This evolution adds a few final features to your calendar, mostly expanding on the capabilities of existing features. The rules for evolution 4 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 an 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.
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.

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).

A user shall be able to add an expiry time for a PUD, at which point it is automatically marked as done.

The user shall be able to specify “priority escalation” as the deadline approaches. For example, a user might specify an initial priority of 5, but that the priority should increase by 2 each day once within a week of the deadline.

4. Slot signups

A user shall be able to enter a special type of event (a “signup block”), allowing other users (or groups of users) to sign up for meeting (or other) slots during that time (think advising signups).

The user creating the signup block shall be able to specify one or more signup blocks, and set a maximum number of slots that any particular user may signup for during those blocks (cumulatively—e.g., advising may be M 3-4, T 1-3, F9-11, and a user may only be allowed one slot during any of those times).

The user creating the signup block shall be able to specify the minimum/maximum duration of a slot that a user may signup for (e.g., 15 to 30 minutes).

A user may only signup for slots the are “aligned” with respect to the minimum duration—i.e., the difference between the start time of the slot and the start time of the block must be a multiple of the minimum slot duration.

The signup block shall be displayed in a distinct fashion to users who are eligible to signup for it. Within this block, the users should be able to see which times are already taken (but not who has taken them), as well as any slot(s) that they have signed up for.

A user should be able to delete or change his/her signup.

Preference-based signup

When creating a signup block, the user creating the event may opt to allow preference-based signup (rather than “first come first serve”). Such a scheme lets users specify which slot(s) they are available for and their preferences. It then allows the originator to schedule people into slots in whatever way is best for that situation.

This feature is an option when creating a signup block.

You may (but do not have to) require that slots are fixed size (minimum = maximum) for this type of signup block.

The user creating the event may optionally specify that a PUD (where the event is to signup) should be created for each user receiving it, and may specify a priority for that PUD. The PUD shall expire at the time that scheduling is determined. This PUD should be marked as “done” when the user signs up for one or more slots.
v. Each user receiving the request shall be able to enter his/her preferences for slots. A user may enter an availability for any number of slots, and express a preference amongst the slots selected.

vi. Until the scheduling is resolved, the slots the user has selected shall appear on his/her calendar in a distinctive manner.

vii. The user may change his/her selections until the scheduling is resolved.

viii. At the time specified, the system shall allow the user which initiated the event to resolve the scheduling—assigning each slot to a person who indicate they were available for it.

A. The system shall present the user with an initial possible assignment, which maximally fills slots, while optimizing for preferences in a reasonable way.

B. The user shall then be able to alter the assignment, moving people between slots that they indicated were acceptable.

ix. Once assignment is complete, the assigned slots shall appear on each user’s respective calendars as normal events.

5. Find free times

(a) A user specify a set of users and/or groups, and determine who has conflicts at what times.

(b) In creating the request, the user should be able to specify whether he/she wants to know for the purposes of a one-time event, or a recurring event. In the later case, the user should be able to enter a time range to consider. The user should also specify how large of a time slot he/she needs (30 minutes, an hour, etc), as well as what time ranges are valid (e.g., M–F 9-5).

(c) In evaluating what conflicts occur, only events which are visible to the requesting user should be considered.

(d) The system should show a summary of the conflicts (e.g., how many people have a conflict with each time slot), with an ability to see more details (who has the conflict, what it is—if that information is visible to the requesting user, if the conflict is a “one-off” when considering a recurring event, etc).

(e) The user should be able to easily initiate an event request to the group he/she searched on at from the display of the results.

6. The system shall provide a way to enter many events en masse in a textual format (e.g., a large text box, with one event per line.

(a) This method of creating events should allow creation of any of the event types (normal, multiple users, PUD, or sign up blocks (including preference-based)).

(b) The specifics of the input format are up to you, but it should be reasonably human readable (i.e., you may not require a binary encoding of your internal object representation).

(c) If there is a problem parsing the input, your system shall indicate which line(s) the error(s) occurred, on, as well as provide helpful message(s) about the problem to the user. In the event of an error, the system should retain the text, allowing the user to edit it—correcting the mistakes and resubmitting.
(d) Acceptance of the events is all or nothing: if there is any parse error in one description, no events will be created.

7. Get my schedule by email

(a) A user shall be able to request an emailed copy of his/her schedule.

(b) When making this request, the user may specify a date range, and whether he/she wants to receive the schedule in plain text or graphical format.

(c) What graphical format(s) you support are up to you, as long as you support at least one widely recognized format (e.g., pdf, or png).