Bilkent University

Department of Computer Engineering

CS 492 - Senior Design Project II

Unisphere: Global University Catalog

Low 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

Low Level Design Report

February 18, 2019

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

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 all of the relevant websites, they try to find the one, which has all the key information they are looking for. Starting from their high school years, students begin to search universities and they try to find various information about universities 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, they want to compare the universities using the obtained, raw information. Following this urge, instead of manually comparing the universities by searching for the universities separately and using the search results as the comparison criteria, Unisphere offers an alternative way of analyzing universities depending on the preferences of the user. Unisphere aims to be a alternative solution for the tiresome process of searching for the suitable, ideally optimal, universities by automating the search and comparison of candidate universities with respect to criteria determined according to the preferences of the user.

Although, there are some web applications which have a subset of the 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 provide an overview to the low-level design of Unisphere. In the following sections, the trade-offs related to the design of Unisphere, the engineering standards followed by the design and the report, the interface documentation guidelines, the information about the packages, classes and class interfaces of the system and an overall class diagram of the system will be presented respectively.

1.1. Object Design Trade-offs

1.1.1 Functionality vs. Usability

The system design of Unisphere integrates both functionality and usability as two of the crucial design goals of the system. However, in some cases(e.g. if a complex series of interactions are required to achieve a functionality) functionality and usability may not be in coherence. In these cases, Unisphere favors usability over functionality. This decision is due to the fact that the functionalities provided by a program is only important if and only if the user can easily interact with them. On the other hand, some functionalities are complex by their nature. For complex use-cases, extensive guidance to the user is provided.

1.1.2 Space vs. Time

One of the most important design goals of the system design of Unisphere is to present the functionality of the system to the user in the most graspable and fluent way. In order to achieve this property, Unisphere stores the data related to the universities, users, sentiment analysis and routes in the server. This decision provides lesser memory usage of the client machine, higher speed in the web application itself. On the other hand, this decision creates an environment with higher storage space usage at the server side. The storage space in the server side creates additional cost for the application however, the cost is reasonable for better user experience. In brief, The time of the user is prioritized over the cost arises from the use of server storage.

1.1.3 Supportability vs. Cost

Unisphere system mostly deals with dynamic data. Therefore, the data stored in the server must be up to date most of the times. Data coherency is an prioritized design goal of the system. In order to make data coherent with their actual source, the APIs used in the back end side of the system must be purchased in a periodic manner and the server side storage space of the system must continuously be purchased. Since the aim of Unisphere is to provide an alternative way to manually searching and comparing universities, the cost for server side storage and APIs are a lesser of a concern.

1.1.4 Scalability vs. Cost

Unisphere system can be scaled to a higher number users if the server machine is powerful enough and the requests are handled in a stable manner. The cost for scalability is to provide data traffic over-the-air by upgrading the server machine, for a higher cost per usage. If Unisphere becomes popular, the backend upgrade cost will not be a concern since the popularity is the dream of any web application.

1.2. Interface Documentation Guidelines

In this report all of the diagrams are coherent with the following guidelines:

- The names of the classes are a string in the camel-case format where the first character of a class name is uppercase(e.g. ‘ClassName’). The names of the classes in the system are not repetitive and they refer to a single class whenever used.
- The names of the variables start with a lowercase character and also follow camel-case format(e.g. ‘varName’)
- The naming convention for the functions are similar to the naming of variables except the function identifier end with a pair of parentheses(e.g. functionName()).
- The names of the variables and functions have a prefix, namely public or private.

A sample class is represented as follows:

Class SampleClass	
The description of the class interface of SampleClass	
Attributes	
private typeOfAttribute attributeName	The description of the attribute declared at the left column
Methods	
public returnType methodName(parameters)	The description of the function declared at the left column

1.3. Engineering standards

1.3.1. UML

UML is an engineering standard which defines a standard set of notations for representing models based on a software project(Brugge, 750).In this report, UML standard is used for representing class interfaces, hardware-software components and decomposing system to packages.

1.3.2. IEEE

IEEE provides various engineering standards. In this report, the references follow IEEE citation standard.

1.4. Definitions, Acronyms and Abbreviations

API: Application Program Interface

AWS: Amazon Web Service

JSON: JavaScript Object Notation

JS: JavaScript

UI: User Interface

DOM: Document Object Model

SQL: Standardized Query Language

NoSQL: Not only SQL

2. Packages

The low level design is spread across four packages. While View and Controller packages represent the client side, Logic and Data packages represent the server side operations.

The client side is responsible for displaying and allowing navigation through the UI of the application. The server side hosts the website; it handles user requests and manages data.

2.1 Client Package

The client side of the application runs on browsers of the users as a dynamic React JS application. It has two packages: View and Controller. View package is responsible for creating and exporting components for the React application to render.

Once the application is launched, users are directed to the Landing page. The landing page displays basic information about the application and has redirect links to Login and Signup pages. There is also an option to move to the main application page without having an account. Once the user is viewing the main application, they are on a single page that is dynamically altered in order to navigate through the application and view information. This main page has a navigation bar that holds a search bar and search results and a bigger main content components that hosts the globe, map and university information. The different classes and functional components within the View subsystem provide different visual components and surface level dynamic functionality for the UI. The controller package is responsible for providing the communication between the user's browser and the application host server. It talks to the Logic package within the server side in order to retrieve or transmit data per the interactions within the View package. It handles users login and signup operation and also the verification of user entered email address domains to make sure that they indeed belong to a valid university.

2.1.1 View Package

The View package hosts several JS files that are responsible for exporting TypeScript that will be rendered in user's browsers. Each file is a JS class with its own constructor and unique state variables. JS handles these state changes in an efficient way in order to dynamically alter DOM's.View package's responsibilities include: create and display visual components, hold basic surface level dynamic functionality such as handling UI element interactions, host listeners and forward entered inputs to the appropriate controller functionality in order to get requested data from or send data to the database.

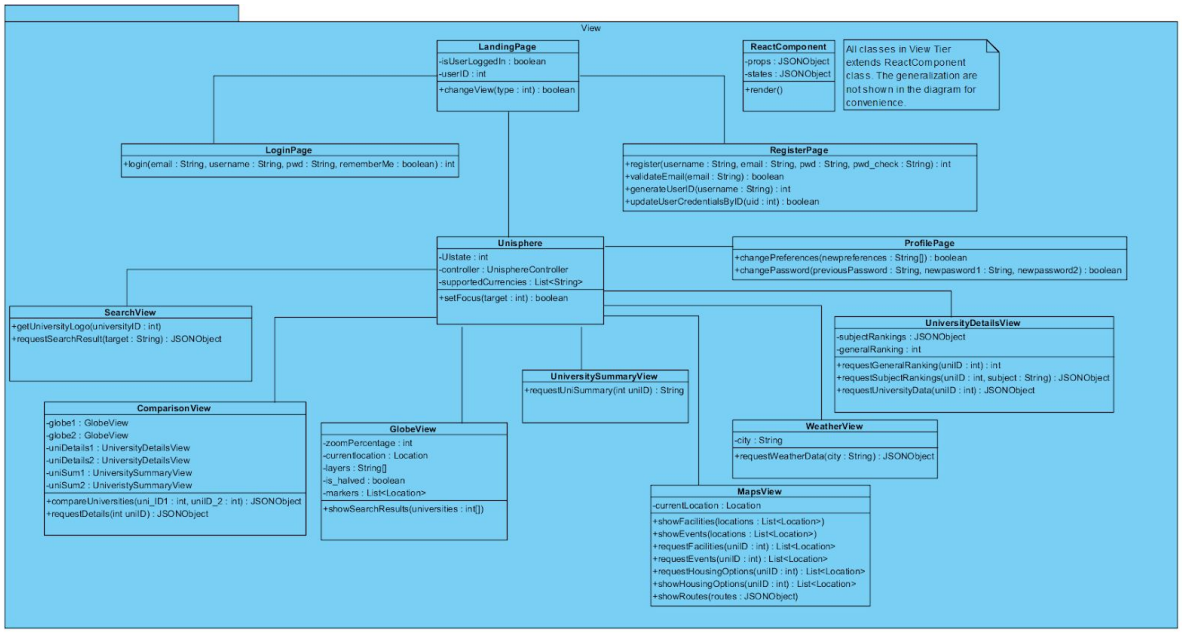

Figure 1. View subsystem in Client

- **LandingPage:** LandingPage is the initial page which welcomes the user, and according to user input, changes the view to Login, Register or Unisphere page.

- **LoginPage:** LoginPage allows registered users to login to the system.

- **RegisterPage:** RegisterPage is used for sign up activities of the users, it gets the email address of the user and checks whether it is a valid one.

- **Unisphere:** Unisphere is the home page of the application. It includes all the views inside and by holding UIState, it shows one of the views.

- **ProfilePage:** ProfilePage displays the profile details of the user and includes the settings. It allows user to select and change their preference list.

- **SearchView:** SearchView is a search bar which allows users to find the universities that they are looking for. The search results appear here as well.

- **ComparisionView:** ComparisionView includes two GlobeViews, according to the selected search results by user, it displays the details of these universities and shows some comparison results.

- **GlobeView:** GlobeView shows a globe,which user can zoom in and out. It can show the locations of universities as a pinpoint on the Earth.

- **UniversitySummaryView:** When user clicks one of the universities that is displayed as a search result, UniversitySummaryView appears on top of GlobeView as a pop-up. It shows a summary information about the university and allows user to view detailed information by a button.

- **MapsView:** MapsView includes GoogleMaps, and according to user's selection it displays the locations of social facilities, social events, dorms/houses in the neighbourhood of the university and it can display the routes from the selected university to these events or facilities.

- **WeatherView:** WeatherView handles visualization of statistical data about weather conditions of a city.

- **UniversityDataView:** UniversityDataView presents all the detailed information related to a university, it displays a general information , departments in that university, Ms and PhD. links of each department, the weather conditions in the city that the university belongs, the MapsView etc. on the user interface.

2.1.2 Controller Package

The controller package has classes that provide network connectivity and data transfer to the server hosted on DigitalOcean. It is not responsible for any operations or modifications to the data, that is handled by the Logic package on the server side. Controller package's responsibilities include: initiate a check if user entered email in sign up page is of a university domain, initiate the authentication process in which a user with the given credentials is logged in or not, transmit data to the server side based on what the user interacted with in the view package and handle data requests from user in the view package.

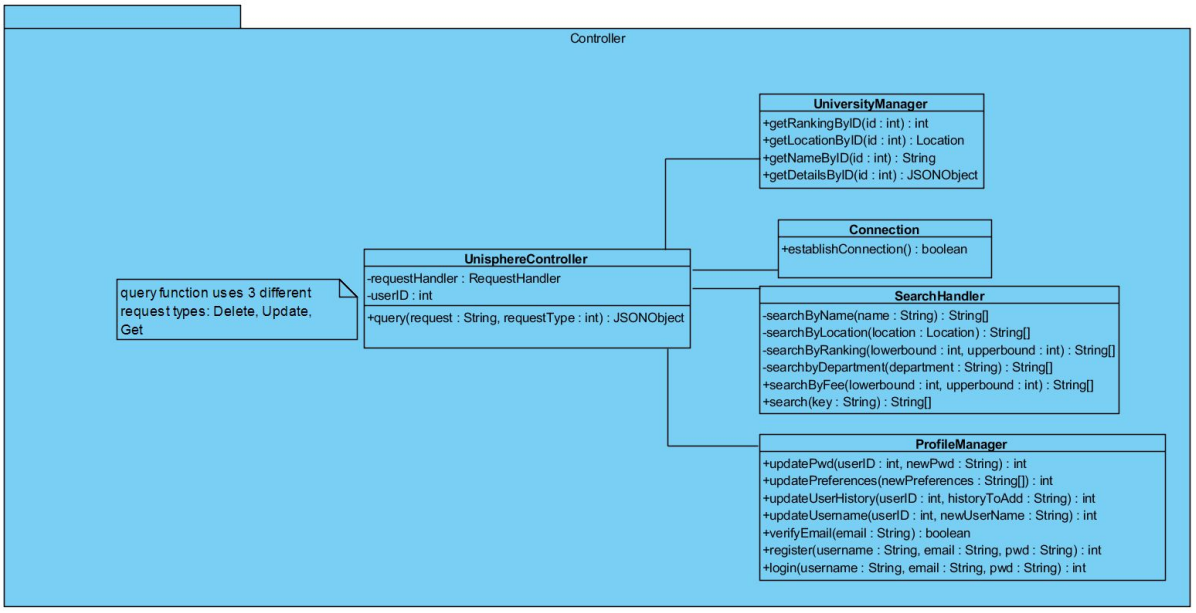

Figure 2. Controller subsystem in Client

- **UnisphereController:** This class interacts with View package and receives user requests and sends these in the expected format to the server, Logic package.

- **UniversityManager:** UniversityManager handles the interaction between Logic and the Controller packages, it retrieves the data requested by UnisphereController.

- **Connection:** This class establishes a connection with the server.

- **SearchHandler:** In order to provide a fast and advanced search by keywords, this class allows different search options and according to these keywords retrieves the data from Logic package.

- **ProfileManager**: This class receives the change requests from UnisphereController and sends these updates to Logic package. It is responsible for the user's search history updates. As well as update operations, it verifies the user account by using an API, checks whether it is a valid university domain or not.

2.2 Server Package

The server side of the application runs on the server machine. The server package consists of two subsytems which are Data and Logic. Data subsystem is responsible for the efficient storage of the data related to the application (e.g. user data, university data) in a NoSQL database. In Logic package, the data in Data layer is transformed into a format which can be used by Client side of the application. Logic subsystem generates and maintains the data related to the application by running website crawling scripts. Furthermore, it analyzes the comments and tweets about a university to understand the opinion of the society formed around the university using machine learning models and fine grained sentiment analysis approaches. This functionality undertaken is achieved by running several scripts sequentially. By using GoogleMaps API, the location of the nearby facilities/events and houses/dorms are detected in the server side in the Logic subsystem as well. Lastly, the routes between these facilities and the university is determined..

2.2.1 Logic Package

Logic package is responsible for handling data requests from the client side's controller package. It provides an interface to both the server side scripts and client side application logic to communicate with the database.

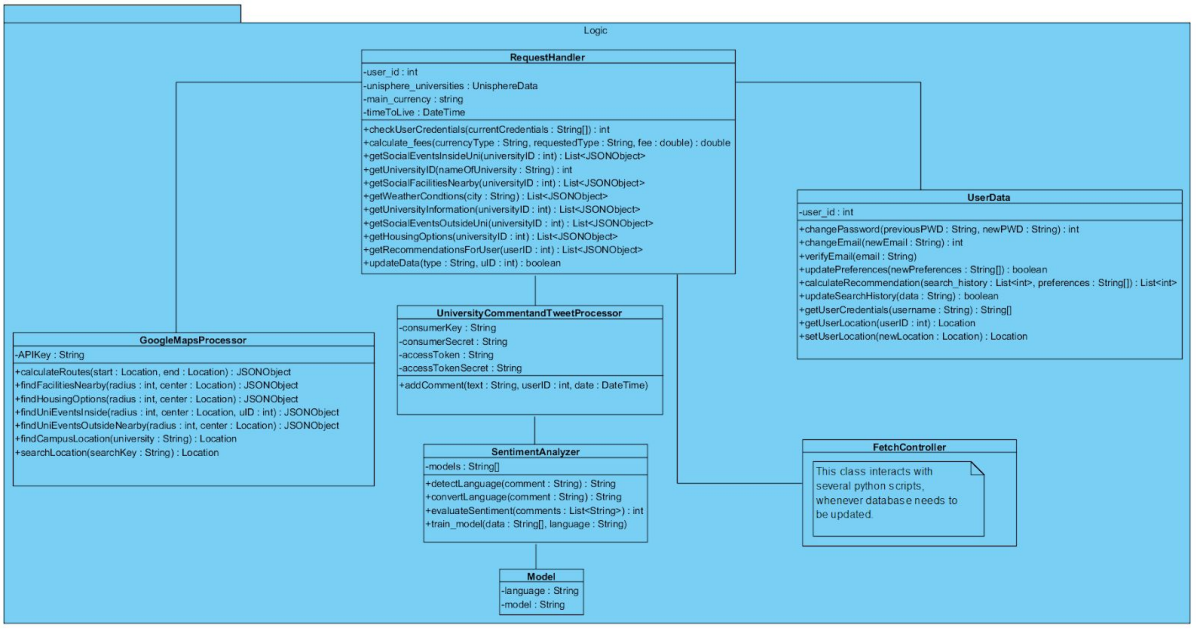

Figure 3. Logic subsystem in Server

- **RequestHandler:** RequestHandler class receives the request from view package and sends back the manipulated data or raw data.

- **UserData:** UserData class detects the current location of the user and changes any user data on request. Besides, according to user's previous search history, it calculates a list of universities as a recommendation.

- **GoogleMapsProcessor:** This class is responsible for communicating with Google's APIs. When the user wants to see 2D maps in order to get information related to places nearby the selected university, locational data will be provided by this class.
- **UniversityCommentTweetProcessor:** This class is the adapter between the Twitter API and the server side. The main functionality of this class is to fetch tweets related to different university daily and store them in the database. Furthermore, the stored tweets are then classified in terms of sentiment and the statistics about the sentiment are updated by the following class.
- **SentimentAnalyzer:** This class is responsible for the sentiment analysis of the tweets and comments related to a university using machine learning and sentiment analysis techniques. Currently the list of supported languages are not finalized however, each language will have a sentiment classifier on scale from 1 to 5 with 5 being the most positive.

- **Model:** This class is a container for a supported language and the classifier corresponding to it.

- **FetchController:** FetchController class interacts with several python scripts whenever data needs to be updated. Since this class denotes set of scripts, there is no clear division of methods and attributes, therefore, the instance denoting this.

2.2.2 Data Package

Data package is the lowest level of the design, which stores the raw data from the database. This layer stores the data in an organized manner and provides it to the Logic layer, and updated the necessary information, when Logic Layer demands it.

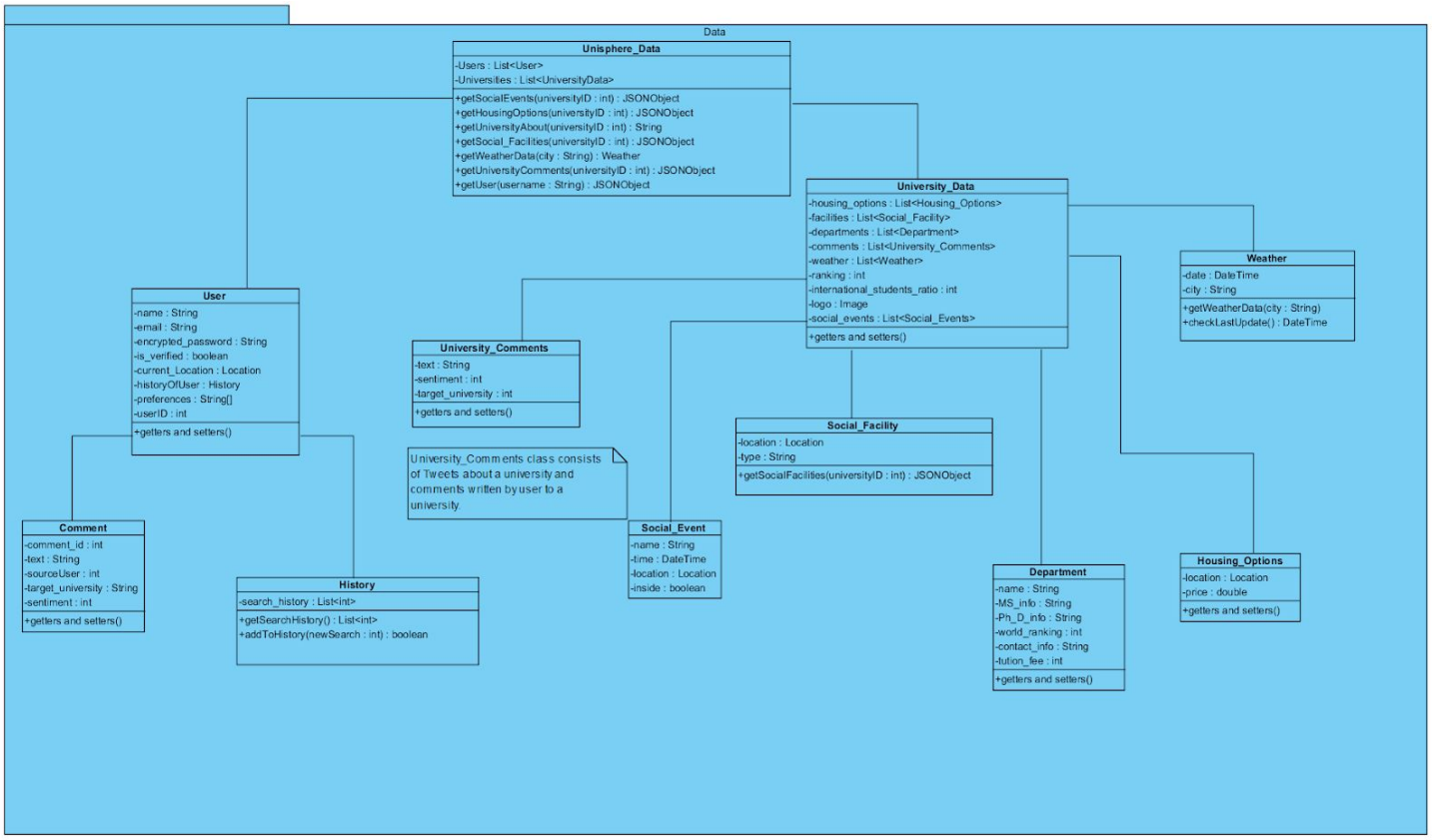

Figure 4. Data subsystem in Server

- **UnisphereData:** UnisphereData class is the Facade class of this package, it interacts with above Logic package and provides raw currently available data in the database by using User and University_Data classes.

- **User:** User class holds user credentials such as username, email address, password and etc. The preferences of the user and their search history are stored in this class as well.

- **Comment:** This class stores the comments made by user to a university.

- **History:** History class keeps a list of recent search history of the user.

- **University_Data:** UniversityData class is responsible for providing any information related to university by using other storage objects such as UniversityComments, SocialFacility, Departments and etc.

- **University_Comments:** UniversityComments is a storage class for Tweets as well as comments made by user to a university.

- **Social_Event:** The properties of social events inside and outside of the university is stored in this class such as the location of the university, name of the event and etc.

- **Social_Facility:** The locations and type of social facilities in the neighbourhood of the university is kept in this class.

- **Department:** Department class stores the information of a department of an university like it's name, tuition fee, world ranking etc.

- **Housing_Options:** This class holds data about housing options nearby the university such as it's location and price.

- **Weather:** Weather class holds the statistical data about the weather conditions in the city, where the university is located.

3. Class Interfaces

In this section of the report each class within a package will be outlined in terms of its high level purpose and low level structure; including private and public functions and variables. The contributions and motives behind the variable and functions in relation to the high level purpose of the class will be described.

3.1 Client Package

3.1.1 View

Class ReactComponent	
ReactComponent allows to create complex UI elements in an easy way, each page extends this class.	
Attributes	
public props	Props stands for the public properties of the component
private state	State holds the information which is controlled only by the component itself.
Methods	
render()	This method updates user interface.

Class Landing Page	
Initial page which welcomes the user, and according to user input, changes the view to Login, Register or Unisphere page.	
Attributes	
private boolean isUserLoggedIn	True if a user was logged in in previous sessions. This variable is used to skip the landing page after a login is made.
private int userID	The user ID that is associated with the login is saved in this variable.
Methods	
public boolean changeView(int type)	Based on high level UI interactions this

	method decides which page to switch to; log in, sign up or Unisphere main app page.

Class LoginPage	
Provides the UI elements that enable a user to login with correct credentials.	
Methods	
public int login(String email, String username, String pwd, boolean rememberMe)	Notifies the controller package of the entered credentials.

Class RegisterPage	
Provides the UI elements that enable a user to register with given credentials.	
Methods	
public int register(String username,String email, String pwd, String pwd_check)	Notifies the controller package of the entered credentials.
public boolean validateEmail(String email)	Notifies the controller of the entered email in order to verify that it is of a valid university or higher education domain.
public int generateUserID(String username)	Notifies the controller that the user needs a new ID assigned to it.

Class Unisphere	
The main application JS file that provides the UI elements that host majority of the applications functionality. It holds other view JS files that provide these functionalities separately.	
Attributes	
private int UIstate	Each view is uniquely encoded as an integer.
private UnisphereController controller	it is an instance of the UnisphereController class to send requests to Controller subsystem.
private List<String> supportedCurrencies	The list of currently supported currencies are used while displaying any tuition fee.

Methods	
public boolean setFocus(int target)	Given int UIState,this method sets focus to any view that will be displayed,

Class ProfilePage	
ProfilePage displays the profile details of the user and includes the settings. It allows user to select and change their preference list.	
Methods	
public boolean changePreferences(String[] newpreferences)	Given list of preferences selected by the user, the view is changed accordingly.
public int changePassword(String previousPassword, String newpassword1, String newpassword2)	Given previousPassword, newpassword2 and newpassword1, user can change their password, the success of the operation, or the fail reason is as an integer as error code.

Class SearchView	
SearchView class is a search bar which allows users to find the universities that they are looking for. The search results appear here as well.	
Methods	
public getUniversityLogo(int universityID)	When user clicks any search results, in order to show the selected university's information, the logo of the university is requested from below layer.
public JSONObject requestSearchResult(String target)	Given target key searched by the user, the list of universities is returned as a JSONObject.

Class ComparisonView	
ComparisonView includes two GlobeViews, according to the selected search results by user, it displays the details of these universities and shows some comparison results.	
Attributes	
private GlobeView globe1	The first globe that is on the left side of the split screen view.
private GlobeView globe2	The second globe that is on the right side of the split screen view.
private UniversityDetailsView uniDetails1	The university details view instance that belongs to the currently displayed university in globe1.
private UniversityDetailsView uniDetails2	The university details view instance that belongs to the currently displayed university in globe2.
private UniversitySummaryView uniSum1	The university summary view instance that belongs to the currently displayed university pop-up in globe1.
private UniversitySummaryView uniSum2	The university summary view instance that belongs to the currently displayed university pop-up in globe2.
Methods	
public JSONObject compareUniversities(int uni_ID1, int uni_ID2)	Given two integer university ID's, the university comparison is requested and the results are returned as JSONObject.
public JSONObject requestDetails(int uniID)	Given universityID, which is selected university by the user, the detailed information is requested.

Class GlobeView	
GlobeView class shows a globe,which user can zoom in and out. It can show the locations of universities as a pinpoint on the Earth.	
Attributes	
private int zoomPercentage	This attribute shows the zoom percentage selected by the user.
private Location currentLocation	currentLocation is the location of the user.
private String[] layers	An array of layers is used to show layers on the globe.
private boolean is_halved	is_halved is a boolean used to check whether view is splitted into two as ComparisonViews or not.
private List<Location> markers	markers are the pinpoint locations which will be added on top of the Earth.
Methods	
public showSearchResults(int[] universities)	Given integer universityID's, this method displays the university locations on the surface of the globe.

Class UniversitySummaryView	
This class provides the UI elements required to display the university summaries when a pin drop on the map is clicked.	
Methods	
public String requestUniSummary(int uniID)	Asks the controller to fetch the university summary information to be displayed in the UI elements.

Class MapsView	
MapsView includes Google Maps, and according to user's selection it displays the locations of social facilities, social events, dorms/houses in the neighbourhood of the university and it can display the routes from the selected university to these events or facilities.	
Attributes	
private Location currentLocation	currentLocation is the location of the user
Methods	
public showFacilities(List<Location> locations)	Given the locations of the facilities, displays these facilities on GoogleMaps.
public showEvents(List<Location> locations)	Given the locations of the events, displays these events on GoogleMaps.
public List<Location> requestFacilities(int uniID)	Given the university ID requests the list of facilities from the controller package.
public List<Location> requestEvents(int uniID)	Given the university ID requests the list of events from the controller package.
public List<Location> requestHousingOptions(int uniID)	Given the university ID, asks the controller to fetch the housing options to be displayed on GoogleMaps.
public List<Location> showHousingOptions(int uniID)	Given the university ID, displays the housing options nearby the university.
public showRoutes(JSONObject routes)	Given the routes in a JSON format, renders the routes on the UI.

Class WeatherView	
This class is responsible for providing the UI elements required to display weather information of university's city.	
Attributes	
private String city	Holds the city that the weather information belongs to.
Methods	
public JSONObject requestWeatherData(String city)	Given the city of the university, asks the controller to fetch weather data.

Class UniversityDataView	
This class is responsible for providing the UI elements for displaying a university's academic information.	
Attributes	
private JSONObject subjectRankings	Holds the rankings per subject within a university.
private int generalRanking	Holds the overall academic ranking of the university.
Methods	
public int requestGenerateRanking(int uniID)	Given the ID of the university, asks the controller to generate ranking of the given university.
public JSONObject requestSubjectRankings(int uniID, String subject)	Given the ID of the university and the subject, asks the controller to fetch subject rankings as a JSONObject.
public JSONObject requestUniversityData(int uniID)	Given the ID of the universtiy, asks the controller to fetch data of the given university in a JSONObject format.

3.1.2 Controller

Class Connection	
This class is responsible for establishing the connection between the server and the client.	
Methods	
public boolean establishConnection()	This method gets no parameters since all of the possible parameters are hardcoded in the function itself. Using the hardcoded variables, this method establishes connection between Client and Server, and returns the success of this operation as boolean.

Class UnisphereController	
This class is the Façade of the controller subsystem since all of the functionality of the controller subsystem can be achieved using UnisphereController. This class uses a previously established connection to the server for exchanging data(by querying) and directing the incoming data to the certain classes in View package.	
Attributes	
private RequestHandler requestHandler	UnisphereController gets data from the server side according to the specified query using this attribute. RequestHandler is the server end of the previously established Client-Server connection. RequestHandler processes the query string and does the specified changes in the server side, gets the existing data or create the specified data.
private int userID	The data exchange between client and server can only be done if the user has already successfully logged in. userID attribute holds the ID of the currently logged in user.
Methods	
public JSONObject query(String request, int requestType)	Given the request itself as a String and the type of the request, this method request a data operation from the server side using the RequestHandler instance stored as the attribute. The request types are enumerated. 1 denotes a GET request, 2 denotes an UPDATE request and 3 denotes a DELETE request. The method returns the specified data as a JSONObject. If the request is of type UPDATE or DELETE, then the JSONObject returned has only one data member whose value is set to True.

Class UniversityManager	
This class is responsible for the data operations that are related to the data about universities. The main goal of this class is to generate queries from the client side information about the needed data item.	
Methods	
public int getRankingByID(int ID)	In the server level, all the universities are represented with a unique integer. Given the university ID, this method generates the related query string for requesting the general ranking of the specified university from the server. Then UnisphereController uses this query string and gets the data from server. This method returns the ranking of the specified university among the other universities in the world.
public Location getLocationByID(int ID)	Using a similar procedure defined for getRankingnbyID method, this method returns the location of the specified university encoded in a Location object
public String getNameByID(int ID)	Using a similar procedure explained above, this method returns the name of a university as a JSONObject given the ID of the university.
public JSONObject getDetailsByID(int ID)	Using a similar procedure explained above, this method returns all of the details about a university as a JSONObject given the ID of the university.

Class SearchHandler	
This class is devoted to handling search related operations at the same place. The search results are altered with respect to the preferences of the current user.	
Methods	
private String[] searchByName(String name)	Given the name of candidate university, returns all of the names of possible universities (both partial and full search, and the deviations from the exact name).
private String[] searchByLocation(Location location)	Given the name of candidate location, returns all of the possible names of locations (both partial and full search, and the deviations from the exact name).
private String[] searchByRanking(int lowerbound, int upperbound)	Given the range of rankings of the candidate search results(both general ranking and the ranking by subject are handled), returns all of the possible universities having rankings in the range(both ends inclusive).
private String[] searchByDepartment(String department)	Given the department name(also handles the similar department names using deviation from the exact department name and similarity), returns all of the possible universities with a department specified as the search key.
private String[] searchByFee(int lowerbound, int upperbound)	Given the lower and upper bounds of the fee in terms of the main currency specified by user, this method returns the candidate universities in sorted order(specified by the user beforehand)
public String[] search(String key)	Given key, entered by the user and passed to this method by UnisphereController, this method will call the necessary search methods defined by this class and return the result as a String array, which is a list of university names.

Class ProfileManager	
This class is devoted to changing and maintaining information about the user's profile.	
Methods	
public int updatePwd(int userID, String newPwd)	Given the ID of the user and a new password, notifies the Logic package on the server side that a password change by the user is made. It returns an integer to indicate an error.
public boolean updatePreferences(int userID, String[] priorityList)	Given the ID of the user and a priority list, notifies the Logic package on the server side that a change to the priority list has been made by the user. Returns a boolean to indicate if the operation was successful or not based on Logic package's response.
public int updateUserHistory(int userID, String historyToAdd)	Given the ID of the user and new history to be added, this method is called whenever the user's history is expanded. This happens when user views details about a university. Returns an integer to cover for several possible error types.
public int updateUsername(int userID, String newUsername)	Given the ID of the user and the desired new username, replaces the current username with the newUsername.
public boolean verifyEmail(String email)	Given email, returns a boolean to indicate if the email is accepted by the server or not.
public int register(String username, String email, String pwd)	Given username or email and user password, registers to the system if the operation is successful.
public int login(String username, String email, String pwd)	Given username or email and user password, notifies the Logic package that a login action was taken by the user. The credentials are sent and Logic package's response is waited. Returns an integer to indicate for several results such as a successful login, wrong credentials etc.

3.2 Server Package

3.2.1 Logic

Class RequestHandler	
This class is responsible for handling requests of any nature. The client will communicate with this part of the server when the user requests to access any kind of data that needs to be computed or processed.	
Attributes	
private int user_id	Each user is assigned a unique id for convenience of the internal operations. The user_id instance here holds the id of the user for which an instance of this class is generated.
private UnisphereData unisphere_universities	This instance is used to reach the data of the packages below.
private String mainCurrency	The user may be from any nation therefore, a main currency is selected using the location of the user. This currency is used to display the tuition fees of the universities. If the tuition fee is not in the unit of the main currency, the fee is converted to the main currency using the most up to date rates.
private DateTime timeToLive	The date time settings of the user are set using the location of the user. This attribute is used to display the date and time on the location of the selected university.
Methods	
public int checkUserCredentials(String [] currentCredentials)	Given the credentials of the current user, the validity of these credentials are checked and an integer that signals whether it is valid or not, or the reason behind,which is encoded as int, it is returned.
public double calculate_fees(String currency_type, String requested_type, double fee)	Given the currency type of the region of the selected university, requested type of the fee and the amount of fee, a new amount of fee is calculated and returned

	as double.
public JSONObject getSocialEventsInsideUni(int universityID)	Given the ID of the university, social events within that university are returned inside a JSONObject.
public int getUniversityID(String nameOfUniversity)	Given the name of the university, retrieves the university ID.
public JSONObject getSocialFacilitiesNearby(int universityID)	Given the ID of the university, social facilities nearby the university are returned inside a JSONObject.
public JSONObject getWeatherConditions(String city)	Given the city of the university,weather conditions about the university are returned inside a JSONObject.
public JSONObject getUniversityInformation(int universityID)	Given the ID of the university, information about the university are returned inside a JSONObject.
public JSONObject getSocialEventsOutsideUni(int universityID)	Given the ID of the university, social events outside the university are returned inside a JSONObject.
public JSONObject getHousingOptions(int universityID)	Given the ID of the university, housing options available are returned inside a JSONObject.
public JSONObject getRecommendationsForUser(int userID)	Given the ID of the user, recommended universities are returned inside a JSONObject.
public boolean updateData(String type, int uniID)	Given type of data related to university as well as the university ID, the data needs to be fetched again, the success of the operation is returned as boolean.

Class GoogleMapsProcessor	
This class is responsible for communicating with Google's APIs. When the user wants to see 2D maps in order to get information related to places nearby the selected university, locational data will be provided by this class.	
Attributes	
private String APIKey	This is the API key required for queries with Google's API's on Google Cloud Platform. It is attached to the end of HTTP requests and needs to be accessible in run-time on the server.
Methods	
public JSONObject calculateRoutes(Location start, Location end)	This function runs when a university is selected. Calculates and returns information related to routes around the university as a JSONObject.
public JSONObject findFacilitiesNearby(int radius, Location center)	This function finds the nearby facilities that are within a certain radius that is given as an int. Returns a JSONObject that hold these facilities.
public JSONObject findHousingOptionsNearby(int radius, Location center)	This function finds the nearby housing options that are within a certain radius that is given as an int. Returns a JSONObject that hold these housing options.
public JSONObject findUniEventsInside(int uniID):	This function takes an integer university ID as input and returns a list of events that are held inside the university.
public JSONObject findUniEventsOutside(int radius, Location center)	This function takes radius and location of the university and returns the nearby events that are held by the university outside of the university as a JSONObject.
public Location findCampusLocation(String university)	This function returns the location of the campus given the name of the university.
public Location searchLocation(String searchKey)	This function directly takes a searchKey as input and searches for a matching location and returns it.

Class UniversityCommentTweetProcessor	
This class is an adapter between the Twitter API and the server side. The main functionality of this class is to fetch tweets related to different university daily and store them in the database.	
Attributes	
All of the attributes are required in order to use the Twitter API.	
private String consumerKey	
private String consumerSecret	
private String accessToken	
private String accessTokenSecret	
Methods	
public boolean addComment(String text, int userID, DateTime date)	Given String text, an integer userID and date as DateTime type, the comment made by user is added to the Data package.

Class UserData	
This class is responsible for the manipulation of user related data, in case of any request made by upper layer, it changes the data stored in below package, besides it calculates recommended universities based on user's search history .	
Attributes	
private int user_id	Each user is assigned a unique id for convenience of the internal operations.
Methods	
public int changePassword(String newPwd, String previousPWD)	Given previous and new password, this method changes the password of the current user. The method returns integer values depending on the error occurred during the update operation(e.g. the user tried the same password as the new password) Each integer code stand for a different type of error or no error.
public int changeEmail(String newEmail)	Changes the stored email of the user with a new one. As the changePwd

	method , integer codes are returned depending on the execution result of the method. On the other hand, as an extra step, each email is verified if it is institutional email using the verifyEmail method described below.
public int verifyEmail(String email)	Checks if the input email is an institutional email using anj off-the-shelf Ruby gem.
public boolean updatePreferences(String[] preferences) newPreferences)	Updates the priority list of the user and a boolean that signals the success of the update is returned.
public List<int> calculateRecommendation(List<int> search_history, String[] preferences)	Given list of universityID's as integer, which is the search history of the user, and the preferences selected by the user, it calculates a list of integers, universityID's as recommended universities.
public boolean updateSearchHistory(String data)	Updates the search history and returned a boolean that signals the success of the update.
public String[] getUserCredentials(String username)	Given the username, user credentials are returned.
public Location getUserLocation(int userID)	Given userID, it returns the last updated location of the user.
public Location setUserLocation(Location newLocation)	Changes the location of the current user. The new location of the user is given as parameter to the method and the method returns old location of the user.

Class SentimentAnalyzer	
This class generates models for trained with phrases in different languages and uses these models to evaluate tweets or comments about a certain university in terms of their positivity and negativity using a scale of 5 where 5 is the most positive reaction and vice versa.	
Attributes	
private String[] models	This attribute holds the trained models for classification of the tweets according to their sentiment. Each model is trained with tweets in a certain language.
Methods	
public String detectLanguage(String comment)	This method is used to detect the language of the input tweet. Comment parameter is either a tweet of Unisphere Comment. The resulting language is used to determine the model to evaluate the sentiment of the comment
public String convertLanguage(String comment)	This method is used to convert comments or tweets written in a language which is not supported by the Unisphere to English. Possibly during the conversion there will be data loss in terms of emotions and sentiments however, this approach is more solid than discarding tweets written in an uncommon language. The method returns the converted version of the input.
public int evaluateSentiment(List<String> comment)	Given comments as Strings, it calculates it's bias whether it is a positive or negative comment, and returns an integer as a result.
public String train_model(String[] data, String language)	By using Machine Learning techniques, this method will train the model in order to evaluate comments and return a sentiment result.

Class Model
This class is used to provide a systematic way of matching languages and corresponding trained models.
Attributes
private String language
private String model

Class FetchController
These set of classes class interacts with several python scripts whenever data needs to be updated. Since this class denotes set of scripts, there is no clear division of methods and attributes. Therefore, the instance denoting this

3.2.2 Data

Class UnisphereData	
It is the Facade class of Data Package, it deals with requests from Logic Tier and provides raw data to Logic Package.	
Attributes	
private List<User> users	
private List<UniversityData> universities	
Methods	
public JSONObject getSocialEvents(int universityID)	Given universityID this method returns the information about social events, which are stored in database, as a JSONObject.
public List<Housing_Options> getHousingOptions(int universityID)	Given universityID this method returns the data about housing options, which are stored in database, as a JSONObject.
public String getUniversityAbout(int universityID)	Given universityID this method returns general university information stored in database as String.
public JSONObject getSocial_Facilities(int universityID)	Given universityID this method returns the information about nearby social facilities, that are stored in database, as a JSONObject.
public JSONObject getWeatherData(String city)	Given city name, this method returns the weather related data that are stored in database as a JSONObject.

public JSONObject getUniversityComments(int universityID)	Given universityID this method returns the university comments that are stored in database as JSONObject.
public JSONObject getUser(String username)	Given username, this method returns the user credentials that are stored in database as JSONObject.

Class User
Data related to the users is stored in this class.
Attributes
private String name
private String email
private String encrypted_password
private boolean is_verified
private Location current_location
private History historyOfUser
private String[} preferences
private int userID
Methods
getters and setters()

Class Comment	
Comments written by user is stored in this class.	
Attributes	
private int comment_id	Unique identification number for a comment. This is implemented for ease of addressing each comment individually.
private String text	The comment itself committed by the user.
private int sourceUser	Username of the user that committed the comment.
private String target_university	Name of the university that the comment belongs to.
private int sentiment	A normalized integer value within a range that indicates the sentiment of a given comment. Low values indicate negative sentiment while high values indicate positive sentiment.
Methods	
getters and setters()	

Class History	
Previous search history of user is stored in this class.	
Attributes	
private List<int> search_history	This attribute holds the list of integer university ID's that are searched by the user recently.
Methods	
public List<int> getSearchHistory()	This method returns the universityID's, which are searched before by user, as a list of integers.
public boolean addToHistory(int newSearch)	Given univeristyID. this method adds recently searched university to user's search history in database, and returns whether this operation is successfully completed as a boolean.

Class University_Data
University_Data class stores all the information related to a university and provides these data to Unisphere_Data.
Attributes
private List<Housing_Options> housing_options
private List<Social_Facility> facilities
private List<Department> departments
private List<University_Comments> comments
private List<Weather> weather
private int ranking
private int international_students_ratio
private Image logo
private List<Social_Events> social_events
Methods
getters and setters()

Class University_Comments
This class stores the comments fetched from Twitter as well as the comments written by Unisphere users.
Attributes
private String text
private int sentiment
private int target_university
Methods
getters and setters()

Class Social_Events
Social_Events class stores the information about social events organized inside and outside of the the university.
Attributes
private String name
private DateTime time
private Location location
private boolean inside
Methods
getters and setters()

Class Social_Facility	
Social_Facility class stores the information about social facilities close to the university.	
Attributes	
private Location location	
private String type	
Methods	
public JSONObject getSocialFacilities(int uniID)	Given univeristyID, this method returns the location and type of the facilities as JSONObject.

Class Department
This class stores all the information related to the department of a university.
Attributes
private String name
private String MS_info
private String Ph_D_info
private int world_ranking
private String contact_info
private int tution_fee
Methods
getters and setters()

Class Housing_Options
Housing_Options class stores the information related to dorms or houses close to the university.
Attributes
private Location location
private double price
Methods
getters and setters()

Class Weather	
Weather class stores weather data of a city that contains a university.	
Attributes	
private DateTime date	
private String city	
Methods	
public JSONObject getWeatherData(String city)	This method returns all the information related to weather as a JSONObject.
public DateTime checkLastUpdate()	This method returns the time of last update made to weather data..

4. Glossary

All classes within the View package are JS files that extend the functionality of the React Component class. Each view file has its state variable and its render method. The state variables provide dynamicity to the application. React automatically calculates an efficient transformation to the virtual DOMs that best reflect the changes in states. For example, a state change can be induced by a button click and once the state is changed, any UI elements that rely its content on that state will have their virtual DOMs automatically updated.

The render method always returns a TypeScript type and is called everytime it should be initially rendered or a corresponding state change is induced.

These JS files can also include other private methods that handle clicks or provide further functionality to the states however we have left those out of the scope of this report as they are more of an implementation detail and are very subject to change. [3]

5. References

[1] *Object-Oriented Software Engineering, Using UML, Patterns, and Java, 2nd Edition*, by Bernd Bruegge and Allen H. Dutoit, Prentice-Hall, 2004, ISBN: 0-13-047110-0.

[2] Bird, Steven, Edward Loper and Ewan Klein (2009).
Natural Language Processing with Python. O'Reilly Media Inc.

[3] "Virtual DOM and Internals – React," – *A JavaScript library for building user interfaces*. [Online]. Available: https://reactjs.org/docs/faq-internals.html. [Accessed: 18-Feb-2019].