Jorge Blanco Rey
Ángel Casanova Bienzobas
S.Xiao Fernández Marín

jorge.blancor@estudiante.uam.es
angel.casanova@estudiante.uam.es
sofiax.fernandez@estudiante.uam.es

Object Oriented Design Document

Application: Citicide

Citicide

Date: 03/03/2020
Pair 2

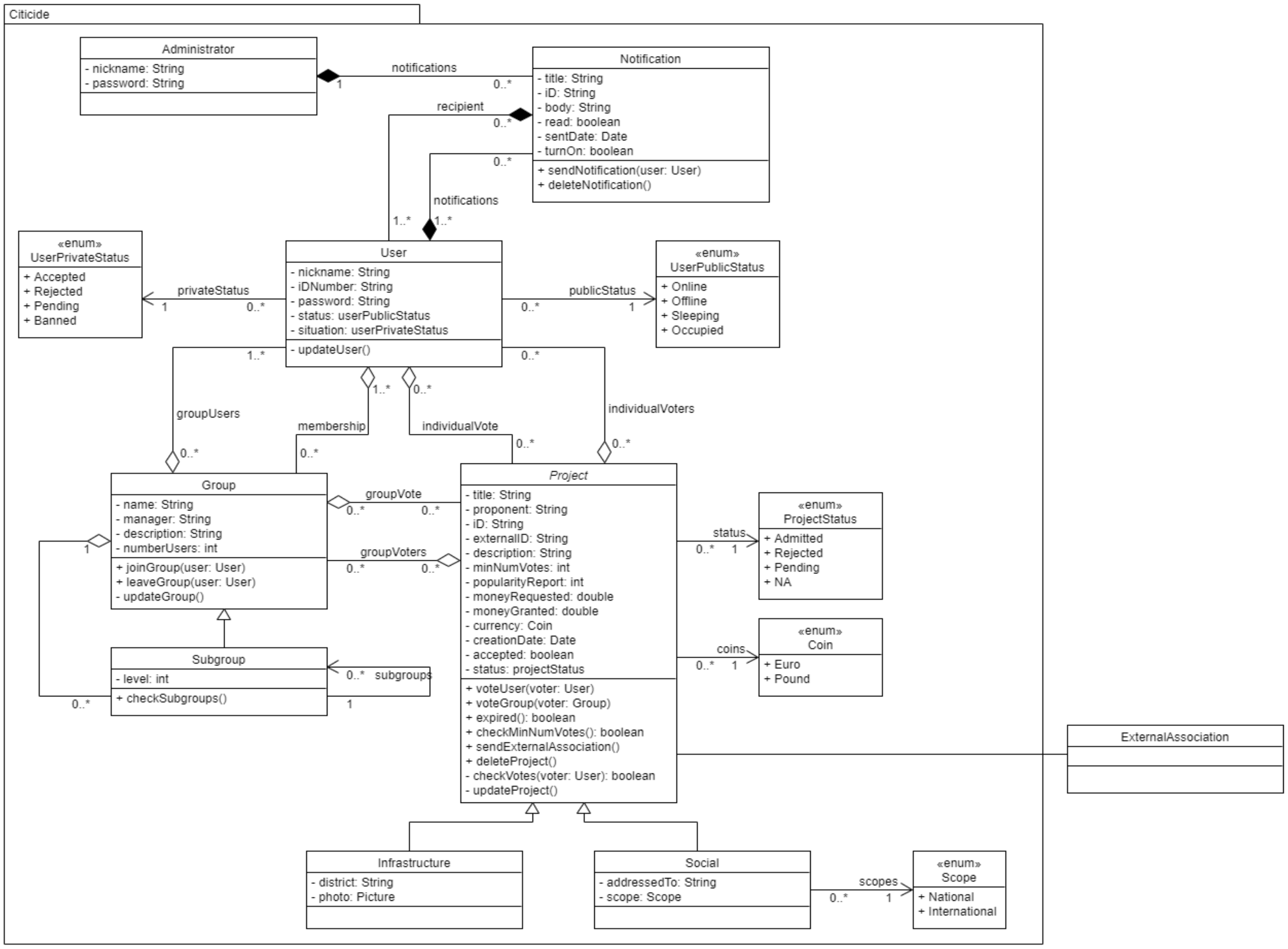
Jorge Blanco Rey jorge.blancor@estudiante.uam.es
Ángel Casanova Bienzobas angel.casanova@estudiante.uam.es
S.Xiao Fernández Marín sofiax.fernandez@estudiante.uam.es

Index

Jorge Blanco Rey
Ángel Casanova Bienzobas
S.Xiao Fernández Marín

jorge.blancor@estudiante.uam.es
angel.casanova@estudiante.uam.es
sofiax.fernandez@estudiante.uam.es

1. Class diagram



This diagram has several classes, which we are going to explain one by one in detail.

The Administrator class has as attributes a string for the name, the password and an array of notifications. The Notification class has a string of characters for the title, the unique ID and the body. It also has a boolean attribute to check whether it has been read and another one to let users set the notifications on or off. In addition, it has a date attribute and an array that contains all the users that have received that notification.

User is another class that is composed of a string for the name, its ID number and the password. It also has two enumerations, one is to set its private status (if it is accepted, rejected, pending or banned) and another one to set its public status (online, offline, etc.).

Finally, it has an array of the groups he belongs to and another where all the projects he has voted for are stored.

The Group class has a string of characters for saving its name, the manager and the description. It also has a counter to know how many users are there in the group, and an array to know which projects it has voted for. In the methods we can find that we have one for joining and another for leaving the group, we have also another one for updating it.

The Group class has also an array to set all its subgroups, for doing this, we have created a Subgroup class. This class inherits from Group all its attributes and methods, and has a new one called level, because users can only be members of sibling groups. Subgroups can have as many Subgroups as they want, but a Subgroup can only have one parent.

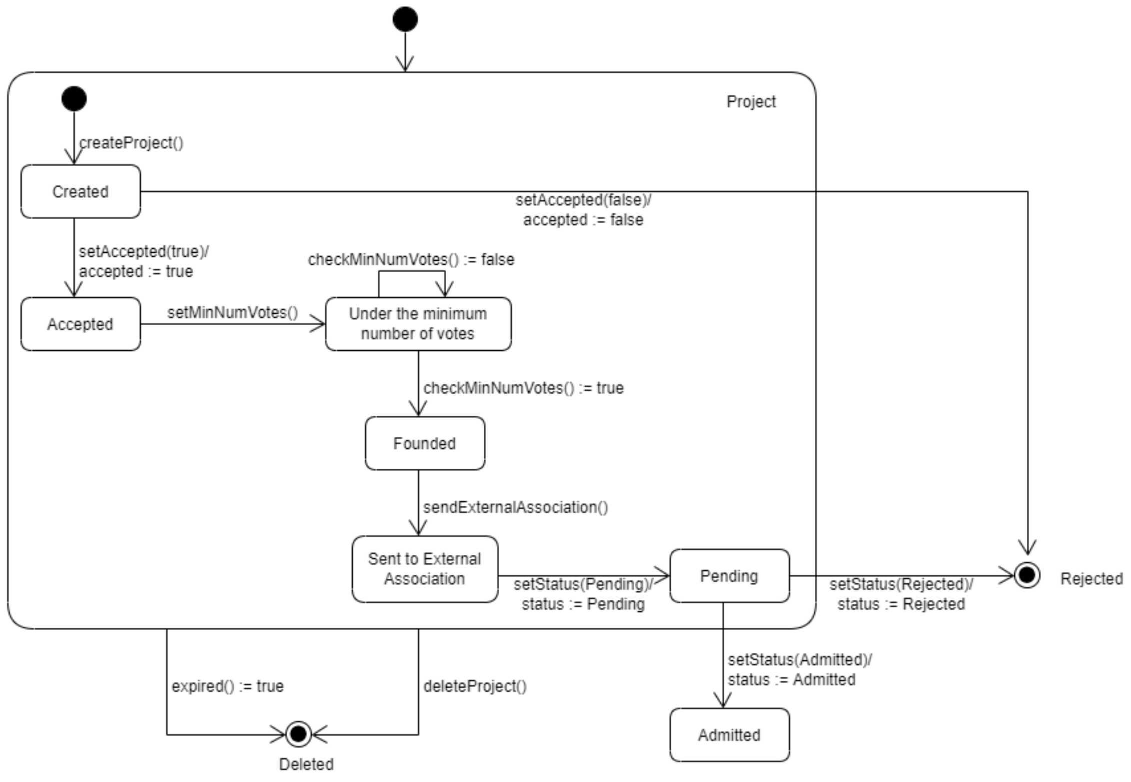
The Project has a string for saving the title, proponent, ID, external ID that is returned by the External Association and the description. There are also integers to save the minimum number of votes that the administrator of the application will set if he approves the project, and to store the number of votes that it receives. It also has two double variables, one to store the money requested and another one to save the money granted. We have set an enumeration, so the proponent can select the currency that he wants to use (Euro or Pound) and a creation date. If it is accepted or not it is stored in a boolean variable and the status of the project is saved on a enumeration (Admitted, rejected, etc.). It also has two arrays, one to store the individual voters and another one for the group voters.

Then Infrastructure and Social classes inherit from Project. The Infrastructure class has a string to store the district where the user wants to perform the project and a matrix that will be a picture of the project. The Social class has a string to whom it is addressed to and an enumeration to set whether is it national or international.

In this whole implementation, as it can be seen in the containments between Group, Project and User, we are going to duplicate information among the different classes, by generating doubles associations between them. We know that this could seem tautological, but we reckon that this design decision is worthy, because we generate a quite important improvement on the system performance.

2. State transition diagrams

2.1 Project

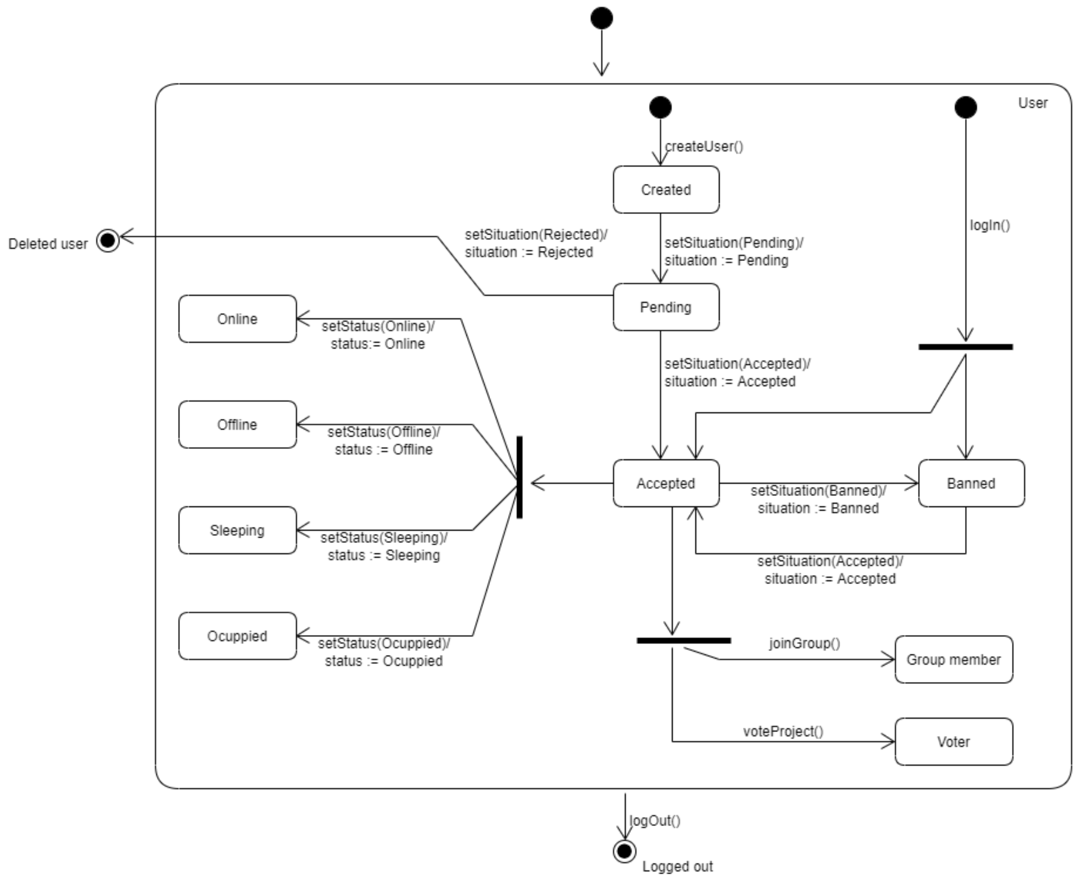


In this diagram we can observe all the possible states of a project. Firstly, the project is on the “Created”, from this point, it can be “Rejected” or “Accepted” by the administrator, in this last case, he has to establish a minimum number. In this state of the diagram, the project is visible to anyone who wants to vote for it and will stay in “Under the minimum number of votes” until enough number users support it. Its next state, when it receives the required votes, is to wait for the proponent to send the project to an external association, once there, this association can reject it or admit it. Besides, if the time threshold is over, independently from the state of the project at that time, it will be “Deleted”.

Jorge Blanco Rey
Ángel Casanova Bienzobas
S.Xiao Fernández Marín

jorge.blancor@estudiante.uam.es
angel.casanova@estudiante.uam.es
sofiax.fernandez@estudiante.uam.es

2.2 User



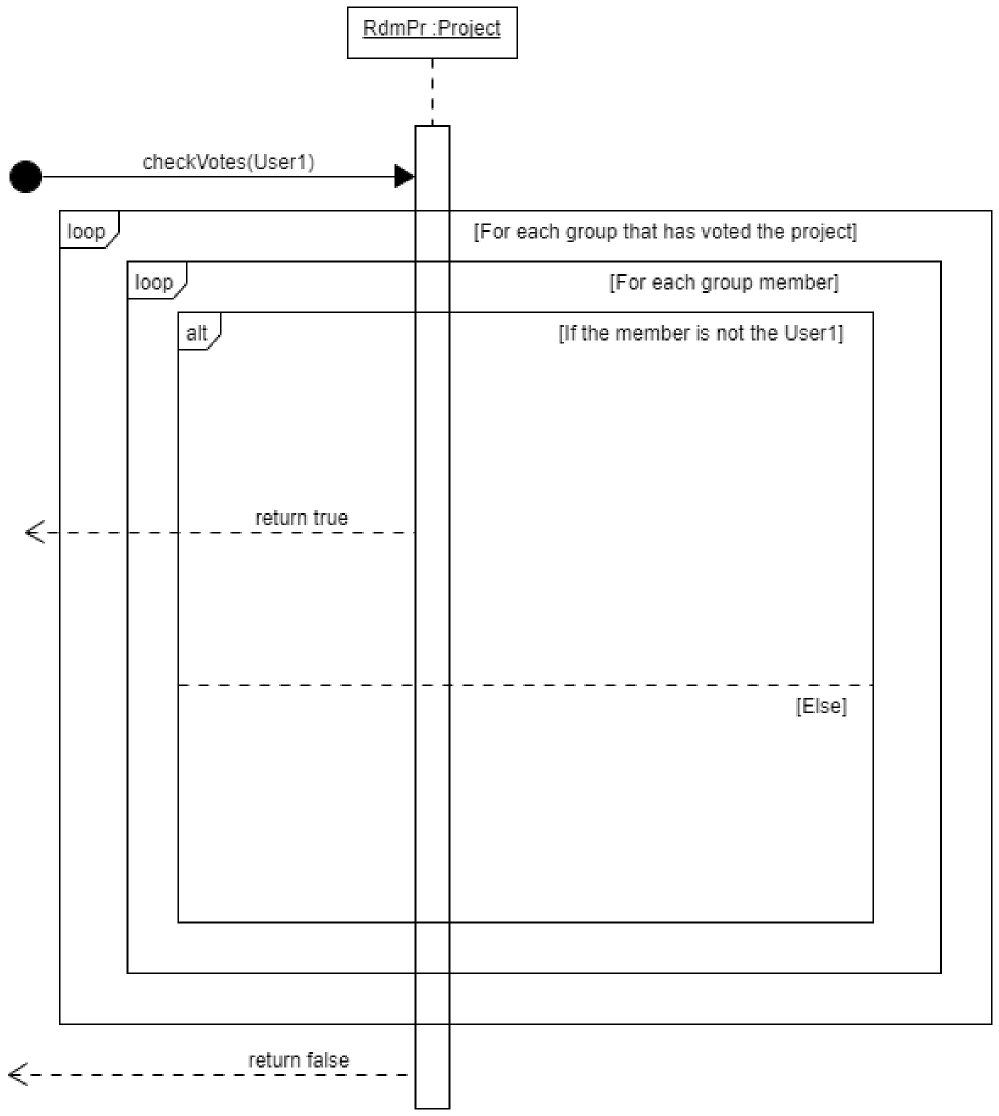
In this diagram we can see the different states that a user can have. Once a user is registered, “Created” state, he is in the “Pending” state waiting for the approval or rejection of the administrator, then he passes to one of these mentioned states. After that, if the administrator considers it, he can ban the user, so he turns to the “Banned” state, because of this, when a user logs in he can be “Banned” or “Accepted”. It is important to remark that a user in the “Banned” state cannot do anything.

Then, when the user is on the “Accepted” state, he can change his status to the four different ones and can also become a “Group member” when joining a group or “Voter” when voting for a project as an individual. We should not forget that although an “Accepted” user can change his state to the ones mentioned in this paragraph, he does not leave the “Accepted” state. Also, when a user is “Banned”, its states will be left as they were the last time he was on the “Accepted” state and will not be changed until he returns to this state.

Finally, from all the possible states previously mentioned the user can log out.

3. Sequence diagrams

3.1 checkVotes

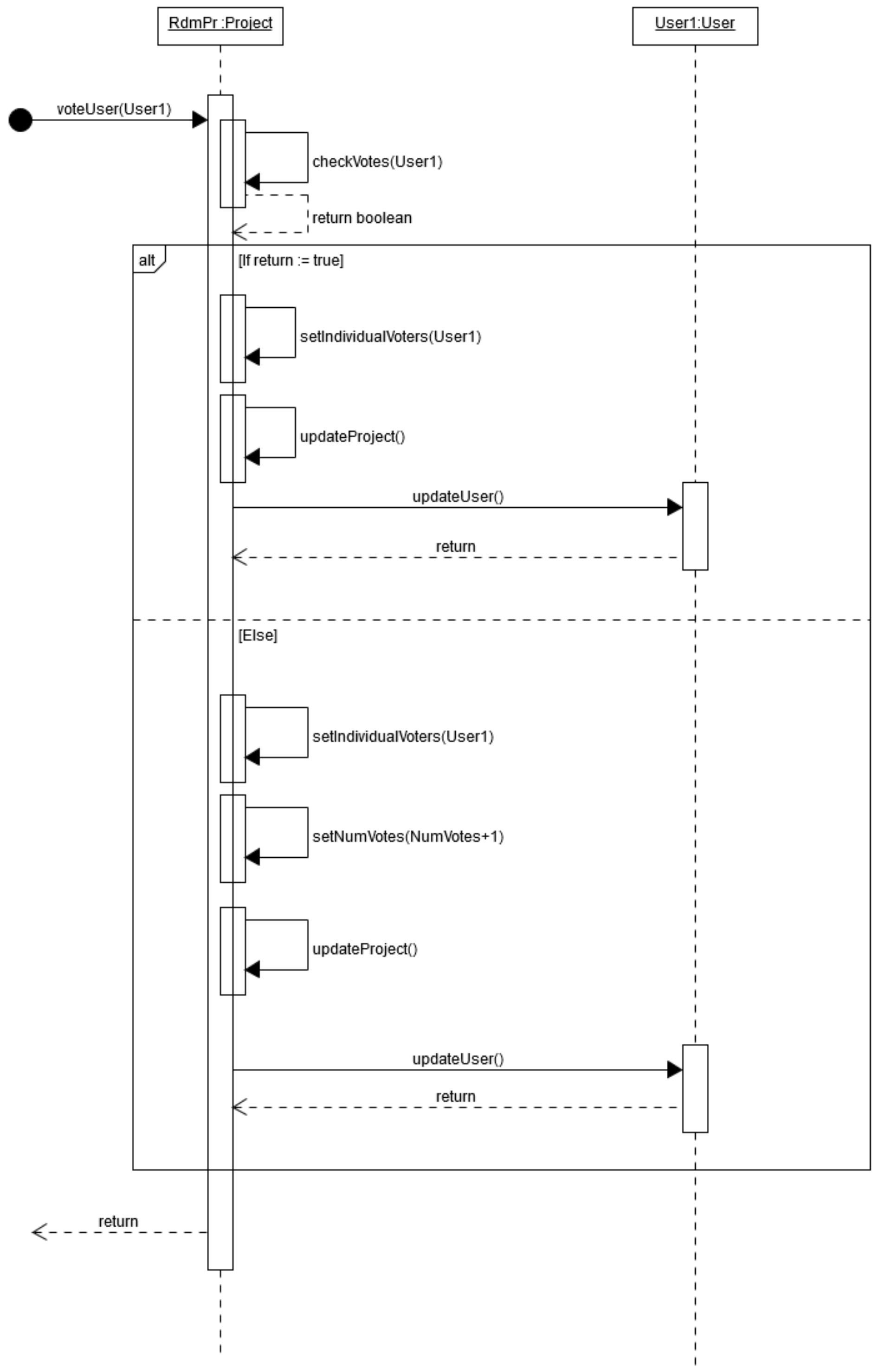


This sequence diagram develops the scenario where the system must determine if a user has already voted for a project since he is a group member.

First, the method checkVotes() iterates through the collection of groups that have supported the project, then for each group, it iterates again in each container of members in order to compare each of them with “User1”, who is the one passed as argument (it is the new voter). If “User1” does not belong to any group, then the function returns true, otherwise it returns false.

Jorge Blanco Rey
Ángel Casanova Bienzobas
S.Xiao Fernández Marín
jorge.blancor@estudiante.uam.es
angel.casanova@estudiante.uam.es
sofiax.fernandez@estudiante.uam.es

3.2 voteUser



Jorge Blanco Rey
Ángel Casanova Bienzobas
S.Xiao Fernández Marín

jorge.blancor@estudiante.uam.es
angel.casanova@estudiante.uam.es
sofiax.fernandez@estudiante.uam.es

This sequence diagram shows the scenario where a user wants to vote for a project individually. The method voteUser() first makes a call to checkVotes() and depending on the return of this subroutine, it updates the number of votes of the project or not.

Both paths in the "alt"(conditional) square, first, includes the "User1"(the new voter), to the array of individual voters of the group and at the end of the method, they both update the collections of the project and the one in user.

As we are duplicating information by duplicating information among the classes, in the aim of gaining performance, it is highly important to call the update functions due to, those functions are ones that syncronis the system.

4. Traceability matrix

In this matrix we can see all the requirements of our project and all the class methods we have created. We have marked with an "X" all the methods that have to do with each requirement. There are some that do not have any mark, this is because they use as methods setters and getters, which were not asked to be added on the diagram.

The first requirement is that "Only the administrator can ban or unban a user" and has "updateUser" and "sendNotification", because we need to modify the User class and send a notification telling why he was banned. The requirement "Only the administrator can reject a petition to join the app" as well as "Groups are created by users" have the "updateUser" method.

"Only the manager of a group can create a subgroup" needs "updateGroup" to change the number of people and subgroups that are in the main group. Users are also allowed to be in several groups if they are siblings, so we need a method called "checkSubgroups" that compares the level of each subgroup the user wants to join.

Users can unsubscribe from a group and for that, we will need the methods "upadateUser" (for updating the numbers of groups he belongs to), "leaveGroup" (that will call updateGroup) and "updateGroup" that will change the array of the people that are in that group.

All users can receive notifications if they want to. That notification will be sent by the administrator of the application using the "sendNotification" method, so they all have the option to turn them on or off (except from the proponent, who will always have the notifications on for the project he created). Users can also delete the notifications, for this, the "deleteNotification" method is used.

"A project is created by a user", so we need the "updateUser" method to maintain up to date its array.

"Only the administrator of the application can approve or reject a project". So, if the project is accepted, the user becomes the proponent and we will need to change its array of projects, as well as if there was a group that wants to create a project, the votes of all the people of that group will be set to the project, and in case that the minimum number of votes for the project is reached, the project will be sent to the external association.

The administrator is also in charge of setting the minimum number of votes when he accepts the project. In that case, a notification will be sent to the proponent.

If a project has not reached the minimum number of votes in 30 days since it is approved, it expires and is deleted. On the other hand, if it has reached the required number of votes and it has not expired, the proponent of the project is in charge of sending it to the external association.

When voting, the user can do it individually or as a member of a group, in any case, the vote or votes will be added to the project and the array of users for the projects he has voted will be updated. If he votes as member of a group, he will need to specify which group he is voting for, to update the group votes and add the them to the project. And if the user votes individually, the application will need to check if he has already voted as group member using "checkVotes", if that is the case, its vote will not be count again.

Also, if a user gets banned, we have to update all the projects in order to remove all his votes, but all the information about the user is kept in the system, since he can be unbanned in the future and then all the votes will be returned to the proper projects. In a similar way, if a user leaves a group, all the projects that are supported by this group will lose one vote.

Finally, we show in the matrix that we have also implemented the popularity and the affinitive reports, the first one returns the number of votes of the project if the user who asked for it has previously voted individually and the last one, just computes a coefficient related to the "relativity association" between two groups.

Class / Method / Requirement	User	Notification		Group			Subgroup	Project							
	updateUser()	sendNotification(u: User)	deleteNotification()	joinGroup()	leaveGroup()	updateGroup()	checkSubgroups()	voteUser (v: User)	voteGroup (v: Group)	expired ()	checkMinNumVotes()	checkVotes(v: User)	sendExternalassociation()	deleteProject()	updateProject()
The user must be registered															
Only the administrator can ban or unban	X	X													
Only the administrator can reject a petition to join the desktop app	X														
A person can just register once															
Users can only see the groups other users are in															
Groups are created by users. Then are managers	X														
Only the manager of a group can create a subgroup						X	X								
A user can be manager of several groups															
Users are let to be in several subgroups if they are siblings							X								

Class / Method / Requirement	User	Notification		Group			Subgroup	Project							
	update User()	sendNotification(u: User)	delete Notification()	join Group()	leave Group()	update Group()	checkSub groups()	vote User (v: User)	vote Group (v: Group)	expired ()	checkMin Num Votes()	check Votes(v: User)	send External association()	delete Project()	update Project()
Users can unsubscribe from a group	X				X	X									
If a user wants to receive a notification he will have to turn it on		X													
A user receives a notification if a project created by a group he is in, is approved or rejected		X	X												
A manager of a group will always receive notifications			X												
A project is created by a user	X														
The system adds an ID to the project															
The administrator can reject or approve a project	X					X		X	X		X			X	X

Class / Method / Requirement	User	Notification		Group			Subgroup	Project							
	updateUser()	sendNotification(u: User)	deleteNotification()	joinGroup()	leaveGroup()	updateGroup()	checkSubgroups()	voteUser (v: User)	voteGroup (v: Group)	expired ()	checkMinNumVotes()	checkVotes(v: User)	sendExternalassociation()	deleteProject()	updateProject()
When accepted, the administrator should set the minimum number of votes		X	X												
A project expires in 30 days since it is approved										X				X	
The project manager submits the project for founding										X	X		X		
Votes can be done as an individual	X							X				X			X
Votes can be done as a manager of a group	X								X			X			X
If a user votes as a manager, he must specify which group he is voting for	X					X			X						X
If a user votes as an individual, he cannot unvote	X							X				X			X

Class / Method / Requirement	User	Notification		Group			Subgroup	Project							
	updateUser()	sendNotification(u: User)	deleteNotification()	joinGroup()	leaveGroup()	updateGroup()	checkSubgroups()	voteUser (v: User)	voteGroup (v: Group)	expired ()	checkMinNumVotes()	checkVotes(v: User)	sendExternalassociation()	deleteProject()	updateProject()
If a user is in a group and he unsubscribes from it, and that group voted on a project, his vote will be removed	X				X	X									X
If a user wants to see the number of votes that a project has, he needs to have voted that project	X							X				X			X
If a user gets banned, its votes are removed	X														X
It exists popularity reports											X				
It exists affinity reports															