Bilkent University

Department of Computer Engineering

CS 491- Senior Design Project I

Unisphere: Global University Catalog

High Level Design Report

Arkın Yılmaz	21502080
Doruk Çakmakçı	21502293
Hakan Sarp Aydemir	21501331
İrem Ural	21502278
Umut Berk Bilgiç	21502757

Supervisor: Uğur Doğrusöz

Jury Members: Fazlı Can and Çiğdem Gündüz Demir

High Level Design Report

December 31, 2018

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.

Table of Contents

1. Introduction

Today in the era of Internet, people are looking for the easiest and fastest way to access information. Instead of looking for every website, they try to find the one, which has all the information they are looking for. Starting from the high school years, students start to search universities and they try to find enormous and variety of information such as the ranking of the university, academic programs, price of the dorms/universities and some location-based data, for example the weather conditions in that city, social events and facilities near the university, campus photos etc. Besides, all that raw information, they want to compare these universities in these several ways. Therefore, by following this urge, instead of searching all the universities in different tabs and then try to compare them by yourself or by considering for some narrow aspects, Unisphere will overcome and abandon this tiresome searching for hours to find the best universities that suit people well.

Though there are some web applications which has some features of Unisphere, this application will gather all that information and will provide some new features. Different from other applications, it will combine both academic and social aspects of universities. It will display all the routes from selected university to all social facilities such as museums, concerts, shopping malls, hospitals, airports, subways etc. Besides, it will inform users about the events in and near the university. To feel the atmosphere in the university, 360 tour in the campus will be offered and general public opinion about the university will be analyzed and presented visually to the users.

This report aims to outline the inner software architecture via realizing its subsystem decomposition and it's design goals.

1.1. *Purpose of the System*

Unisphere aims to gather various information related to universities all around the world. In this web application, undergraduate, master’s and PhD. programs for different departments will be displayed for each selected university. Users will easily access all universities around the world and they will be able to compare the universities side by side on a user-friendly interface. Besides the educational aspects, by using Google Street view, users can travel inside the campus and they will be able to see the close by facilities such as hospitals, shopping malls, airport, subway. They can learn how to go nearby places by the color-coded fastest routes displayed on an interactive map.

Besides all the above features, this application will provide some additional data related to the location of the university such as the weather conditions in that city, dorms/ university prices according to the stock market. Additionally, Unisphere will inform users about the social events (cinemas, concerts, exhibitions etc.) inside and outside of the university by filtering EventBrite events.

By combining both academic and social features of Unisphere, users can prioritize their needs and desires for the university in a priority list and they can get an instant result which will fit their requests. For instance, they can select whether they give importance to academic success, sociability, whether the university is close to the city center etc. By using this list created by the user and by looking at the users search history, a list of recommended universities will be presented.

Users can utilize this application with or without registering. The registered users will be privileged to access the general public opinion about universities in a visually attractive way. By using Twitter and Reddit comments the thoughts and emotions about the universities will be illustrated. Secondly, these verified users can leave comments about the university.

1.2. Design Goals

In this section, the design goals of Unisphere will be presented and the rationale behind the goals will be explained.

1.2.1. Reliability

- The application must not provide incorrect or unreferenced information. The information used by the application must be verified.
- The application must regularly update information regarding the events for a university more often than institutional information, since the information about an event are subject to asynchronous change(location change, time change etc.).
- The application must recommend universities to a user based on their preferences. A generic approach for recommendation system should not be in the design constraints.
- The application must accurately perform sentiment analysis on the information for a university to accurately represent public opinion towards the university.
- The application should correctly determine the verified users from their email addresses.

1.2.2. Usability

- The application should have an easy to use interface without discarding any of the proposed functionality.
- The application should recommend universities to a user in accordance with the user’s preferences.

1.2.3. Efficiency

- The application should be quick to navigate around the user interface.
- The application’s user interface elements should not lag and provide smooth and snappy animations.

- The information present in the application must not be too deep into the menu structure. It should not take a lot of backtracking of menus to get back to another user interface element.
- The application should display only the relevant information on the screen according to the menu or user interface elements present.

1.2.4. Flexibility

- The application's software (both client and server side) must be completed in a way that allows for feature updates, addition of new features. Changing/refactoring the code for one module should not break another.
- The application's user interface should scale in size based on the user's browser windows size.

1.2.5. Security

- The application should sanitize the user text input in order to prevent injection exploits or server side errors that could happen while trying to process bad or malicious text input.
- User's passwords will not be kept as plain text, they will be encrypted.
- User's data such as browser used, OS used, email and password will be kept within the database and not be publicly available.

1.3. Definitions, Acronyms and Abbreviations

HTTP:	Hypertext Transfer Protocol
API:	Application Program Interface
AWS:	Amazon Web Service
JSON:	JavaScript Object Notation
JS:	JavaScript

1.4. Overview

Internet provides the fastest access to information in an easiest way. Though when people try to find a certain information, it can really become tiresome. One of the examples is definitely the case when people try to find a university, masters or PhD. program.They want to search all of the aspects of that university. Starting from the academic standing of the university to living conditions in that city. They can find the general information about the university in one tab, and then the living conditions, social facilities around the university in other tabs… The tabs starts to accumulate, they want to learn more about different universities as well and the information starts to become more complex and hard to compare. To avoid all these complexity, Unisphere will provide a simple platform for users. It will gather both academic and social aspects of the universities, and most importantly it will provide a centralized and up to date reliable data.

Both academic and non-academic information will be collected in this platform. PhD. and master programs of the universities, university rankings, departments of the universities and etc. Besides all these academic side, the social events and social facilities around the universities will be displayed with color coded routes to the user on a map. The weather conditions in that city and the housing options will be available to users as well. Lastly, by using Google Street View users will feel the atmosphere in the university by looking the 360° view in the campus.

2. *Current Software Architecture*

Though there are several applications which have some features of Unisphere, this project aims to gather all information from several websites. Besides, instead of focusing only on the academic information, this web application will combine both academic and social features about universities.

Google Street View [1]:
Google Street View allows 360° tour around the world. By using Google Street View API, Unisphere will provide street tour inside the campus.

Google Maps [2]:
Google Maps shows the location of universities and nearby places around the selected place. It provide routes from a selected point to another. By using Google Maps API, Unisphere will provide information about nearby places around the university and show these places on the map.

EventBrite [3]:
EventBrite gathers several events around the world, including some events organized by universities. The events near and inside the university will be fetched and listed in Unisphere. Each university website has every information related to their university.

Twitter [4]:
Twitter has a huge database, which includes comments all around the world. This application will fetch the comments related to the universities, and it will apply sentiment analysis on the comments and visualize them.

Reddit [5]:

Reddit is the most popular web content aggregation website on the internet. By using Reddit, people leave comments on several topics under “subreddits”. Reddit comments which are related to selected universities will be fetched, and sentiment analysis will be applied on these comments.

OpenWeatherMap.org [6]:

Up to date weather conditions of a certain area is provided hourly, weekly and monthly bases. Unisphere will collect the data according to the location of the university and visualize them on graphs.

Yelp [7]:

Yelp provides reviews based entirely on places. Users are able to submit images, ratings and comments. Unisphere aims to aggregate this kind of social information with academic information to paint a more complete picture.

3. Proposed Software Architecture

3.1. Overview

To be able to achieve the design goals, which are explained above, the system should be designed in a neat and proper way. The whole system has an enormous complexity and to overcome this chaos, the system can be decomposed into smaller chunks, which gathers similar functionalities together and collects the classes which has more associations with each other into one package. Unisphere sticks to those design goals and gives importance to subsystem decomposition. While Unisphere aims to decouple the specified subsystems and tries to collect the classes which are relatively coherent with each other. Decomposition provides some abstraction and modularity, hence it will allow different teams to work simultaneously and optimize the implementation process. As each subsystem provides an interface to others, it will ease the error revisions as well and minimize the dependency of each team to others. It will minimize the communication between teams during implementation. Furthermore, in a well-defined subsystem hierarchy, when a change is applied, other subsystems will not be affected by this change. As a result, these advantageous aspects of subsystem decomposition are realized and used. In latter sections, the subsystems, their relations and their functionalities will be explained. In section 3.3, the hardware to software mappings will be detailed, explaining which software runs on what hardware.

3.2. Subsystem Decomposition

Since Unisphere is a web application, its main functionalities are divided into two parts, the functionalities related to user interface and the functionalities related to data preservation and manipulation. In order to provide this separation Client/Server architecture will be used as a core subsystem decomposition. In this case, client requests services of server, while server hosts the data and makes calculations using the data and provides those services to Client. The central database connection will be established in server side.

To separate pure data storage, from manipulation of these data, three tier architecture will be used. The client-side will include the presentation layer. The presentation layer is the top-most level of Unisphere. The main function of this layer is to provide an interface between the user and the application, which translates and propagates user’s requests to the server side for request processing. Also, the data provided from the server side will be visualized in this layer. By requesting necessary data from server, it will display the information related to universities, show the events, facilities and routes to these in an attractive way. Furthermore, the presentation layer will be partitioned to two subsystems, View subsystem will hold classes of high coupling with view-level objects. Controller subsystem will hold the controller classes which have the responsibility of manipulating the view-level objects when a particular change has occurred in the database. The server-side will be divided to two layers: Logic and Data layer. Data layer is the layer that the information related to universities and user's themselves are stored. The information will be propagated to Logic layer for processing. The Logic layer will be responsible for the calculations on the stored data.

Since the subsystem decomposition diagram of Unisphere is too large to show fully we have separated the subsystem contents and overall subsystem decomposition. The general subsystem decomposition of Unisphere is below:

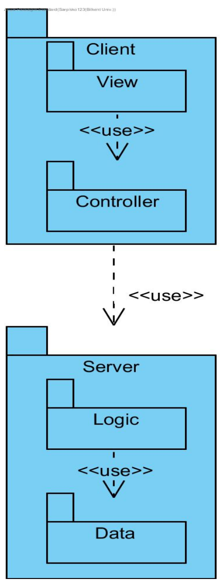

Figure 1. Subsystem Decomposition

3.3. *Hardware/Software Mapping*

Users will run our application on their browsers. Even though many devices can run man different browsers such as Safari, Chrome, Opera, Firefox etc. we will treat these as one virtual machine and refer to it as ClientBrowser. The web server runs on Unix machine and it runs an Apache web server.Thus the server side device name is WebHost and the web server name is Apache. The server and client will communicate with HTTP requests. Some API calls will return a JSON file but these will not be used as they are and will be converted and interpreted in a way that our server-side JavaScript code can understand and work with. The application will not be coded using pure JavaScript but rather a JS development framework called React will be used. For additional server-side scripts, python and ruby can be utilized.The database lives in the cloud and is a NoSQL type database.

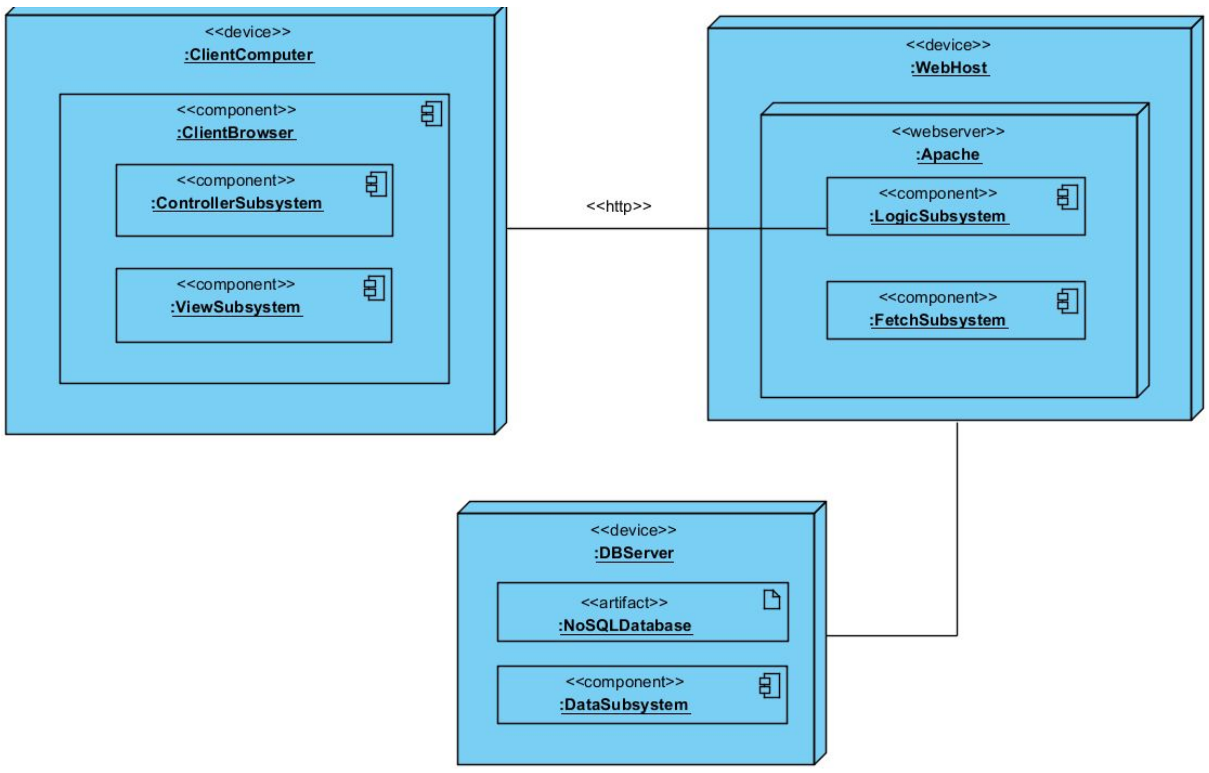

Figure 2. Hardware Software Mapping

3.4. *Persistent Data Management*

Though the system shuts down, some data should live longer than an execution of the program. These are considered as persistent data in this web application. To illustrate, all the user related information such as username, password, email and priority list will be kept in the database. Additionally, the database will also store information which is specific to each university such as the name of the university, logo of the university, departments in the university, location of the university etc. Some statistical information, for instance average weather statistics of a city that university belongs to, does not change frequently in our system, it will be updated periodically. Social facilities, such as supermarkets, museums etc. around the university can be considered as persistent data as well. However, the route from the university to the region of interest will be computed whenever user requests it. In other words, dynamically calculated data will not be persistent. Note that, if the computation of a non-fundamental data is computationally expensive, it will also be stored in the database. For the periodically updated data, a certain time limit will be defined and to serve a request the data will be fetched from the database. On the other hand, some of the data will be cached to the database whenever, the data is requested by the user. This approach, reduces the computational load on the server when a user does repetitive calls to a specific data.

Since the information that will be stored consist of different data types and SQL queries are not well suited for the object oriented data structures, non relational database (NoSQL) will be used to store persistent data. Also AWS services, namely DynamoDB and S3, will be used for our data storage. As these services are efficient, and high in performance, it will minimize the delays.

3.5. Access Control and Security

Users are free to use the application without signing in. However if signed in, they will be able to access unique features only available to those who signed up such as reading about the sentiment analysis for a given university and comment on the university that they are a current member of. In the sign-up process; Lee Reilly's “swot” [ref. swot github] Ruby tool will be used in order to verify that the user has entered a valid high-education email domain. A verification email will be sent to this email address if it is indeed a valid educational domain. After the email address is verified and the user is created, their university will be saved so that they can use the commenting feature.

Some of the user information can be changed in the future such as the password, username or the priority list by the user that the information belongs to. The email cannot be changed as it is unique for every user instance. The passwords will be encrypted.

System admins can view every information present in the database (including the encrypted versions of user passwords) and the source code. The source code and the database is not publicly available however some parts of the database is of course publicly available through the web application interface.

Any user input that is in the form of text will be sanitized before is used. Even though the cloud service could be internally avoiding this, there are instances in which the user text will be used without passing it directly to the cloud as a request. This way, the application will be resistant to injection and will not suffer from bad data.

3.6. Global Software Control

The global software control of our application is centralized and mostly dependent on the server. The control is dependent to the events of the user.

Unisphere provides different kinds of data(i.e. academic statistics of a data, social sides of universities) to the user, some of which are obtained using API's. Since one of the design requirements of Unisphere is reliability, the data provided to the user must be up-to-date. To satisfy this requirement, the server must call different API's in different periods to maintain the truthness of the data. The global software control of API services will be event driven and asynchronous. Whenever newer data is available, the server must make API calls to update the data.

When user requests sentiment analysis results for the first time, comments which are stored in the database will fetched and the results will be calculated in server side. Though after that, for a certain time limit, these results will be stored in database. When user requests it for the second time, it will avoid unnecessary computation and fetch the results from database. There will be a periodic update for comments which will allow some comments to accumulate, after a certain time the results will be recomputed based on the new larger data. Besides, according to user's search history and the predefined priority list, in the server side, university recommendations will be computed.

3.7. Boundary Conditions

3.7.1. Initializing the Application

If time allows an android version of this app will be implemented. In this case Android user will need to download the app. If users login in to the system, they can enter wrong credentials, in this case an error message will be displayed. Furthermore, in order to use the website, an internet connection is required.

3.7.2. Terminating the application

The verified user will be able to logout of the system with the logout button. Both non-verified and verified users will be able to quit the application via the quit button on any page.

3.7.3. Usage of unauthorized e-mail

The application will give an error and prompt the user to enter a valid email address when the user tries to enter an email address that isn’t allowed by the system.

3.7.4. Failure in the application

Due to AWS itself or requests overloading the server machine, server can crash and the results of requested queries, the data about the universities may not be demonstrated. If the connection goes down on client side, users cannot access our application.

4. Subsystem Services

This section of the report aims to outline the components of each subsystem and explain the services offered by each subsystem.

4.1. Client

Client side of the system interacts with the end user, it is basically what the end user sees. When the user enters login information, chooses universities to compare, wants to view sentiment analysis or changes the settings on his/her account, then a request will be sent to the server. The client subsystem will be implemented with Model View Controller pattern.

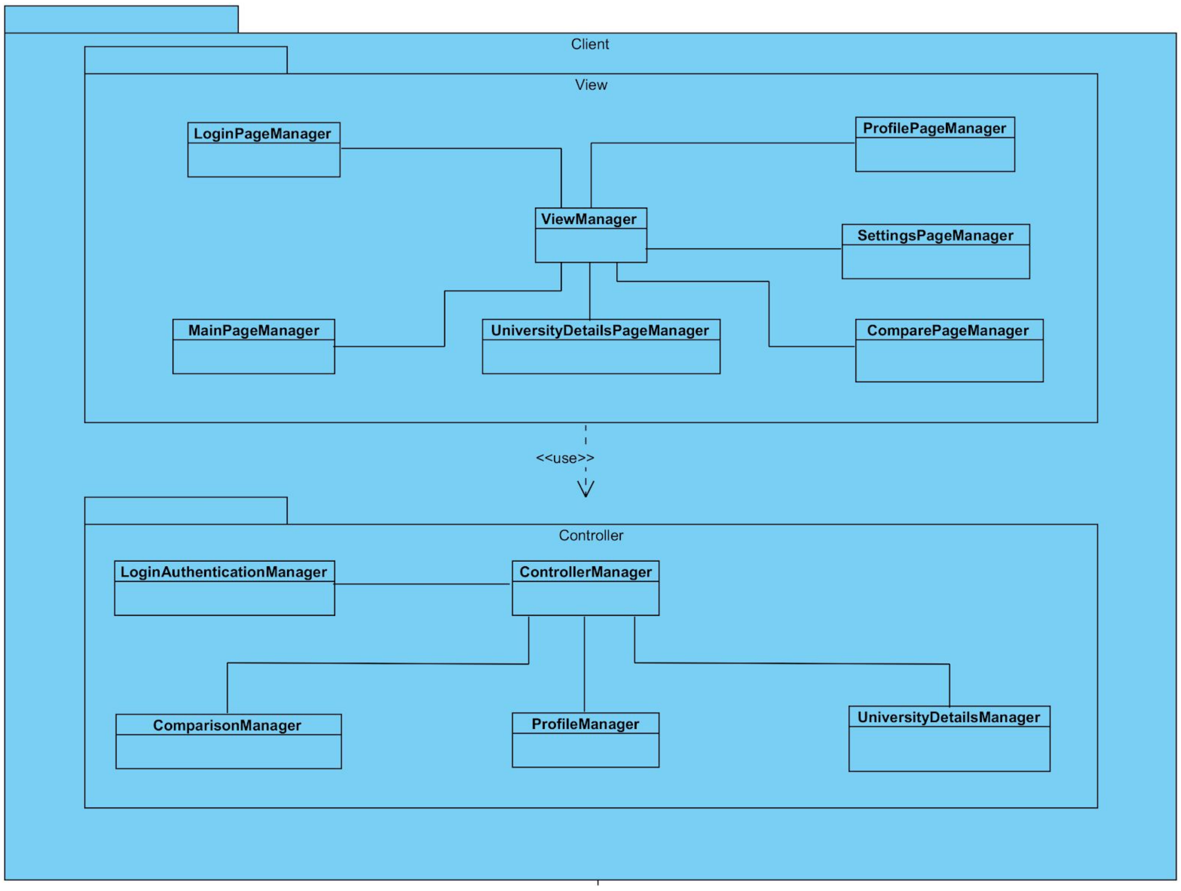

Figure 3. Client Subsystem

4.1.1. View Subsystem

The main functionality of view subsystem is to interact with the user. The web pages, login page, user profile page, settings page, all those different views will be supplied by this subsystem. It only interacts with the controller, and requests the necessary data from controller and shows the result to the end user.

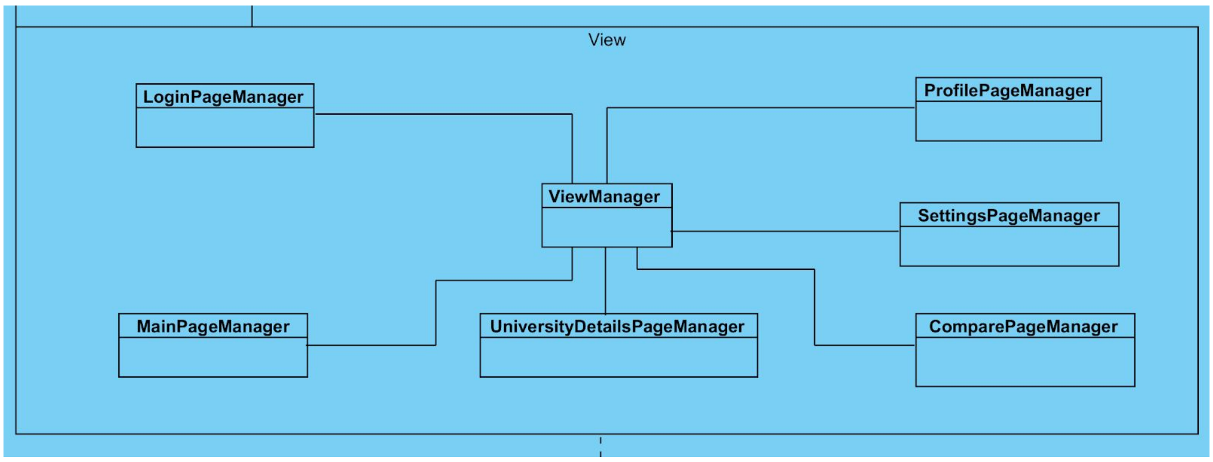

Figure 4: View Subsystem

ViewManager: It initializes the current user interface, and it redirects requests to different pages.

LoginPageManager: It shows login, sign up pages to user.

ProfilePageManager: User information, its priority list, email address are displayed.

SettingsPageManager: All settings changes are handled by this class. Also, this

MainPageManager: This class controls the main page shown to the user, it contains the information of the current state of the globe(the state of the globe includes: current zoom factor, current angle with the greenwich meridian, currently pinned universities, etc.) shown in the main page of Unisphere. Also, the search operation is handled in this class.

UniversityDetailsPageManager: Displays the detailed university information, such as academic information (departments, ranking etc.) as well as the social information (draws routes from university to social facilities, shows social events, housing options nearby on Google Maps) in a visually appealing way.

ComparePageManager: It splits the view into two, and displays general information about selected two universities. The statistical information is demonstrated in graphs to provide more comprehensive results to the user.

4.1.2. Controller Subsystem

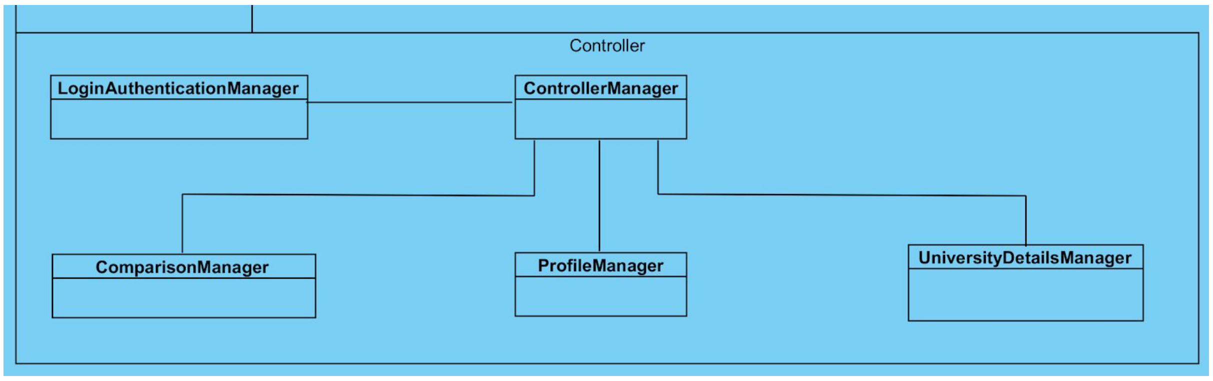

Figure 5: Controller Subsystem

The controller subsystem is responsible for the data transmission between the client and the server subsystems.

ControllerManager: It interacts with both view and the server. According to changes made by user, it can send data to server, and provide the requested information to the view.
ComparisonManager: Information exchange about the details of universities that are being compared are handled by this class.
ProfileManager: Information about the user’s details such as his/her username, comments, etc. will be handled by this class.
LoginAuthenticationManager: Checking a user’s credentials during the login process and updating the login information will be done by this class.
UniversityDetailsManager: The details about a university such as its academical conditions, social conditions, etc. will be handled by this class.

4.2. Server

The crucial and main functionality of this subsystem is to store and manage data, including both persistent and dynamic data. As a database Amazon Web Services and its cloud infrastructure will be used. To separate raw data storage, from manipulation of these data, the server side is decomposed into two parts, ie. Logic and Data. Most of the user actions requires the connection to the server. Starting from the login events to each university search by user, each activity will require data manipulation or data fetching.

When user sign ups to the system, it will store user data and save it to the database. When user logins to the system, the credentials of the user will be checked by using user information in database. When user clicks to a university or searches a university, general information such as logo of the university, departments etc. will be provided by using database connection. These operations require an access to the data classes in the Data layer. When user tries to access more information about a university, as well as location based informations, such as social facility, social event locations and weather conditions in that city, data manipulation is required . For instance, in order to show locations of the facilities, several routes from a university must be drawn as well. Hence in Logic subsystem, the raw data will be manipulated further to draw these routes in the client side. Besides, the general opinion about universities are displayed as well, in order to show these, the comments fetched from twitter and reddit will be analyzed in Logic layer.

In general, server subsystem is decomposed into two parts, Logic and Data. When user requests data, controller will interact with DataProcessor in Logic layer. Data subsystem stores user data as well as university related information. In Data layer DataFetcher will be used for dynamic data, by using crawler, necessary information will be fetched from websites.

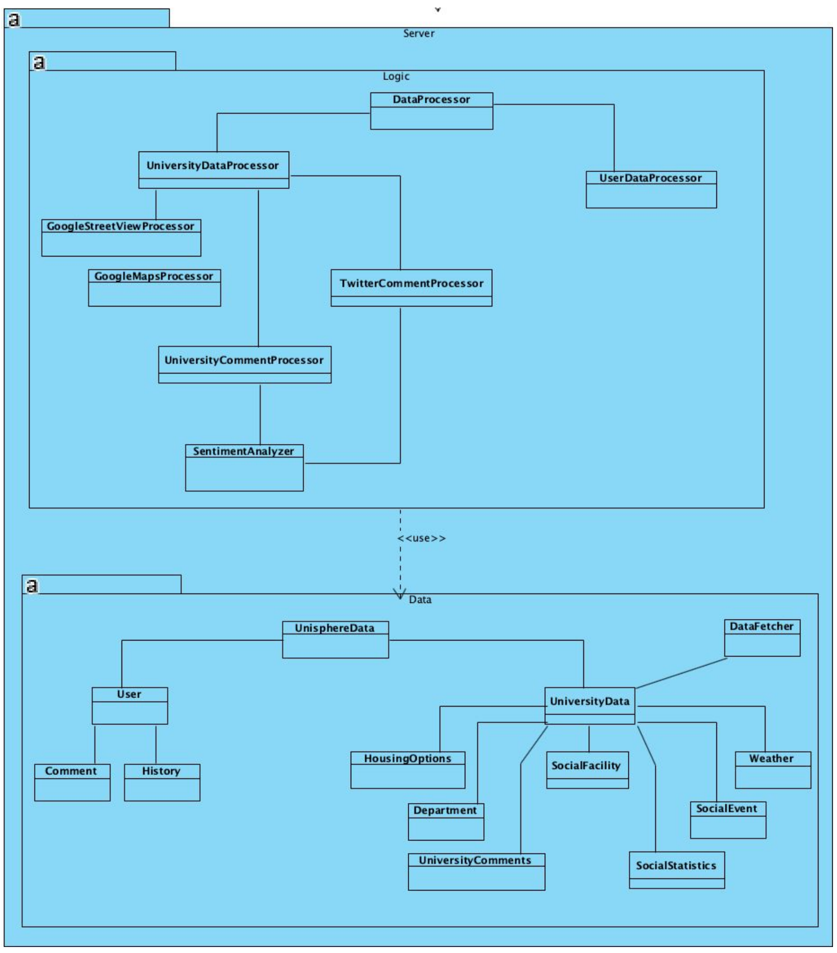

Figure 6: Server Subsystem

4.2.1. Logic Subsystem

The main functionality of Logic subsystem is to manipulate raw data.

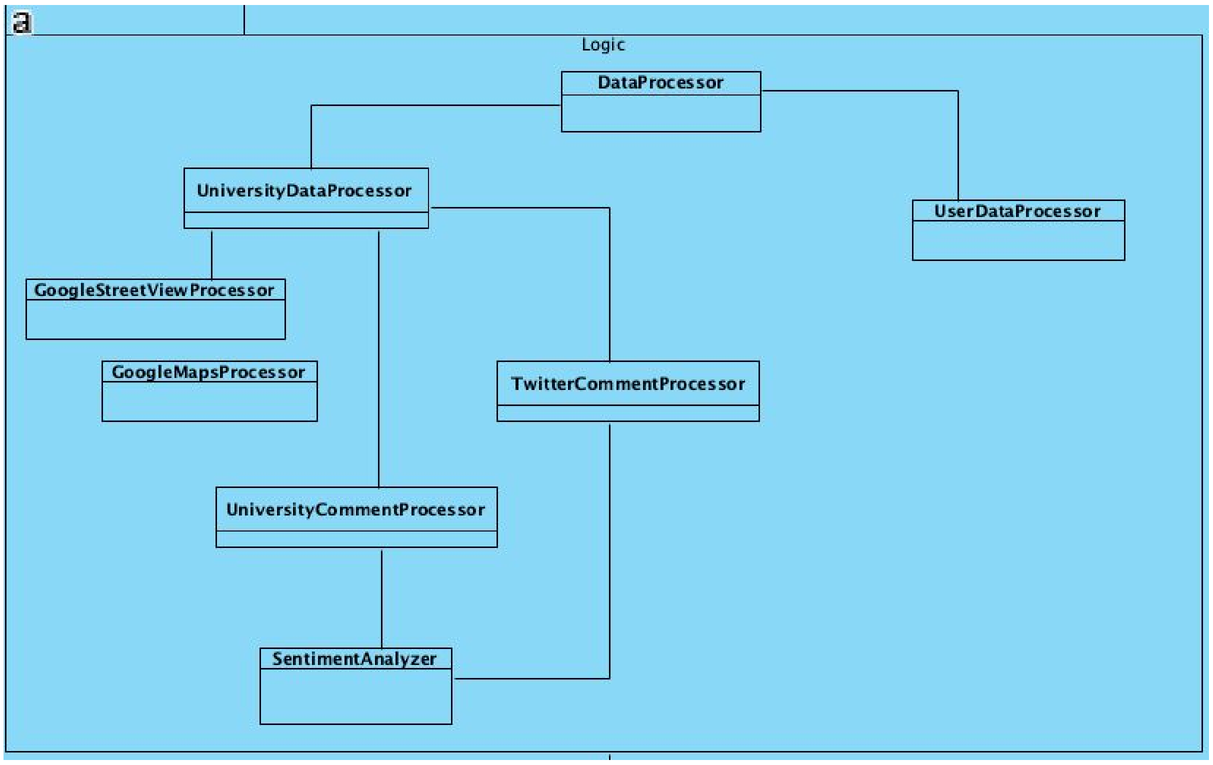

Figure 7: Logic Subsystem

DataProcessor: Data processor is the Facade class for Logic subsystem.

UserDataProcessor: User sign in/login operations will be checked through this class. Furthermore, when user searches new universities, it will update user history in user in Data layer. Based on the user history and priority list selected by user, a list of recommendations will be selected for user.

UniversityDataProcessor: This class is responsible for managing university data. This class and the data subsystem communicates to get data related to universities and create data that is meaningful to the end user.

GoogleStreetViewProcessor: Based on the user requests, it will get the Google Street View data from the Data layer, and will transform in a format which will be accepted by the view subsystem.

GoogleMapsProcessor: This class is the Façade class for Google maps services. It also serves as an adapter between Unisphere internal representation of locations and google maps api.

UniversityCommentProcessor & TwitterDataProcessor & SentimentAnalyzer: Sentiment of the tweets and comments about a university is determined by work done by these three classes SentimentAnalyzer calls UniversityCommentProcessor and TwitterDataProcessor which process the corresponding items and find it's sentiment.

4.2.2. Data Subsystem

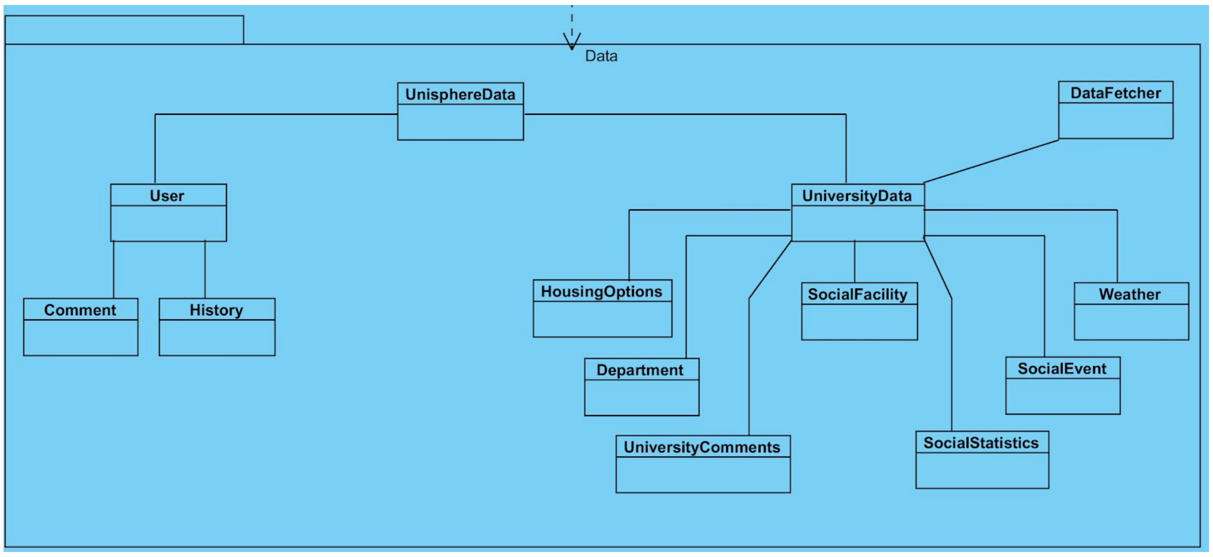

Figure 8: Data Subsystem

UnisphereData: This class manages User and DataFetcher classes.

User: This class is responsible for user-related data(priority list). It is also responsible for the handling of search preferences, account preferences, etc.

Comment: User comments will be handled and stored in this class. New comments will be saved into database.

History: User search history will be fetched from database, or recently searched results will be added by using this class.

UniversityData: This class manages UnisphereData, HousingOptions, Department, UniversityComments, SocialStatistics, SocialFacility, SocialEvent and Weather classes.

HousingOptions: This class will manage information about housing options near the university that is selected.

Department: All the information related to the departments of the university, specifically the list of departments, the PhD., masters programs of the university, academic cadros of those departments will be managed in this class.

UniversityComments: It is responsible for Twitter and Reddit comments related to the university.

SocialFacility: Social facilities, i.e. museums, shopping centers, nearby restaurants etc. will be managed by this class, by using Google Maps API the locations and facilities will be stored.

SocialStatistics: This class will manage information about social statistics of universities.

SocialEvent: Social events near the university will be stored and updated periodically on user request by using this class.

Weather: This class will manage information about the weather conditions at the area of the selected university.

DataFetcher: From several websites, in order to fetch dynamic data, such as social events, academic statistics of a university, current weather data, this class is used.

5. *References*

[1] “Street View Service,” Google, 25-Sep-2018. [Online]. Available: https://developers.google.com/maps/documentation/javascript/streetview. [Accessed: 12-Nov-2018].

[2]“Maps JavaScript API,” Google, 25-Sep-2018. [Online]. Available: https://developers.google.com/maps/documentation/javascript/tutorial. [Accessed: 12-Nov-2018].

[3] “Eventbrite APIv3 Developer Documentation¶,” Eventbrite APIv3 Developer Documentation - Eventbrite Developer Center. [Online]. Available: https://www.eventbrite.com/developer/v3/. [Accessed: 12-Nov-2018].

[4] “Keyword Insights - Twitter Developers,” Twitter. [Online]. Available: https://developer.twitter.com/en/docs/ads/audiences/api-reference/keyword-insights.html. [Accessed: 12-Nov-2018].

[5] reddit.com: api documentation. [Online]. Available: https://www.reddit.com/dev/api/. [Accessed: 12-Nov-2018].

[6] OpenWeatherMap.org, “Weather API,” openweathermap. [Online]. Available: https://openweathermap.org/api. [Accessed: 12-Nov-2018].

[7] “Yelp,” *Yelp*. [Online]. Available: https://www.yelp.com/. [Accessed: 31-Dec-2018].