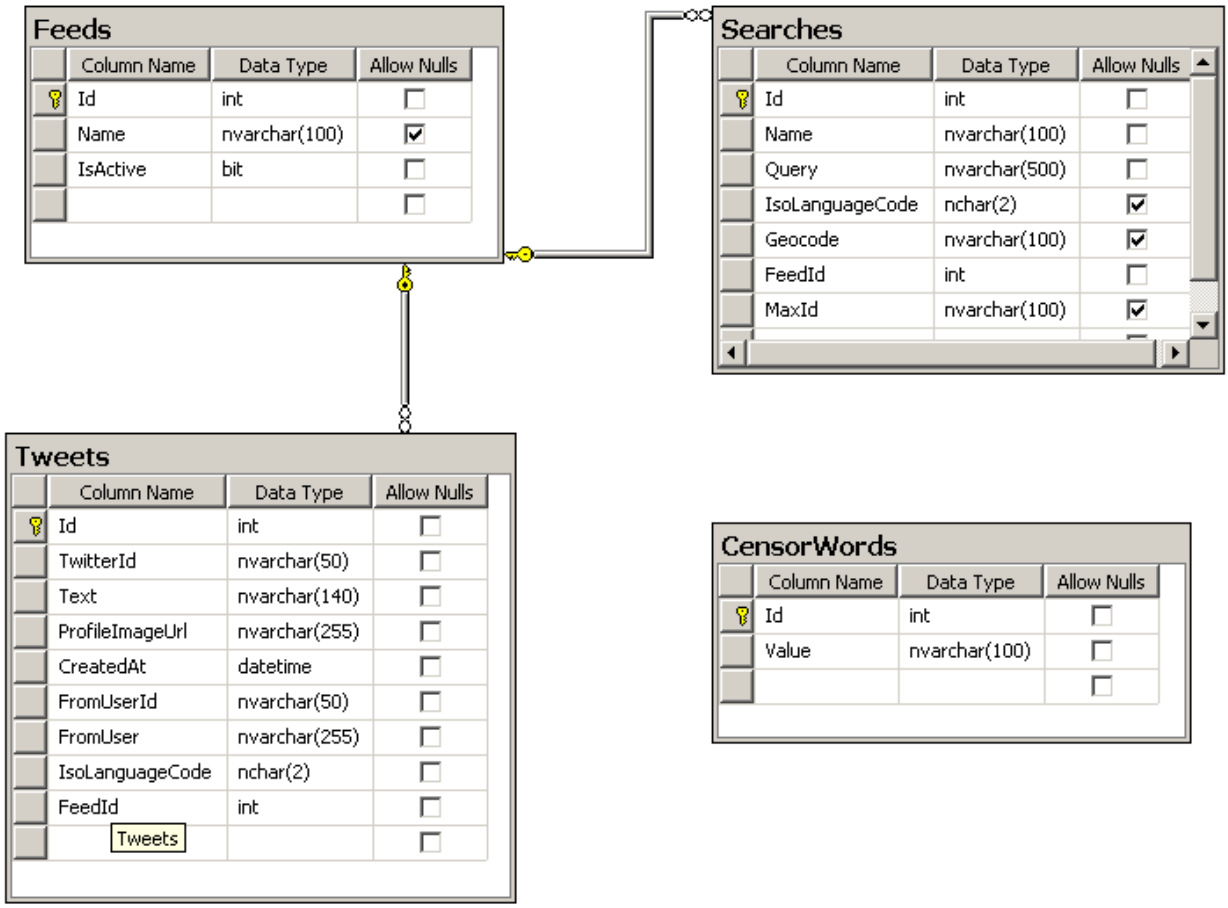
DEVELOPMENT ENVIRONMENT

Windows XP, Windows 7
.NET 3.5
SQL Server 2005, 2008

TWITTER API ACCESS

Twitter API access will be done through the Twitterizer open source library that abstracts HTTP-based access to the Twitter API using objects.

DATA MODEL



The .NET 3.5 Entity Framework is being used as an ORM for abstracting database access in the C# code. The primary entities are Feeds, Searches, and Tweets. Entity Linq will be used to pull information from the database and the Entity Models and ObjectContext will be used to store new and modify existing records in the database.

SEARCH

The search requests are abstracted using the aforementioned Twitterizer library. A direct API request for a Twitter search is:

http://search.twitter.com/search.json?q=%23ios5&rpp=10

This same search executed through the Twitterizer library is:

```
string query = "#ios5";

int pageNumber = 1;

SearchOptions options = new SearchOptions()

{

    PageNumber = pageNumber,

    NumberPerPage = 10

};

TwitterResponse<TwitterSearchResultCollection> searchResult =
TwitterSearch.Search(query, options);
```

The SearchOptions for a given Search object can be retrieved via a property.

STORAGE

SQL Server 2008 will be used as the data store for the application.

RETRIEVAL

The consuming application will pull data by directly connecting to the SQL Server instance. A View will be created for the consumer to query in order to simplify DB access.

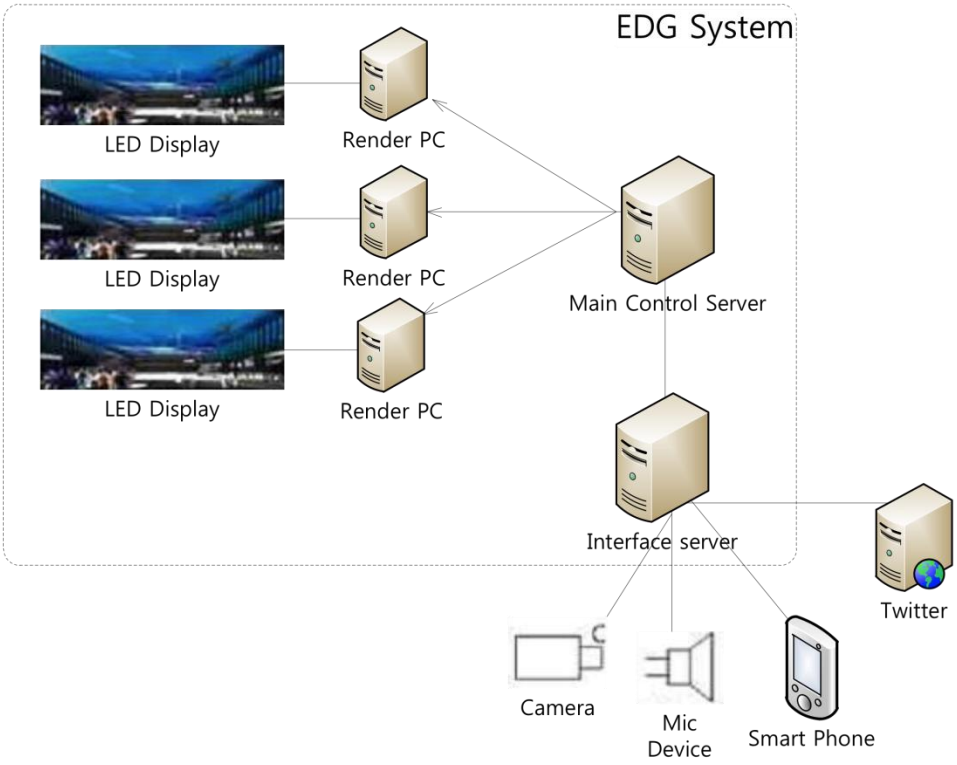
CENSOR FILTER

Given the nature of the Internet, the requirement is present to be able to filter out Twitter content that contains particular words such as profanity. A list of words to censor is stored in the database and is accessible via the CensorWord Entity class. An EFRepository<CensorWord> can be declared to pull the list of words. The CMU.Yeosu.Twitter.Util.Censor class can be instantiated with a List<string> of censor words that are internally converted to a regular expression for matching by using the IsSafe() method.

PHYSICAL VIEW

DEPLOYMENT VIEW

PRODUCTION ENVIRONMENT



PROCESS VIEW

SECURITY

Authentication to Twitter is done using OAuth, but the search calls do not require authentication so this will not be necessary for this project.