



CS 319 - Object-Oriented Software Engineering
Design Report

v.map

Group 3-C

Yağız Gani

Kanan Asadov

Ömer Faruk Karakaya

Mert Osman Dönmezyürek

| | |
|--|----------|
| 1.0 Introduction | 2 |
| 1.1 Purpose Of The System | 2 |
| 1.2 Design Goals | 2 |
| End User Criteria | 2 |
| Maintenance Criteria | 3 |
| Performance Criteria | 3 |
| Trade-offs | 4 |
| 1.3 Definitions | 4 |
| 1.4 References | 4 |
| 1.5 Overview | 4 |
| 2.0 Proposed Software Architecture | 5 |
| 2.1 Overview | 5 |
| 2.2 Subsystem Decomposition and Layered Design | 5 |
| 2.2.1 Layers | 6 |
| 2.2.2 Subsystems | 6 |
| 2.3 Hardware/Software Mapping | 6 |
| 2.4 Persistent Data Management | 6 |
| 2.4.1 Identifying Persistent Objects | 6 |
| 2.4.2 Selecting a Storage Strategy | 7 |
| 2.5 Access Control And Security | 7 |
| 2.6 Boundary Conditions | 7 |
| 3.0 Subsystem Services | 8 |
| 3.1 View Classes | 8 |
| 3.2 E-mail Service | 11 |
| 3.3 Model Classes | 13 |
| 3.3.1. User Class: | 13 |
| 3.3.2. Announcement Class: | 15 |
| 3.4 Database Management | 16 |
| 3.5 Session Management | 18 |
| 3.5.1 SessionManager Class: | 18 |
| 3.5.2 UserManager Class: | 19 |

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 design 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 advantage, 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.

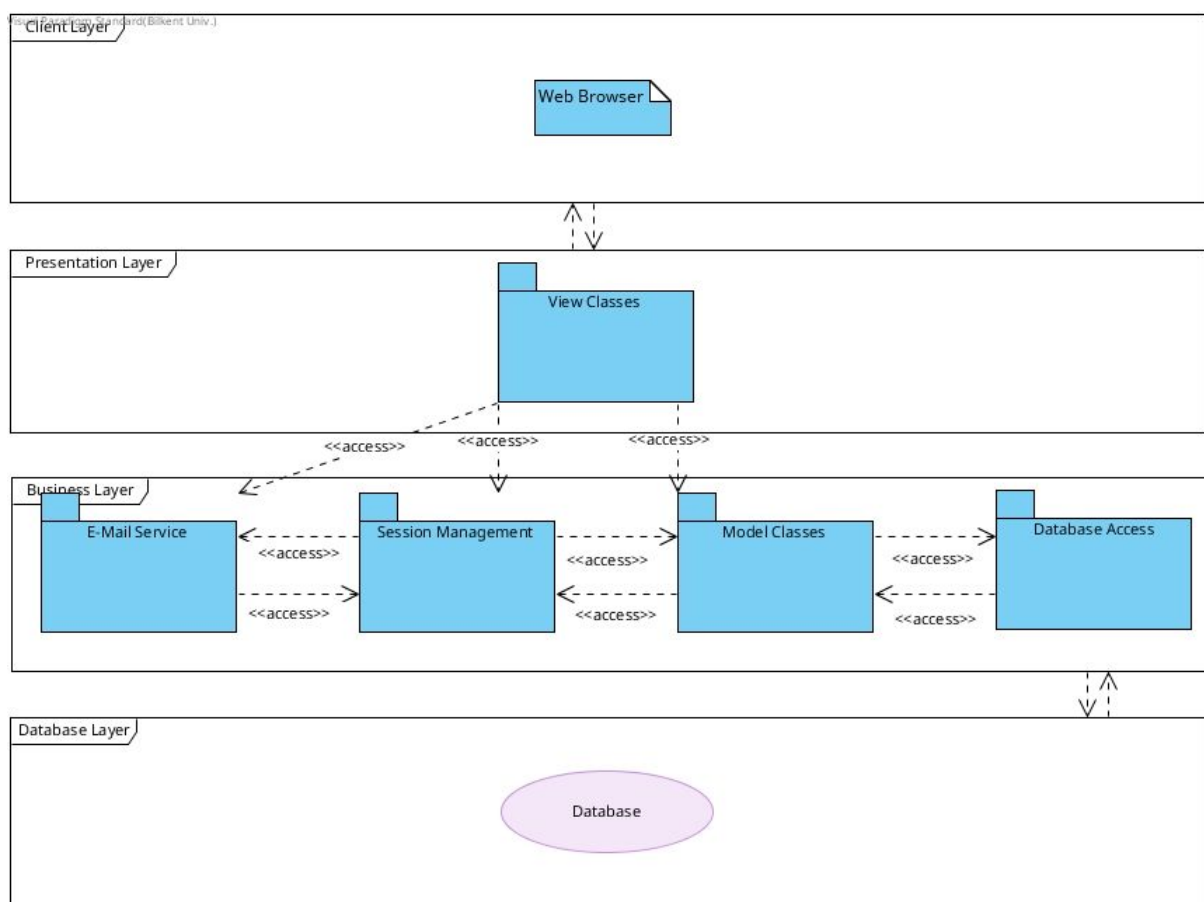
2.0 Proposed Software Architecture

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 Subsystem 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 subsystems 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 subsystem 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 subsystem consist of Java clases to send verification emails tu users. This subsystem have access to Session Management.
- Model : This subsystem consist of Java classes which represent entity objects. This subsystem have acces to Database Management subsystem.
- Database Access: This subsystem 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, javascript compatible web browser, application server like apache Tomcat and Database Server PostgreSQL is needed.

2.4 Persistent Data Management

2.4.1 Identifying 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 automatically creating model from objects. Hibernate ORM is chosen as a ORM software. The reason why Hibernate is chosen is Hibernate is widely used in business applications, it is an opensource software and 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.

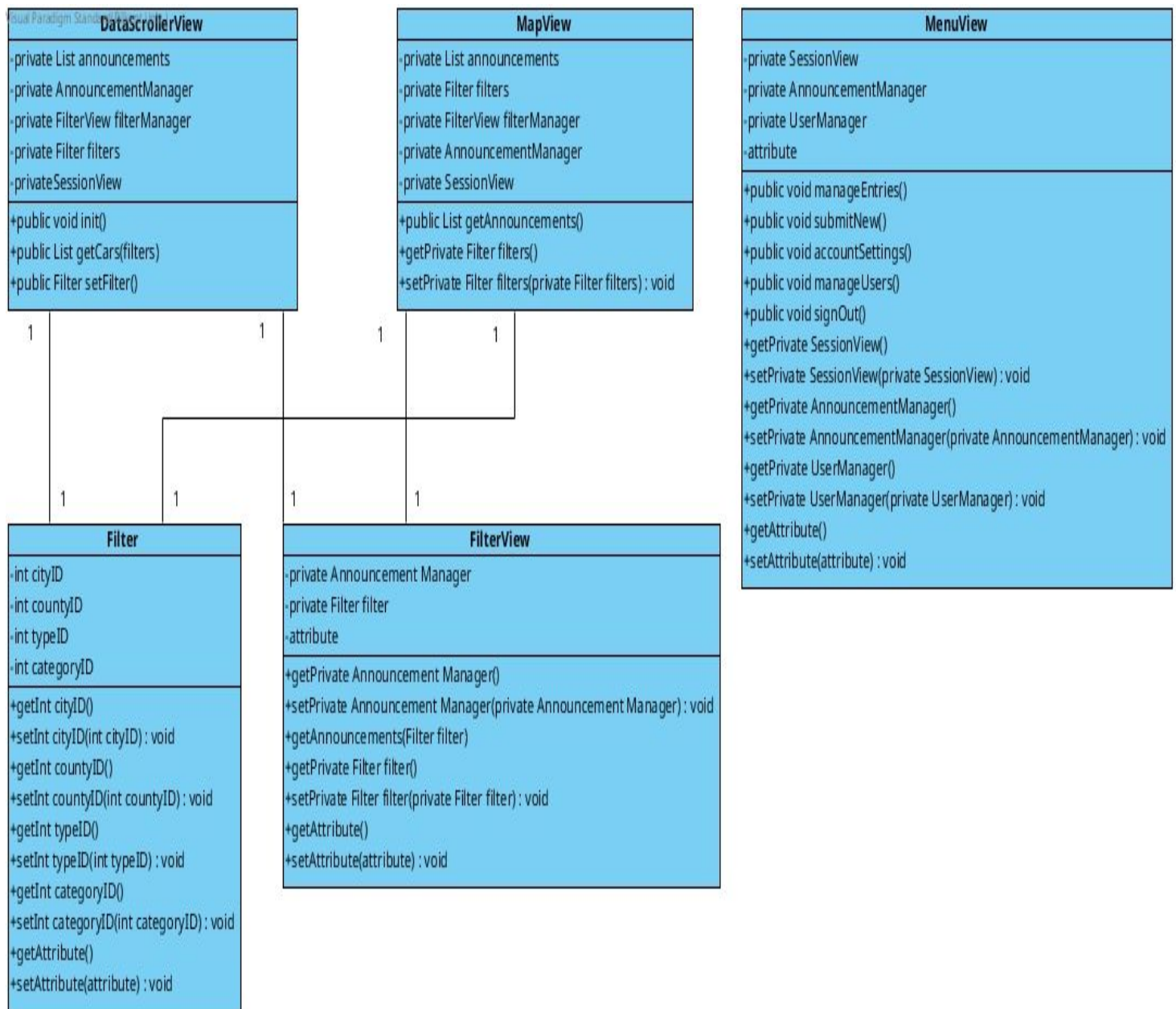
*If the user wants to make announcement, the pop-ups will be shown on the screen.

*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. Since we use PrimeFaces which is a UI framework for JSF, framework let us create User interface without writing anymore code than getters and setters for UI elements.



MapView Class:

Attributes:

private List announcements: Arraylist of Announcements.

private SessionView : SessionView instance to check sessions.

private Filter filters: Filter instance to show Announcements accordingly.

private FilterView filterManager: Filter manager instance reach filter.

private AnnouncementManager: AnnouncementManager instance to reach announcements.

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 to check sessions.

private AnnouncementManager : AnnouncementManager instance to reach announcements.

private UserManager: UserManager instance to reach Users..

Methods:

public void manageEntries: Initialize manage Entries screen.

public void submitNew: Initialize submit new screen.

public void accountSettings: Initialize account settings screen.

public void manageUsers: Initialize manage users screen.

public void signOut : Initialize 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 to show Announcements accordingly.

privateSessionView: SessionView instance to check sessions.

Methods:

public void init : Initialization of attributes. This method called by framework even if object not created.

FilterView Class

Attributes:

private Announcement Manager: AnnouncementManager instance.

private Filter filter: Filter instance to show Announcements accordingly..

Methods:

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

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

setfilter(Filter filter) : Sets filter, this method is used when user choose filter in UI.

Filter Class

Attributes:

int cityID : ID of filtered city, -1 if nothing filtered.

int countyID : ID of filtered county, -1 if nothing filtered.

int typeId: ID of filtered type, -1 if nothing filtered.

int categoryID: ID of filtered category, -1 if nothing filtered.

Methods:

getcityID: return filtered cityID.

setcityID(int cityID) : sets cityID to ID of city which is chosen by user.

getcountyID: return filtered countyID.

setCountyID(int countyID) : sets countyID to ID of county which is chosen by user. Only active is cityID different than -1.

getInt typeId: returns typeId:

setTypeID(int typeId): sets typeId to ID of type which is chosen by user.

getInt categoryID: returns categoryID.

setInt categoryID: sets categoryID to ID of category which is chosen by user.

3.2 E-mail Service

| EmailService |
|---|
| -emailAddress : String -emailPassword : String -props : Properties -session : Session |
| +EmailService() -startConnection() : void -prepareVerificationEmailInHTML(userFullName : String, verificationCode : String) : String -sendEmail(recieverEmail : String, mailSubject : String, contentOfEmail : String) : boolean -sendVerificationEmail(userFullName : String, userEmail : String, verificationCode : String) : boolean |

Attributes:

emailAddress: This attribute will store the e-mail address of v.map to send emails to its users.

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

Methods:

public EmailService(): It is the constructor of the class. It sets the “emailAddress” and “emailPassword” of v.map app. Also, it creates a new Properties for “props” and adjust the smtp server before starting the connection.

private void startConnection(): The method starts the connection of SMTP by initiating the “session”.

private String prepareVerificationEmailInHTML(String userFullName, String verificationCode): It creates an email template in HTML for verification email by inserting the “userFullName” and the “verificationCode” into the HTML codes and returns string of the codes.

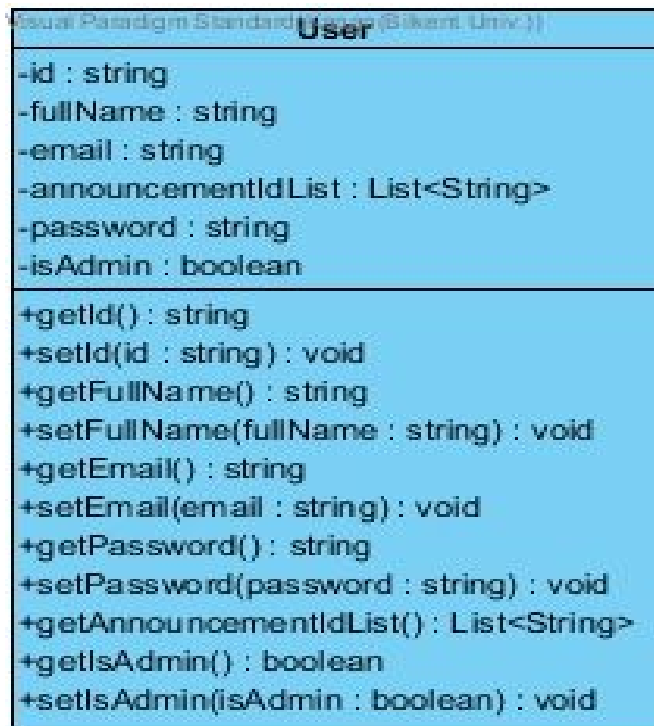
private boolean sendEmail(String recieverEmail, String mailSubject, String contentOfEmail): The function takes the e-mail address of the users to send, the subject of the e-mail, and the content of the e-mail as inputs. It creates an e-mail object (new MimeMessage(session)) by utilizing the email library. Then, the method tries to transport the e-mail to the server. If the process is successful, the method returns true, otherwise false.

public boolean sendVerificationEmail(String userFullName, String userEmail, String verificationCode): the connection to SMTP is established by using **startConnection()** method to send email. 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 (userFullName) , e-mail address (userEmail), and verification code (verificationCode) 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” in the

end of the mail. This generated e-mail text will be converted into HTML codes by using **prepareVerificationEmailInHTML** 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:

| Usual Paradigm Standard (Khan, Ghani, Usmani) |
|---|
| Announcement |
| Attributes: -city : string -category : string -district : string -latitude : float -longitude : float -title : string -needOrDonation : boolean -userID : string -announcementID : string |
| Operations: +getCity() : string +setCity(city : string) : void +getCategory() : string +setCategory(category : string) : void +getDistrict() : string +setDistrict(district : string) : void +getLatitude() : float +setLatitude(latitude : float) : void +getLongitude() : float +setLongitude(longitude : float) : void +getTitle() : string +setTitle(title : string) : void +getNeedOrDonation() : boolean +setNeedOrDonation(needOrDonation : boolean) : void +getUserID() : string +setUserID(userID : string) : void +getAnnouncementID() : string +setAnnouncementID(announcementID : string) : void |

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:

| UML Paradigm Standard (Kanan/Bikem, UML 2.0) |
|---|
| DatabaseManager |
| -sessionFactory : sessionFactory |
| +createUser(user : User) : boolean +deleteUser(userID : string) : boolean +getUser(userID : string) : User +getUser(email : string, password : string) : User +createAnnouncement(announcement : Announcement, userID : string) : boolean +getAnnouncementsConcise() : Map<String, String> +getAnnouncementsConcise(userID : string) : Map<String, String> +getAnnouncementDetails(announcementID : string) : Announcement +deleteAnnouncement(announcementID : string) : boolean |

- 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.

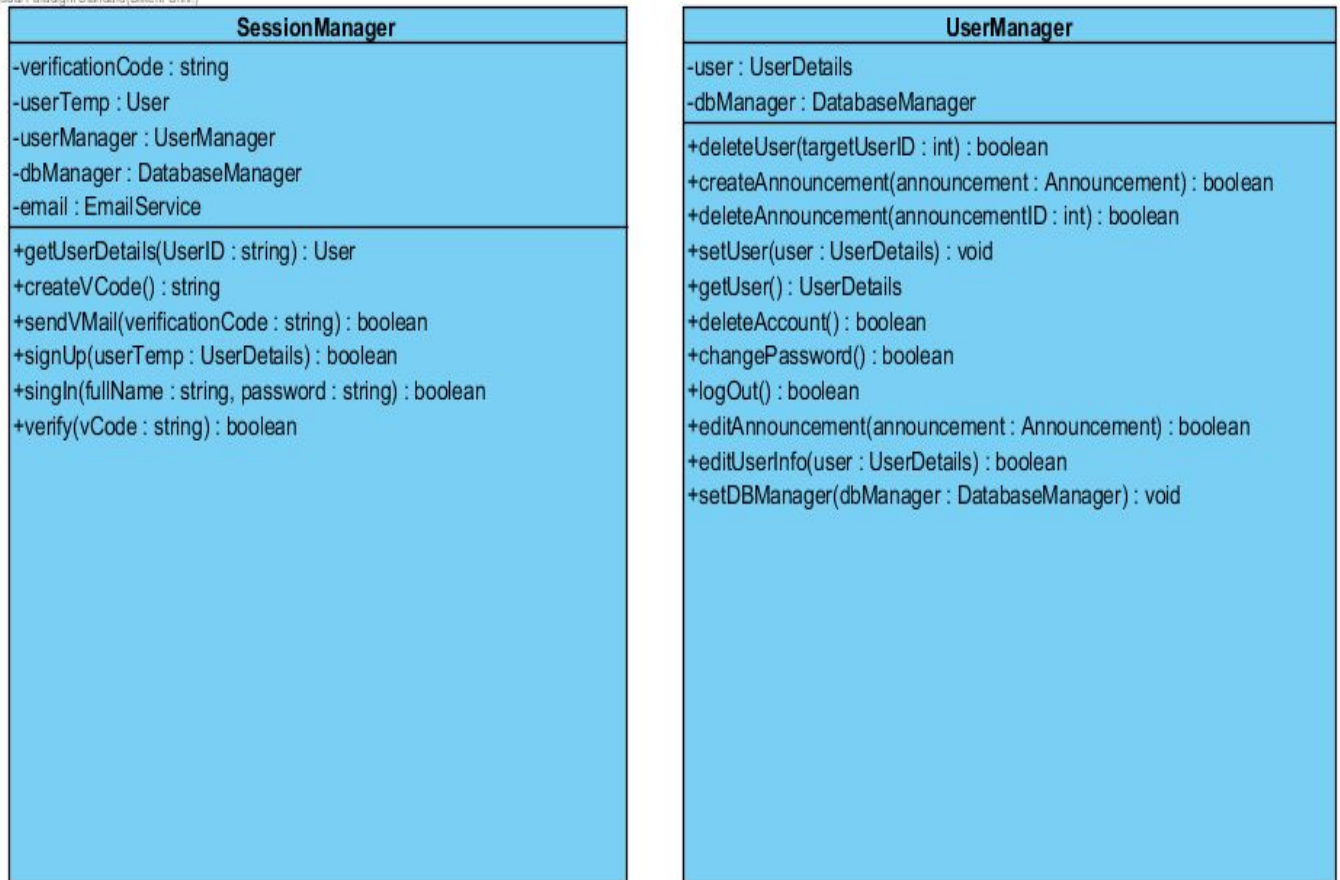
public Map<String, String> getAnnouncementsConcise(String userID): returns a map containing only the location, the title and the id of the announcements related to the user whose id is given.

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

Visual Paradigm Standard (Bilkent Univ.)



3.5.1 SessionManager Class:

Attributes:

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

private UserDetails 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 reference to `UserManager` class.

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

private EmailService email: This attribute is created in order to access the EmailService class.

Methods:

public UserDetails 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 boolean signUp(UserDetails userTemp): createVCode(), 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): 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 UserDetails 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(int 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): 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(int 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(UserDetails user): //setter

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

public void setDBManager(DatabaseManager dbManager): Sets a reference to the dataBaseManager of the sessionManager's dbManager object.