CS 319 - Object-Oriented Software Engineering

Analysis Report

v.map

Group 3-C

Yağız Gani

Kanan Asadov

Ömer Faruk Karakaya

Mert Osman Dönmezyürek

1.0 Introduction

1.1 Purpose Of The System

v.map is a web application that will serve for Community Volunteer Foundation.This foundation has more than 400,000 volunteers all around Turkey and they need to communicate with each other when they require some help for their events or efforts for the society. They used to inform their needs by reference or old-fashion e-mailing. At this point, v.map will be useful for all members of the foundation. The application will allow them to find a nearby announcement of all community or announce their request based on their locations on Turkey map. It will also give them a huge opportunity to share their contact information in a very efficient and easy way to meet the needs. The user-friendly and perfectly simple interface of v.map will make all of them happen. In addition to those purposes, our application will work on every web browser, every operating system or every device such as mobile, desktop, etc. When there is Internet connection, this will enable the users to access the system at any time and on any device.

1.2 Design Goals

End User Criteria

Usability: v.map is a helpful application designed for Community Volunteers Foundation. While helping the foundation members to communicate with each other, we also aim to design a fluent and user-friendly interface by eliminating endless process pages. That is why

v.map contains some pop-up windows to show all in one with a simple appearance rather than has numberless different web pages.

Learnability: Since v.map aims to make the usage of the application very fluent and simple, it will not take any remarkable time to learn. Just few steps will lead them to what they want. They will not get lost among the complex and countless web pages, all steps continue on several pop-up windows. This will allow the user not to spend so much time to learn how to use.

Adaptability: Java® has a significant advantage among the other programming languages, which is cross-platform portability. This enables the users to access the application from anywhere. They will just need a web browser because our application works on a web server.

Maintenance Criteria

Extensibility: v.map is meant to provide usability and easy communication for its users so that we will make it proper for adding new features and modifying the current ones according to the use or need of members. In order to sustain the interest of the users and usability, our desing of the application will be suitable to add new features or functions.

Portability: Portability is the one of the most important subject when it comes to software. It enables the applications to transfer into a different system that already exists without requiring any new configuration of a system. Since we want to utilize that adventage, our application is

implemented in Java®. With the power of JVM, our application can be moved into different servers, devices or operating systems without a concern about compatibility.

Performance Criteria

Efficiency: Announcements have the largest data to download since there are many of them and they contain pictures etc. Thus, we focused on implementing this part as efficient as possible. The system downloads only the must have data such as coordinates and titles of the announcements and downloads the rest only when the user wants to see the details. This way we have less data downloaded and stored on the device at the same time which makes the application run smoother.

Reliability: The system will be as bug-free as possible. It will have only one page and many pop-up windows. We will be checking everything that the system can produce after closing the pop-up window and take care of it. Since the organization has over 400,000 volunteers we will focus on the backend so that it will be able to hold this amount of pressure.

Trade-offs

Dynamic UI vs. CPU load: Since we are using Java Server Faces, our interface on the browser side (HTML code) is created dynamically by the server itself. This will allow us to provide a very user-friendly and simple

interface for our users. On the other hand, those kind of features for a fluent interface will require a better CPU which will be costly.

1.3 Definitions

Cross-platform: ability to run on different platforms such as Microsoft Windows, Android, Linux and Mac OS X.

MVC: Model View Controller

JRE: Java Runtime Environment

JDK: Java Development Kit

JVM: Java Virtual Machine

1.4 References

[1] Object-Oriented Software Engineering, Using UML, Patterns, and Java, 3rd Edition, by Bernd Bruegge and Allen H. Dutoit, Prentice-Hall, 2010, ISBN-10: 0136066836.

1.5 Overview

We have represented the purpose of our application. The aim is becoming an efficient utility for the member of the Community Volunteer Foundation with a user-friendly, fluent, and simple interface while they are communicating each other. when we determined definition of v.map, we considered the portability, usability, learnability, adaptability, extensibility, efficiency, and reliability.

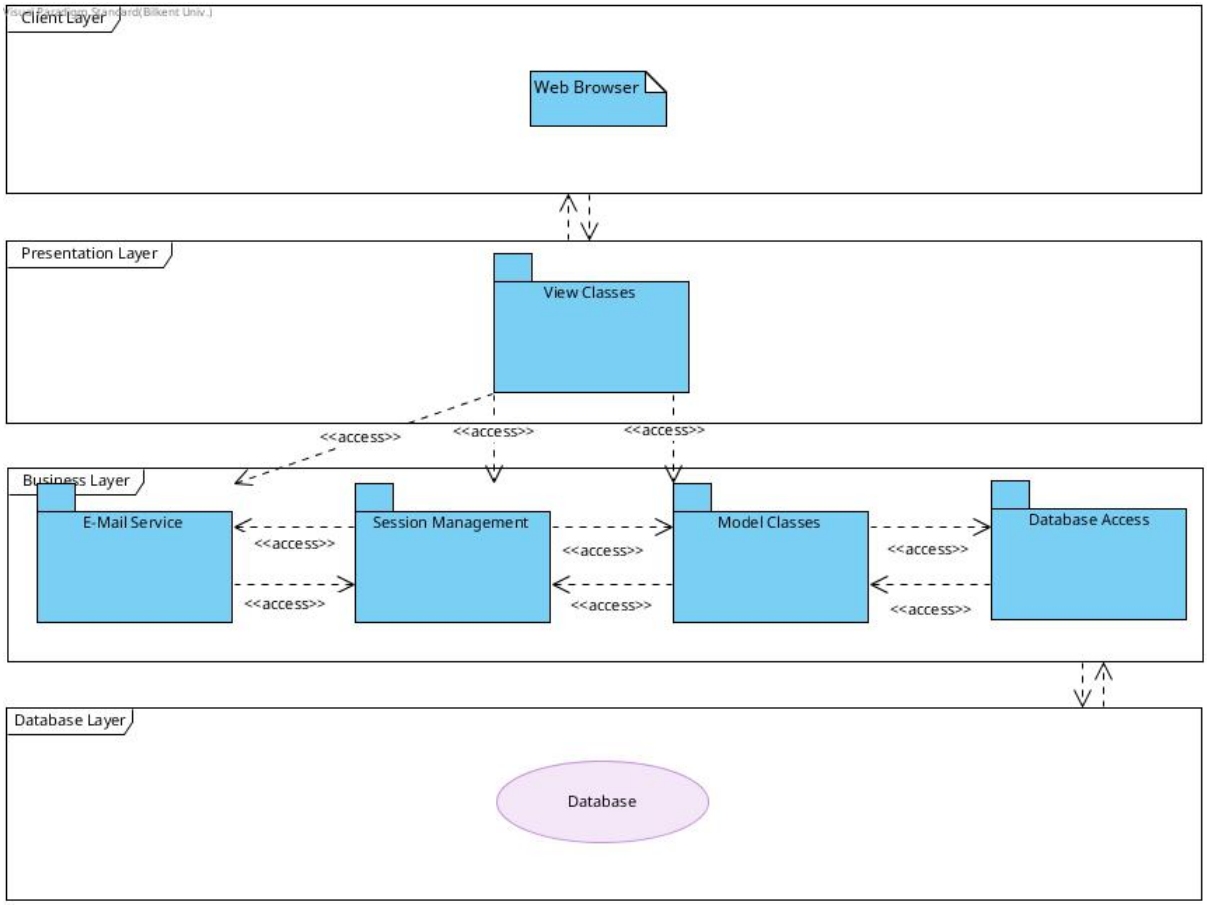
2.0 Proposed Software Architecture

2.1 Overview

In this section, we will decompose our system into maintainable subsystems. By dividing whole system to 5 subsystems, we aim to create 5 different sub-projects. On the other hand we tried to decompose our system in order to apply MVC(Model View controller) architectural style on our system.

2.2 Subsytem Decomposition and Layered Design

In this section, the system is divided into relatively independent parts to clarify how it is organized. In Figure-1 it is shown that how layers and subsytems are arranged and how relation between them are managed.



2.2.1 Layers

- Client Layer : The Layer where View Classes already translated to HTML file by application server.
- View Classes : The Layer where UI instances are initialized and this layer is bridge between business layer and user interfaces.
- Bussiness Layer: The Layer where logic of the project is initialized.
- Databes Layer: The layer where whole Data is kept in PostgreSQL .

2.2.2 Subsystems

- View: This subsytem is consist of JSF clases in which dynamic UI HTML creating clases initialized. Viev Clases have access to E-mail Service, Session Management, Model Clases.
- E-Mail Service: This subsytem consist of Java clases to send verification emails tu users. This subsystem have access to Session Management.
- Model : This subsytem consist of Java classes which represent entity objects. This subsytem have acces to Database Management subsytem.
- Database Access: This subsytem consist of Java classes which are accessing database by using Hibernate Framework.

2.3 Hardware/Software Mapping

v.map will be implemented in Java Enterprise Edition 8. As hardware configuration, v.map needs just a mouse and keyboard to manage entries. Since w.map is and web project javascript compatible web browser is sufficient for using v.map. On the development side a computer with java JDK, javascriot compatible web browser, application server like apache Tomcat and Database Server PostgreSQL is needed.

2.4 Persistent Data Management

2.4.1 Identiying Persistent Objects

v.map has two set of objects that must be stored. One set is Announcement’s which are created and accessed by users and they need to be persistent because they will be accessed just when they needed, also they shouldn’t be lost even server shout down. The other set is User’s which represent data of signed in users. They also should be persistent because of the same reasons of Announcements.

2.4.2 Selecting a Storage Strategy

Relational database will be used for store data. PostgreSQL database is chosen as a relational database software for this project. The reason why PostgreSQL is chosen is PostgreSQL is widely used in business application and it is opensource. An Object Relational Mapping will be used for database control and automatcly creating model from objects. Hibernate ORM is choosen as a ORM software. The reason why Hibernate is chose is Hibernate is widely used in business applicaitons, it is an opensource software an it has very large amount of community support.

2.5 Access Control And Security

Access control management is very important for our web application because we have authentication for different type of users such as admin, users or guests. the range of the permissions differs for every group. For example, guests have very limited permissions while accessing the other users’ information or the

details of the announcements that the users or the admin shared. The admins of the system, on the other hand, have extra features and permissions such as deleting other users' accounts or managing their posted announcements. To ensure the security of the application and the database, we will be controlling the authentication for every step of every user. Moreover, we will be checking how often a user creates an announcement to protect our application and server from the spammers.

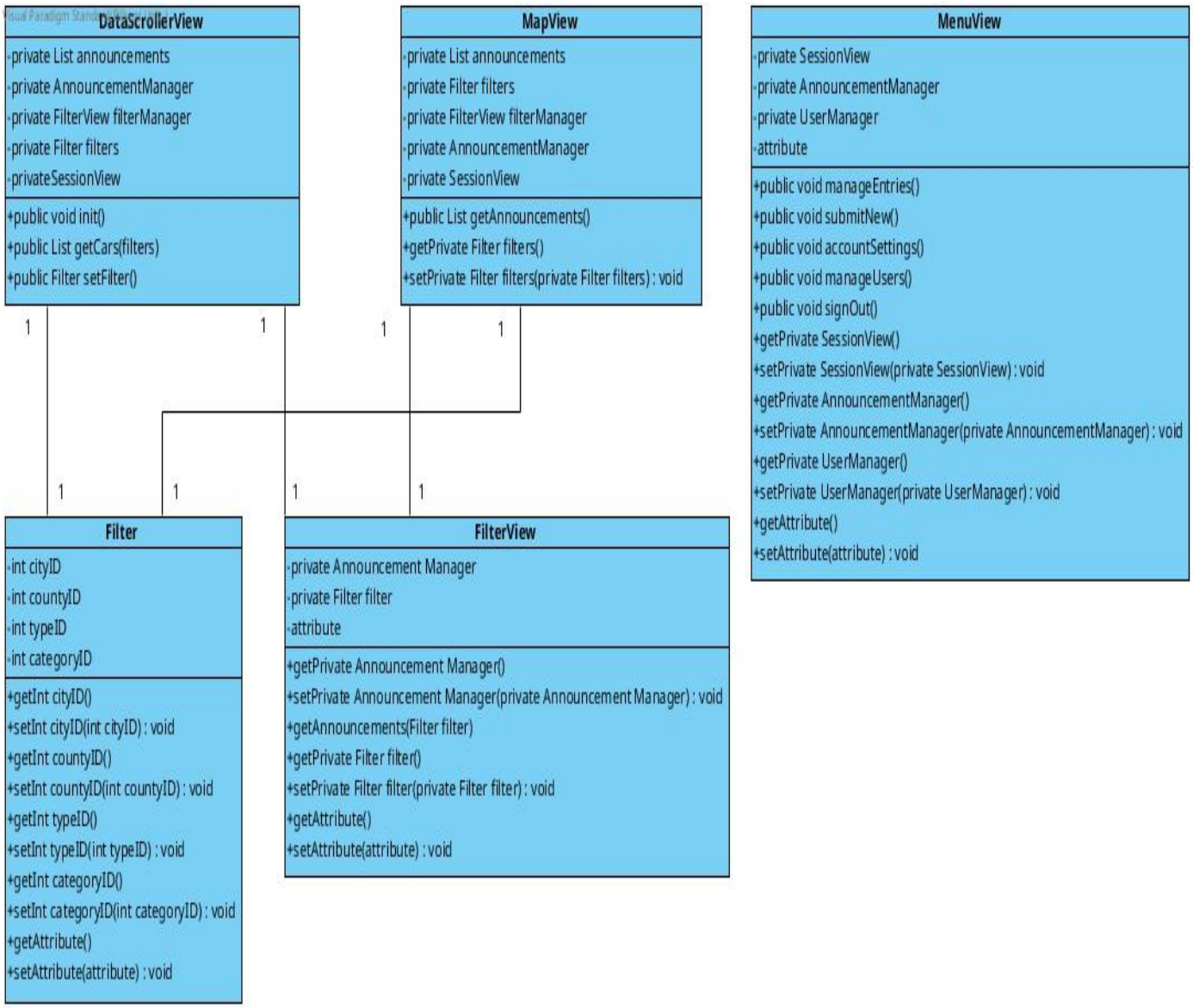
2.6 Boundary Conditions

*When the user signs in, the system will show some additional selection types, button or additional functions such as make announcement, logout, show, etc.
*If the user signs out, then the window changes to the guest screen.
*If the user wants to make announcement, the pop-ups will be shown on the screen.
*If the user is an admin, he will be able to use manage users session and he can manage all the users of the website.

3.0 Subsystem Services

3.1 View Classes

View Classes consist of 5 classes as shown below.



MapView Class:

Attributes:

private List announcements: List of Announcements.

private SessionView : SessonView instance.

private Filter filters: Filter instance.

private FilterView filterManager: Filter manager instance.

private AnnouncementManager: AnnouncementManager instance.

Methods:

public List getAnnouncements() : Returns list of announcements.

getPrivate Filter filters() : Returns Filter instance.

setPrivate Filter filters() : Create Filter Object.

MenuView Class

Attributes:

private SessionView: SessionView instance.

private AnnouncementManager : AnnouncementManager instance.

private UserManager: UserManager instance.

Methods:

public void manageEntries: Inıtialize manage Entries screen.

public void submitNew: Inıtialize submit new screen.

public void accountSettings: Inıtialize account settings screen.

public void manageUsers: Inıtialize manage users screen.

public void signOut : Inıtialize sign out screen.

getPrivate SessionView : returns SessionView instance.

setPrivate SessionView: sets SessionView instance.

getPrivate AnnouncementManager(): returns AnnouncementManager instance.

setPrivate AnnouncementManager(private AnnouncementManager) :

sets AnnouncementManager instance.

getPrivate UserManager: returns UserManager instance.

setPrivate UserManager(private UserManager) : sets UserManager instance.

DataScrollerView Class

Attributes:

private List announcements: List of announcements

private AnnouncementManager: AnnouncementManager instance.

private FilterView filterManager: FilterManager instance.

private Filter filters: Filter instance.

privateSessionView: SessionView instance.

Methods:

public void init : Initialization of attributes.

FilterView Class

Attributes:

private Announcement Manager: AnnouncementManager instance.

private Filter filter: Filter instance.

Methods:

setAnnouncement Manager(private Announcement Manager) : set AnnouncementManager instance.

getAnnouncements(Filter filter): returns list of filtered announcement.

setfilter(Filter filter) : Sets filter.

Filter Class

Attributes:

int cityID

int countyID

int typeID

int categoryID

Methods:

getcityID

setcityID(int cityID)

getcountyID

setCountyID(int countyID)

getInt typeID

seTypeID(int typeID)

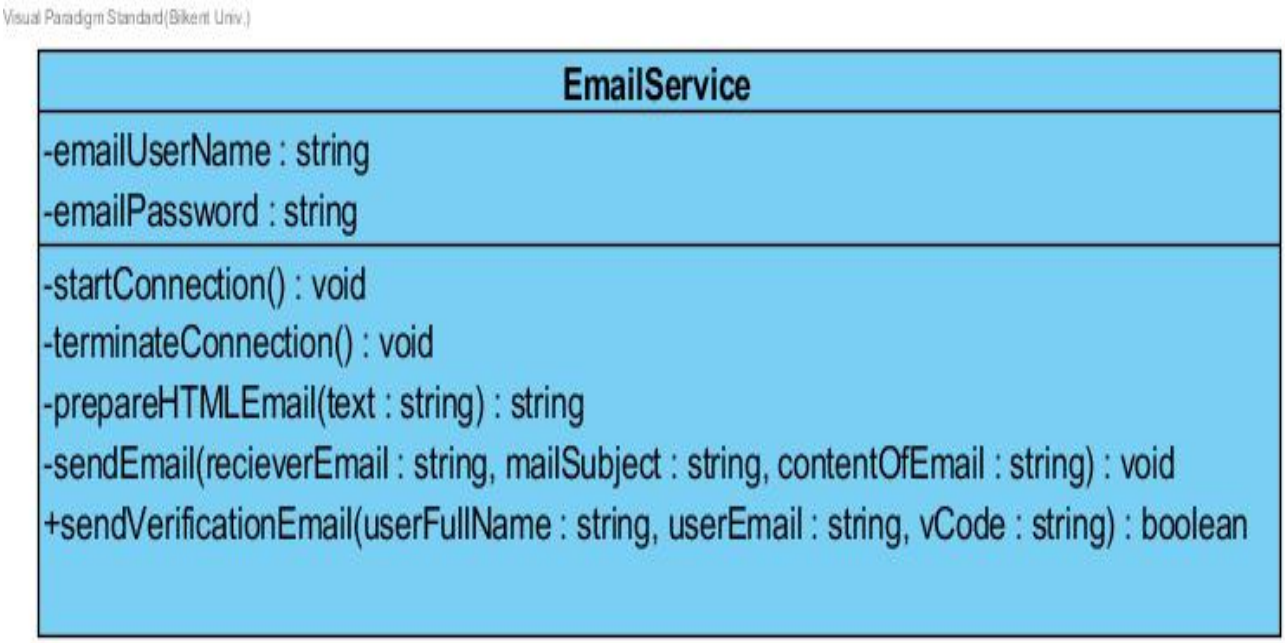
getInt categoryID

setInt categoryID

getAttribute

setAttribute(attribute)

3.2 E-mail Service



***Attributes*:**

emailUserName: This attribute stores the e-mail address of v.map to send emails to its users.

emailPassword: This attribute stores the e-mail account password of v.map to send emails to its users.

Methods:

private void startConnection(): The method starts the connection of SMTP by using the emailUserName and emailPassword.

private void terminateConnection(): The function closes the connection of SMTP that have been used.

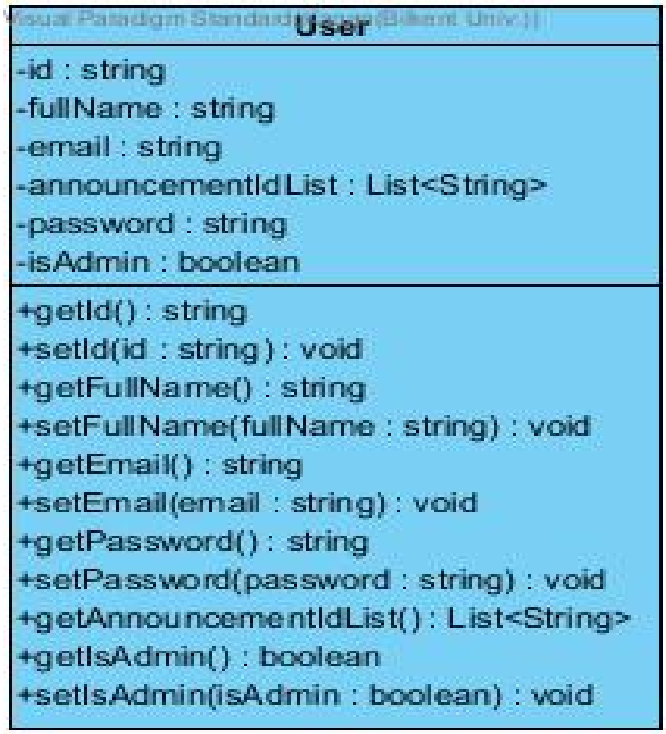
private string prepareHTMLEmail(String text): It creates an email template in HTML by inserting the given text into the HTML codes and returns string of the codes.

private void sendEmail (String recieverEmail, String mailSubject, String contentOfEmail): The function takes the e-mail address of the users to send, the subject of the e-mail, and the content of the e-mail as inputs. It creates an e-mail object by utilizing the email library. Then, the connection to SMTP is established by using **startConnection()** method and email is sent. The connection is closed by using **terminateConnection()** method.

public boolean sendVerificationEmail (String userFullName, String userEmail, String vCode): The method will be used by SessionManager to send a verification e-mail to the guests that want to sign up. The guest's full name, e-mail address, and verification code generated by SessionManager are taken. The method produces an e-mail which has the full name of the guest in the beginning (userFullName), a short information about e-mail, and the verification code (vCode) in the end of the mail. This generated e-mail text will be converted into HTML codes by using **prepareHTMLEmail** and sent by using **sendEmail**.

3.3 Model Classes

3.3.1. User Class:



Attributes:

private String id: id property of the user class.

private String fullName: fullName property of the user class.

private String email: email property of the User class.

private List <String> announcementIdList : An arraylist consists of the userAnnouncement id's.

private String password: password attribute of the user class.

private boolean isAdmin: Check if admin or user.

***Methods*:**

getters and setters

public String getID(): //getter of id

public void setId(String id): //setter of id

public String getFullName(): //getter of fullName

public void setFullName(String fullName): //setter of fullName

public String getEMail(): // getter of email

public void setEMail(String email): //setter of email

public String getPassword(): //getter of password

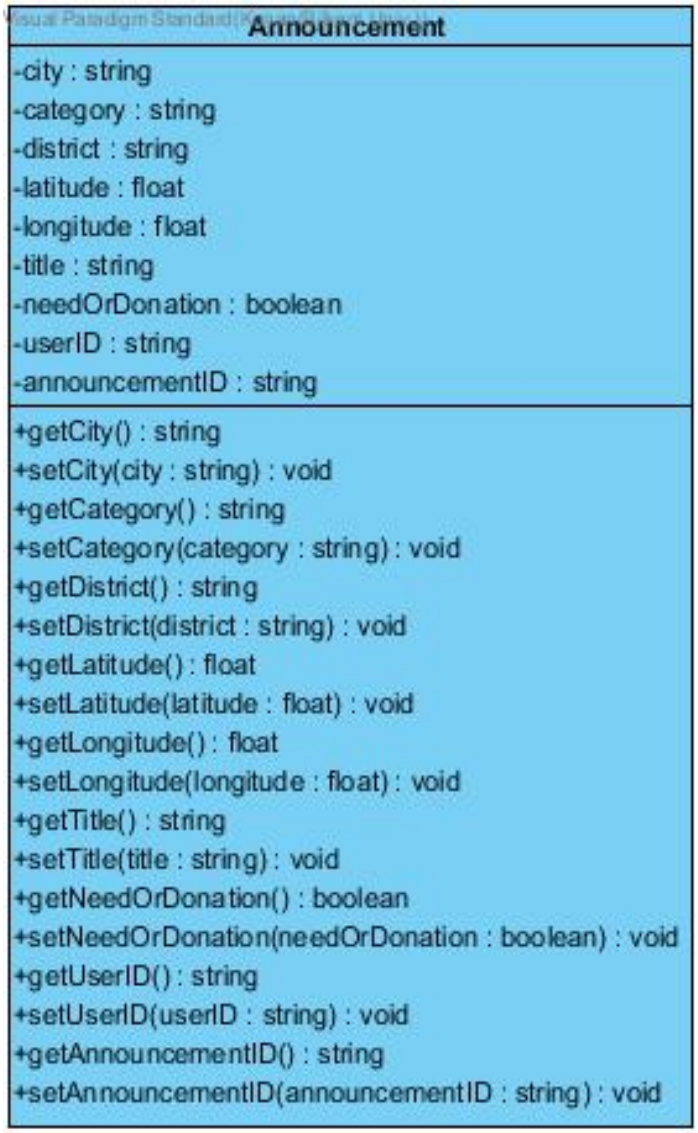
public void setPassword(String password): //setter of password

public List <String> getAnnouncementIdList(): //getter of announcementIdList

public boolean getIsAdmin(): //getter of isAdmin

setIsAdmin(boolean isAdmin): //setter of isAdmin

3.3.2. Announcement Class:



Attributes:

private String city: City in which the announcement is located

private String category: Category of the announcement

private String district: District of the announcement

private float latitude: Latitude of the announcement

private float longitude: Longitude of the announcement

private String title: Title of the announcement

private boolean needOrDonation: Type of the announcement. If true - need. If false - donation.

private String userID: id of the user that created the announcement

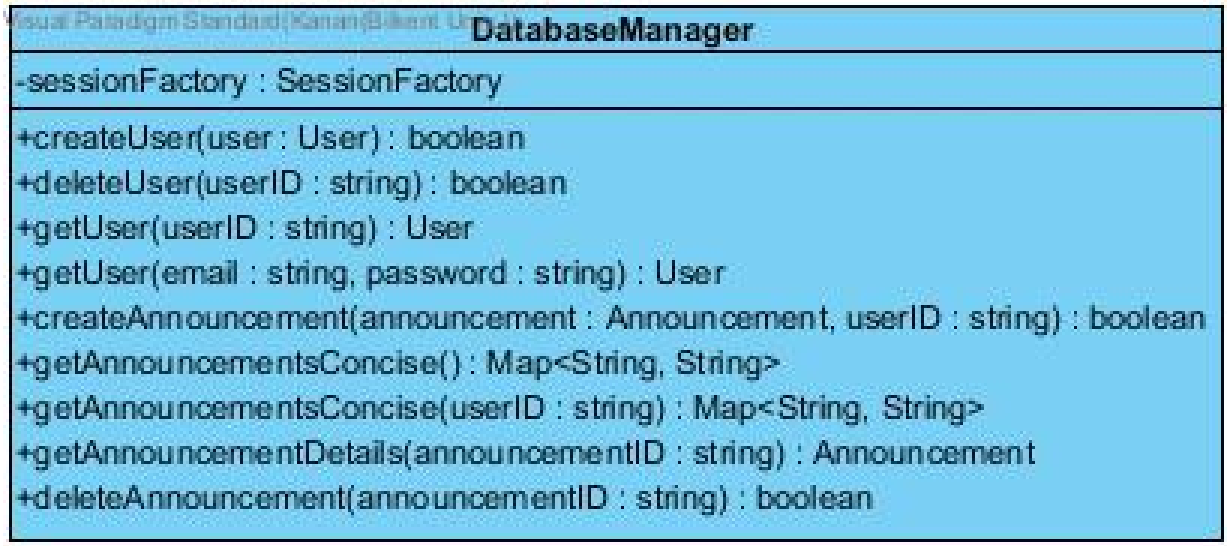
private String announcementID: id of the announcement

Methods:

getters and setters

3.4 Database Management

DatabaseManager Class:



- This is the main class of the Database that provides the connection between the database and the application. The requests come from the SessionManager and UserManager Classes.

Attributes:

private SessionFactory sessionFactory: his attribute is a *Hibernate* object that we must have in order to create sessions and transactions to store the object in our database.

Constructors:

public DatabaseManager(): initializes the DatabaseManager for the first time in the system. It will be called from the SessionManager constructor.

Methods:

public boolean createUser(User user): takes the user object as a parameter and tries to store it in the database. Returns true upon success. If the user already exists returns false

public boolean deleteUser(String userID): takes the id of the user that needs to be deleted. Checks if the user exists and deletes him/her. Returns true upon success. Returns false if the user does not exist.

public User getUser(String userID): takes the id of the user as a parameter and returns the object from the database corresponding to that id. Returns null if there is no user with that id.

public User getUser(String email, String password): takes the email and the password of the user and returns the object corresponding to these parameters from the database.

Returns null if the data does not match.

public boolean createAnnouncement(Announcement announcement, String userID): takes the announcement object and store it in the database and connects it to the user whose id was given. Returns true upon success. Returns false if the user with the id given does not exist.

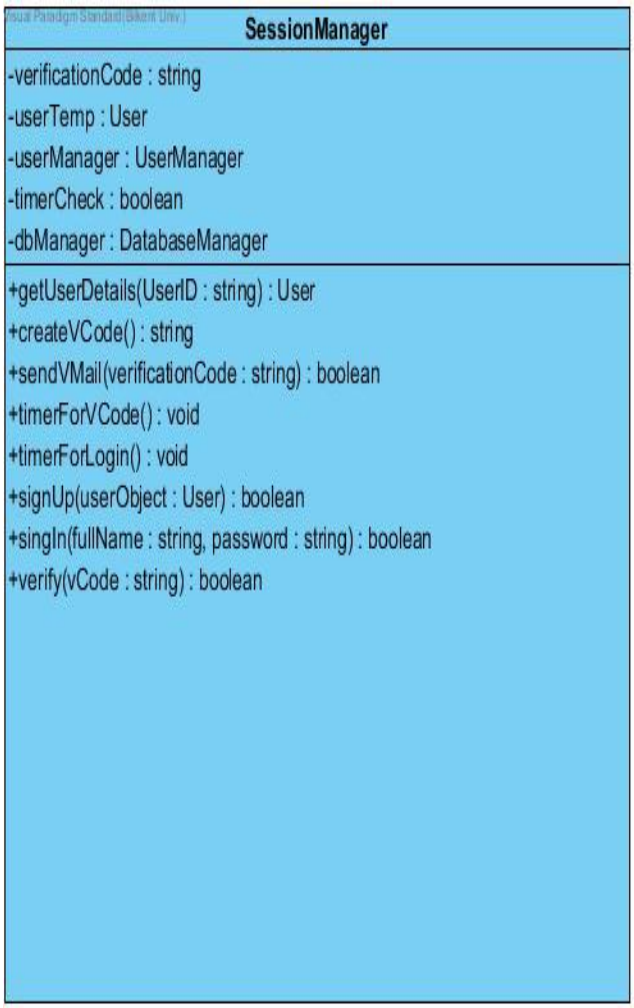
public Map<String, String> getAnnouncementsConcise(): returns a map containing only the locations, the title and the id.

public Map<String, String> getAnnouncementsConcise(String userID): returns a map containing only the location, the title and the id of the announcements related to the user whose id is given.

public Announcement getAnnouncementDetails(String announcementID): takes the id of the announcement as a parameter and returns the announcement object from the database corresponding to that id.

public boolean deleteAnnouncement(String announcementID): deletes the announcement from the database corresponding to the id given. Returns true upon success. Returns false if announcement does not exist.

3.5 Session Management



3.5.1 SessionManager Class:

***Attributes*:**

private String verificationCode: When createVCode() is called the return value is stored in the verificationCode variable.

private User userTemp: It's a User object that is created either in signUp or signIn methods.

private UserManager userManager: UserManager object is created. This is a referance to UserManager class.

private boolean timerCheck: It checks whether vCode timer is finished for the verification code.
*When the verification code writing time for the guest is over, timeCheck becomes true.

private DatabaseManager dbManager: This attribute is created in order to access the database.

Methods:

public User getUserDetails(String userID): The caller enters the userID, parameter and the function returns the User object corresponding to the entered userID.

public String createVCode(): Randomly generates a verification code and assigns it to the **verificationCode** attribute.

public boolean sendVmail(String verificationCode): The function gets the verification code produced in createVCode function, sends an e-mail to the user, if the operation is successful, then the function returns true, otherwise it returns false.

public void timerForVCode(): It's a method, which uses timer class of Java, if the verification code is sent to the guest, (he is not a user yet), the timer function

begins and when the time runs out, the vCode is deleted and the guest can’t use that VCode again.

public void timerForLogin(): It implements the timer class of java to compute a time limit for login process. If a user stays logged in and inactive within that time interval, then the **timerForLogin()** ends and automatically, the user is logged out.

public boolean signUp(User userTemp): createVCode(), timerForVCode(), sendVmail(), verify() methods are checked and implemented respectively. If the signUp process is successful, the method returns true, else case, it returns false.

public boolean signIn(String fullName, String password): This methods calls the getUser method from the DatabaseManager. If getUser() returns null, this method returns false. Otherwise, it assigns the user that is returned to the userTemp attribute and returns true.

public boolean verify(string vCode): If the guest enters the verification code sent to him before the timeCheck becomes true, the verify function is called and it returns true if the verification process is succesfuly done.
Verification process: the userTemp object is sent to database. If verify is successful, then the function returns true.

3.5.2 UserManager Class:

***Attributes*:**

private User user: This is a reference to the user object from the SessionManager class.

private DatabaseManager dbManager: This is a reference to dbManager object from SessionManager class.

***Methods*:**

public boolean deleteUser(String targetUserID): This method calls the deleteUser method of the DatabaseManager class and passes the targetUserID there. Returns the value obtained from the method called.

public boolean createAnnouncement(Announcement announcement, String userID): This method calls the createAnnouncement method of the dbManager object. Passes the announcement and the userID to the method and returns the value obtained from it.

public boolean deleteAnnouncement(String announcementID): This method calls the deleteAnnouncement method of the dbManager object and passes the announcementID to it. Returns the value obtained from it.

public void setUser(User user): //setter

public User getUser(): //Getter

public boolean deleteAccount(): This method calls the deleteUser method of the dbManager object and passes the id of the user from the attributes section.

public boolean changePassword(): It calls the createUser method in the dataBaseManager class that recreates the same user with a different password.

public boolean logOut(){ }: Change the state of the user from user - to guest. Set the user object of the userManager class to NULL.
* user object in the UserManager class and the tempUser object in the SessionManager class points to the same object.

public boolean editAnnouncement(Announcement announcement): Calls the createAnnouncement method of the dbManager object and passes the announcement taken from the parameters.

public boolean editUserInfo(User user): Calls the createUser method of the dbManager object. Passes the user and recreates it with unique values being unchanged.

public void setDBManager(): Sets a referance to the dataBaseManager of the sessionManager’s dbManager object.