This evolution introduces more events. First, users can create events involving other users. Second, users can create events that persist until they are done (sort of like a “todo list”), and schedule time to work on the highest priority of those events. The requirements for this evolution are:

1. Server
   (a) Your software must have a server that supports an arbitrary number of users.
   (b) Each user must create an account before using the system.
   (c) Passwords must be kept in a secure manner (e.g., salted + hashed)
   (d) All communication between the clients and server must be encrypted
   (e) The server must maintain state in a persistent fashion.

2. Client: Basic Calendar Functionality
   (a) A user shall be able to create events on his/her calendar, including the date/time of the event, the name of the event, and other information they may wish to include.
   (b) A user shall be able to indicate that an event repeats at regular intervals (weekly, daily, monthly, etc) for a specified period.
   (c) The client shall display the user’s calendar with its events in a intuitive manner.
   (d) The user shall be able to elect to have the system send him/her an e-mail reminder about an event at a time of his/her choosing.
   (e) A user shall be able to control which other user’s can see the events on his/her calendar
      i. By default, events shall be private to the user (not visible to others)
      ii. A user shall be able to create named groups of other users (e.g., “Coworkers”, “Friends”, “Family”).
      iii. A user shall be able to change the level of access for any combination of users and/or groups to any of his/her calendar events. The levels of access shall be Private (no access/no visibility), Busy Only (shows the time as occupied, but no details), See All (show all information), Modify (can see and modify the event).
      iv. The user shall be able to enter multiple rules, which take precedence in the order that they appear. For example, if “Frodo” is in the group “Hobbits”, the user shall be able to enter a rule for Frodo first, then a rule for the Hobbits group after it, with the former applying to Frodo, and the later applying to other members of Hobbits.
      v. The user shall be able to re-order or delete existing rules for an event.
      vi. You may (but do not have to) organize events into multiple logical calendars, and control the access of events at this granularity if you wish.
      vii. Events shared from other users’ calendars shall be displayed in such a way that they can be easily distinguished from the user’s own events.
   (f) A user shall be able to send an event creation request to any combination of other users/groups’ calendars.
      i. Each user who receives the event request shall be able to confirm, decline, or be removed from the event.
ii. If user confirms for an event, the event appears on his/her calendar normally.

iii. If a user declines for an event, the event either does not appear, or appears in a distinct (less obtrusive—e.g., grayed out) fashion. However, the user will receive any change requests.

iv. If a user removes him/herself from an event, the event will not appear at all, and the user will not receive any notifications of changes.

v. An event for which the user has not responded shall appear on the calendar in a distinctive fashion.

vi. The originator of the event shall be able to determine who has confirmed/declined/removed themselves from an event.

vii. The default permissions for such an event shall be that anyone who received it can modify it.

viii. If someone other than the originator of the event attempts to modify its date/time/information, the request shall be sent to the originator for his/her approval.

ix. If the originator of the event modifies it (either directly, or by approving another’s request), that change shall be sent to all users participating in the event. They again must confirm or decline.

x. Each user shall be able to set his/her own notifications on e-mail reminders for the event (without the involvement of other users).

3. Present-until-done Events

(a) The calendar shall support the concept of a “present until done” (PUD) event—that is, an item on the calendar that is not fixed to any particular time, but rather a persistent reminder until it is done.

(b) The user shall be able to enter a priority for each PUD event.

(c) The system shall have a way to display a prioritized list of PUD events.

(d) The system shall have a way for the user to indicate that a PUD event is completed.

(e) The user shall be able to specify that a PUD event can recur after it is completed (e.g., “Call Your Grandparents” might have be set to “recur weekly”). In such a case, the recurrence occurs after the user indicates that the first occurrence is completed.

(f) The user shall be able to enter an amount of time, indicating the smallest amount of time that this task may be meaningful worked on.

(g) The user shall be able to enter calendar events which reserve time to work on PUD events.

i. These events will display the highest priority event which fits into the time allocated to them.

ii. If the user marks the currently displayed event as done during, the displayed event should be updated to the next appropriate event.

iii. If no appropriate events are available, the block on the calendar should indicate the situation.
iv. The time allocated to “work on PUDs” should not permit “Modify” access for other users. If the user shares it so that it shows information, you should only show that it is reserved as time to work on PUD events.

(h) You may (but do not have to) allow a user to share his/her PUD event list (or a subset of it) with other users.

(i) The user shall be able to specify that email reminders be sent to him/her for a PUD until it is completed. The user should be able to specify the occurrence of the reminders in terms of a start time (relative to the creation, or recurrence) of the PUD event, and a frequency after the reminders being (e.g., 3 days after creation, remind me every other day).