

sked-it

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:

Dela Sierra, Joshua Joseph Riki V.
Garcia, Patric Charles
Granda, Justin Tristan Gabriel R.

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



This work is licensed under a [Creative Commons Attribution-ShareAlike 4.0 International License](https://creativecommons.org/licenses/by-sa/4.0/).

Unique Reference:

The documents are stored in the

<https://github.com/jvdelaSierra/sked-it/tree/master/02-Requirements%20Engineering/Project%20Deliverables>
referenced 3-sked-it-Analysis Model.pdf..

Purpose:

The purpose of this document is to formalize the Analysis Method of sked-it by deriving it from the Use Case Model and Specifications

Audience:

The target audience of this document includes the developers of the sked-it project and any person who wishes to learn more about the project.

Revision Control:

<i>Revision Date</i>	<i>Person Responsible</i>	<i>Version Number</i>	<i>Modification</i>
09/28/2019	Joshua V. Dela Sierra	1.0	Prepared Initial Document
10/01/2019	Joshua V. Dela Sierra	1.1	Added Boundary Classes
10/02/2019	Joshua V. Dela Sierra	1.2	Added Entity Classes
10/02/2019	Justin Tristan Gabriel R. Granda	1.3	Added Control Classes
10/02/2019	Justin Tristan Gabriel R. Granda	2.0	Updated Boundary Class Descriptions Added Initial Data to Behavioral Model

System Name: Sked-it

Description: Sked-it is an app that we are proposing to make to help people manage their own personal schedules while being able to set-up meetings with other people. The app aims to provide users with a way to make a “routine” schedule which they could update if they have other things to do. One of the main features that we will implement is a Group Scheduling system where people in the same group will be allowed to view a portion of another person’s schedule (controlled by the owner of the schedule) within the same group so that the people involved in the group can easily set-up their meetings based on their common free time. A person will then be notified of their schedule with a reminder sent to their phones.

Analysis Model:

Place here the class diagram of the analysis model.

Boundary Classes:

Class Name	Description
MonthCalendarUI	<p>This is the interface of the end user to the system whenever he/she needs to manage his/her monthly calendar.</p> <p>Responsibilities:</p> <pre>public void addEventCalendar(boolean addOrDelete, String date, String time, String description, String groupPrivacy) public void updateEventCalendar(String eventID, String date, String time, String description, String groupPrivacy) public void deleteEventCalendar(String eventID, boolean addOrDelete)</pre>
WeeklyTemplateCalendarUI	<p>This is the interface of the end user to the system whenever he/she needs to manage his/her weekly template calendar.</p> <p>Responsibilities:</p> <pre>public void addEventCalendar(boolean addOrDelete, String date, String time, String description, String groupPrivacy) public void updateEventCalendar(String eventID, String date, String time, String description, String groupPrivacy) public void deleteEventCalendar(String eventID, boolean addOrDelete)</pre>
WeeklyCalendarUI	<p>This is the interface of the end user to the system whenever he/she needs to manage his/her weekly calendar.</p> <p>Responsibilities:</p> <pre>public void addEventCalendar(boolean addOrDelete, String date, String time, String description, String groupPrivacy) public void updateEventCalendar(String eventID, String date, String time, String description, String groupPrivacy) public void deleteEventCalendar(String eventID, boolean addOrDelete)</pre>
DailyCalendarUI	<p>This is the interface of the end user to the system whenever he/she needs to manage his/her daily calendar.</p> <p>Responsibilities:</p> <pre>public void addEventCalendar(boolean addOrDelete, String date, String time, String description, String groupPrivacy) public void updateEventCalendar(String eventID, String date, String time, String description, String groupPrivacy) public void deleteEventCalendar(String eventID, boolean addOrDelete)</pre>
LoginUI	<p>This is the interface of the end user to the system whenever he/she needs to enter the system.</p> <p>Responsibility:</p> <pre>public String enterLoginDetails(String email, String password)</pre>
CreateProfileUI	<p>This is the interface of the end user to the system whenever he/she needs to create an account for the system.</p> <p>Responsibility:</p> <pre>public String enterProfileDetails(String email, String password, boolean agreeOnTerms)</pre>
ForgetPasswordUI	<p>This is the interface of the end user to the system whenever he/she forgot the password for the system.</p> <p>Responsibility:</p> <pre>public String enterDetails(String email)</pre>
LinkScheduleUI	<p>This is the interface of the end user to the system whenever he/she needs to link a</p>

	<p>schedule to a group.</p> <p>Responsibility:</p> <p>public String enterDetails(String groupCalendarID, string eventID)</p>
ConfirmScheduleIO	<p>This is the interface of the end user to the system whenever he/she needs to link a schedule to a group.</p> <p>Responsibility:</p> <p>public String confirmDetails(String groupCalendarID, string eventID)</p>

Control Classes:

Class Name	Description
ManageCalendarController	<p>This controller handles creating, updating and deleting calendars for the user.</p> <p>Responsibilities:</p> <pre>public void addCalendar(boolean addOrDelete, String date, String time, String description, String groupPrivacy) public void updateCalendar(String eventID, String date, String time, String description, String groupPrivacy) public void deleteCalendar(String eventID, boolean addOrDelete)</pre>
NewAccountController	<p>This controller handles the creation of new user accounts.</p> <p>Responsibilities:</p> <pre>public String createProfile(String email, String password, boolean agreeOnTerms)</pre>
AccountController	<p>This controller is responsible for maintaining existing user accounts, including logging in and managing forgotten passwords.</p> <p>Responsibilities:</p> <pre>public String loginSystem(String email, String password) public void RenewPassword(String email)</pre>
LinkScheduleController	<p>This controller handles schedule linking (i.e. finding common free time between schedules).</p> <p>Responsibilities:</p> <pre>public String LinkSchedules(String eventID_1, String eventID_2)</pre>

Entity Classes:

Class Name	Description
Individual	<p>This is the entity class Individual, which contains the data about an individual user.</p> <p>Attributes: private String userID private String email private String password private Schedule schedules</p>
Schedule	<p>This is the entity class Schedule, which contains the data about an individual user.</p> <p>Attributes: private String eventID private String date private String time private String description private String groupPrivacy</p>
Group	<p>This is the entity class Group, which contains the data about groups.</p> <p>Attributes: private Individual individual private String nameOfLeader</p>

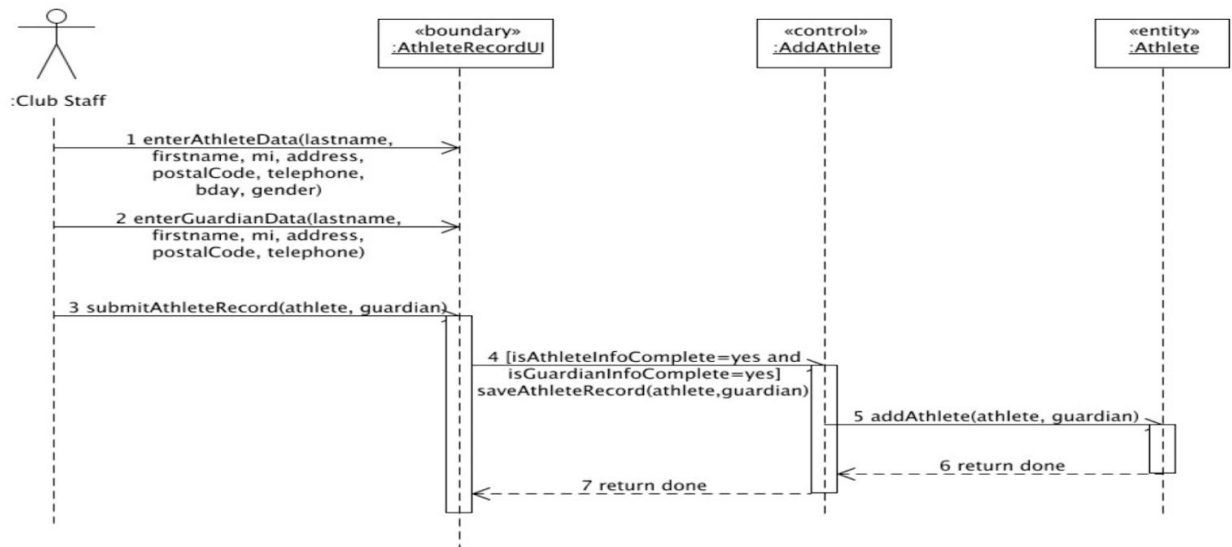
Behavioral Model:

Use-Case Name: 1.0 Notify Individuals

Description:

Individual users can receive notifications pertaining to their schedule. This notification may come from a group that they are a part of. The user can also be notified if they have been added to a new group.

Scenario 1: Successful Notification

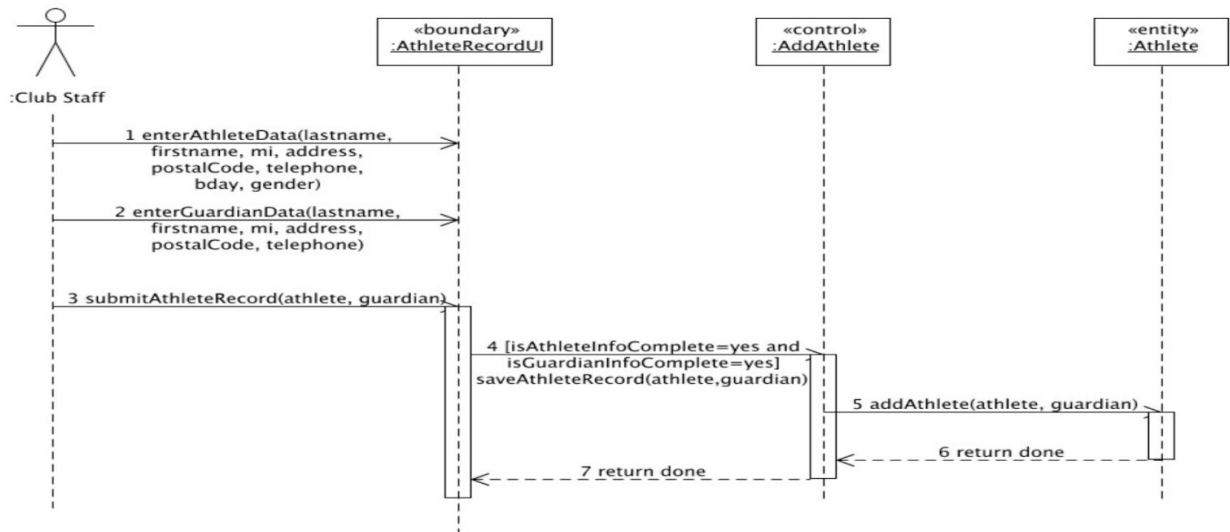


Use-Case Name: 1.1 Create Profile

Description:

Individual users are able to create and upload their profile. They simply input their emails and their password for their profile. The user will then receive a confirmation through their email and must confirm it.

Scenario 1: Invalid email



Scenario 2: Email already used

[diagram]

Scenario 3: Successful Creation of Profile

[diagram]

Scenario 4: Invalid Password

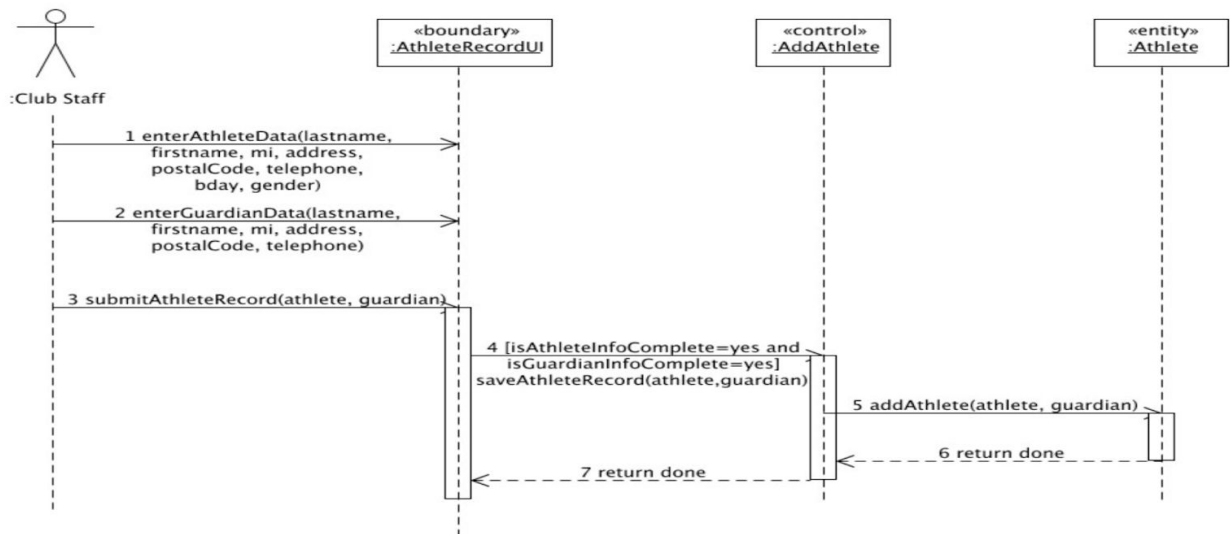
[diagram]

Use-Case Name: 1.2 Create Group

Description:

Individual users can create their own groups within the app. The group creator can add members (individual users) to this group. All added members are notified when the group is created.

Scenario 1: Successful Creation of Group

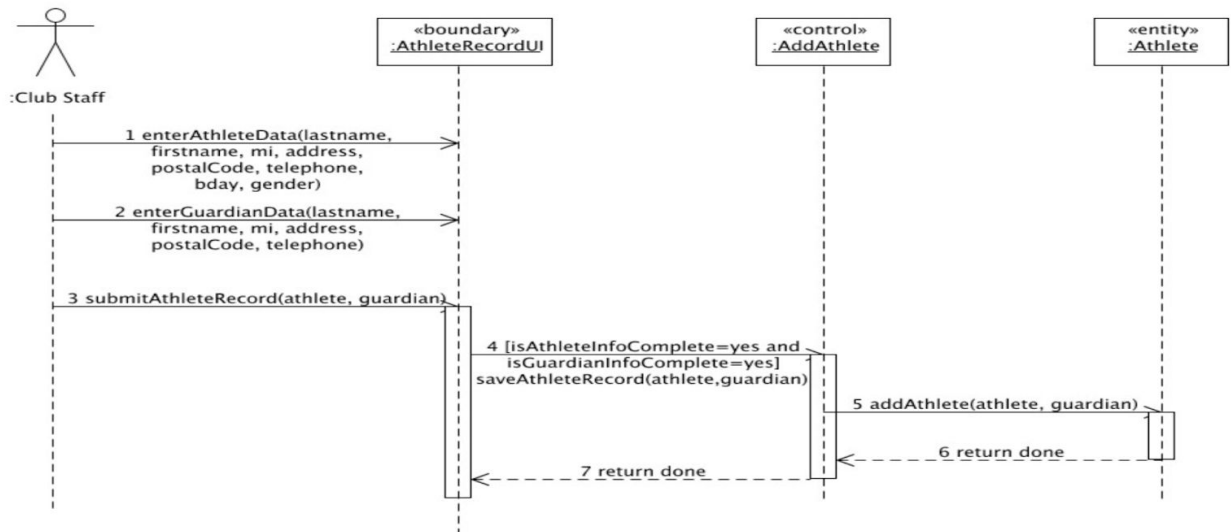


Use-Case Name: 1.3 Create Schedule

Description:

Individual users are able to create and upload their schedule/s. They simply input the times per day of the week that they are free/busy. The user can create a general schedule and have different schedules for different groups.

Scenario 1: Successful Creation of Routine Schedule



Scenario 2: Successful Creation of Group Free Time Schedule

[diagram]

Scenario 3: Successful Creation of Date-specific Schedule

[diagram]

Scenario 4: Wrong Time Intervals

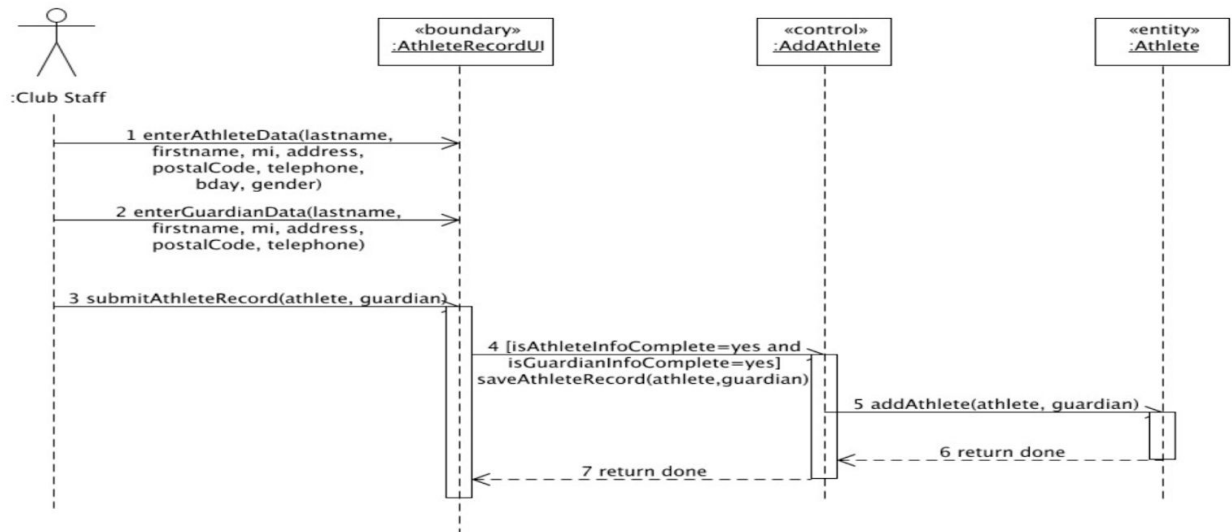
[diagram]

Use-Case Name: 2.0 Appoint Leaders

Description:

The members can assign leaders for their group. The leaders are in charge of managing the schedules/activities of their group. There can be one or many leaders per group.

Scenario 1: Everyone agrees and a member is appointed



Scenario 2: Most of the members agree and a member is appointed

[diagram]

Scenario 3: Most of the members disagree and the process is repeated

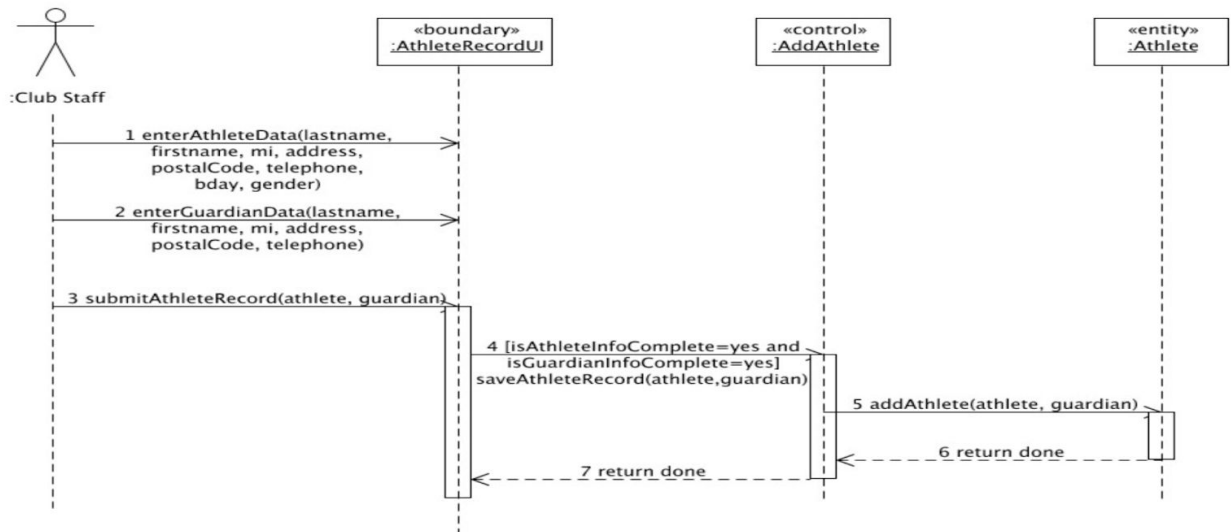
[diagram]

Use-Case Name: 2.1 Notify Group

Description:

Members can notify each other within the group. Each notification will be concerning the time of the meeting. It can also be about the venue of the meeting

Scenario 1: Successful Notification

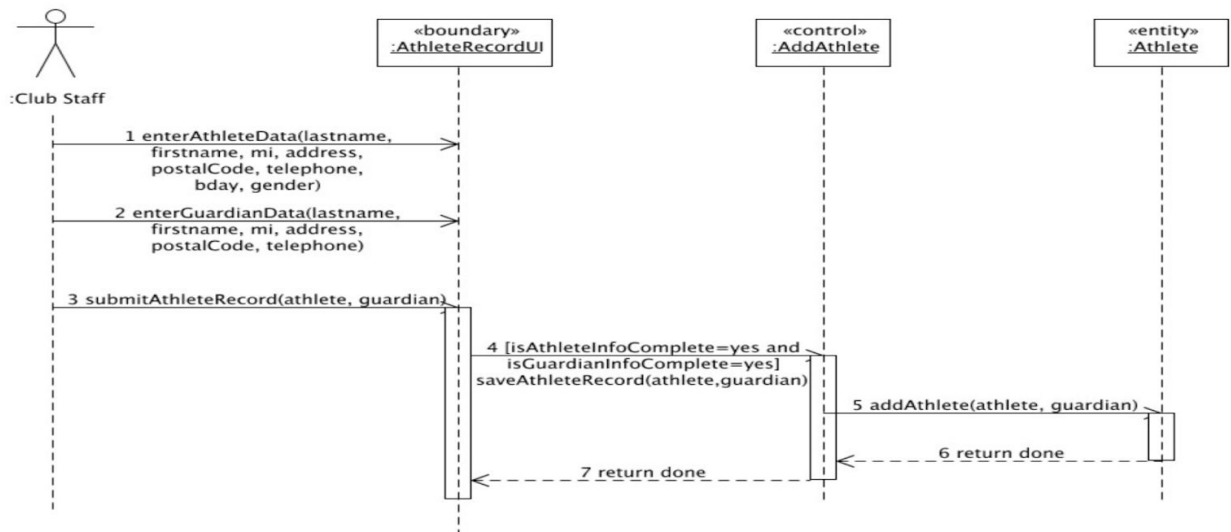


Use-Case Name: 2.2 Confirmation of Common Schedule

Description:

The group can confirm their common schedule for a certain activity/event. This requires the Link Schedule functionality of the system. An automatic notification (notify group) will be queued shortly.

Scenario 1: Everyone Agrees and a time is confirmed



Scenario 2: Some of the members are not available but have confirmed

[diagram]

Scenario 3: Some of the members have confirmed but redacted their confirmation after

[diagram]

Scenario 4: Some of the members have confirmed but overrides the schedule with a schedule of a higher priority

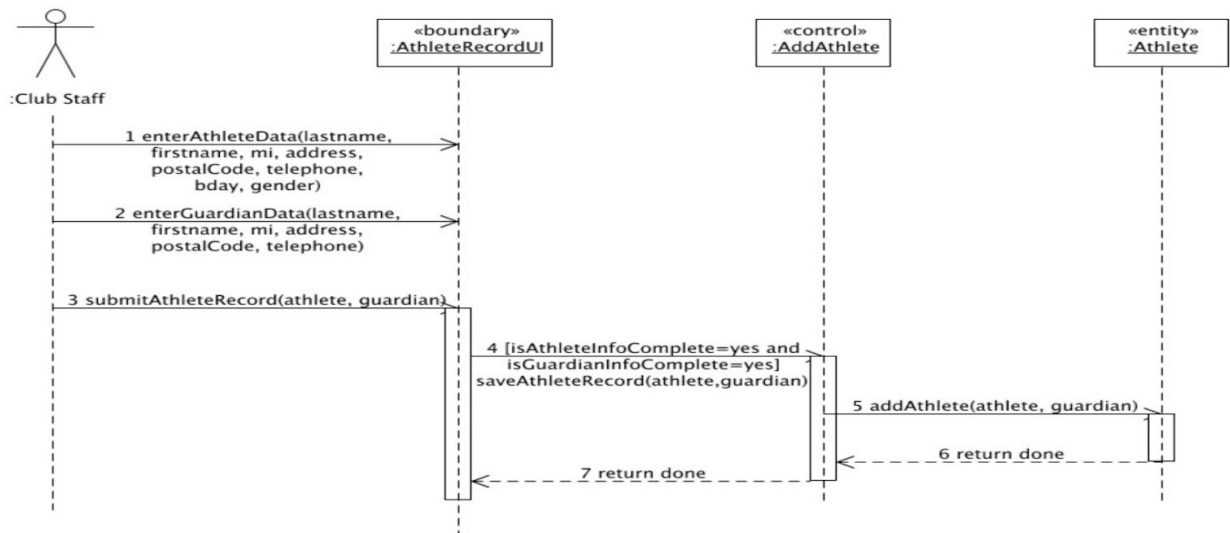
[diagram]

Use-Case Name: 3.0 Link Schedules

Description:

This is the system's main functionality. Given some schedules, it returns the common free time found between them. This can be done between individual users or within a group.

Scenario 1: Common free time is successfully found between individual users



Scenario 2: Common free time is successfully found between group members

[diagram]

Scenario 3: No common free time was found between the members

[diagram]