
CS 319 - Object-Oriented Software Engineering

Analysis Report

# v.map

Group 3-C

Yağız Gani

Kanan Asadov

Ömer Faruk Karakaya

Mert Osman Dönmezyürek

# Table of Contents

# 1. Introduction

v.map is a web application that will serve for Community Volunteer Foundation. “Community Volunteers Foundation (TOG) is a change and transformation project designed to translate the youth’s energy into social benefit. Established in December 2002, TOG aims at achieving social peace, solidarity and change under the pioneership of the youth and the guidance of adults” [1].

This foundation has more or less 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.

Another undeniable feature is that v.map is a web application that can 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.

# 2. Overview

v.map is a project made for the "Community Volunteers Foundation/Toplum Gönüllüleri Vakfı (TOG)". TOG is an organization with around 65k volunteers and the majority of them are students working from their cities/towns/villages.
Our web application aims to help them answer/create "needs" and organize them in an efficient manner the way "İhtiyaç Haritası" does it.
Volunteers will be able to create their needs in a specific way: We need "x" amount of notebooks in school "y". and it will be displayed on the map of Turkey. Moreover, if volunteers have things for a giveaway they will also be able to enter it into this system.
Users will also be able to filter the announcements depending on city, county, donate/need.

# 3. Requirement Specification

## 3.1. Functional Requirements

### 3.1.1. Create Need

v.map is a web application which is made for making announcements all over the Turkey and it is made to correspond the foundation members' needs. "Submit New" button, which is located on the top of the webpage, allows the user to create an announcement according to the need and the user will be able to give specific information about the need, by typing in the text field in announcement-form. Also, he/she will be able to give the specific location where the delivery can be made and the location is also pointed on the v.map, with a location icon. Only the users are able to see more detailed information about the announcement.

*Guests are not allowed to see the contact information about the announcement and about the announcer.

*Only the site members/users have permission to make announcements.

*Guests will only be able to see a basic information about the announcements.

### 3.1.2. Delete

This function allows the user to delete the announcement he made. Each user is able to delete only their announcements.

*Only the admin has the permission to delete everyone's announcements.

### 3.1.3. Show

Show function maintains detailed information about the opened announcement. When the user clicks on the show button he will be able to reach more detailed information about the announcement. When the user pulls the mouse button towards the location icon shown in the v.map, the show button occurs on the right side of the information box.

*Also the guests are able to use the show function, which provides a general information about the announcement. However, the guests will not have permission to see the contact information of the announcer. Only the users are able to see the personal contact information of the owner of the announcement.

### 3.1.4. Manage users

Only the admin will have this function when he opens the website. With this function, the admin will be able to delete the announcements that are unnecessary or irrelevant or the admin can delete kick out user from the website.
*Only the admin will have the permission to kick out a user if he faces with unwanted or unexpected situations.

### 3.1.5. Edit Announcement

Each user is able to edit his announcements himself, if the announcements need correction or if they need any updates.

### 3.1.6. Announcement Filter Bar

The filter has the fundamental selections like donations or needs.
Donations: The user will be able to see only the donation announcements on the v.map.
Needs: The user will be able to see only the need announcements on v.map.
Also, filter service will consist of different types of topics according to the need fields. These fields will be:

- Education
- Health
- Technology
- Clothing
- Sports
- Handicapped
- Other

*Each subhead will involve their icons which will be relevant to the topic, on the v.map.
*For example, if the user wants to see only the education announcements, he will select education and click on the filter button and he will see all the

announcements about education on the v.map. Each education announcement will be shown on the map with a book icon.

## 3.2. Non-Functional Requirements

### 3.2.1. Performance

Since v.map is a web application, it should work on a browser and use the Internet connection. Therefore, the quality of the Internet and stability of the browser might affect the performance of the application. While coding the app, we will consider the worst situations such as poor Internet connection or an unstable browser, and take precautions to make our app work as fast as it can on major systems in the market while designing v.map.

### 3.2.2. System

Although theoretically v.map can support on every operating system that has a web browser featured by current technologies such as CSS or html5, it will be tested on major operating systems and browsers. The system requirements will be following:

Windows, Mac, Linux / Ubuntu, Google Chrome OS (Chromebook)
Google Chrome v34 or later, Mozilla Firefox v34 or later, Internet Explorer v9 or later, Apple Safari v6 or later

An Intel Pentium 4 processor or later
1 Mbps or better Internet connection

### 3.2.3. User Friendly Interface

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

### 3.2.4. Backup

Our application handles a huge database that contains users' and server's information or needs and requests shared by the members of the foundation. Thus, the database will need to be given importance while being used by the administrators of the foundation. It needs to be backed up in certain periods not to lose the valuable information.

### 3.2.5. 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.

# 4. System Model

## 4.1. Use Case Model

This section provides information about the main use case model of the v.map. Detailed explanations are included below.

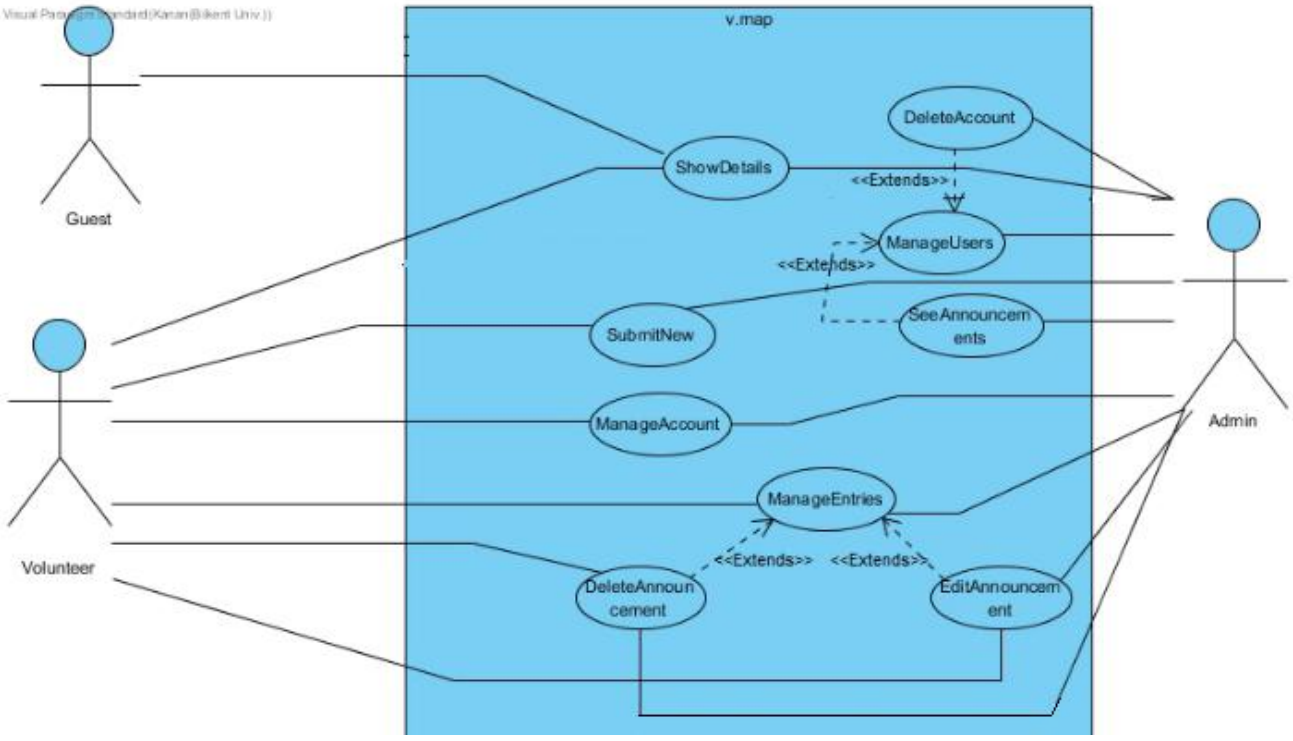



Figure 1

### 4.1.1. Use Case Name: Submit New

**Primary Actor:** Volunteer, Admin

**Stakeholders and Interests:**
- User wants to create a need or a donation
- System creates the need or a donation

**Pre-conditions:** User should be logged in and be on the homepage
**Post-condition:** If the user put a tick next to donation then the system sets the type of the submission to donation, otherwise it is a need

**Entry Condition:** User presses the submit new button
**Exit Condition:** User presses the submit button after filling the information

**Success Scenario Event Flow:**

1. System saves the submission in the database and shows it on the map
2. "Your submission was successful" pop up appears

**Alternative Flows:**
A. If the user enters wrong information:
    A.1. System prompts the user to check information

B. If the user decides not to make a submission:
    B.1. the user clicks on the close button
    B.2. Pop up is closed

### 4.1.2. Use Case Name: Show Details

**Primary Actor:** Guest, Volunteer, Admin

**Stakeholders and Interests:**
- The user wants to see the details of the announcement
- System shows a pop up with the details

**Pre-conditions:** User should be on the homepage
**Post-condition:** If user is a Guest then do not show the contact information

**Entry Condition:** User presses the show button
**Exit Condition:** User presses the close button

**Success Scenario Event Flow:**

1. System shows a pop up with submission information

**Alternative Flows:**
A. If the need was deleted by the time user presses on it:
A.1. Prompt user that the need was deleted

### 4.1.3. Use Case Name: Manage Account

**Primary Actor:** Volunteer, Admin

**Stakeholders and Interests:**
- The user wants to manage his information
- System shows a pop up with the settings

**Pre-conditions:** User should be logged in and be on the homepage
**Post-condition: -**

**Entry Condition:** User presses the Manage Account button
**Exit Condition:** User presses save and close button

**Success Scenario Event Flow:**

1. System shows a pop up with user information

Alternative Flows:
A. If user presses the close button:
A.1. System closes the pop-up window

### 4.1.4. Use Case Name: Manage Entries

**Primary Actor:** Volunteer, Admin

**Stakeholders and Interests:**
- User wants to manage his announcements
- System shows a pop up with the list of announcements

**Pre-conditions:** User should be logged in and be on the homepage
**Post-condition:** -

**Entry Condition:** User presses the Manage Entries button
**Exit Condition:** User presses “save and close” button

**Success Scenario Event Flow:**

1. System shows a pop up with user announcements

**Alternative Flows:**
A. If user presses the close button:
    A.1. System closes the pop-up window

### 4.1.5. Use Case Name: Edit Announcement

**Primary Actor:** Volunteer, Admin

**Stakeholders and Interests:**
- User wants to edit his/her announcement
- System edits the announcement and updates the map

**Pre-conditions:** User should be logged in and be in Manage Entries pop-up window and have at least one announcement
**Post-condition:** -

**Entry Condition:** User presses the edit button

**Exit Condition:** Operation is successfull

**Success Scenario Event Flow:**

1. System edits the announcement

**Alternative Flows:**
A. If user decides not to edit the announcement and presses the close button:
    A.1. System closes the pop-up window

### 4.1.6. Use Case Name: Delete Announcement

**Primary Actor:** Volunteer, Admin

**Stakeholders and Interests:**
- User wants to delete the announcement
- System deletes the announcement and updates the map

**Pre-conditions:** User should be logged
**Post-condition:** -

**Entry Condition:** User presses the delete button
**Exit Condition:** The announcement was deleted succesfully

**Success Scenario Event Flow:**

1. System deletes the announcement
2. System updates the map

**Alternative Flows:**
A. If user decides not to delete the announcement and presses the close button:
    A.1. System closes the pop-up window
*Volunteers can only delete their own announcements, while Admins can delete any announcement

### 4.1.7. Use Case Name: Manage Users

**Primary Actor**: Admin

**Stakeholders and Interests:**
- User wants to manage the user list
- System shows a pop up with user list

**Pre-conditions:** User should be logged in and be an admin
**Post-condition:** -

**Entry Condition:** User presses the manage users button
**Exit Condition:** User presses the close button

**Success Scenario Event Flow:**

1. System shows a pop-up with users

**Alternative Flows:**
A. If user presses the close button:
    A.1. System closes the pop-up window

### 4.1.8. Use Case Name: Ban User

**Primary Actor:** Admin

**Stakeholders and Interests:**
- User wants to ban the user
- System deletes the user and puts his email into a blacklist

**Pre-conditions:** User should be logged in and be an admin and be in the Manage Users pop-up
**Post-condition:** -

**Entry Condition:** User presses the ban button
**Exit Condition:** User confirms the action

**Success Scenario Event Flow:**

1. System deletes the user from the database
2. System puts the email to a blacklist
3. System updates the map

**Alternative Flows:**
A. If user decides not to edit the continue the action and presses the close button:
A.1. System closes the pop-up window

### 4.1.9. Use Case Name: See Announcements

**Primary Actor:** Admin

**Stakeholders and Interests:**
- Admin wants to see announcements of the user
- System filters the announcements on the map and shows only the ones created by that user

**Pre-conditions:** User should be logged in and be an admin and be in the Manage Users pop-up
**Post-condition:** -

**Entry Condition:** User presses the show posts button
**Exit Condition:** User presses the close button

**Success Scenario Event Flow:**

1. System closes the Manage Users pop up
2. System filters the announcements and shows only the ones that were created by a specific user

3. System adds a close button that can be used to cancel filtering and show all announcements without filtering

**Alternative Flows:**

A. If user decides not to continue the action and presses the close button:

A.1. System closes the pop-up window

## 4.2. Dynamic Models

### 4.2.1. Activity Diagrams

This activity diagram shows what happens when the user presses the announcement marker on the map. First, the system checks if the announcement belongs to the user. If it does belong to him/her it shows the box with an edit and a show button. However, if it does not belong to the user, it shows a pop up with a show button only.

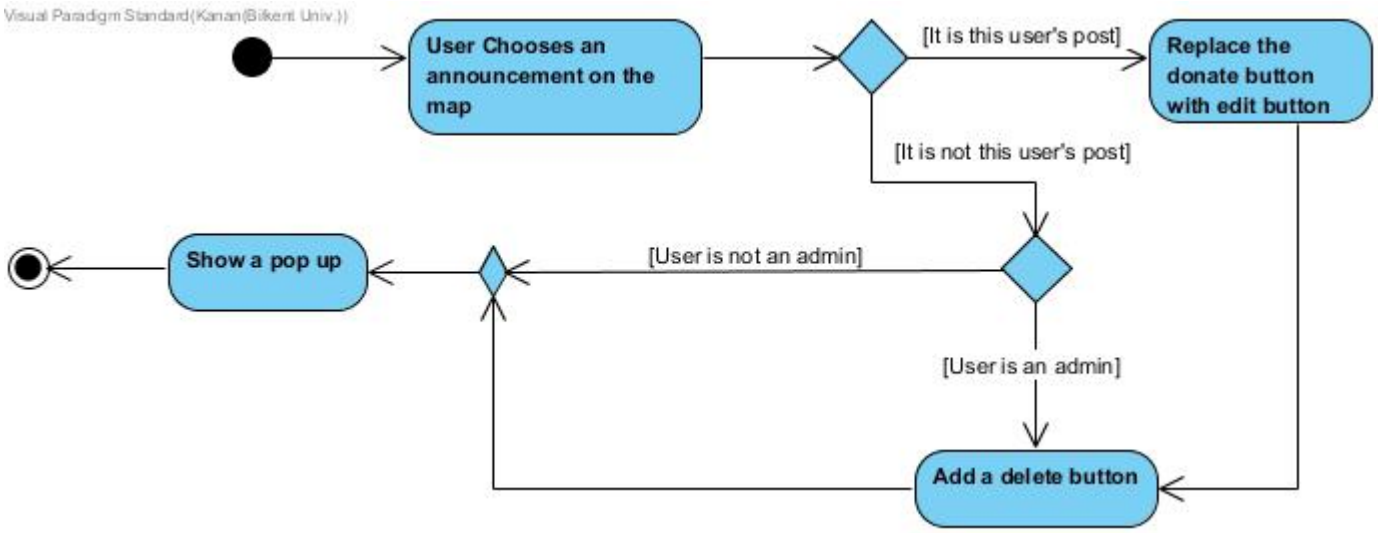
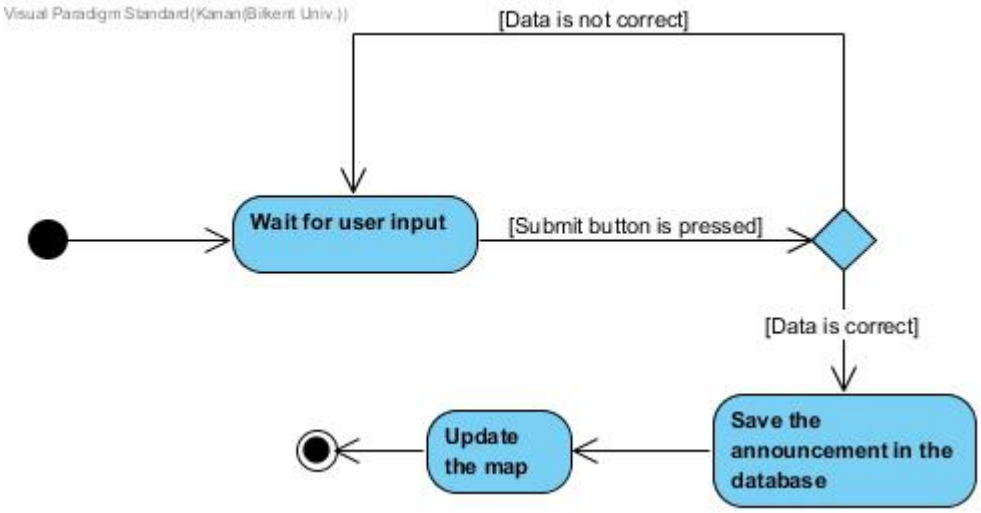



Figure 2

This diagram shows what happens when the user submits a new need/donation. It waits for the user to fill the information needed and press the submit button. When the button is pressed the system checks if all the information is correct. If it is, the system updates the database and the map. In case the information is not correct or full, the system prompts the user to




revise the information and submit the need/donation again.

Figure 3

## 4.2.2. Scenarios an Sequence Diagrams

### 4.2.2.1. Scenario 1 and Related Sequence Diagram

A guest enters web page and all announcements are displayed. When guest filter announcements, filtered announcements are displayed.

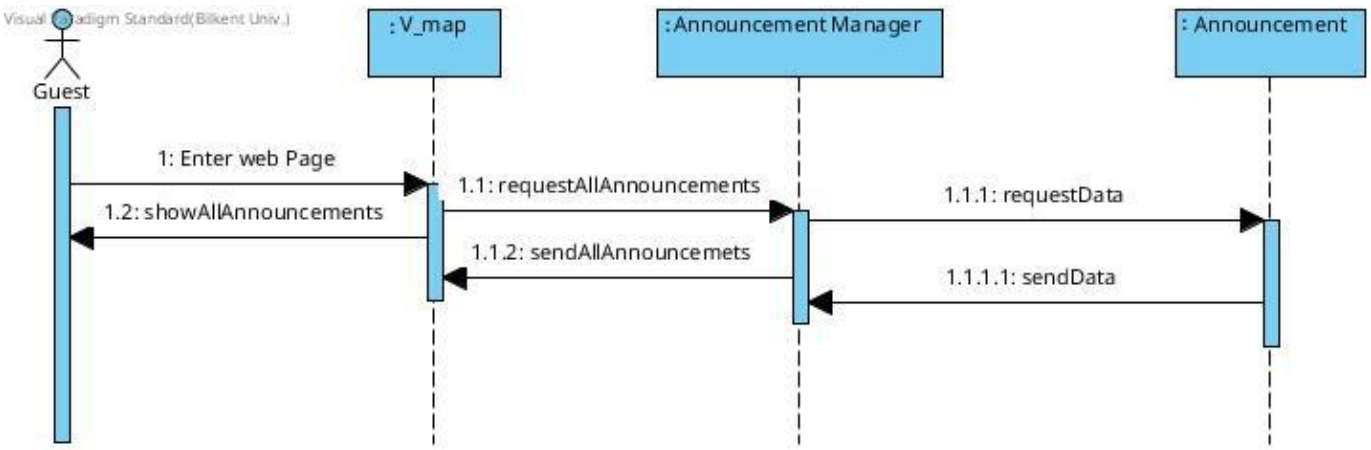


Figure 4

#### 4.2.2.2. Scenario 2 and Related Sequence Diagram

Guest Sign In and Log In. When guest Log In main page refreshed.

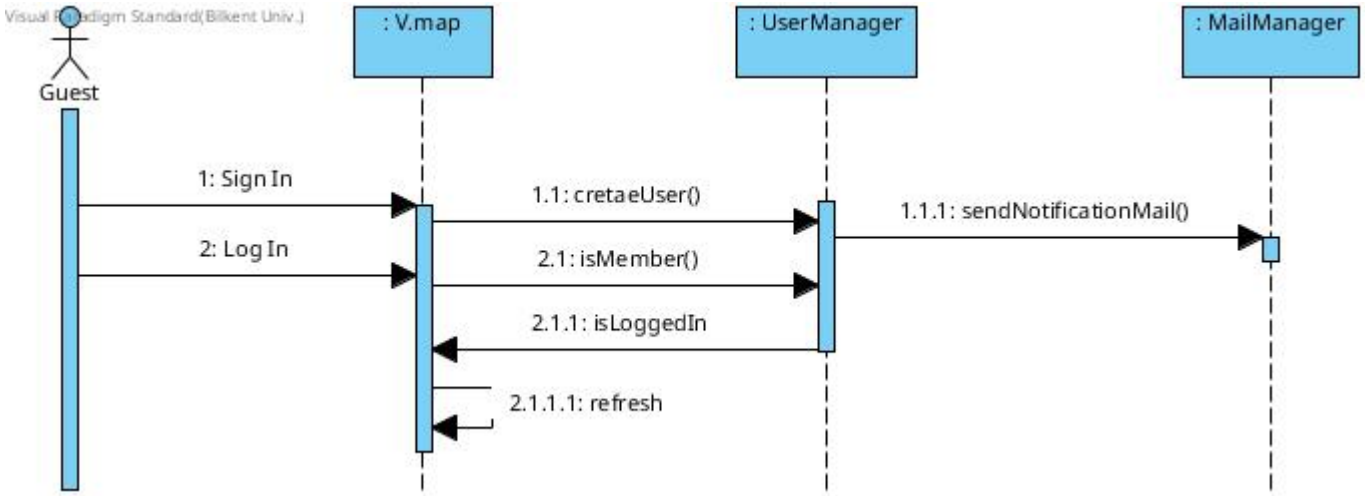



Figure 5

#### 4.2.2.3. Scenario 3 and Related Sequence Diagram

User add new announcement and then user deletes his/her previous announcement.

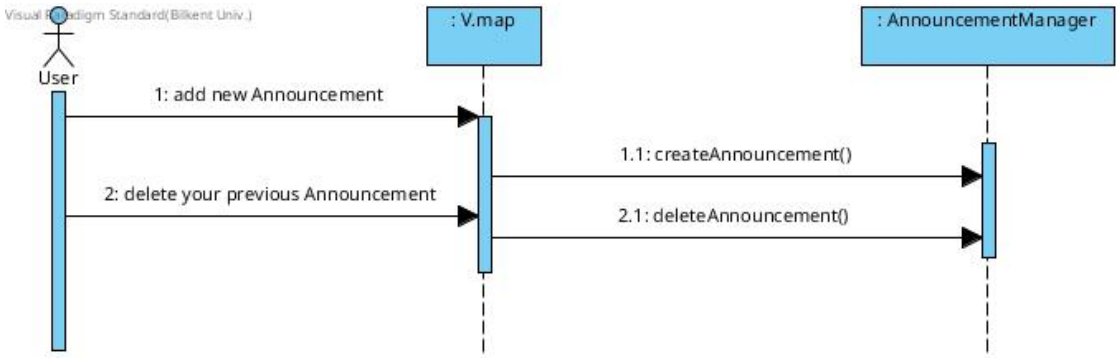



Figure 6

#### 4.2.2.4. Scenario 4 and Related Sequence Diagram

Admin deletes a User and deletes a Announcement.

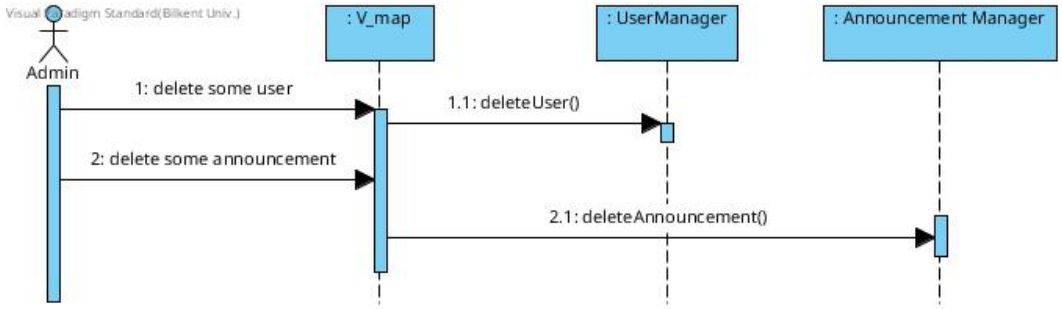



Figure 7

## 4.3. Object And Class Model

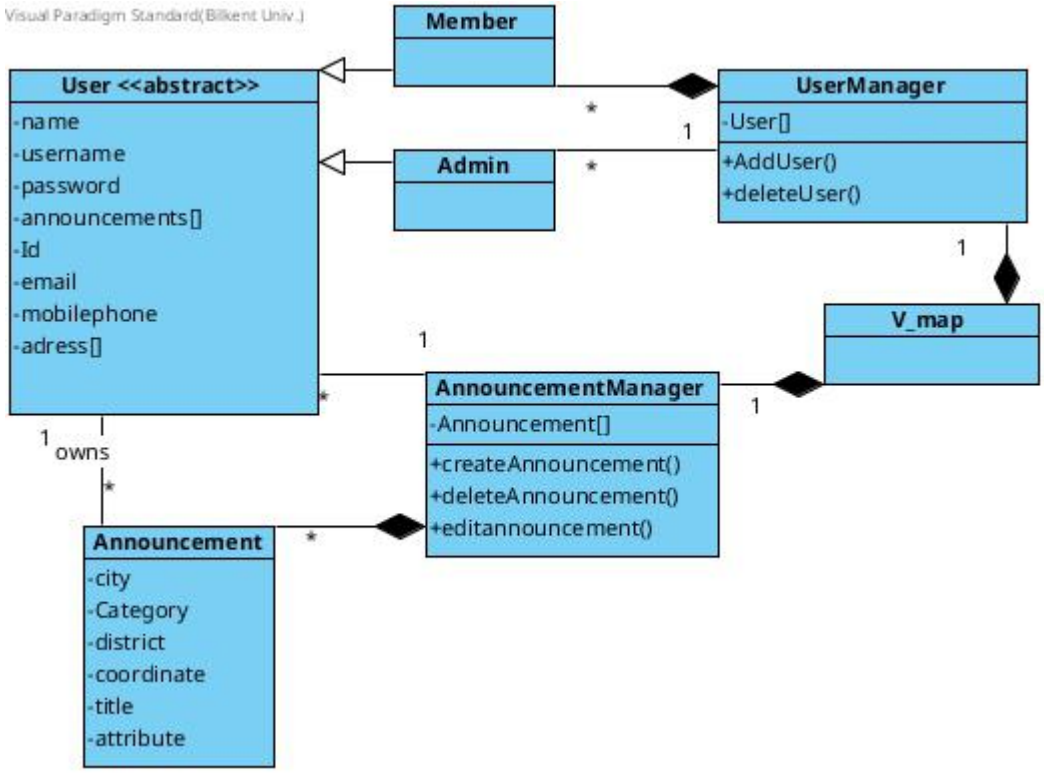



Figure 8

The object model of v.map is illustrated above and classes explained briefly below.

- User class is an abstract class specifies common properties of admin type and member type of user.

- Member class represent member type of user who can create announcements and manage their announcements.
- Admin class represent admin type of user who has the ability to edit/delete all announcement and delete members additionally to member abilities.
- UserManager class is to create, delete users and keep them in list.
- Announcement class specifies properties of announcements.
- AnnouncementManager class is to create/delete/edit announcements and keep them in list.
- V_map class is the main class of program which has whole functionality.

# 5. User Interface

## 5.1. Screen Mock-ups


Figure 9

*When a guest clicks the Sign-up button, a form page is opened, which indicates full name, email address, password, and confirm password sections. The guest must give each of these information correctly in order to be a member in the website.

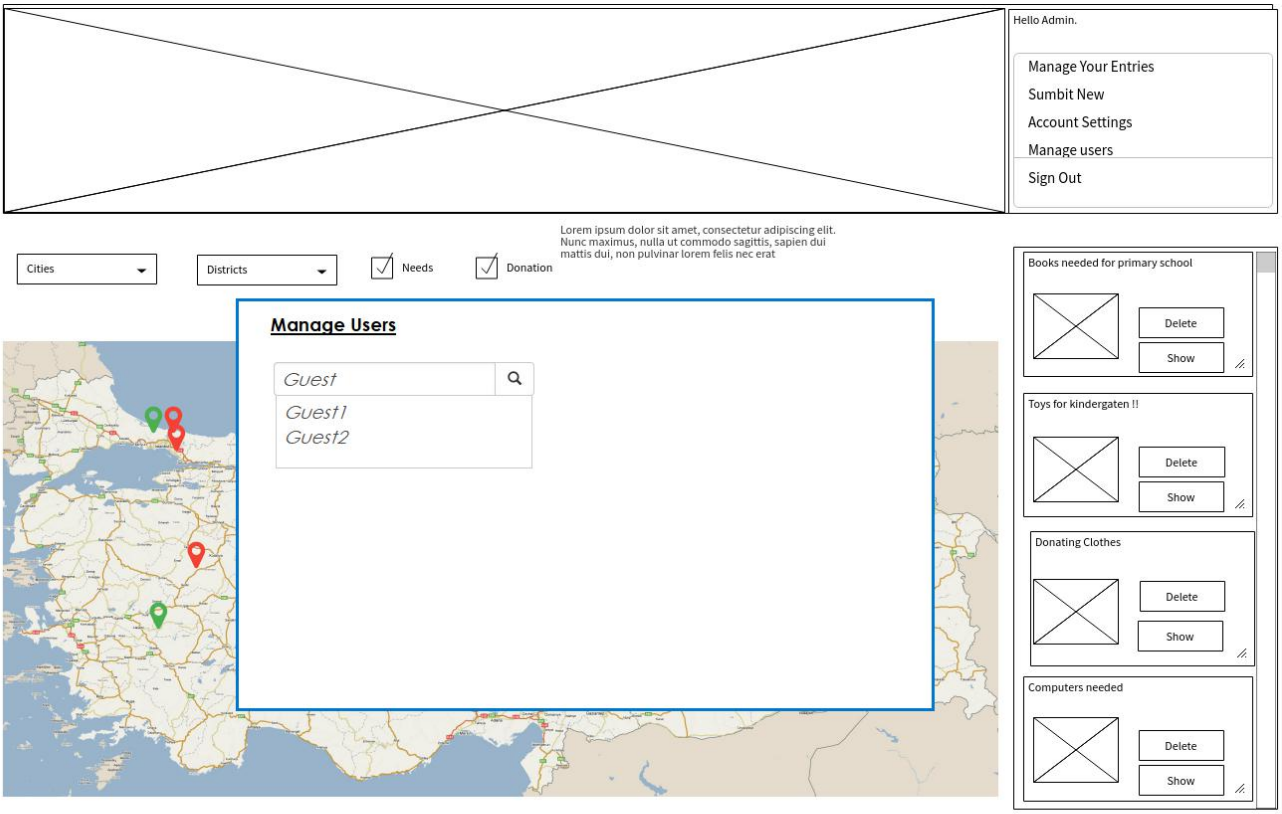



Figure 10

*This page comes up, when the admin clicks the manage users button. The admin can find a user easily, using the" filter" feature. This way he can see the user's profile and investigate it.

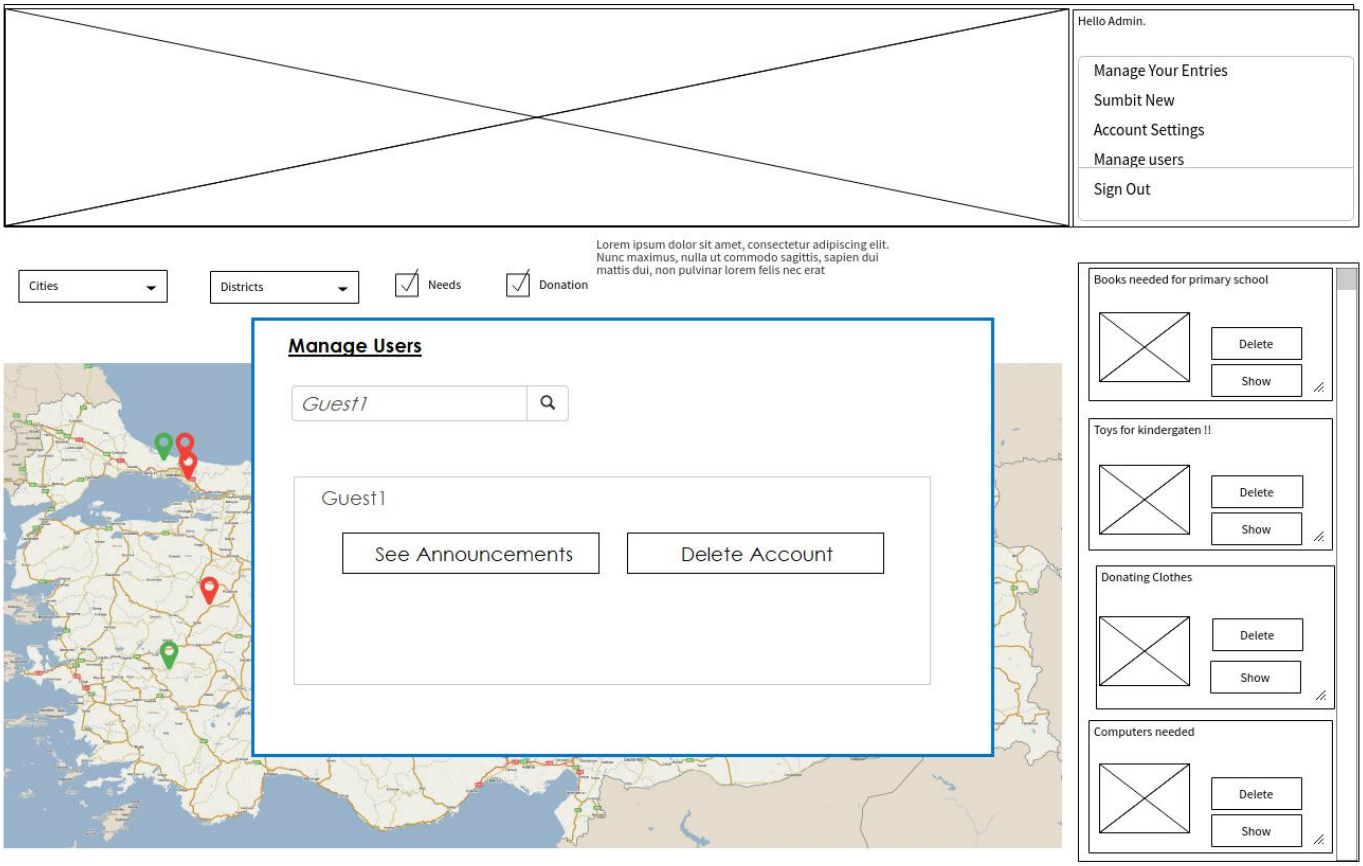



Figure 11

*The admin has the permission to see and delete any user's announcements or accounts. He can also be able to search for a specific user's name and reach his profile quickly. In this picture he selects user1 and he can see his announcements or delete his account.

* When the admin clicks on the "see announcements" button, the announcement box will be filtered according to user1's submissions only.

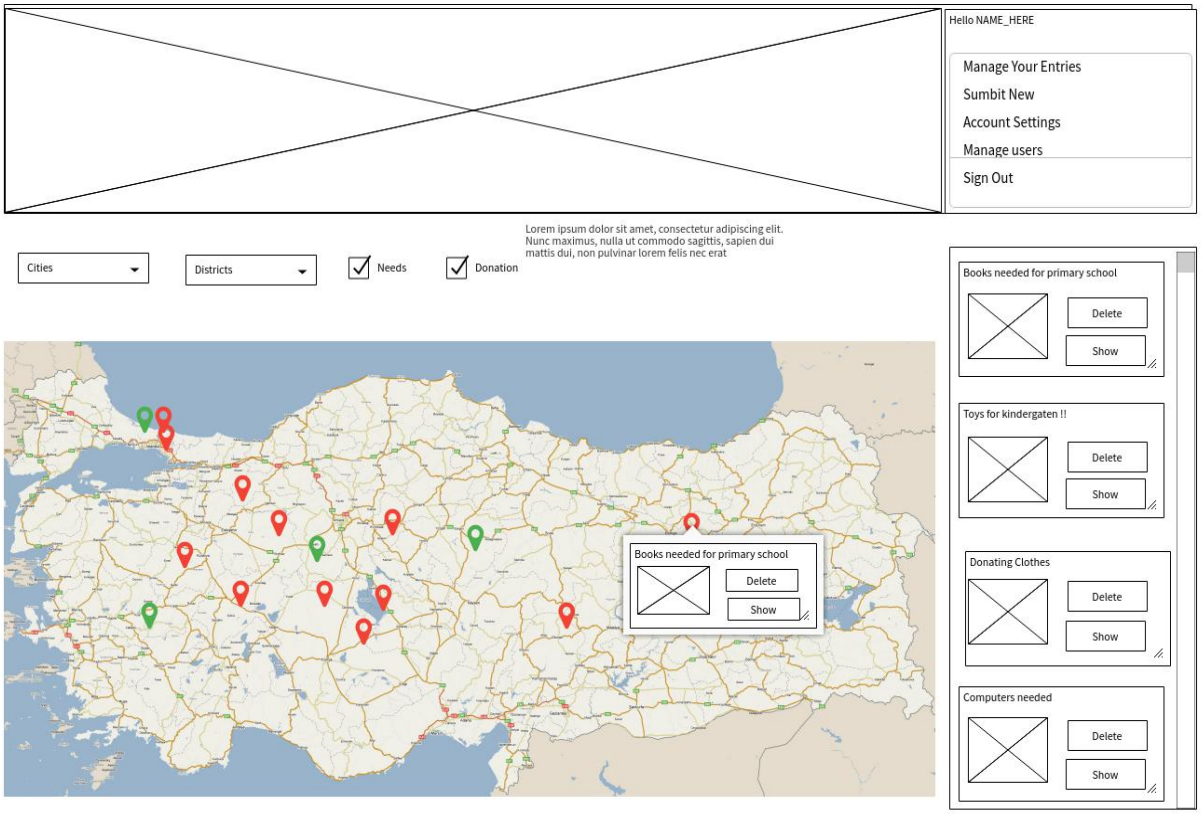



Figure 12

*This page shows the page, when the admin signs up and when he pulls his mouse towards a location on the v.map.

*The ‘manage users’ section will not be shown on the screen if the user is not admin.

* A user can delete only his own announcements or he can revise/ edit them if he needs.

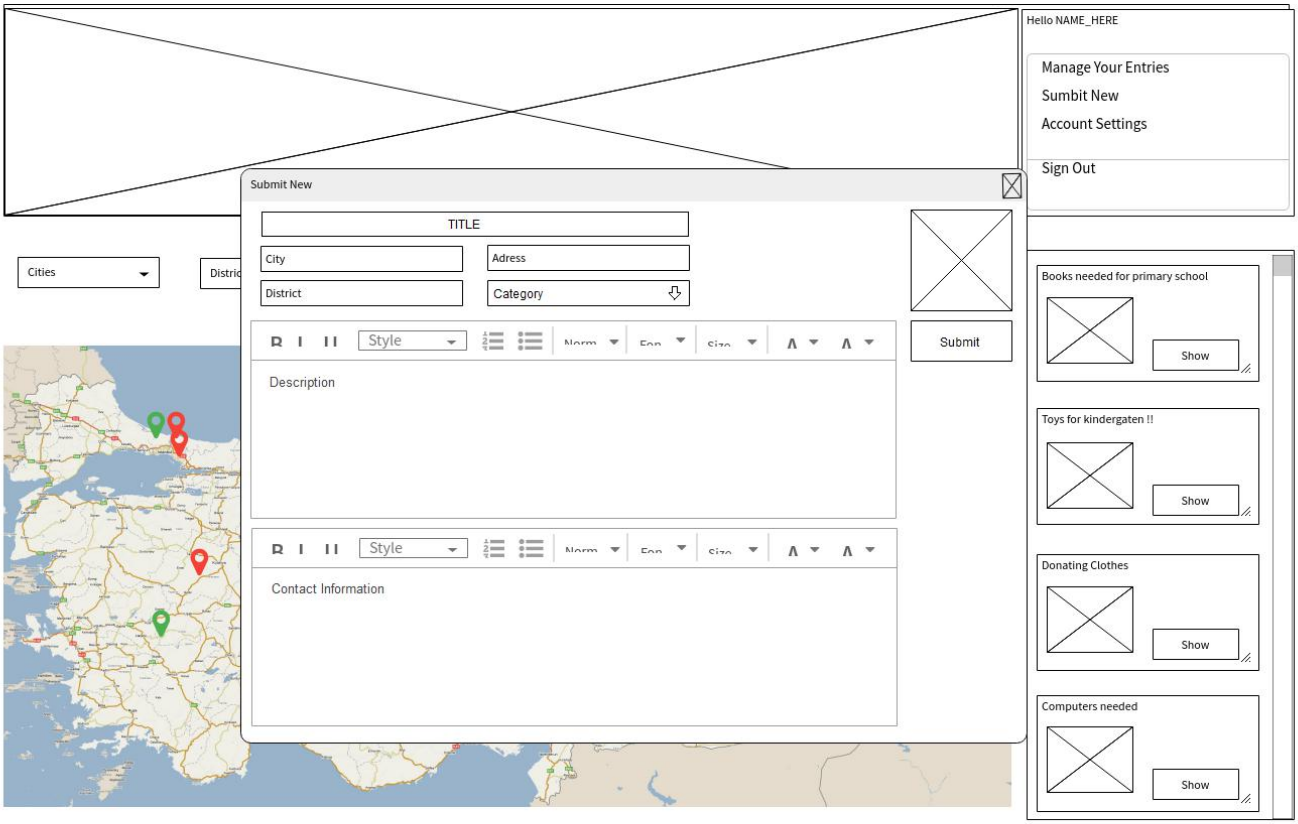



Figure 13

*Random user submission pop-up screen. (Announcement submission)

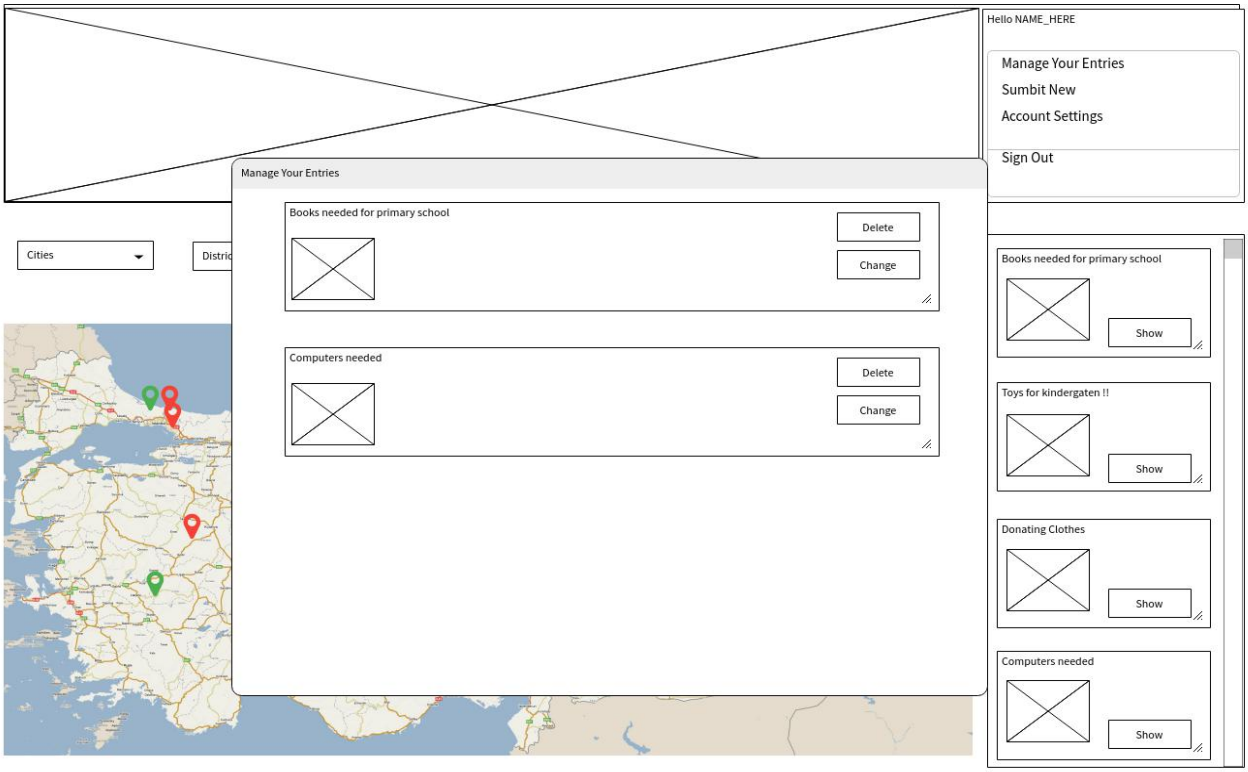



Figure 14

*A user can manage his entries. He can delete his own submission or edit it according to the situation.

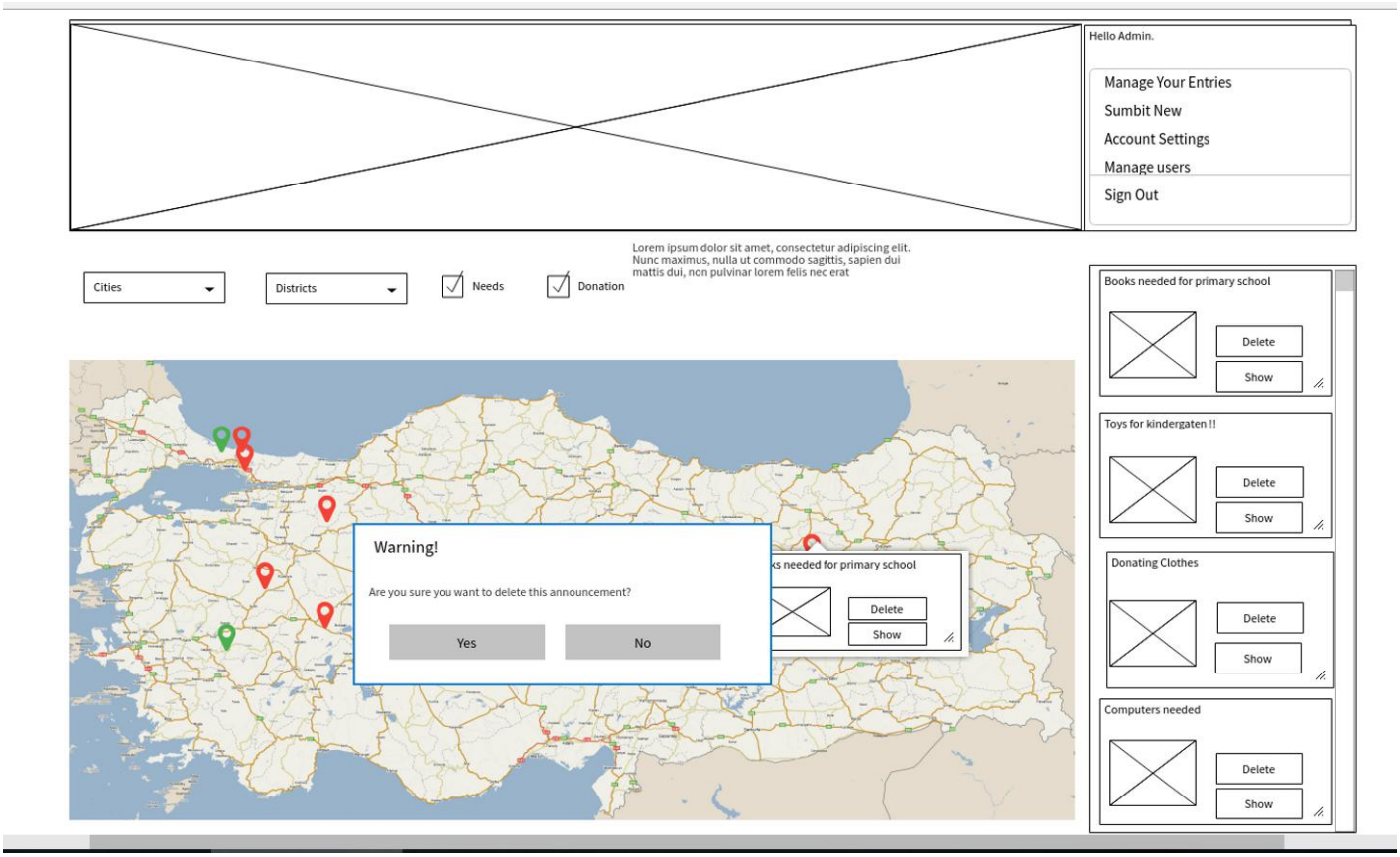



Figure 15

*Admin can delete a user account or an announcement.

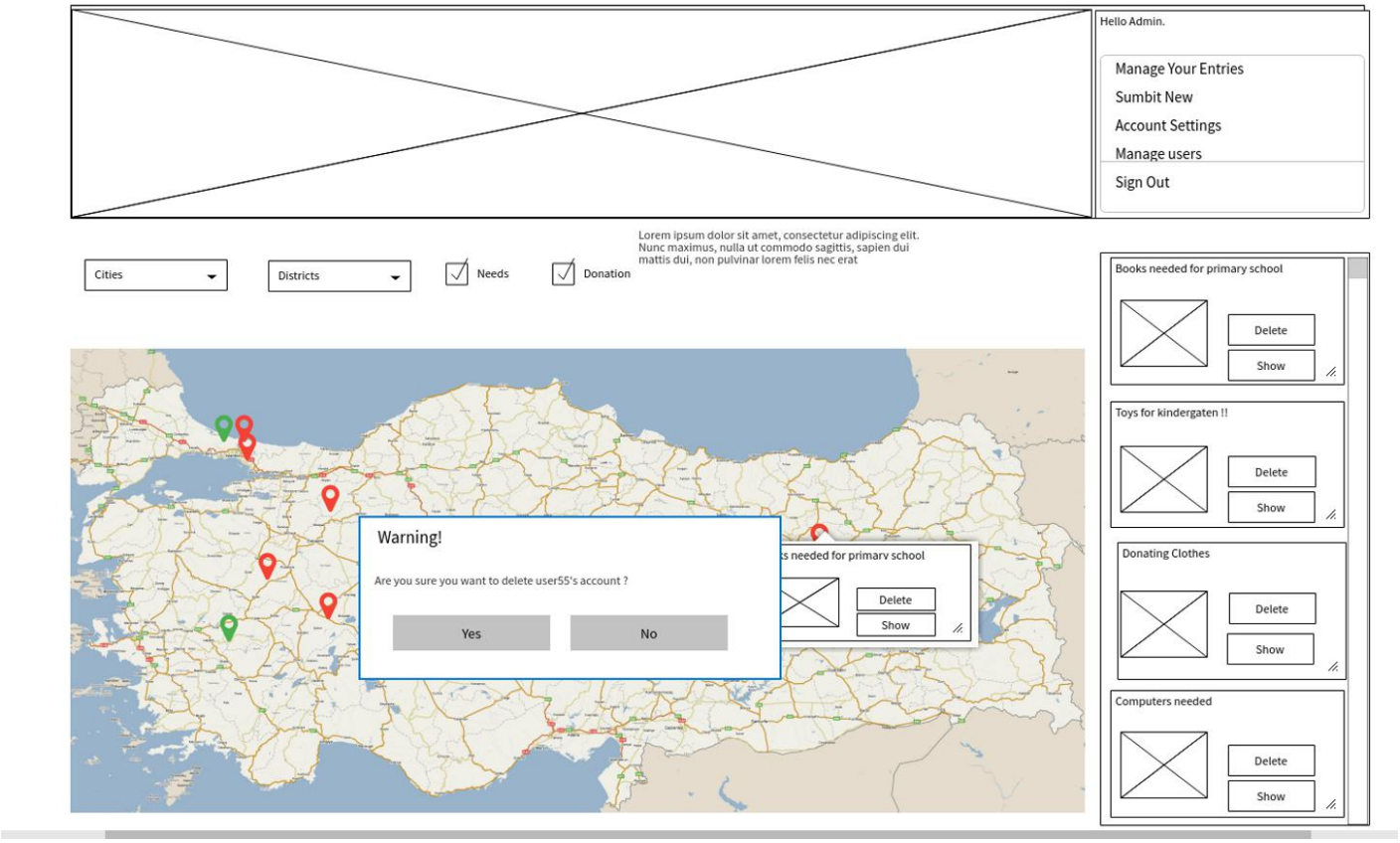



Figure 16

# 6. Conclusion

This document is our analysis report for our web application. Since our project is made for such a big organization as TOG with around 65000 volunteers from various age groups it was important for us to make something simple and user-friendly.

Our use case model has gone through several iterations before we finally decided what use cases and actors we should or should not have. During this part we contacted the organization couple of times and asked some questions make sure we know what they want. At first they wanted us to prevent non-members from creating needs and donations, however, they changed their mind later and we had to remake our use case diagram to suit their needs.

We tried showing only the most important parts of our system in the dynamic model. It shows what happens when users open announcements or create them when Admins ban the Volunteers etc. This was one of the hardest parts to finish in this report. It made us think about the things that can go wrong and ways of preventing it. Although it took us a lot of time to finish it, it sharpened our web application and taught us a lot.

We also added our user interface in this report. As we mentioned before, since the organization was changing their mind we had to redraw the user interface to satisfy their needs. Our group tried focusing on simplicity. We did not want to have many pages, thus, we tried including everything in pop up windows. It made our user interface fancy and less annoying since it would be irritating for the user to go back to the map every time he/she finishes an action.

To sum up, we think that this report was worth our time since it will help us a lot in the future. In fact, it already made us contact the organization and ask questions in order to make some parts of the project clear.

# 7. References

[1] “Toplum Gönüllüleri Vakfı”. https://www.tog.org.tr. [Accessed: Oct 7, 2107].