Bilkent University

Department of Computer Engineering

Senior Design Project

Project short-name: Augma

Low Level Design Report

Ahmet Burak Şahin, Can Bayraktar, Çağdaş Han Yılmaz, Fırat Sivrikaya, Utku Oymak

Supervisor: Ercüment Çiçek

Jury Members: Bülent Özgüç and İbrahim Körpeoğlu

Progress Report

February 12, 2018

Table of Contents

Low Level Design Report

Project short-name: Augma

1.Introduction

New tools and ideas emerge in the profession of computer science with advancements in technology. Some of these tools or ideas start to grow and become more popular over time. Augmented Reality [1] [2] [3] is one of these emerging technologies which makes it possible to look at the world from a new perspective. Even though it is a relatively new technology, it holds many great possibilities like helping doctors in a surgery, a visual navigation that doesn't need for the driver to look away from the road, just to name a few. Unfortunately, being new also comes with many drawbacks like most users not willing to change what they are using or lack of tool kits and native platforms. Most AR applications today are designed to work on mobile devices [4] [5] like tablets or phones but true potential of this technology can only be unlocked with devices like Google Glass. We expect to see much more advanced AR applications [6] [7] [8] as native AR platforms [9] like Google Glass starts to become affordable by the public and this can only happen if developers keep creating new and interesting Augmented Reality applications, further increasing the value of the technology and attracting more investors to the area.

Social media applications are the most used day-to-day mobile applications in the market. This is why we saw a need and a potential for a new kind of social media application in an augmented space. Augma creates a world where people can interact with each other in an exciting and creative way. Users will be able to post location-based notes on anywhere in the world. Other people who are near one of these notes will be able see it using Augma like a scope, looking into another dimension from their phone's camera. With Augma, we aim to achieve a deeper level of empathy between the users than any other social media application today by putting the readers into the exact same environment where the post was written.

In this report, we aim to provide an overview of the low-level architecture and design of the system we will develop. Firstly, the design trade-off, engineering standards and documentation guidelines are described. Afterwards, the packages in our system and their functionalities are described along with detailed class diagram views. Furthermore, interfaces of all classes in all packages are included.

1.1. Design trade-offs

1.1.1.Functionality vs. Usability

Functionality and usability are two core factors in Augma's design just like they should be in every software. First iterations of the system's design featured a loaded functionality list comprised from a combination of all the popular features currently available in social media applications on the market, but it was quickly abandoned. Reason behind this was to ensure that we don't lose out on the usability coming from simplicity. Since Augma is a social media app in its core, we decided to lean towards ease of use rather than bloat the application with features and functionalities very few people will use.

1.1.2.Security vs. Cost

Augma is a social media platform and this is why security is one of our main concerns. All the user info and Augma Note data will be stored in cloud using AWS. In order to ensure the security of the database and the S3 storage, they will not have direct connections to clients. All the communications and requests will pass through our API Gateway and these connections will be encrypted by AWS. Since these AWS services are not free, providing this level of security will come with a cost.

1.1.3.Cost vs Functionality

Augma's most unique feature is its AR capabilities. This made choosing the right AR module one of the most important decisions we had to make. Even though there are a lot of great AR modules with a ton of features and ease of use, they were all paid libraries with price tags starting at 1000$. Because of this we had to make a trade-off and settled on using EasyAR in Augma's AR module.

1.2. Engineering Standards

In the report, UML design principles are used in the description of class interfaces, diagrams, scenarios and use cases, subsystem compositions, and hardware-software components depictions. UML is a commonly used standard that allows simpler description of the components of a software project. The reports follow the IEEE citation guidelines for the references since they are easy to understand and very commonly used.

1.3. Interface Documentation Guidelines

This report follows the convention where all class names are singular and named with the standard ‘ClassName’ form. The variable and method names follow the same convention ‘variableName’ and ‘methodName()’. The class descriptions follow the order where the class name comes first, the attributes follow, and lastly the methods are listed. After the class names, a brief description and function of the class can be found. The detailed outline is provided below:

Class Name

· Description of class

Attributes

· Attribute name
· Type of attribute

Methods

· Method name
· Parameters
· Return value

1.4. Definitions, Acronyms, and Abbreviations

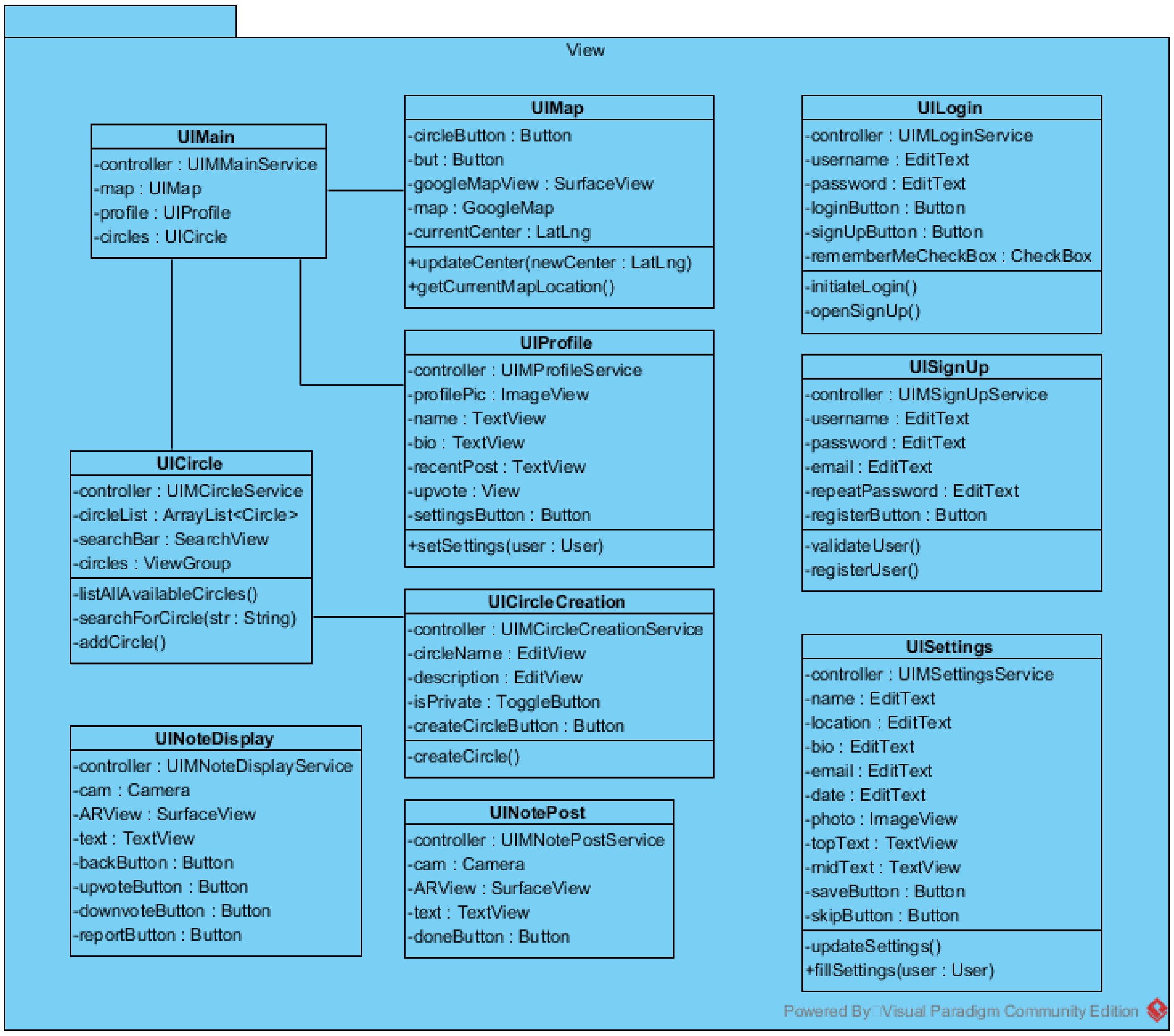
API: Application Programming Interface
AR: Augmented Reality
AWS: Amazon web services
S3: Simple storage service on AWS
Client: The part of the system the users interact with
HTTP: Hypertext Transfer Protocol
TCP: Transmission Control Protocol
UI: User Interface

2.0. Packages

The system is composed of two main packages as client and server. In client package there are three subpackages called View, Controller and AR module. In server package there are two subpackages called Logic and Data.

2.1.Client

2.1.1. View



View package contains the classes related to Graphical User Interface.

UILogin: At the initial startup it is the class that User will encounter its rendered instance. It asks for basic authentication information.

UISignUp: In the case where user has not been registered to the system already, this class will take care of accumulating necessary information and registering the new user to the system. It asks for basic authentication information such as username, password and email.

UIMain: UIMain is the class that will be rendered right after authentication is completed. It consists of three tabs which are Map, Circle and Profile.

UICircle: This class represents the page that provides main operations on circles that user is currently in.

UICircleCreation: UICircleCreation class represents the page that enables user to create a circle.

UINotePost: This class' render provides the main functionality of Augma. In this page, user will be able to prepare and post any note he/she desires. It utilizes the camera so that user can take a snapshot for his/her note.

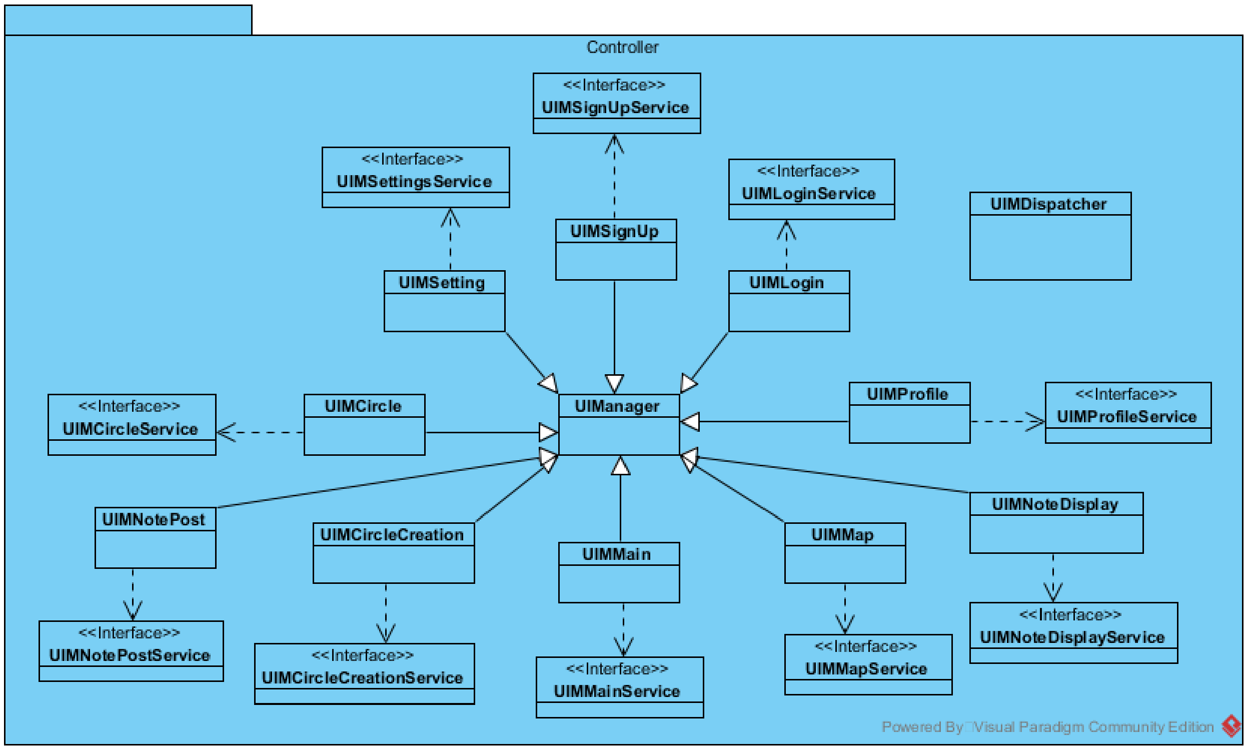
UINoteDisplay: This class' render enables users to see a particular note that is posted on a certain location.

UIMap: UIMap is the class that its render enables users to see whether there are any nearby notes while they roam around their location. It displays the data collected from Google Maps API.

UIProfile: This class represents the profile page of a particular user.

UISettings: UISettings is the class that its render enables users to see/change the settings of their account.

2.1.2. Controller



Controller package contains classes related to managing the Graphical User Interface.

UIManager: Abstract UI manager class that contains the most fundamental logic shared between all UI managers.

UIMSignUp: Manager class that manages the sign up page of the program.

UIMLogin: Manager class that manages the UI operations of user login.

UIMProfile: Manager class that manages the profile pages of the users.

UIMMap: Manager class that is responsible of keeping the map up to date and processing inputs coming from the map interface.

UIMMain: Manager class that manages the main menu of the program.

UIMCircleCreation: Manager class that contains the circle creation aspect of the program.

UIMCircle: Manager class that contains the logic behind circle operations.

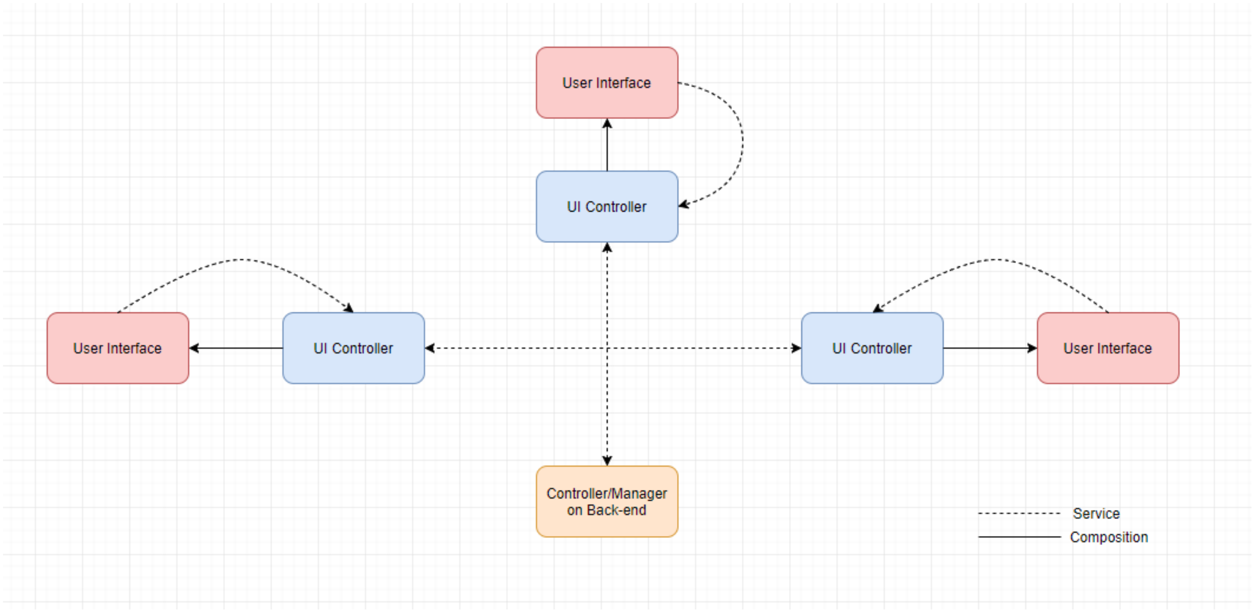
UIMSetting: Manager class that manages the settings of the software.

UIMNotePost: Manager that handles the operations regarding note posting.

UIMNoteDisplay: Manager class that manages the operations upon displayed note post, such as upvoting, reporting, etc.

UIMDispatcher: Manager that switches between the pages or tabs.

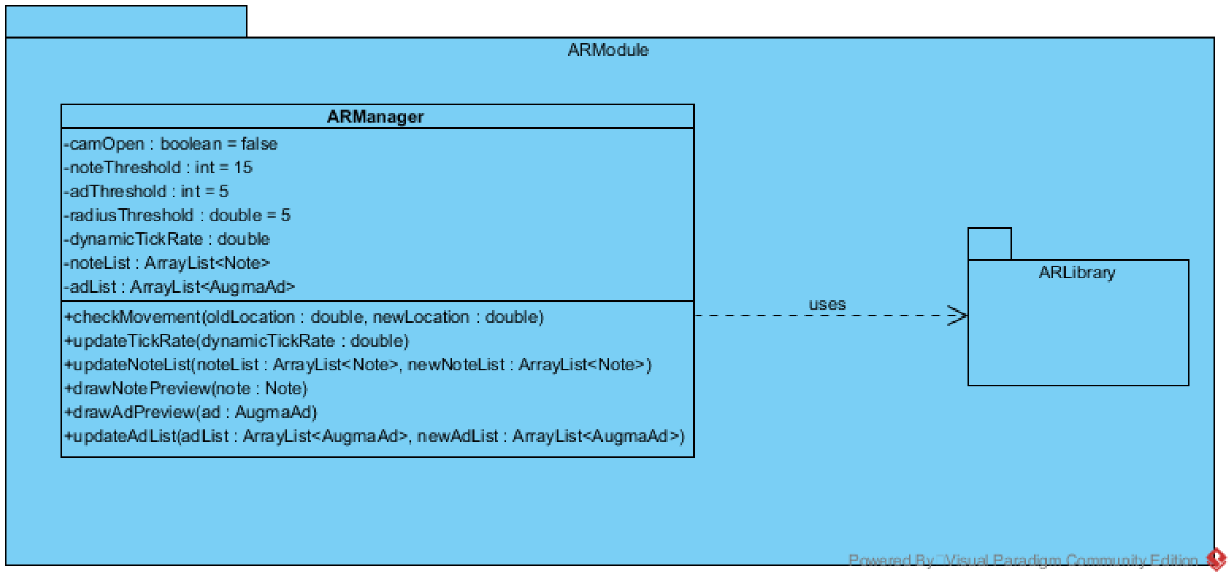
2.1.2.1. Communication through Services



The figure above shows the fundamental controller-to-controller and controller-to-UI communication channels implemented in Augma. The logic behind this setup is that every User Interface will have their own specific controller which enables them to function and communicate with the rest of the software's back-end. To establish the communication back, in a case where a button is pressed and the controller needs to be warned, a service, which is essentially an interface implemented by the controller, is provided to the User Interface element so that a backwards communication can be established. Similarly, communication between both backend and front-end controllers are established through services. If controller A wants to warn controller B about a particular situation, controller A can use controller B's services.

Java Spring for Android technology will be used to ensure that there will only be one copy of any controller instance at any given time. In classes that use services of a controller, a reference for that particular service interface will be sufficient since its implementation will reside in the controller class and Java Spring will take care of assigning controller instance to that interface reference.

2.1.3. AR Module

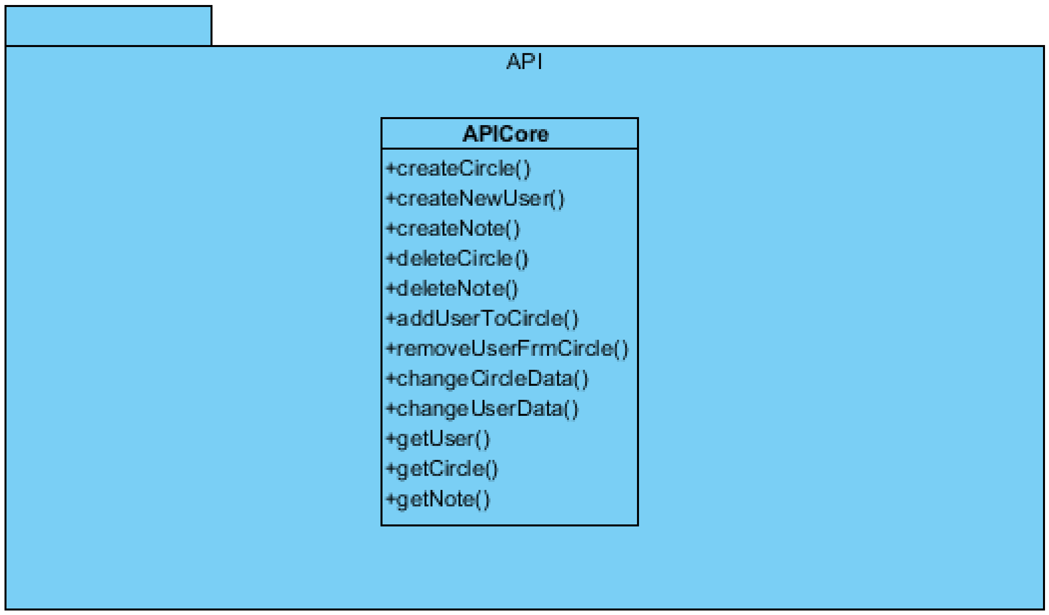

ARModule
ARManager
-camOpen : boolean = false
-noteThreshold : int = 15
-adThreshold : int = 5
-radiusThreshold : double = 5
-dynamicTickRate : double
-noteList : ArrayList<Note>
-adList : ArrayList<AugmaAd>
+checkMovement(oldLocation : double, newLocation : double)
+updateTickRate(dynamicTickRate : double)
+updateNoteList(noteList : ArrayList<Note>, newNoteList : ArrayList<Note>)
+drawNotePreview(note : Note)
+drawAdPreview(ad : AugmaAd)
+updateAdList(adList : ArrayList<AugmaAd>, newAdList : ArrayList<AugmaAd>)
uses
ARLibrary
Powered By Visual Paradigm Community Edition

ARManager: This class represents the manager that is associated with the off-the-shelf AR library and acts as a façade class between UIManager and the off-the-shelf component. Its job is to draw objects such as notes and Augma ads to the camera using AR Library.

ARLibrary: This package represents the off-the-shell AR library that is EasyAr. Bunch of AR libraries are eliminated as Google's required Note 8 and next generations, most of them were too expensive for us and some of them did not have the functionalities that we need. In the end, we decided on EasyAr which suits our application.

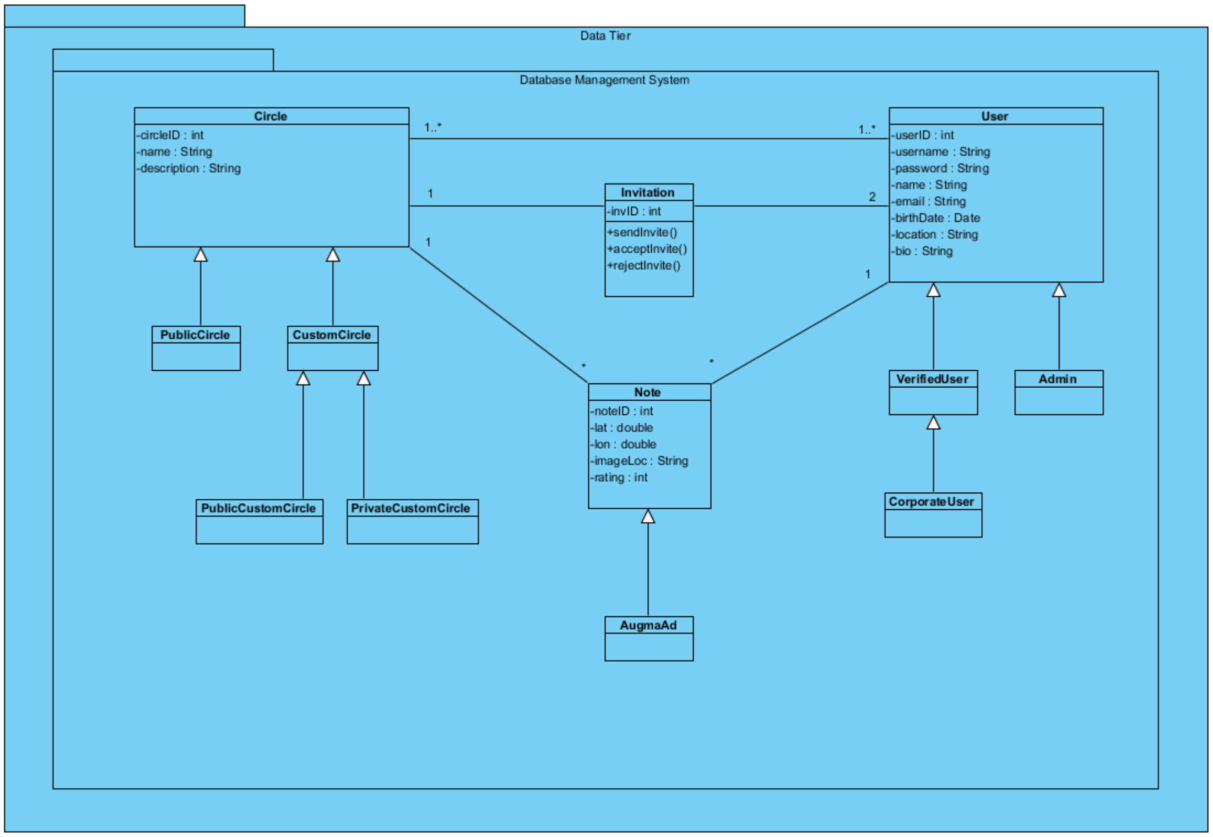
2.2.Server

2.2.1. Logic



API package is the layer responsible for handling the communication between the client side and the data base side of the application. It will be deployed onto the AWS APIGateway service as a RESTful API. It will handle the incoming HTTP requests from the clients and forward them to corresponding Lambda functions, which will parse the requests and talk directly to DynamoDB and S3. APIGateway’s biggest positive will be its integration with the load balancing system in AWS. This will ensure that there aren’t too many requests coming in simultaneously and work like a cushion.

2.2.2. Data



Data tier is responsible for data-related operations. Database management system (DBMS) resides in data tier and keeps the data of our application in server-side.

Circle: This class represents a circle in the system. Each circle has a unique id, and identifying information such as name and description.

User: This class represents a user in the system. Each user has a unique id, and identifying information such as username, password, name, email, birthdate, location and bio. This class also keeps information about the circles that the user is a member of.

Invitation: This class represents an invitation that is sent by a user to another user. It has a unique invitation id and stores information about the circle which is attached.

Note: This class represents a note in the system. Each note is identified by its unique noteID. It keeps location information as latitude and longitude. It also stores the address of the image attached to the note, and rating of the note.

Admin: This class represents a user who has admin privileges.

Verified User: This class represents a user who is verified by an admin.

Corporate User: This class represents a user who is given corporate-level user privileges.

AugmaAd: This class represents a note that has advertisement purpose.

Public Circle: This class represents a public circle in the system.

Custom Circle: This class represents a custom circle in the system.

Public Custom Circle: This class represents a custom circle which is publicly visible.

Private Custom Circle: This class represents a custom circle whose visibility is private.

3.0. Class Interfaces

3.1.Client

3.1.1. View

class UILogin
At the initial startup it is the class that User will encounter its rendered instance. It asks for basic authentication information.
Attributes
private UIMLoginService controller
private EditText username
private EditText password
private Button loginButton
private Button signUpButton
private CheckBox rememberMeCheckBox
Methods
private void initiateLogin() : Initiates login process with the data entered in EditText fields
private void openSignUp() : If user presses sign up button, the program will proceed to sign up process

class UISignUp
In the case where user has not been registered to the system already, this class will take care of accumulating necessary information and registering the new user to the system. It asks for basic authentication information such as username, password and email.
Attributes
private UIMSignUpService controller
private EditText username
private EditText password
private EditText email
private EditText repeatPassword
private Button registerButton
Methods
private void validateUser() : Initiates the validation process which is checking whether or not a

user with such information exists
private void registerUser() : Initiates the registration process

class UIMain
UIMain is the class that will be rendered right after authentication is completed. It consists of three tabs which are Map, Circle and Profile.
Attributes
private UIMMainService controller
private UIMap map
private UIProfile profile
private UICircle circles

class UICircle
This class represents the page that provides main operations on circles that user is currently in.
Attributes
private UIMCircleService controller
private ArrayList<Circle> circleList
private SearchView searchBar
private ViewGroup circles
Methods
private void listAllAvailableCircles() : This method creates a group of View's based on the circleList and puts them on display.
private void searchForCircle(str : String) : This method prompts backend to search for a particular circle.
private void addCircle() : If any circle is added on fly, this method will update the ViewGroup

class UICircleCreation
UICircleCreation class represents the page that enables user to create a circle.
Attributes
private UIMCircleCreationService controller
private EditView circleName

private EditView circleDescription
private ToggleButton isPrivate
private Button createCircleButton
Methods
private void createCircle() : This method prompts backend to register newly created circle.

class UINotePost
This class' render provides the main functionality of Augma. In this page, user will be able to prepare and post any note he/she desires. It utilizes the camera so that user can take a snapshot for his/her note.
Attributes
private UIMNotePostService controller
private Camera cam
private SurfaceView ARView
private TextView text
private Button doneButton

class UINoteDisplay
This class' render enables users to see a particular note that is posted on a certain location.
Attributes
private UIMNoteDisplayService controller
private Camera cam
private SurfaceView ARView
private TextView text
private Button backButton
private Button upvoteButton
private Button downvoteButton
private Button reportButton

class UIMap
UIMap is the class that its render enables users to see whether there are any nearby notes while they roam around their location. It displays the data collected from Google Maps API.

Attributes
private UIMMapService controller
private Button button
private SurfaceView googleMapView
private GoogleMap map
private LatLng currentCenter
Methods
public void updateCenter(newCenter : LatLng) : If any custom location is provided, controller of this class may change the focal point of the map.
public void getCurrentMapLocation() : Whenever the current center is needed by the controller, this method can be called.

class UIProfile
This class represents the profile page of a particular user.
Attributes
private UIMProfileService controller
private ImageView profilePic
private TextView name
private TextView bio
private TextView recentPost
private View upvote
private Button settingsButton
Methods
public void setSettings(user : User) : Settings of the provided user will be displayed.

class UISettings
UISettings is the class that its render enables users to see/change the settings of their account.
Attributes
private UIMSettingsService controller
private EditText name
private EditText location

private EditText bio
private EditText email
private EditText date
private ImageView photo
private TextView topText
private TextView midText
private Button saveButton
private Button skipButton
Methods
private void updateSettings() : Updates the newly entered settings for the user.
public void fillSettings(user : User) : Settings of the provided user will be filled.

3.1.2. Controller

class UIManager
Abstract UI manager class that contains the most fundamental logic shared between all UI managers.
Attributes
Methods
private void switchToMap() : Switches the current tab to map.
private void switchToCircle() : Switches the current tab to circle.
private void switchToCircleCreation() : Switches the current tab to circle creation.
private void switchToCamera() : Switches the current tab to camera.
private void switchToProfile() : Switches the current tab to profile.
private void switchToNotePost() : Switches the current tab to note post.
private void switchToNoteDisplay() : Switches the current tab to note display.
private void switchToSettings() : Switches the current tab to setting.

class UIMSignUp
Manages the sign up page of the program.
Attributes
private String username

private String password
private String name
private String surname
private String email
Methods
private void createNewUser() : Creates a new user based on the information given.
private void addToDb(User:user): Adds the user to the database.

class UIMLogin
Manages the login page of the program.
Attributes
private String username
private String password
Methods
private void authenticate(String username : String password) : Authenticates the user
private void login() : Initiates the login process

class UIMProfile
Manages the profile page of a user.
Attributes
private Image picture
private String name
private String surname
private String email
private String bio

class UIMMap
Manages the map.
Attributes
private ArrayList<Notes> noteList
Methods

class UIMCircleCreation
Manages the circle creation page.
Attributes
private String circleName
private String circleDescription
Methods
private void createCircle(String circleName, String circleDescription) : Creates a circle with the given information.
Private void addtoDb(Circle: circle): Adds the circle to the database.

class UIMCircle
Manages the already created Circles.
Attributes
private ArrayList<Circle> circleList
private String circleName
private String circleDescription
private int numberOfPeople

class UIMSetting
Manages the settings of a user.
Attributes
private String name
private Location location
private String bio
private String email
private Date date
private Image photo
Methods
private void updateSettings() : Updates the settings of the user.

class UIMDispatcher
Dispatches the tabs and pages.

Attributes
Methods
private void dispatchCamera(): Dispatches the camera.
private void dispatchCircles(): Dispatches the circles.
private void dispatchProfile(): Dispatches the profile page.
private void dispatchNotes(): Dispatches the notes.

3.1.3. AR Module

class ARManager
This class represents the manager that is associated with the off-the-shelf AR library and acts as a façade class between UIManager and the off-the-shelf component. Its job is to draw objects such as notes and Augma ads to the camera using AR Library.
Attributes
private boolean camOpen
private int noteThreshold
private int adThreshold
private double radiusThreshold
private double dynamicTickRate private ArrayList<Note> noteList
private ArrayList<AugmaAd> adList
Methods
checkMovement(double oldLocation, double newLocation)
updateTickRate(double dynamicTickRate)
updateNoteList(ArrayList<Note> noteList, ArrayList<Note> newNoteList)
updateAdList(ArrayList<AugmaAd> adList, ArrayList<AugmaAd> newAdList)
drawNotePreview(Note note)
drawNotePreview(AugmaAd ad)

3.2.Server

3.2.1. Logic

class ARManager
This class represents the manager that is associated with the off-the-shelf AR library and acts as a façade class between UIManager and the off-the-shelf component. Its job is to draw objects such as notes and Augma ads to the camera using AR Library.
Attributes
private boolean camOpen
private int noteThreshold
private int adThreshold
private double radiusThreshold
private double dynamicTickRate
private ArrayList<Note> noteList
private ArrayList<AugmaAd> adList
Methods
checkMovement(double oldLocation, double newLocation)
updateTickRate(double dynamicTickRate)
updateNoteList(ArrayList<Note> noteList, ArrayList<Note> newNoteList)
updateAdList(ArrayList<AugmaAd> adList, ArrayList<AugmaAd> newAdList)
drawNotePreview(Note note)
drawNotePreview(AugmaAd ad)

3.2.2. Data

class Circle
This class represents a circle in the system. Each circle has a unique id, and identifying information such as name and description.
Attributes
private int circleID
private String name
private String description

class User
This class represents a user in the system. Each user has a unique id, and identifying information such as username, password, name, email, birthdate, location and bio. This class also keeps information about the circles that the user is a member of.
Attributes
private int userID
private String username
private String password
private String name
private String email
private Date birthDate
private String location
private String bio
private List<Circle> joinedCircles
private List<Circle> ownedCircles
private List<Note> ownedNotes
private List<Invitation> pendingInvites

class Invitation
This class represents an invitation that is sent by a user to another user. It has a unique invitation id and stores information about the circle which is attached.
Attributes
private int invID
private User sender
private User receiver
private Circle circle
Methods
public boolean sendInvite(User sender, User receiver, Circle circle): Sends an invite to the receiver user from sender user, adds the invitation to the pending list of receiver user.
public boolean acceptInvite(Invitation invitation) : The caller user accepts the given invitation in the pending list and joins the circle, returns false if invitation could not be found.

public boolean rejectInvite(User sender, User receiver, Circle circle) : The caller rejects the given invitation in the pending list if found, returns false if the invitation could not be found.

class Note
This class represents a note in the system. Each note is identified by its unique noteID. It keeps location information as latitude and longitude. It also stores the address of the image attached to the note, and rating of the note.
Attributes
private int noteID
private double lat
private double lon
private String imageLoc
private int rating

class Admin extends User
This class represents a user who has admin privileges.

class VerifiedUser extends User
This class represents a user who is verified by an admin.

class CorporateUser extends User
This class represents a user who is given corporate-level user privileges.

class AugmaAd extends Note
This class represents a note that has advertisement purpose.

class PublicCircle extends Circle
This class represents a public circle in the system.

class CustomCircle extends Circle
This class represents a custom circle in the system.

class PublicCustomCircle extends Circle
This class represents a custom circle which is publicly visible.

class PrivateCustomCircle extends Circle
This class represents a custom circle whose visibility is private.

References

[1] WoodrowBarfield, Fundamentals of Wearable Computers and Augmented Reality, CRC Press, 2016.

[2] Matt Bower, Cathie Howe, Nerida McCredie, Austin Robinson & David Grover, «Augmented Reality in education – cases, places and potentials,» *Educational Media International,* 2014.

[3] Mark Billinghurst, Adrian Clark, Gun Lee, «A Survey of Augmented Reality,» *Foundations and Trends in Human-Computer Interaction,* cilt 8, pp. 73-272, 2014.

[4] Clemens Arth, Lukas Gruber, Raphael Grasset, Tobias Langlotz,Alessandro Mulloni, Dieter Schmalstieg, Daniel Wagner, «The History of Mobile Augmented Reality- Developments in Mobile AR over the last almost 50 years,» Inst. for Computer Graphics and Vision Graz University of Technology, Graz, 2015.

[5] Thomas Olsson, Kaisa Väänänen, «Expected user experience of mobile augmented reality services: A user study in the context of shopping centres,» *Personal and Ubiquitous Computing,* 2011.

[6] Philip Geiger, Marc Schickler, Rudiger Pryss, Johannes Schobel, Manfred Reichert, «Location-based Mobile Augmented Reality Applications,» 2014.

[7] Panos E. Kourouthanassis, Costas Boletsis, George Lekakos, «Demystifying the design of mobile augmented reality applications,» *Multimedia Tools and Applications,* 2013.

[8] Rüdiger Pryss,Philip Geiger,Marc Schickler,Johannes Schobel,Manfred Reichert, «Advanced Algorithms for Location-Based Smart Mobile Augmented Reality Applications,» *Procedia Computer Science,* cilt 94, pp. 97-104, 2016.

[9] Zhihan Lv ,Alaa Halawani , Shengzhong Feng , Shafiq ur Re´hman, Haibo Li, «PreprintTouch-less Interactive Augmented Reality Game on Vision Based Wearable Device,» 2014.