DCS Codex Notification System
Analysis Model

Submitted to:

Asst. Prof. Ma. Rowena C. Solamo
Faculty Member
Department of Computer Science
College of Engineering
University of the Philippines, Diliman

Submitted by:
Galano, Anica
Isidro, Rogiella
Tabagan, Ken

In partial fulfillment of Academic Requirements
for the course
CS 191 Software Engineering I
of the
1st Semester, AY **2019-2020**

Unique Reference:

The documents are stored in the https://github.com/theswinghobo/DCSCodexApp referenced with **Group 5-DCS Codex Notification System-Analysis Model.pdf**.

Purpose:

The purpose of this document is to describe the use case analysis through the use of an analysis model, which will aid the group in completing their project requirement for CS 191.

Audience:

The target audience is Asst. Prof. Ma. Rowena C. Solamo as she will be approving of this analysis model for our project in CS 191. This document is also for ourselves, as it serves as a guide for how to proceed with our project.

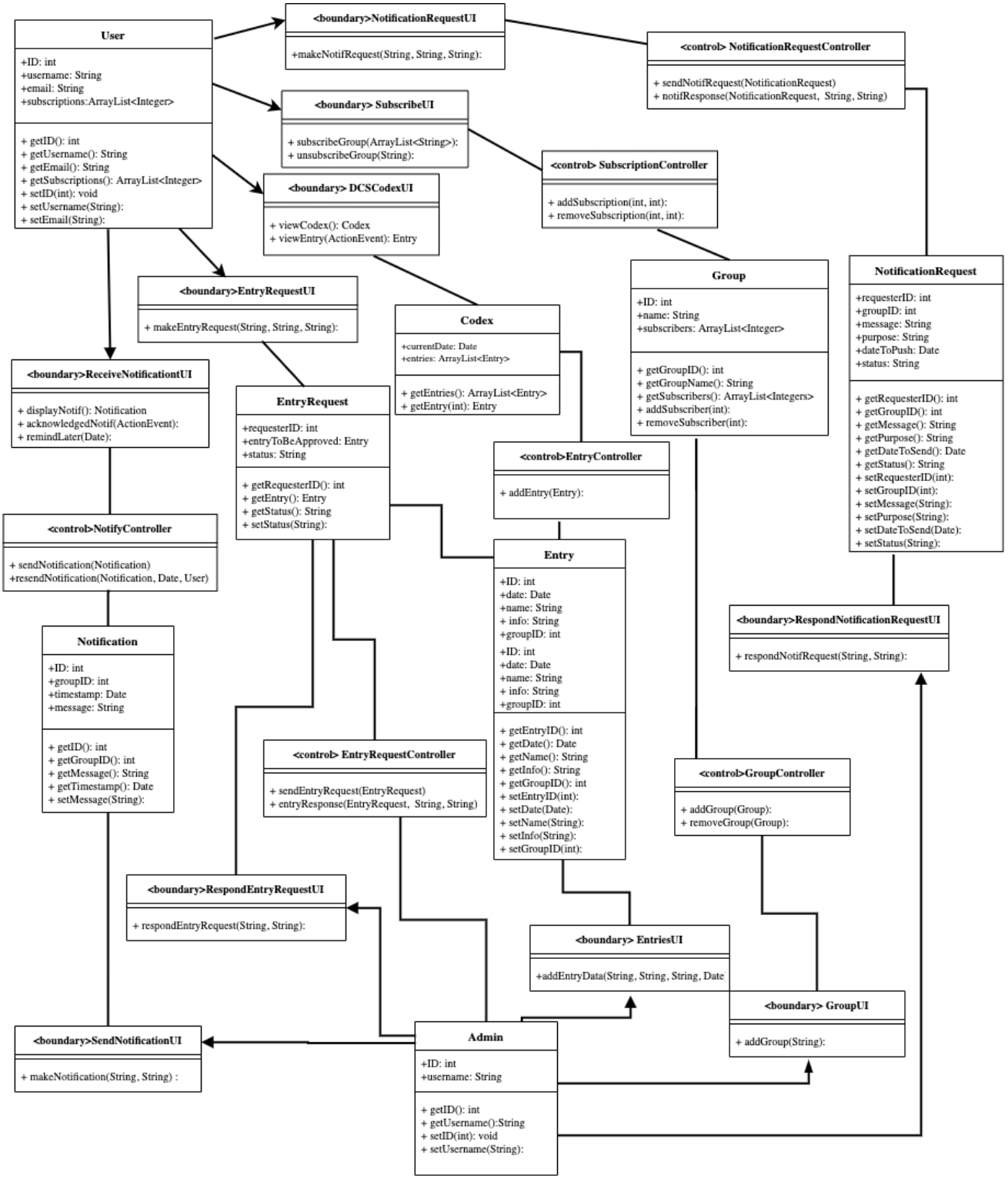
Revision Control:

Revision Date	*Person Responsible*	*Version Number*	*Modification*
10/02/2019	Anica Galano	1.0	Initial document edits; Edits to: all boundary classes;
10/03/2019	Ken Tabagan	1.1	Edits to all entity classes and control class SubscriptionController.
10/03/2019	Anica Galano	1.2	Edits to (User, Admin, Group, Notification, NotificationRequest, Entry, Codex) entity classes and to all control classes;
10/04/2019	Rog Isidro	1.3	Edits to Behavioral Model (Notify)
10/04/2019	Ken Tabagan	2.0	Edits to entity classes (responsibilities) and Behavioral Model (Request Notification)
10/04/2019	Anica Galano	3.0	Added Analysis Model
10/04/2019	Rog Isidro	3.1	Edits to Behavioral Model (Subscribe - Scenario 1)
10/04/2019	Anica Galano	4.0	Edits to Behavioral Model (Subscribe - all scenarios), edits to Analysis Model

***System Name*:** DCS Codex Notification System

***Description*:** This system aims to improve the current DCS Codex system by adding the feature of notifications. Notifications will be used to publicize upcoming events and as reminders regarding academic deadlines and exam schedules. The system is also open to entry and notification requests from users, so they can add events into the codex and notify other users about those events.

***Analysis Model*:**



Boundary Classes:

Class Name	**Description**
DCSCodexUI	This is the main interface to the system the **users** interact with to view the DCS Codex. *Responsibilities*: public Codex viewCodex() public Entry viewEntry(ActionEvent onClick)
GroupUI	This is the interface to the system that the **admin** interact with to create groups. *Responsibilities*: public addGroup(String name)
SubscribeUI	This is the interface the **users** interact with to subscribe to a group (class or organization). *Responsibilities*: public void subscribeGroup(ArrayList<String> groups) public void unsubscribeGroup(String groupName)
EntriesUI	This is the interface of the **admin** to the system whenever he or she needs to maintain the codex entries. *Responsibilities*: public void addEntryData(String name, String info, String group, Date date)
SendNotificationUI	This is the interface the **admin** interacts with to create notifications to send to users. *Responsibilities*: public void makeNotification(String groupName, String message)
NotificationRequestUI	This is the interface of the **user** to the system in order to make and send notification requests. *Responsibilities:* public void makeNotifRequest(String groupName, String message, String purpose)
RespondNotificationRequestUI	This is the interface of the **admin** to the system in order to approve or reject notification requests. *Responsibilities:* public void respondNotifRequest(String status, String message)
EntryRequestUI	This is the interface of the **user** to the system in order to make and send entry requests. *Responsibilities:* public void makeEntryRequest(String groupName, String entryName, String info)
RespondEntryRequestUI	This is the interface of the **admin** to the system in order to approve or reject entry requests *Responsibilities:* public void respondEntryRequest(String status, String message)

ReceiveNotificationUI	This is the interface the user interacts with when receiving notifications. *Responsibilities*: public Notification displayNotif() public void acknowledgedNotif(ActionEvent onClick) public void remindLater(Date datetime)

Control Classes:

Class Name	**Description**
GroupController	This is the control that handles group maintenance. *Responsibilities*: public void addGroup(Group g) public void removeGroup(Group g)
SubscriptionController	This is the control that makes changes to subscription relationships between users and groups (subject or organizations). public void addSubscription(int groupID, int userID) public void removeSubscription(int groupID, int userID)
NotificationRequestController	This is the control that handles notification requests from users. *Responsibilities:* public void sendNotifRequest(NotificationRequest n) public void notifResponse(NotificationRequest n, String status, String message)
EntryRequestController	This is the control that handles entry requests from users. *Responsibilities:* public void sendEntryRequest(EntryRequest e) public void entryResponse(EntryRequest e, String status, String message)
NotifyController	This is the control that handles notifications sent to users from the admin. *Responsibilities*: public void sendNotification(Notification n) public void resendNotification(Notification n, Date datetime, User u)
EntryController	This is the control that handles entries in the DCS Codex as maintained by the admin. *Responsibilities*: public void addEntry(Entry e)

Entity Classes:

Class Name	Description
User	This is the entity class User, which contains the data about the user. *Attributes:* private int ID private String username private String email ArrayList<Integer> subscriptions = new ArrayList<Integer>(); //contains IDs of groups subscribed to by the user *Responsibilities:* public int getID() public String getUsername() public String getEmail() public ArrayList<Integer> getSubscriptions() public void setID(int ID) public void setUsername(String username) public void setEmail(String email)
Admin	This is the entity class Admin, which contains data about the admin. *Attributes:* private int ID private String username; *Responsibilities:* public int getID() public String getUsername() public void setID(int ID) public void setUsername(String username)
Group	This is the entity class Group, which contains data about subjects or organizations users can subscribe to (e.g. CS 191, DCS Servers). *Attributes:* private int ID private String name ArrayList<Integer> subscribers = new ArrayList<Int>(); *Responsibilities*: public int getGroupID() public String getGroupName() public ArrayList<Integers> getSubscribers() public void addSubscriber(int userID)

	public void removeSubscriber(int userID)
Notification	This is the entity class Notification, which contains data about notifications that get sent out by the admin to the users. *Attributes:* private int ID private int groupID // references Group.ID - what group to send notification to private Date timestamp // when notification is to be sent out private String message; *Responsibilities*: public int getID() public int getGroupID() public void setMessage(String message) public void sendNotification(int groupID)
NotificationRequest	This is the entity NotificationRequest, which contains data about notification requests sent by users to admin for approval. *Attrbibutes:* private int requesterID // references User.ID private int groupID // references Group.ID - what group to send notification to private String message // notification content private String purpose // why notification request was made private Date dateToSend// when the notification will be sent to subscribers private String status // APPROVED or REJECTED *Responsibilities:* public int getRequesterID() public int getGroupID() public String getMessage() public String getPurpose() public Date getDateToSend() public String getStatus() public void setRequesterID(int ID) public void setGroupID(int ID) public void setMessage(String message) public void setPurpose(String purpose) public void setDateToSend(Date date) public void setStatus(String status)
Codex	This is the entity Codex, which contains data about the Codex. *Attributes:*

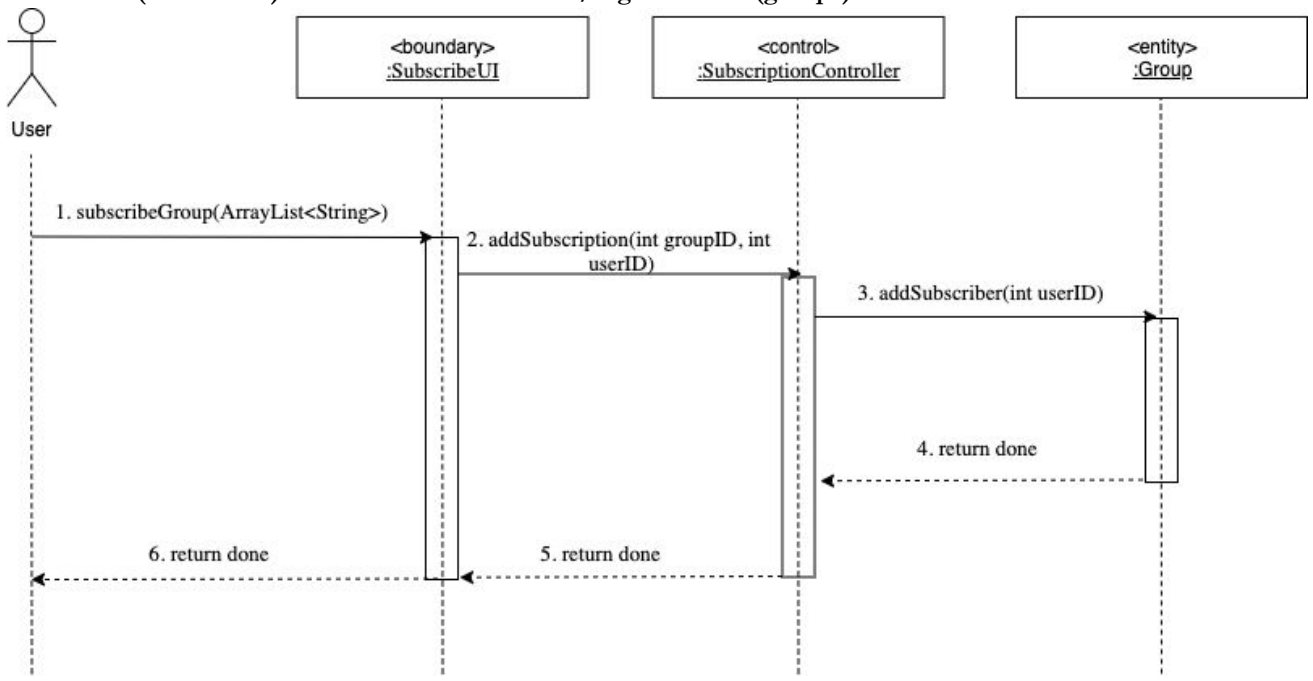
	private Date = currentDate ArrayList<Entry> entries = new ArrayList<Entry>() // list of entries in the Codex *Responsibilities:* public ArrayList<Entry> getEntries() public Entry getEntry(int entryID)
Entry	This is the entity class Entry, which contains data about events and other entries that are displayed in the DCS Codex. *Attributes:* private int ID private Date date private String name private String info private int groupID // references Group.ID - group the entry is for/organized by *Responsibilities:* public int getEntryID() public Date getDate() public String getName() public String getInfo() public int getGroupID() public void setEntryID(int ID) public void setDate(Date date) public void setName(String name) public void setInfo(String info) public void setGroupID(int groupID)
EntryRequest	This is the entity EntryRequest, which contains data about entry requests sent to DCS Codex administrators for approval *Attrbitutes:* private int requesterID // references User.ID private Entry entryToBeApproved private String status // APPROVED or REJECTED *Responsibilities*: public int getRequesterID() public Entry getEntry() public String getStatus() public void setStatus(String status)

Behavioral Model:

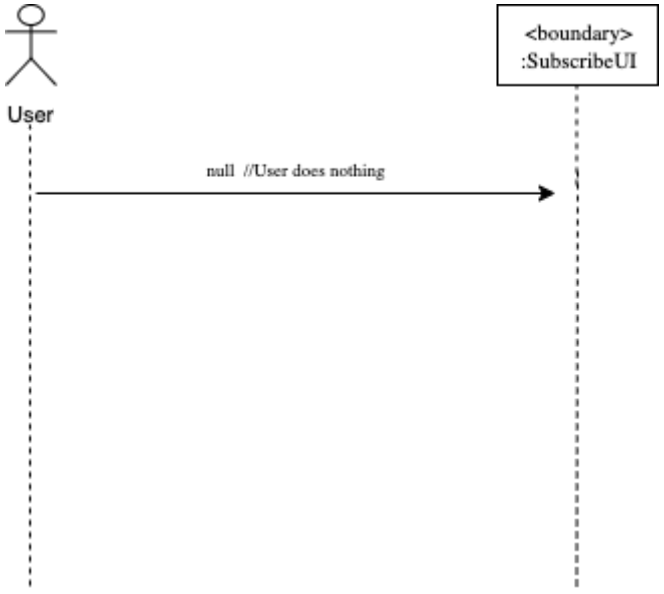
***Use-Case Name*:** 3.0 Subscribe

Description: This use-case occurs after a user registers an account and logs in. With that account they can subscribe to certain classes and CS Network organizations. Through the subscription, users will be able to receive reminders related to the classes and organizations they subscribed to. They can set the frequency, intensity and mode of the reminders.

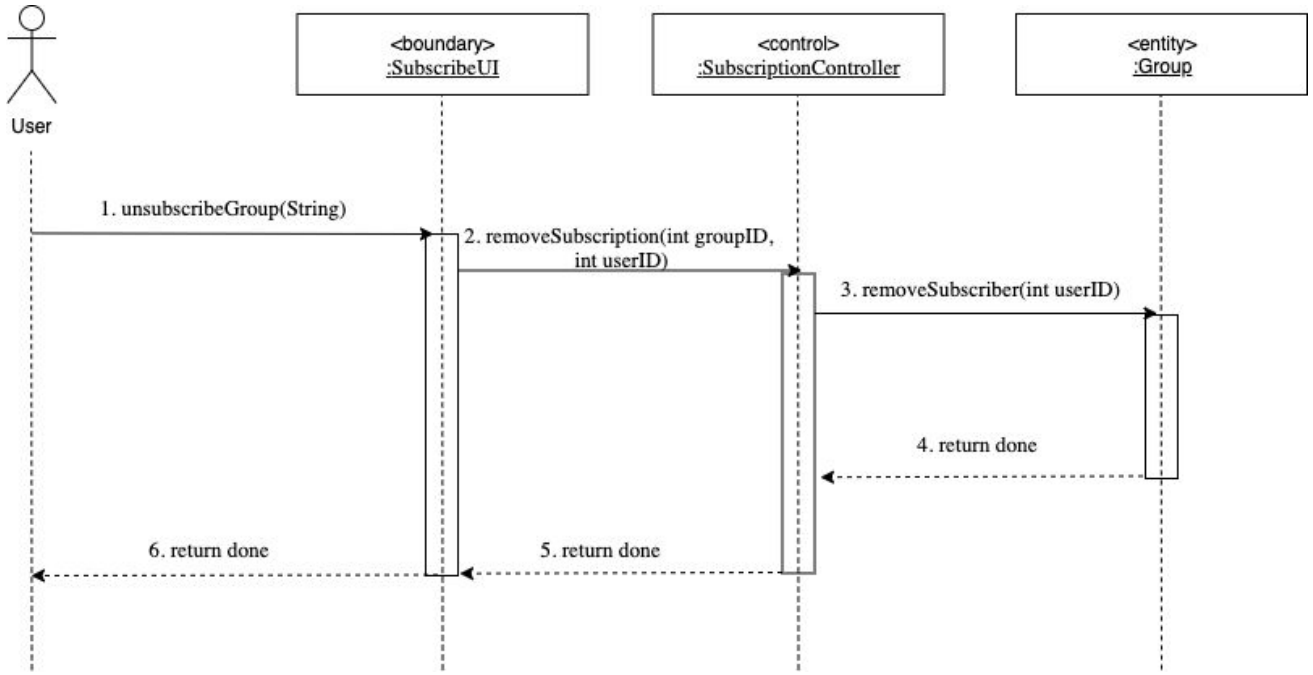
Scenario 1: (Basic Flow) User subscribes to classes/organizations (groups).



Scenario 2: User decides to not subscribe to any classes/organizations (groups).



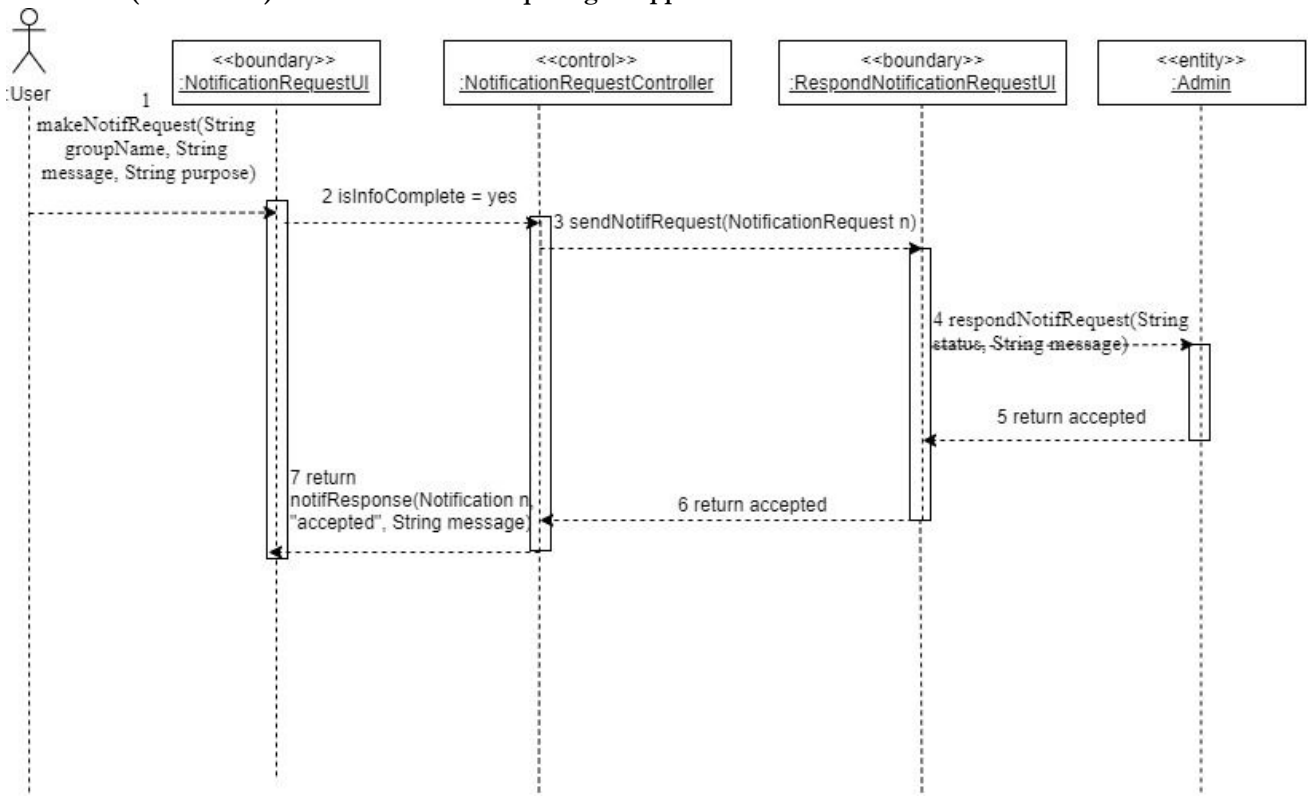
Scenario 3: User decides to unsubscribe from a certain class or organization.



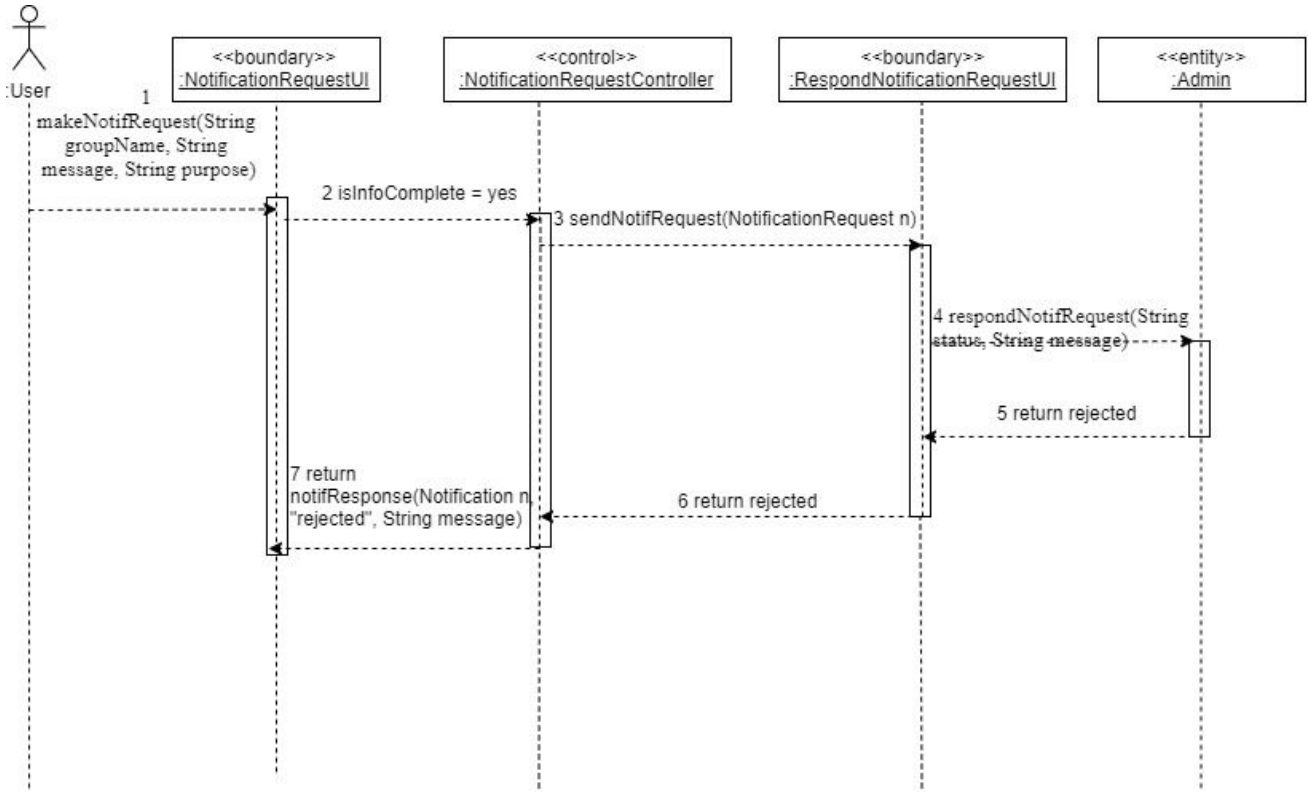
***Use-Case Name*:** 4.0 Request Notification

Description: After the user creates or logs in to his account, he will be able to make notification request. He can answer the notification request form and send it to the DCS Codex admin. The DCS Codex admin will then evaluate the request and decide whether to approve or disapprove it.

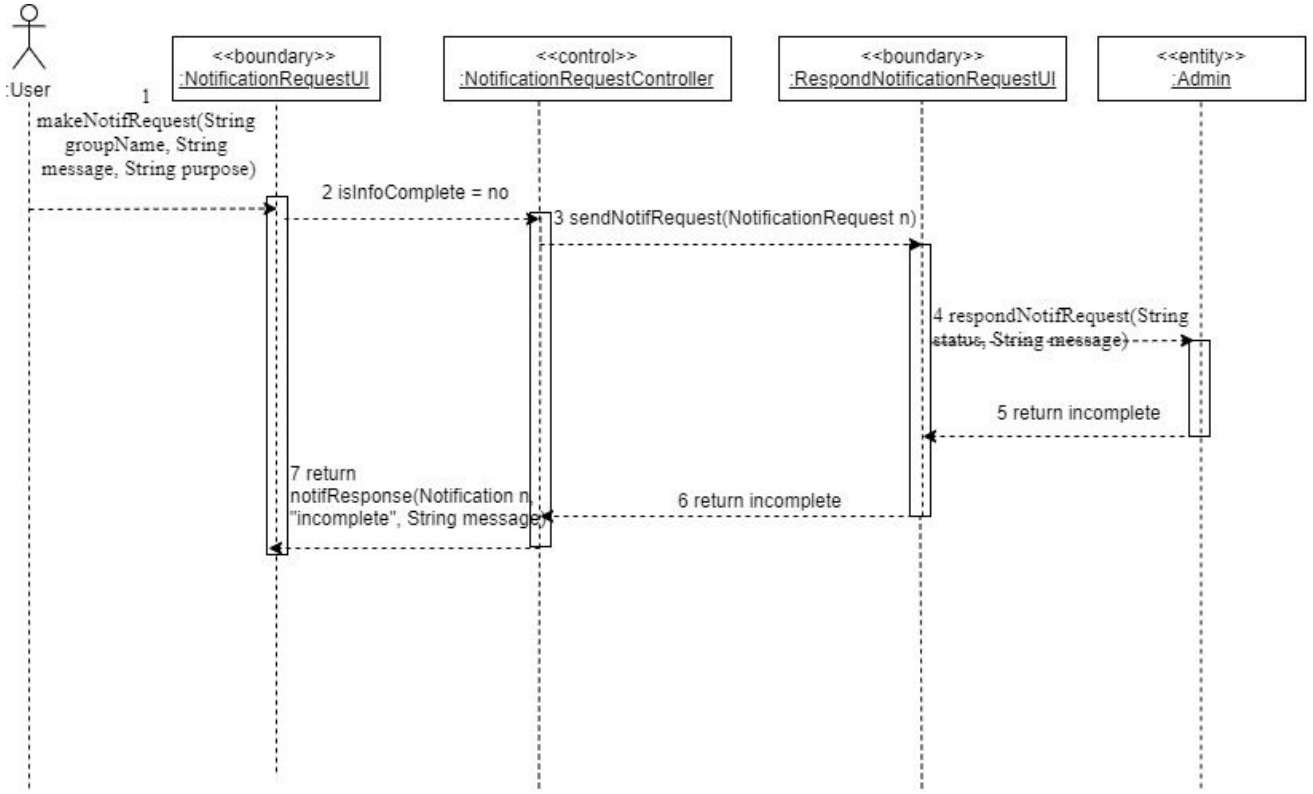
Scenario 1: (Basic Flow) User's notification request gets approved.



Scenario 2: User's notification request get disapproved.



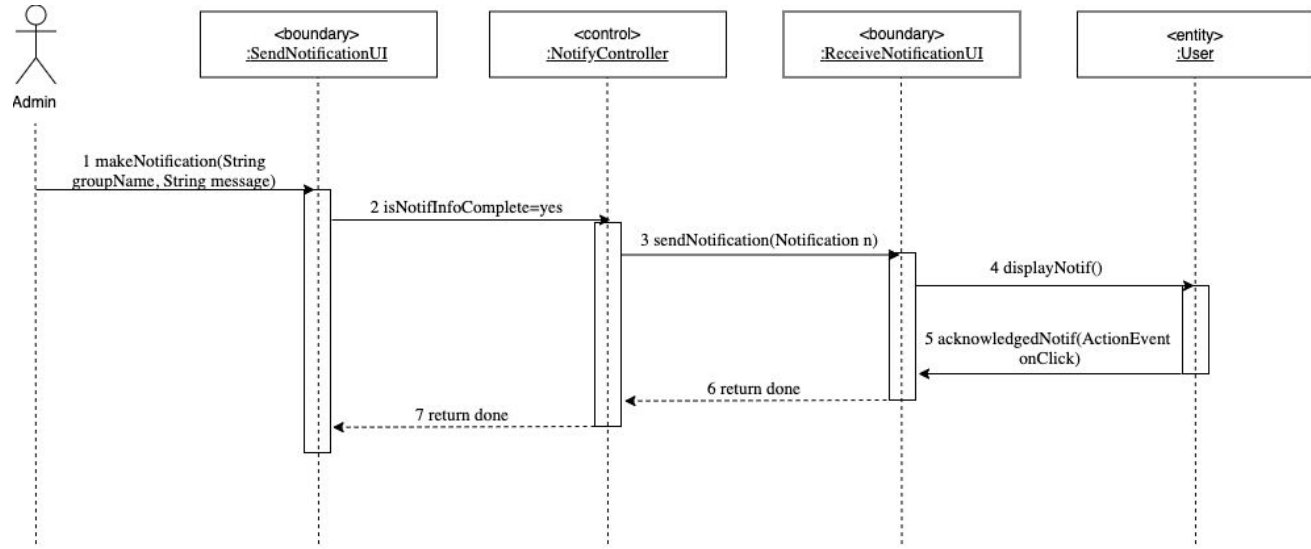
Scenario 3: User's notification request is incomplete.



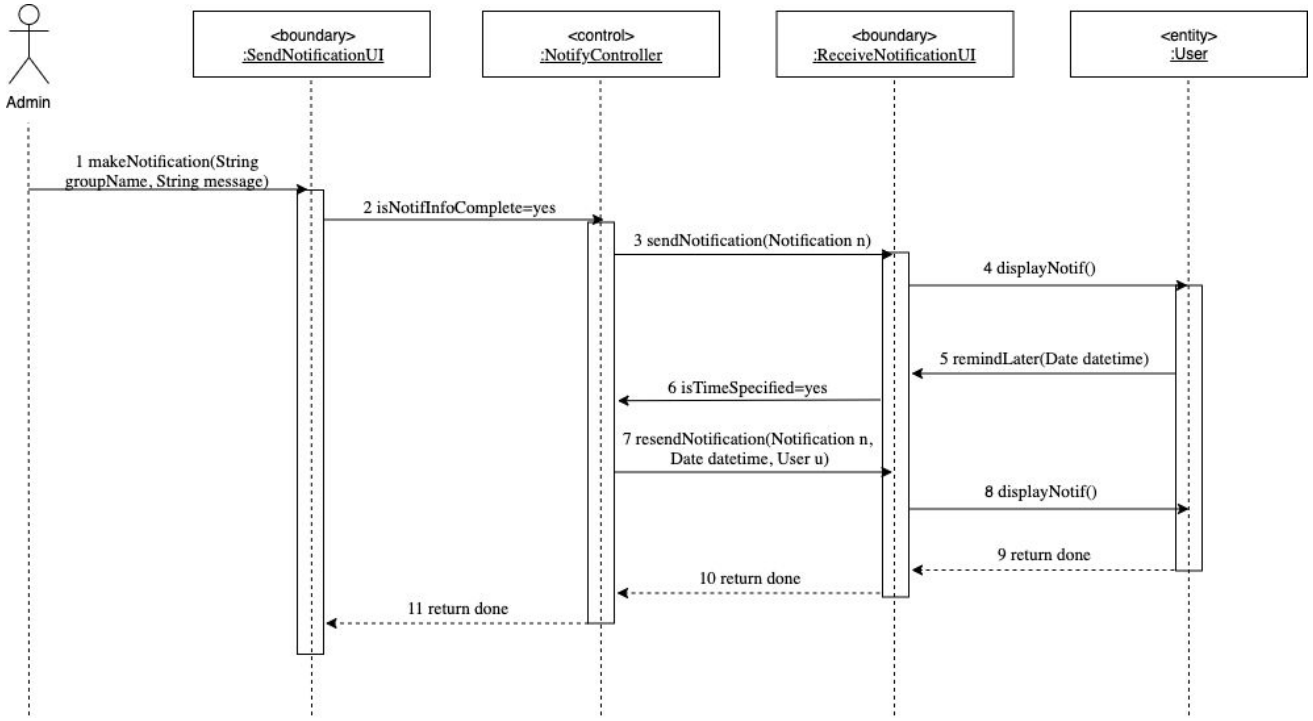
***Use-Case Name*:** 6.0 Notify

Description: The use-case is used for sending relevant notifications to users who are subscribed to particular classes and organizations. Notifications that are pushed to the users are pre-approved by DCS Codex administrators through the Request Notification use-case (Use-Case 4.0).

Scenario 1: (Basic Flow) Admin sends and user receives and acknowledged the notification.



Scenario 2: Admin sends notification and user receives notification and picks remind later.



Scenario 3: Admin sends and user receives but did not acknowledge the notification.