NETime Planner

Project Authors:

- Kevin DeMars
- Trenton Strickland
- Eric Jaroszewski
- Joshua Kanagasabai
- Max Cadeddu
- Samuel Kim

Project Vision

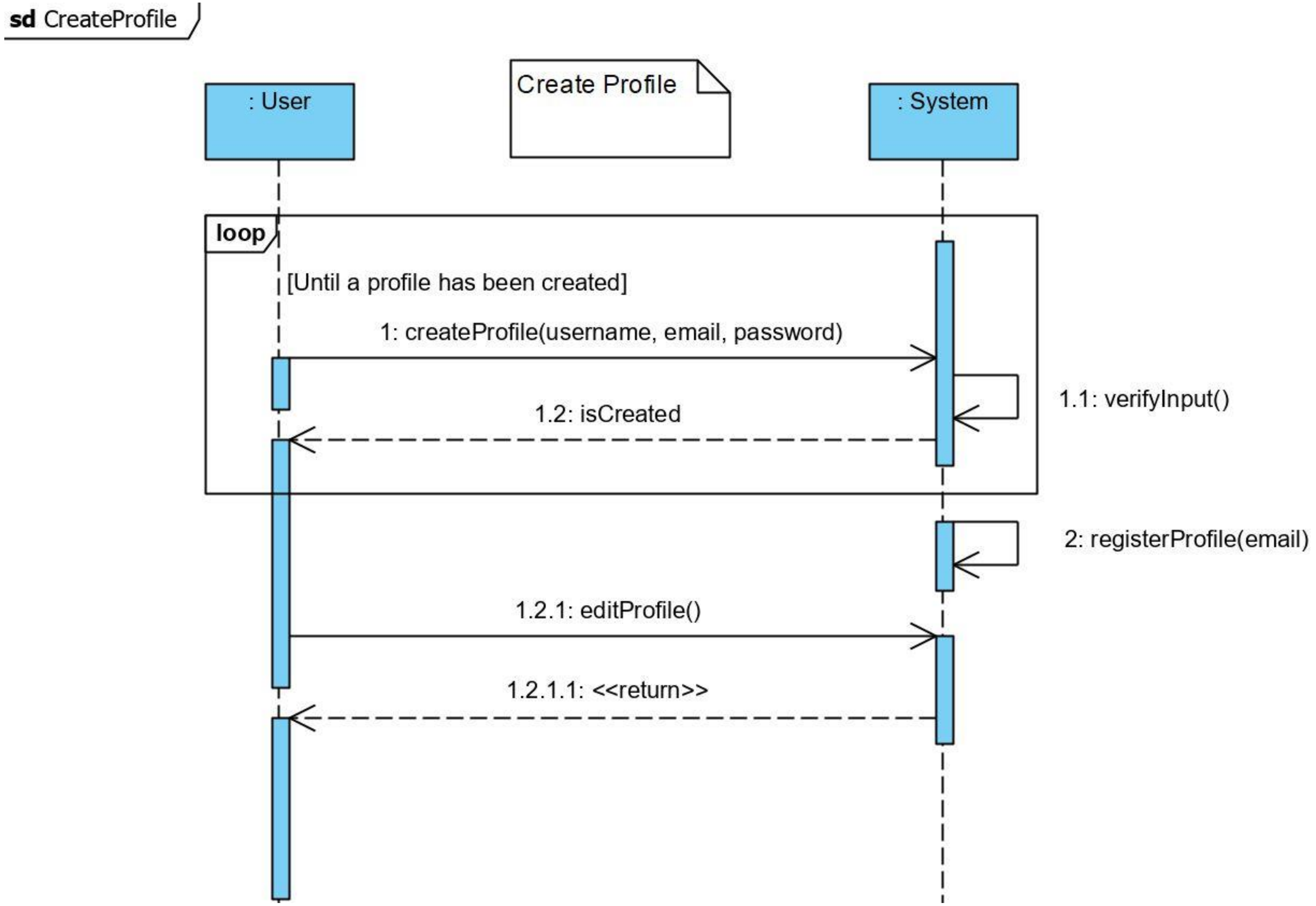
- NETime (pronounced "any time") Planner is a social planner focused on students.
- In addition to the basic actions of managing personal deadlines and activities, it supports sharing schedules and collaborating on schedules with groups. Our goal is to make managing your schedule simple and convenient, with helpful features such as automatically appending events to your next free time.
- The features don't stop as soon as you've entered your info into your schedule. NETime Planner provides features to analyze your schedule, giving you feedback on what percentage of time you have left for deadlines, how much free time you have each week, and more.

Highlighted Features

User Profile

- Create Profile
- Edit Profile
- View Schedule

SSD

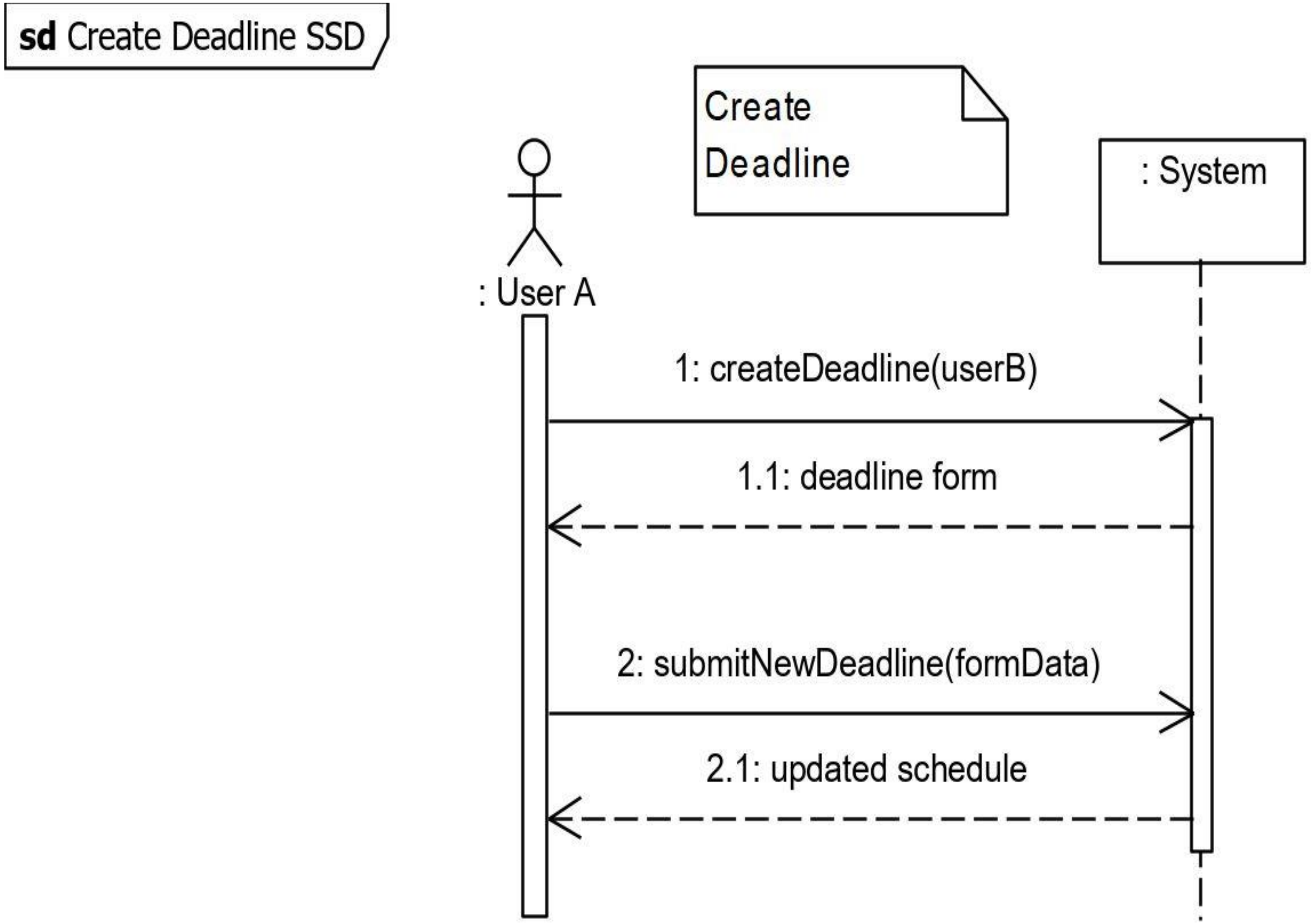

sd CreateProfile
Create Profile
: User
: System
loop
[Until a profile has been created]
1: createProfile(username, email, password)
1.1: verifyInput()
1.2: isCreated
2: registerProfile(email)
1.2.1: editProfile()
1.2.1.1: <<return>>

Highlighted Features

Deadlines & Activities

- Create
- Edit
- Remove

SSD

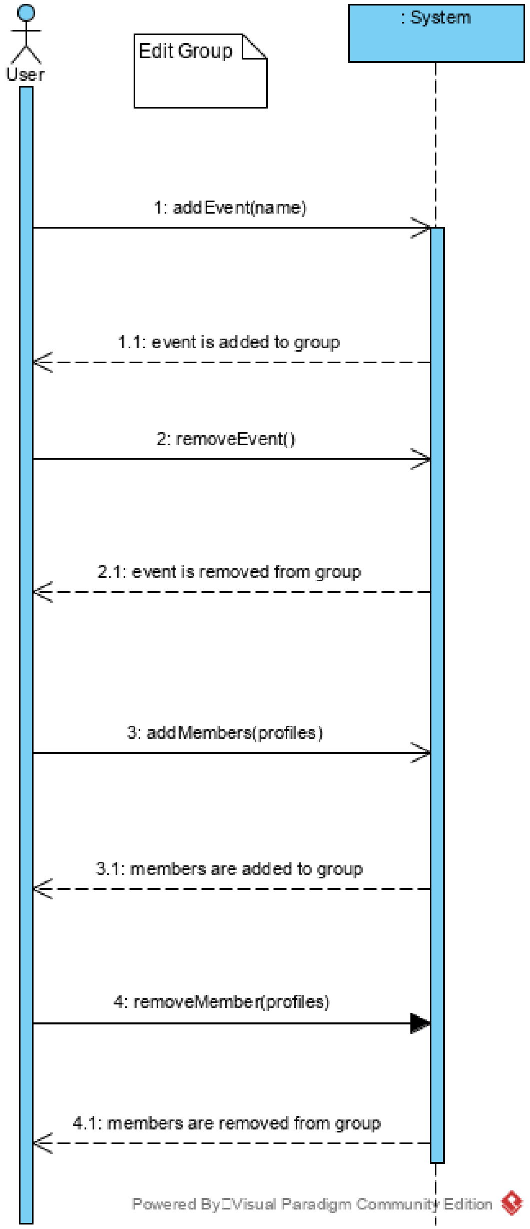

sd Create Deadline SSD
Create Deadline
: User A
: System
1: createDeadline(userB)
1.1: deadline form
2: submitNewDeadline(formData)
2.1: updated schedule

Highlighted Features

Groups

- Create
- Edit
- Delete
- Edit Permissions

SSD



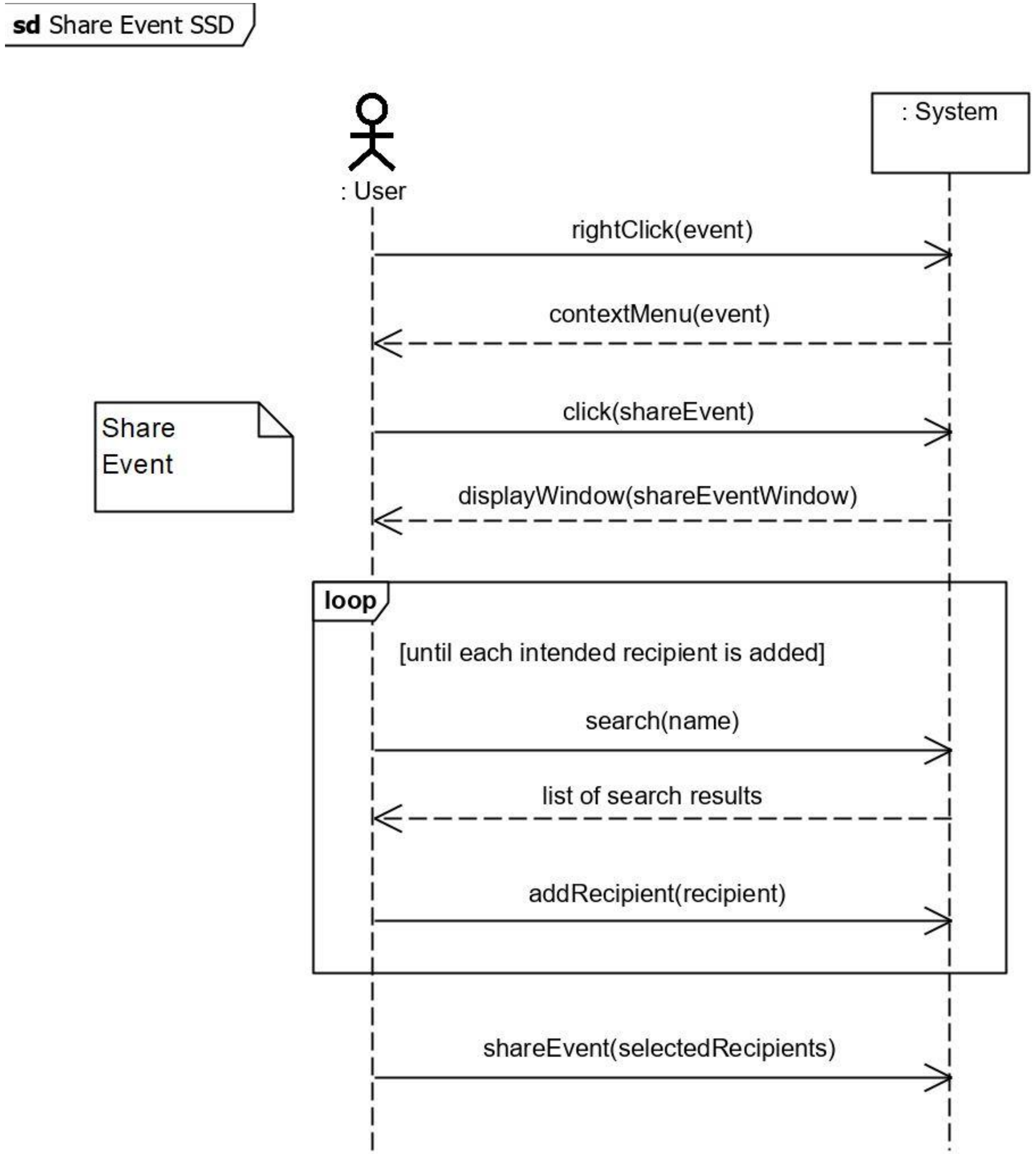
User
Edit Group
: System
1: addEvent(name)
1.1: event is added to group
2: removeEvent()
2.1: event is removed from group
3: addMembers(profiles)
3.1: members are added to group
4: removeMember(profiles)
4.1: members are removed from group
Powered By Visual Paradigm Community Edition

Highlighted Features

<u>Able to Share</u>

  - Schedules
  - Events
- Edit Privacy Settings

SSD

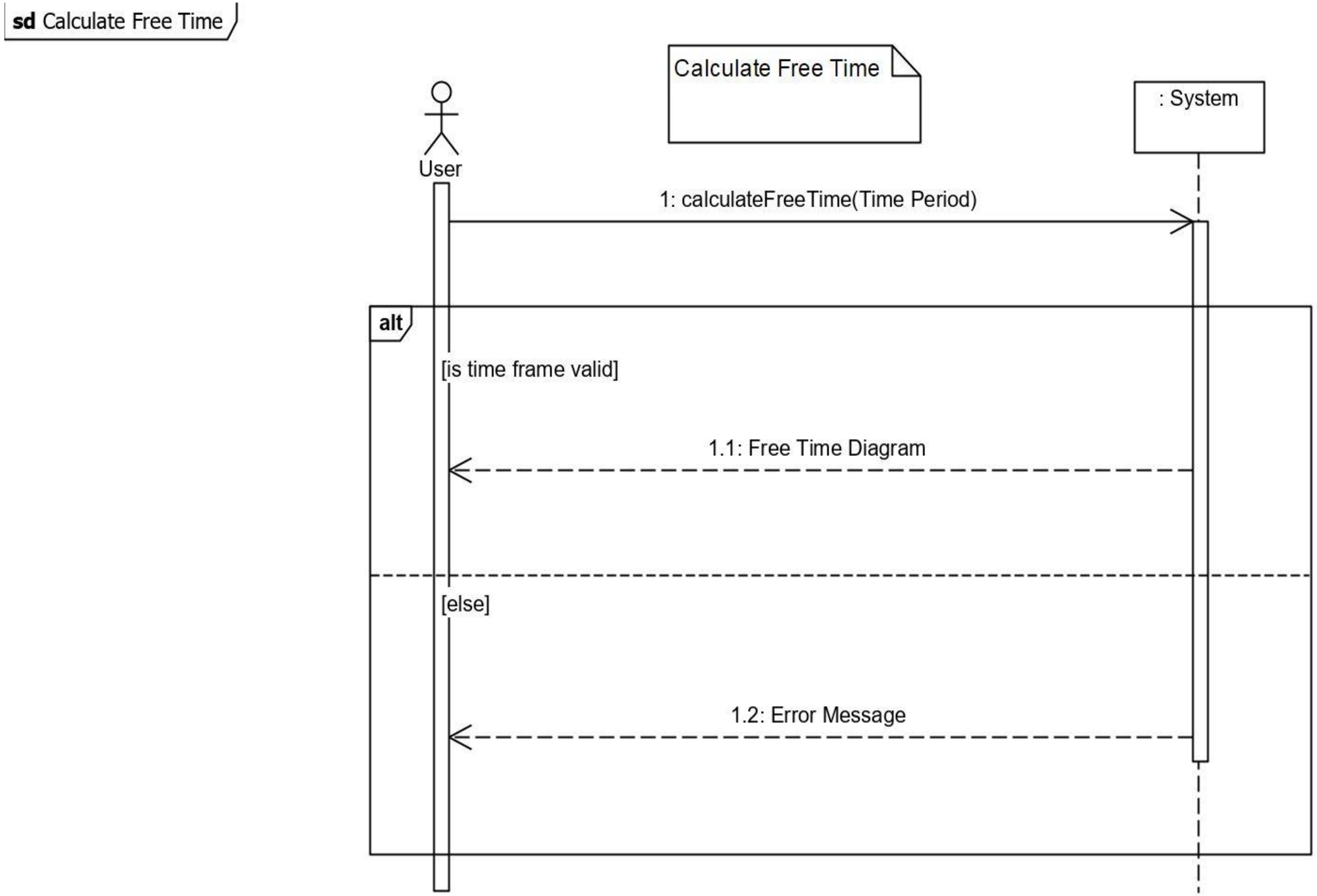

sd Share Event SSD
: User
: System
Share Event
rightClick(event)
contextMenu(event)
click(shareEvent)
displayWindow(shareEventWindow)
loop
[until each intended recipient is added]
search(name)
list of search results
addRecipient(recipient)
shareEvent(selectedRecipients)

Highlighted Features

Time Statistics & Calculations

- Calculate Free Time
- Time left on Task
- Time Spent on Task

SSD

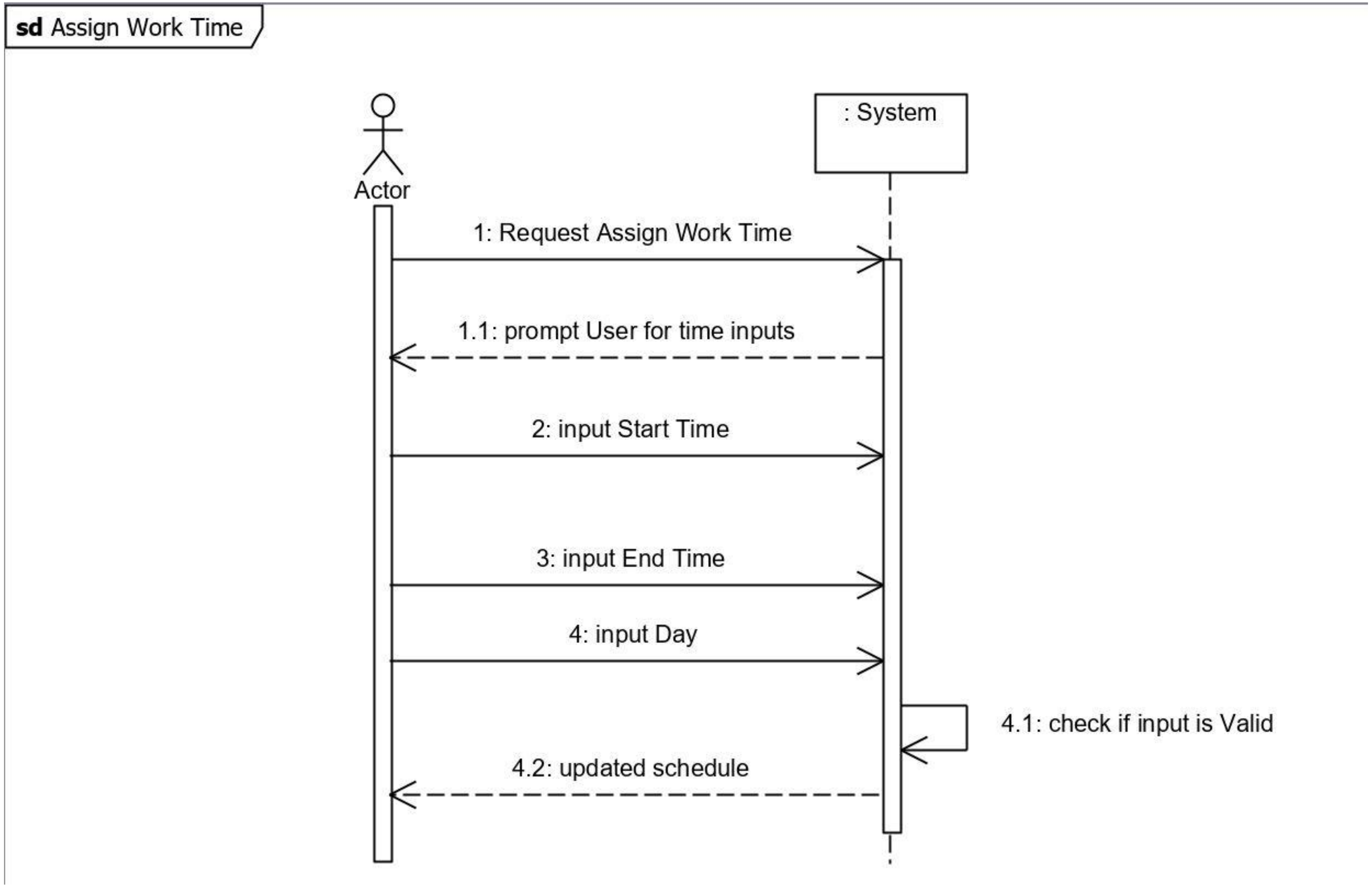

sd Calculate Free Time
Calculate Free Time
User
: System
1: calculateFreeTime(Time Period)
alt
[is time frame valid]
1.1: Free Time Diagram
[else]
1.2: Error Message

Highlighted Features

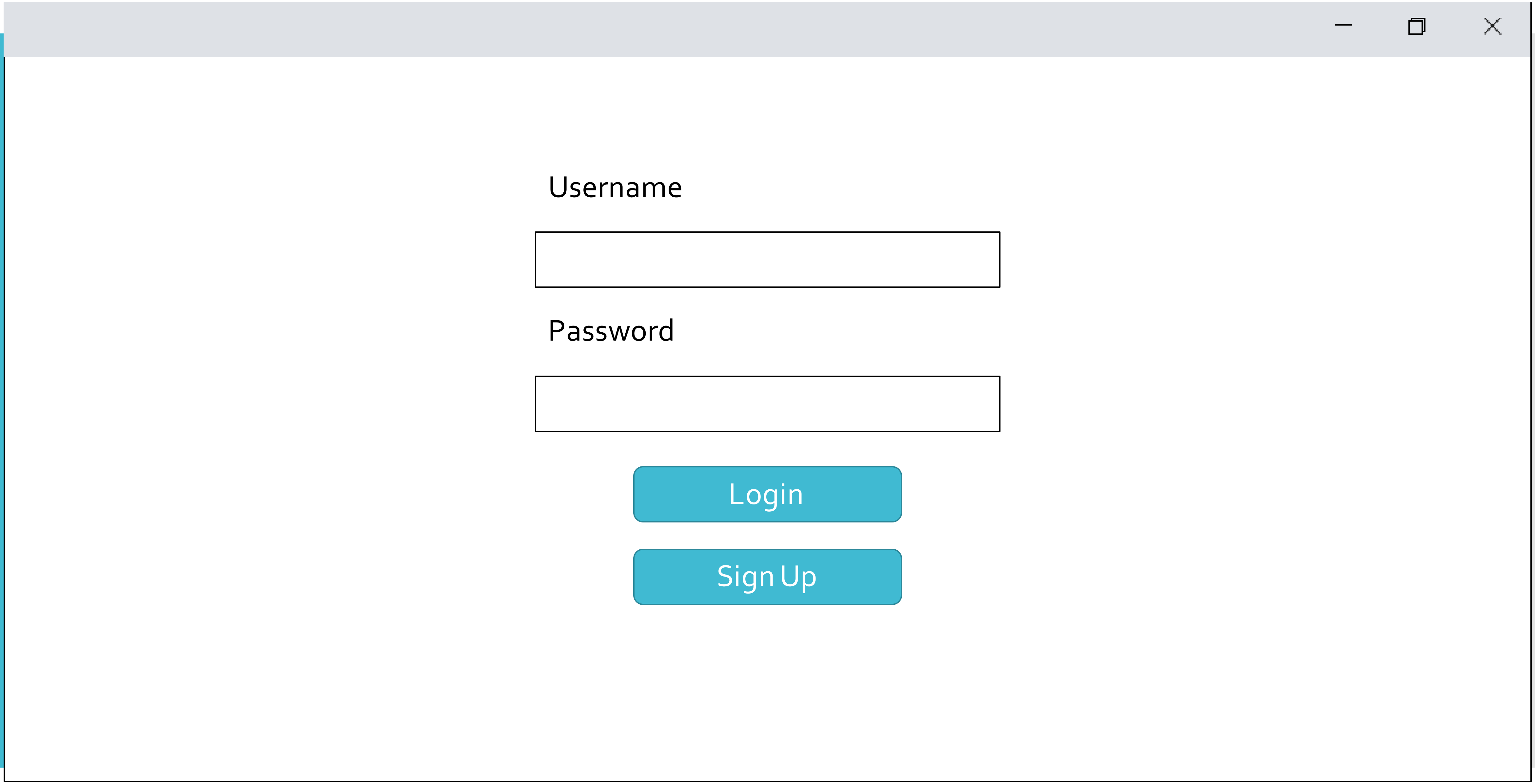
Schedule Operations

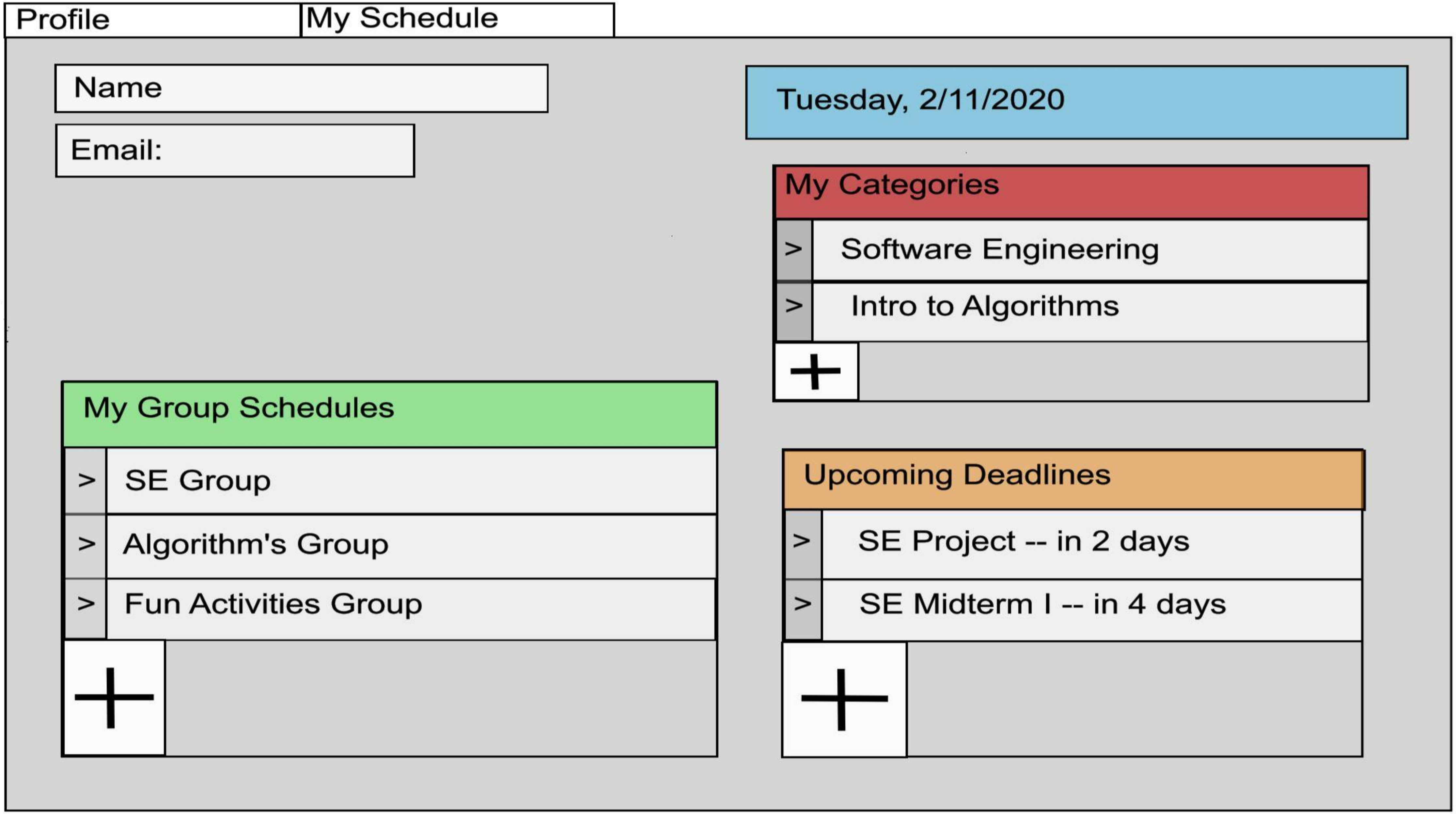
- Assign Work Time
- Auto – Append Event
- Generate To-Do List

SSD

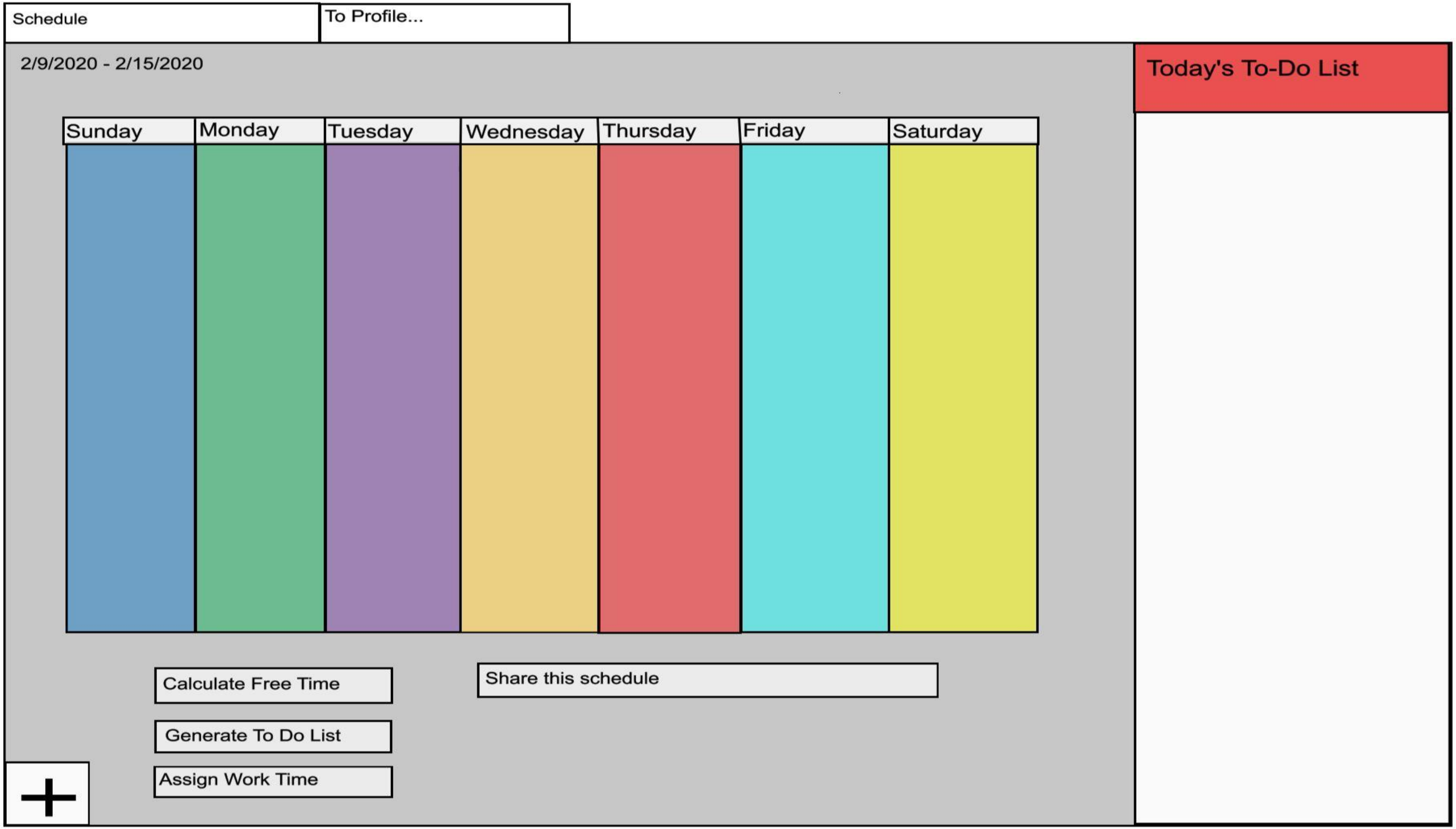

sd Assign Work Time
Actor
: System
1: Request Assign Work Time
1.1: prompt User for time inputs
2: input Start Time
3: input End Time
4: input Day
4.1: check if input is Valid
4.2: updated schedule

Wireframes

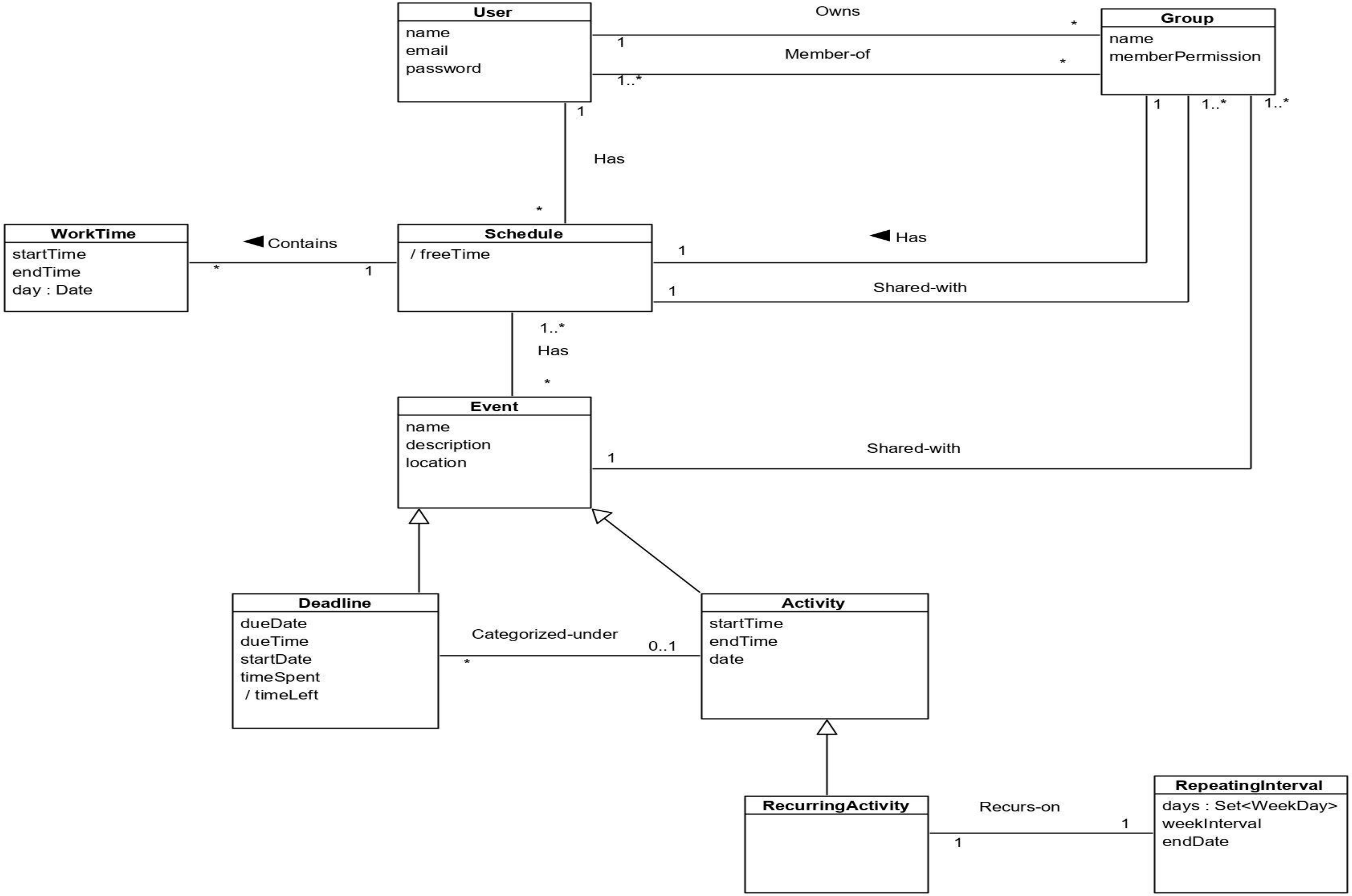

Username
Password
Login
Sign Up



Profile
My Schedule
Name
Email:
Tuesday, 2/11/2020
My Categories
> Software Engineering
> Intro to Algorithms
+
My Group Schedules
> SE Group
> Algorithm's Group
> Fun Activities Group
+
Upcoming Deadlines
> SE Project -- in 2 days
> SE Midterm I -- in 4 days
+


Schedule
To Profile...
2/9/2020 - 2/15/2020
Today's To-Do List
Sunday
Monday
Tuesday
Wednesday
Thursday
Friday
Saturday
Calculate Free Time
Share this schedule
Generate To Do List
Assign Work Time
+

Domain Model


User
name
email
password
Owns
Member-of
Group
name
memberPermission
1
*
1..*
*
1
1..*
1..*
Has
1
*
WorkTime
startTime
endTime
day : Date
◀ Contains
*
1
Schedule
/ freeTime
◀ Has
1
Shared-with
1
1..*
Has
*
Event
name
description
location
Shared-with
1
Deadline
dueDate
dueTime
startDate
timeSpent
/ timeLeft
Categorized-under
*
0..1
Activity
startTime
endTime
date
RecurringActivity
Recurs-on
1
1
RepeatingInterval
days : Set<WeekDay>
weekInterval
endDate

Website & Questions?

Website	https://kevindemars.github.io/netime-planner
GitHub Repo:	https://github.com/KevinDeMars/netime-planner/issues

NETime Planner Time Card

Name	Hours Spent
Joshua Kanagasabai	11
Kevin DeMars	16.5
Trenton Strickland	10
Samuel Kim	11
Eric Jaroszewski	9
Max Caddeddu	11

NETime Planner

Feb 11, 2020

CSI 3471

http://

Project manager	
Project dates	Jan 28, 2020 - Apr 28, 2020
Completion	0%
Tasks	32
Resources	6

Gantt Diagram for first iteration of software project.

Tasks

Name	Begin date	End date
Iteration 1	1/28/20	2/11/20
Decide on project	1/28/20	1/30/20
Group Meeting 1: Describe Use Cases	1/31/20	1/31/20
FD Use Cases	2/3/20	2/4/20
Group Meeting 2: Go over Use Cases	2/5/20	2/5/20
SSD	2/6/20	2/6/20
Group Meeting 3	2/7/20	2/7/20
Edit Use Cases	2/7/20	2/7/20
Domain Model	2/7/20	2/7/20
Wireframes	2/7/20	2/7/20
Create Presentation	2/10/20	2/10/20
Operation Contracts	2/10/20	2/10/20
Website	1/31/20	2/10/20
Project Vision	1/31/20	1/31/20
Issue Tracking Link	2/5/20	2/5/20
Project Download link	2/10/20	2/10/20
Team Time Cards	2/10/20	2/10/20
Give Presentation	2/11/20	2/11/20
Iteration 2	2/14/20	3/20/20
Design Class Diagrams	2/14/20	2/18/20
Sequence/Communication Diagrams	2/19/20	2/21/20
Package Diagram	2/24/20	2/26/20
GRASP	2/27/20	3/2/20
UI Demo	3/3/20	3/17/20
Issue tracking/GIT	3/18/20	3/20/20
Iteration 3	3/23/20	4/27/20
maven	3/23/20	3/27/20

Tasks

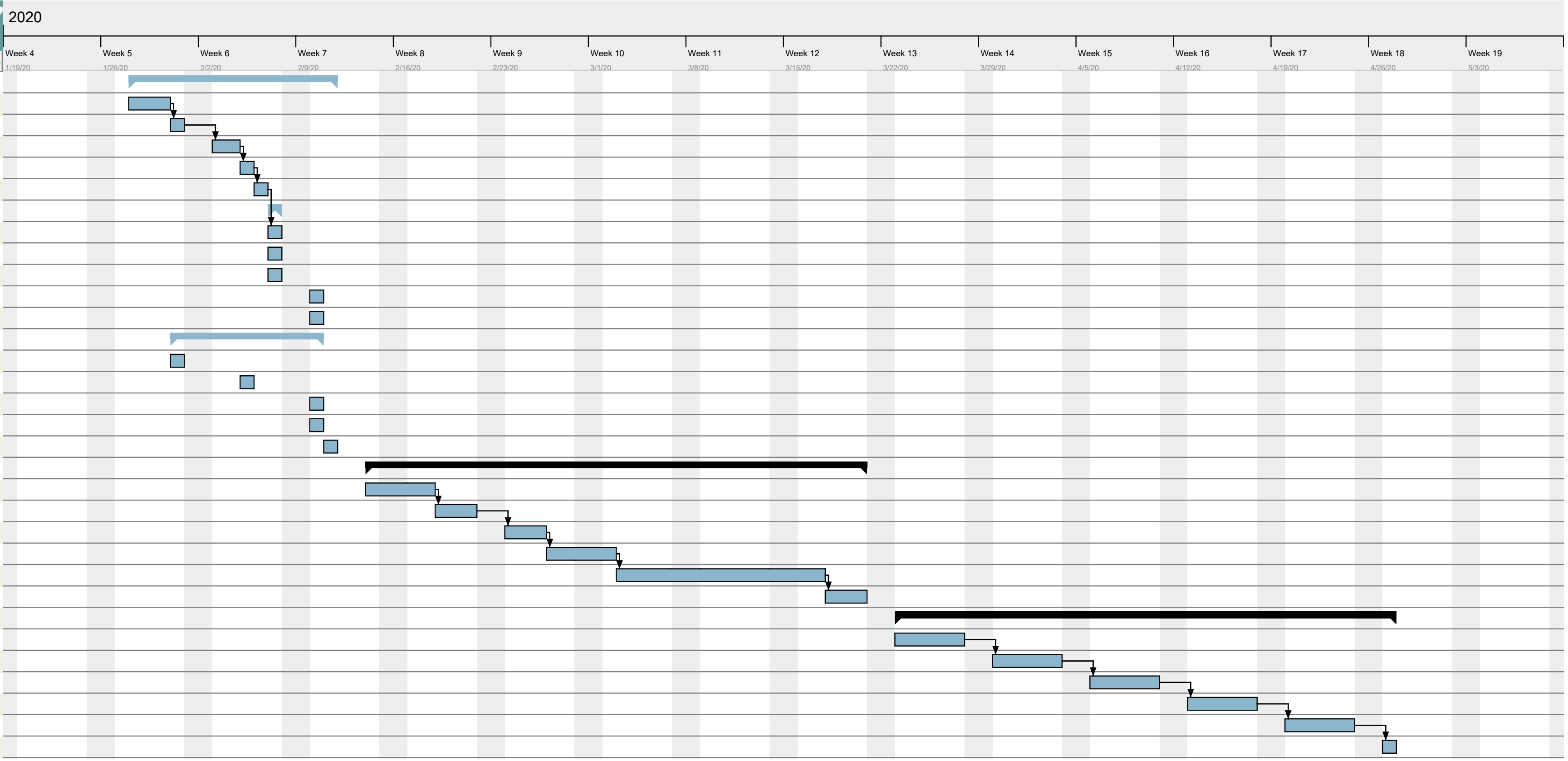
Name	Begin date	End date
git	3/30/20	4/3/20
GUI	4/6/20	4/10/20
Prototype	4/13/20	4/17/20
testing	4/20/20	4/24/20
diagrams	4/27/20	4/27/20

Resources

Name	Default role
Trenton Strickland	developer
Eric Jaroszewski	developer
Samuel Kim	developer
Kevin Demars	developer
Maximilian Cadeddu	developer
Joshua Kanagasabai	developer

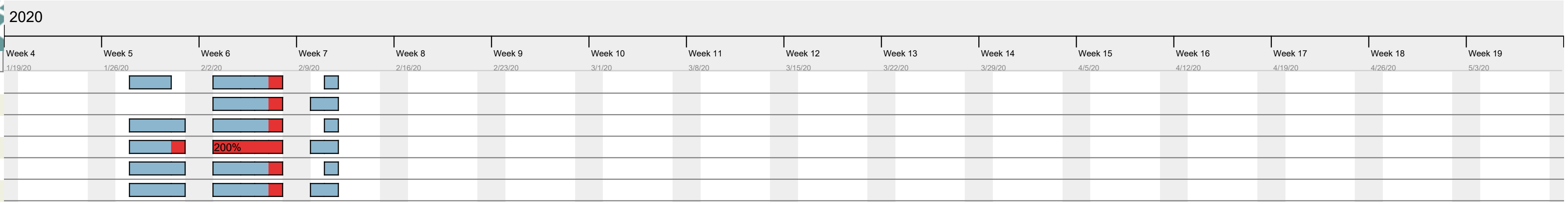
Gantt Chart

Name	Begin date	End date
Iteration 1	1/28/20	2/11/20
Decide on project	1/28/20	1/30/20
Group Meeting 1: Describe U...	1/31/20	1/31/20
FD Use Cases	2/3/20	2/4/20
Group Meeting 2: Go over Us..	2/5/20	2/5/20
SSD	2/6/20	2/6/20
Group Meeting 3	2/7/20	2/7/20
Edit Use Cases	2/7/20	2/7/20
Domain Model	2/7/20	2/7/20
Wireframes	2/7/20	2/7/20
Create Presentation	2/10/20	2/10/20
Operation Contracts	2/10/20	2/10/20
Website	1/31/20	2/10/20
Project Vision	1/31/20	1/31/20
Issue Tracking Link	2/5/20	2/5/20
Project Download link	2/10/20	2/10/20
Team Time Cards	2/10/20	2/10/20
Give Presentation	2/11/20	2/11/20
Iteration 2	2/14/20	3/20/20
Design Class Diagrams	2/14/20	2/18/20
Sequence/Communication D...	2/19/20	2/21/20
Package Diagram	2/24/20	2/26/20
GRASP	2/27/20	3/2/20
UI Demo	3/3/20	3/17/20
Issue tracking/GIT	3/18/20	3/20/20
Iteration 3	3/23/20	4/27/20
maven	3/23/20	3/27/20
git	3/30/20	4/3/20
GUI	4/6/20	4/10/20
Prototype	4/13/20	4/17/20
testing	4/20/20	4/24/20
diagrams	4/27/20	4/27/20



Resources Chart

Name	Default role
Trenton Strickland	developer
Eric Jaroszewski	developer
Samuel Kim	developer
Kevin Demars	developer
Maximilian Cadeddu	developer
Joshua Kanagasabai	developer



Glossary

Event: A timeframe block. It can be plotted by a User in a schedule.

- **Deadline:** Task a specified end time but no specific start time.
- **Activity:** Task with a start time, end time, and date.
- **Recurring Activity:** A specialized form of Activity that repeats every interval and has an end date.

Profile: Contains information pertaining to a unique User.

Group: Collection of profiles that can all be assigned and see the same Events.

Schedule: Contains Events plotted over a calendar.

View: Visualization of a Schedule.

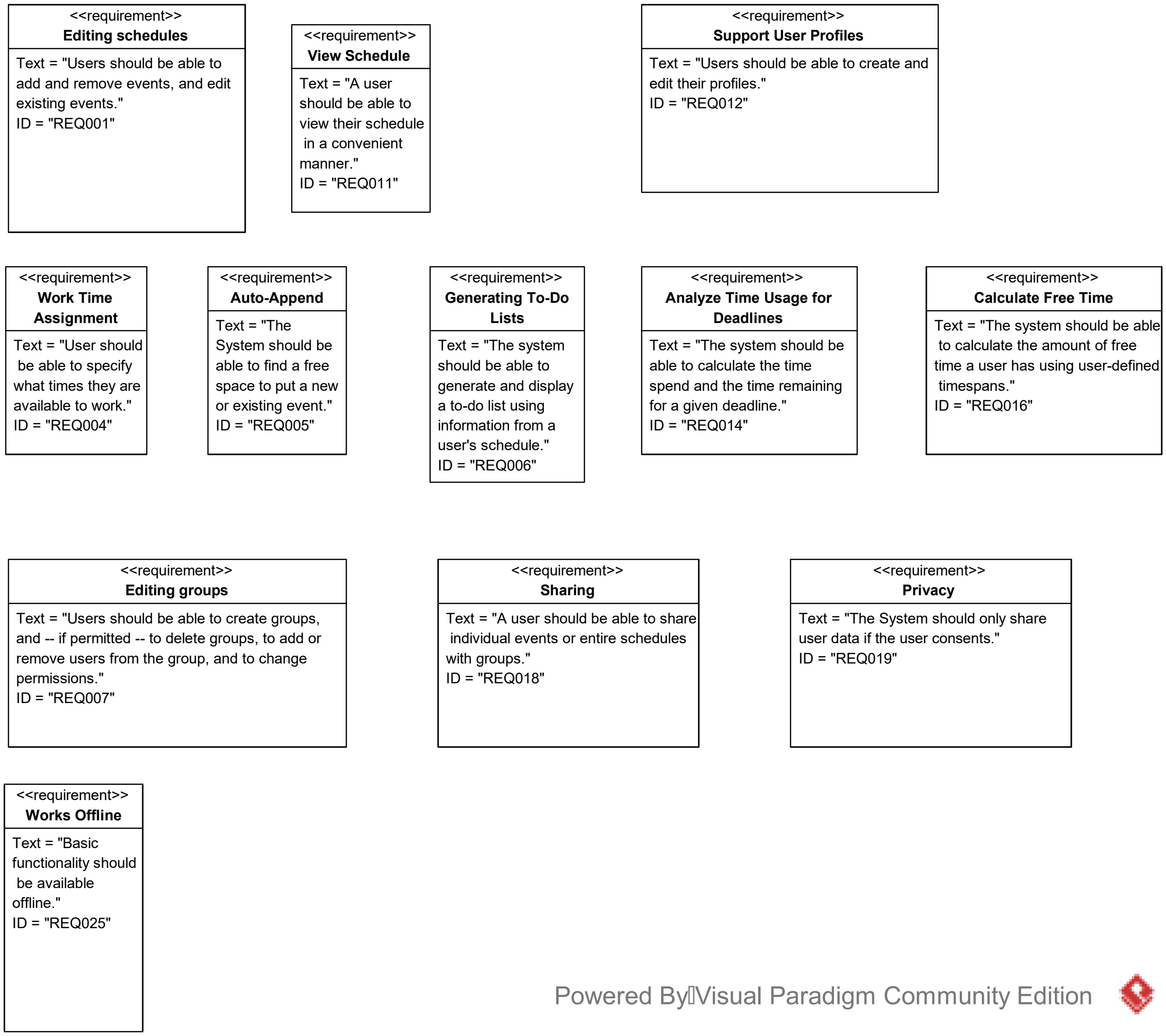
Category: A division of Events.

Privacy Settings:

- **Private:** Only the User can see or edit.
- **Only View Free/Not Free:** Events are Private. Shows a User's free time.
- **Only View:** Shows all a User's scheduled Events.
- **View/Edit:** Shows all a Group's scheduled Events and allows edits.

Date: A specific calendar day**:** mm/dd/yy

WeekDay: One of the following: Monday/Tuesday/Wednesday/Thursday/Friday/Saturday/Sunday



<<requirement>>
Editing schedules
Text = "Users should be able to add and remove events, and edit existing events."
ID = "REQ001"

<<requirement>>
View Schedule
Text = "A user should be able to view their schedule in a convenient manner."
ID = "REQ011"

<<requirement>>
Support User Profiles
Text = "Users should be able to create and edit their profiles."
ID = "REQ012"

<<requirement>>
Work Time Assignment
Text = "User should be able to specify what times they are available to work."
ID = "REQ004"

<<requirement>>
Auto-Append
Text = "The System should be able to find a free space to put a new or existing event."
ID = "REQ005"

<<requirement>>
Generating To-Do Lists
Text = "The system should be able to generate and display a to-do list using information from a user's schedule."
ID = "REQ006"

<<requirement>>
Analyze Time Usage for Deadlines
Text = "The system should be able to calculate the time spend and the time remaining for a given deadline."
ID = "REQ014"

<<requirement>>
Calculate Free Time
Text = "The system should be able to calculate the amount of free time a user has using user-defined timespans."
ID = "REQ016"

<<requirement>>
Editing groups
Text = "Users should be able to create groups, and -- if permitted -- to delete groups, to add or remove users from the group, and to change permissions."
ID = "REQ007"

<<requirement>>
Sharing
Text = "A user should be able to share individual events or entire schedules with groups."
ID = "REQ018"

<<requirement>>
Privacy
Text = "The System should only share user data if the user consents."
ID = "REQ019"

<<requirement>>
Works Offline
Text = "Basic functionality should be available offline."
ID = "REQ025"

Assign Work Time

Use case: Assign Work Time

Scope: NETime Planner Application

Level: User goal

Primary Actor: User

Stakeholders and Interests:

· User: Wants accurate emplacement of Work Time and pre-existing events in his/her schedule to remain untampered.

· System: Wants precise and correctly formatted data to perform time computations and apply schedule operations.

Preconditions: User has a created schedule.

Postconditions: Work Time is assigned to schedule, Start and End Times are saved and emplaced.

Basic Flow:

1. User navigates to schedule
2. User requests schedule to ‘Assign Work Time’.
3. Schedule prompts User to input a ‘Start Time’ and ‘End Time’.
4. User inputs ‘Start Time’ and ‘End Time’.
5. Schedule prompts User to input the ‘Day’ in which the User would like to assign this time frame.
6. User inputs ‘Day’.
7. Schedule checks if ‘Start Time’ to ‘End Time’ overlaps any existing event

8. Schedule records and emplaces ‘Start Time’ and ‘End Time’ on the specified ‘Day’ within the User’s current week schedule

Extensions:

a.) If System recognizes an event is already emplaced within Assigned Work Time

i.) User can cancel assignment

ii.) User can auto emplace the events(s) within the requested time.

b.) If assigned work time already exists for given day

i.) System reports existing Work Time frame.

Information

Rank	Unspecified
ID	
Status	Unspecified
Justification	
Primary Actors	
Suppporting Actors	

Requirements

Work Time Assignment

Works Offline

Auto-Append

Use case: Auto - Append

Scope: NETime Planner Application

Level: User goal

Primary Actor: User

Stakeholders and Interests:

· User: Wants accurate emplacement of events and pre-existing events in his/her schedule to remain untampered.

· System: Wants precise and correctly formatted data to perform time computations and apply schedule operations.

Preconditions: User has a created schedule and assigned Work Time to it.

Postconditions: Event is inserted into the User's schedule at the next available time within the selected day's Assigned Work Time time-frame.

Basic Flow:

1. User navigates to schedule
2. User selects an existing event OR a newly created event
3. User requests to 'Auto-Append' the event.
4. Schedule prompts User to input the 'Day' in which the event should be appended.
5. User inputs 'Day'.
6. Schedule navigates to specified 'Day', then calculates the difference between the day's Assigned Work Time 'End Time' and the time of the last occurring event between 'Start Time' and 'End Time'.

6a. The schedule adds 15 minutes to the calculated difference for spare time / travel time.

6b. If the Event to be auto-appended is less than (the difference + 15 minutes), the event is emplaced in the schedule at (the difference + 15 minutes).

Extensions:

a.) If the System recognizes Assigned Work Time is full:

i.) The user can cancel operation

ii.) The user can request to extend Assigned Work Time to ending time of Event to be appended

Information

Rank	Unspecified
ID	
Status	Unspecified
Justification	
Primary Actors	
Suppporting Actors	

Requirements

Editing schedules

Work Time Assignment

Auto-Append

Works Offline

Calculate Free Time

Scope: NETime Planner Application

Level: User goal

Primary Actor: User A

Stakeholders and Interests:

- User A (logged in user): Wants to check how much free time they have scheduled on their planner

Preconditions: User A has logged into their profile

Postconditions: User A is shown much free time they have

Basic Flow:

1. User A indicates they want to view their free time

2. System prompts user

3. User A enters Time period

4. User A submits request to the Planner System.

5. System searches parses the given time period

6. System adds up anytime that is not otherwise occupied by and event or task

7. System returns data to User A

7. System returns data to User A.

Extensions:

1a. Invalid time period(not formatted correct or not in planner)

1. System signals the time period is invalid and for what reason.
2. Operation is canceled.

Frequency of Occurrence: Medium- useful mostly for heavy users as much data has to be entered in order for free time calculation to be accurate.

Information

Rank	Unspecified
ID	
Status	Unspecified
Justification	
Primary Actors	
Suppporting Actors	

Requirements

Calculate Free Time

Works Offline

Create Activity

Scope: NETime Planner Application

Level: User goal

Primary Actor: User

Stakeholders and Interests:

- User (User changing schedule): Wants Schedule updated for organizational/planning purposes
- Other Users (can be in the same group or not, depending on group's privacy settings): Wants Schedule updated for organization/planning purposes

Preconditions: User is authenticated and has permission to edit Schedule.

Postconditions: Activity is created and saved.

Basic Flow:

1. User requests to add an activity to Schedule.
2. System prompts User for activity details.
3. User fills in the form with activity details.
4. User submits the data to System.
5. System records the new activity.
6. For each user "User B" the schedule is shared with...
    - 6.1. If User B wants to be notified when Schedule changes:
        - 6.1.1. System notifies User B that Schedule changed.
7. System shows User the updated schedule (see View Schedule UC).

Extensions:

1a. Schedule isn't found.

1. System signals error to User.
2. Event creation is cancelled.

1b. User is offline.

1. If Schedule is a group schedule and not User's personal schedule...
    - 1.1. System signals error to User.
    - 1.2. Event creation is cancelled.

4a. The submitted data is invalid.

1. System signals error to User.
2. Event creation is cancelled.

Frequency of Occurrence: Could happen concurrently with many users.

Special Requirements:

- Must work locally for personal schedules if the remote server is unavailable.

Information

Rank	Unspecified
ID	
Status	Unspecified
Justification	
Primary Actors	
Suppporting Actors	

Requirements

Editing schedules

Works Offline

Create Deadline

Scope: NETime Planner Application
Level: User goal
Primary Actor: User
Stakeholders and Interests:

- User (User changing schedule): Wants Schedule updated for organizational/planning purposes
- Other Users (can be in the same group or not, depending on group's privacy settings): Wants Schedule updated for organization/planning purposes

Preconditions: None
Postconditions: Deadline is created and saved.
Basic Flow:

1. User requests to add a deadline to Schedule.
2. System prompts User for deadline details.
3. User fills in the form with deadline details.
4. User submits the data to System.
5. System records the new deadline.
6. For each user "User B" the schedule is shared with...
    6.1. If User B wants to be notified when Schedule changes:
        6.1.1. System notifies User B that Schedule changed.
7. System shows User the updated schedule (see View Schedule UC).

Extensions:

1a. Schedule isn't found.

1. System signals error to User.
2. Event creation is cancelled.

1b. Schedule is owned by another user, and User doesn't have permissions to edit.

1. System signals error to User.
2. Event creation is cancelled.

1c. User is offline.

1. If Schedule is a group schedule and not User's personal schedule...
    1.1. System signals error to User.
    1.2. Event creation is cancelled.

4a. The submitted data is invalid.

1. System signals error to User.
2. Event creation is cancelled.

Frequency of Occurrence: Could happen concurrently with many users.

Special Requirements:

- Must work locally for personal schedules if the remote server is unavailable.

Information

Rank	Unspecified
ID	
Status	Unspecified
Justification	
Primary Actors	
Suppporting Actors	

Requirements

Editing schedules

Auto-Append

Works Offline

Create Group

Use Case:
Create a Group

- Scope- NETime Planner website
- Level: User Goal
- Primary Actor: User
- Stakeholders and Interests:
    - User - Wants reliable, fully functional calendar for planning, and wants to be able to share important tasks with groups to help better plan for how to handle said tasks.
    - Peers/Friends - Want to stay up to date with important tasks being created by user, and be able to contribute in meaningful ways to the tasks at hand.
- Preconditions: The user has created and logged into their profile.
- Success Guarantee (or Postconditions): A group is created. The user is within the group, and has all available permissions to edit the group.
- Main Success Scenario (Basic Flow):

1. User wishes to create a new group.
2. User creates a new group.
3. User provides a name for the new group.
4. User provides profile information for a peer to add to the group.

   User repeats step 5 until they indicate they are done.
5. The group is created.
6. Invitations are sent to all peer profiles provided to join the group.

- Extensions (Alternate flows)
    - a. At any time, the user decides to cancel the group.
        - i. The User presses the “Cancel” button.
        - ii. The group creation is canceled.
    - b. At any time, the system fails.
        - i. The system is restarted.
        - ii. The user restarts the group creation process.
    - c. At any time, the user wishes to go back and modify data in a previous step.
        - i. The user presses the in-site “back” button.
    - 4a. The user does not provide a name for the group.
        - i. The system informs the user a name has not been entered.
        - ii. The system asks the user if they wish to name the group.
        - 3.
            1. The user declines.
            2. The group is named “Group[num]” depending on the number of generic groups that exist.

- 5a. The user does not provide any profiles to the group.
  i. The system asks if the user wishes to add any profiles to the group at the current time.
  2. 1. The user declines.
     2. The group is created and no invitations are sent.
- 5b. The user wishes to remove from the invite list a peer they planned to invite.
  i. The user removes the peer's profile from the list.

- Special Requirements:
  - Group creation and invites by system should be handled within 30 seconds.
- Technology and Data Variations List:
  - Servers to permanently hold user inputted group information and requests.

Information

Rank	Unspecified
ID	
Status	Unspecified
Justification	
Primary Actors	
Suppporting Actors	

Requirements

Editing groups

Sharing

Create Profile

Scope: NETime Planner Application

Level: User goal

Primary Actor: User

Stakeholders and interests:

- User: Wants to create a profile.

Preconditions: A user without a profile exists.

Postconditions: A profile unique to the User has been created. The profile will be usable even after the application has been exited. User has been briefed on the UI following profile creation.

Basic Flow:

1. User indicates they want to create a new profile.

2. User is prompted to enter login credentials

3. User enters their login credentials (username, email, password)

4. A new profile has been created and registered.

5. Application directs User to edit their profile.

6. User finishes editing their profile

7. Tooltips are displayed to introduce the UI to the User

7. Tooltips are displayed to introduce the UI to the User.

8. The User is notified of a means to seek a documentation/usage guide.

9. User dismisses the notice.

10. Application reverts to default behavior

Extensions

2a. The entered email is already attached to an existing profile or is invalid.

    1. An error message is displayed, prompting the User to enter a different email address.

Loop steps 2-3 until a working email address has been entered.

2a. The entered password is less than 8 characters long.

    1. An error message is displayed, prompting the User to enter a new password at least 8 characters in length.

Loop steps 2-3 until a working password has been entered.

2a. The entered username contains invalid characters

    1. An error message is displayed, prompting the User to enter a new username that does not contain invalid characters. Violating character is labeled as such for the user to see.

character is labeled as such for the user to see.

Loop steps 2-3 until a working username has been entered.

Special Requirements:

- If there is going to be a usage guide, it will need to be intelligible.

Frequency of Occurrence:

- Could be nearly continuous across many different machines.

Information

Rank	Unspecified
ID	
Status	Unspecified
Justification	
Primary Actors	
Suppporting Actors	

Requirements

Support User Profiles

Delete Group

Delete a Group

- Scope - NETime Planner Website
- Level - User Goal
- Primary Actor - Group Owner
- Stakeholders and Interests -
  - Group Owner - Wants easy access to all functions of a group, and wants to make sure rest of group is fully informed of his decisions.
  - Peers/Group Members - Want to be up to date with all information regarding the group, and wish to contribute to tasks quickly and efficiently.
- Preconditions - Group Owner has created and logged into their profile, and has created at least one group.
- Success Guarantee(Postconditions) - The group selected by the owner is removed.
- Main Success Scenario -

1. The group owner decides he no longer needs a particular group.
2. Group owner clicks on the group he wishes to remove.
3. Group owner goes to the delete group option.
4. The system asks the owner if he wishes to delete the group.
5. The group owner confirms he wishes to delete the group.
6. All tasks within the group are deleted.
7. All peers and the group owner are removed from the group.
8. The group is deleted.

- Extensions -
  - a. At any time, the system fails.
    - i. The system restarts and the group deletion process is cancelled.
  - 1-4a. At any time, the owner decides to go back one step.
    - i. The owner clicks the in-site “back” button.
  - 6a. The owner declines the request to delete the group.
    - i. The group deletion is canceled.
  - 10a. Within 48 hours of deletion, the group owner wishes to restore the group.
    - i. Group owner clicks on the deleted group.
    - 2.
      - 1. Group owner decides against restoring the group.
      - 2. Group owner clicks in-site “back” button.
    - iii. Group owner restores the group.

- Special Requirements:
    - Group deletion by system should be handled in 48 hours for user convenience.
- Technology and Data Variations List:
    - Servers to permanently hold user inputted group information and requests.

Information

Rank	Unspecified
ID	
Status	Unspecified
Justification	
Primary Actors	
Suppporting Actors	

Requirements

Editing groups

Sharing

Edit Activity

Scope: NETime Planner Application
Level: User goal
Primary Actor: User
Stakeholders and Interests:

- User (User changing schedule): Wants Schedule updated for organizational/planning purposes
- Other Users (can be in the same group or not, depending on group's privacy settings): Wants Schedule updated for organization/planning purposes

Preconditions: User is authenticated and has permissions to modify Schedule.
Postconditions: Activity is modified and saved.
Basic Flow:

1. User requests to edit Activity within Schedule.
2. System displays form containing Activity details.
3. User A edits the form as needed.
4. User A submits data to System.
5. System records the modified data.
6. For each user "User B" the schedule is shared with...
    - 6.1. If User B wants to be notified when Schedule changes:
        - 6.1.1. Notify User B that Schedule changed.
7. System shows User A updated schedule (See View Schedule UC).

Extensions:

1a. Schedule or Activity doesn't exist.

1. System signals error to User.
2. Event editing is cancelled.

1b. User is offline.

1. If Schedule is a group schedule and not User's personal schedule...
    - 1.1. System signals error to User.
    - 1.2. Event creation is cancelled.

4a. The submitted data is invalid.

1. System signals error to User.
2. Event editing is cancelled.

Frequency of Occurrence: Could happen concurrently with many users.
Special Requirements:

- Must work locally for personal schedules if remote server is unavailable.

Information

Rank	Unspecified
ID	
Status	Unspecified
Justification	
Primary Actors	
Suppporting Actors	

Requirements

Editing schedules

Works Offline

Edit Deadline

Scope: NETime Planner Application

Level: User goal

Primary Actor: User

Stakeholders and Interests:

- User (User changing schedule): Wants Schedule updated for organizational/planning purposes
- Other Users (can be in the same group or not, depending on group's privacy settings): Wants Schedule updated for organization/planning purposes

Preconditions: User is authenticated and has permissions to modify Schedule.

Postconditions: Deadline is modified and saved.

Basic Flow:

1. User requests to edit Deadline within Schedule.
2. System displays form containing Deadline's details.
3. User A edits the form as needed.
4. User A submits data to System.
5. System records the modified data.
6. For each user "User B" the schedule is shared with...
   - 6.1. If User B wants to be notified when Schedule changes:
     - 6.1.1. Notify User B that Schedule changed.
7. System shows User A updated schedule (See View Schedule UC).

Extensions:

1a. Schedule or Deadline doesn't exist.

1. System signals error to User.
2. Event editing is cancelled.

1b. User is offline.

1. If Schedule is a group schedule and not User's personal schedule...
   - 1.1. System signals error to User.
   - 1.2. Event creation is cancelled.

4a. The submitted data is invalid.

1. System signals error to User.
2. Event editing is cancelled.

Frequency of Occurrence: Could happen concurrently with many users.

Special Requirements:

- Must work locally for personal schedules if remote server is unavailable.

Information

Rank	Unspecified
ID	
Status	Unspecified
Justification	
Primary Actors	
Suppporting Actors	

Requirements

Editing schedules

Works Offline

Edit Group

Use Case:

Edit a Group

- Scope- NETime Planner website
- Level: User Goal
- Primary Actor: User
- Stakeholders and Interests:
  - User - Wants easy access to all functions of a group, and wants to make sure rest of group is fully informed of his decisions.
  - Peers/Group Members - Want to be up to date with all information regarding the group, and wish to contribute to tasks quickly and efficiently.
- Preconditions - User has created and logged into their profile, is in at least one group, and has the permissions necessary to edit parts of the group.
- Success Guarantee - The group is modified to the user’s liking.
- Main Success Scenario -

1. The User wishes to modify member list of the group.
2. User clicks on desired group.
3. User provides profile information for a peer to add or remove.

   User repeats step 3 until they indicate they are done.

4. Added group members are sent invitations.
5. Members specified to be deleted are removed from the group.
6. User wishes to modify group’s task list.
7. User clicks on desired group.
8. User creates a new task, or selects task to remove.
9. User indicates he is done.
10. The specified task is created or removed, and the assigned group members are notified.

- Extensions -
  - a. At any time, the system fails.
    - i. The system restarts, and the last unsaved edit is canceled.
  - b. At any time, the user wishes to go back, either to cancel an unsaved action or to modify data entered in a previous step.
    - i. The user clicks on the in-site “back” button.
- Special Requirements:
  - Once saved, edits done by system should be handled within 30 seconds.
- Technology and Data Variations List:

    - Servers to permanently hold user inputted group information and requests.

Information

Rank	Unspecified
ID	
Status	Unspecified
Justification	
Primary Actors	
Suppporting Actors	

Requirements

Editing groups

Sharing

Edit Group Permissions

Use Case: Edit Group Permissions

- Scope- NETime Planner website
- Level: User Goal
- Primary Actor: User
- Stakeholders and Interests:
  - Group Owner - Wants easy access to all functions of a group, and wants to make sure rest of group is fully informed of his decisions.
  - Peers/Group Members - Want to be up to date with all information regarding the group, and wish to contribute to tasks quickly and efficiently.
- Preconditions - Group Owner has created and logged into their profile and is in at least one group, and does not wish to modify his own permissions.
- Success Guarantee - The permissions of individuals within the group are modified to the owner's liking.
- Main Success Scenario -

1. The Owner wishes to modify the permissions for group members.
2. Owner clicks on desired group.
3. Owner goes to Permissions setting.
4. Owner finds the group member whose permissions he wishes to modify.
5. Owner wishes to modify the group member's permissions to either add other group members, or to edit group tasks.
6. Owner checks off the boxes for approved permissions, and clears the boxes for denied permissions.
7. The owner indicates he is done.
8. The Owner wishes to transfer ownership of the group to another member.
9. Owner performs steps 2-3.
10. System asks Owner if he wishes to transfer ownership to the selected user.
11. Owner confirms.
12. Notification is sent to specified group member for request to transfer ownership.
13. Other group member confirms.
14. Ownership is transferred from old owner to group member.
15. Former owner loses permission to set permissions.

16. New Owner gains all available permissions to modify the group.

- Extensions-
  - a. At any time, the system fails.
    - i. The system restarts, and the permissions are reset to their last known state.
  - b. At any time, the user wishes to go back, either to cancel an unsaved action or to modify data entered in a previous step.
    - i. The user clicks on the in-site “back” button.
  - 1 or 11a. The Owner is the only member of the group.
    - i. The system will state that there are no other group members with modifiable permissions.
  - 19a. The group member denies.
    - i. System asks group member if he does not want ownership of the group.
    - 2. 1. Group member denies. Back to 19a.
    - iii. Group member confirms.
    - iiv.. Ownership of the group remains with original owner.
- Special Requirements:
  - Once saved, edits done by system should be handled within 30 seconds.
- Technology and Data Variations List:
  - Servers to permanently hold user inputted group information and requests.

Information

Rank	Unspecified
ID	
Status	Unspecified
Justification	
Primary Actors	
Suppporting Actors	

Requirements

Editing groups

Sharing

Privacy

Edit Privacy Setting

ID: Edit Privacy Setting
Scope: NETime Planner Application
Level: User Goal
Primary Actor: Group Owner
Stakeholders and Interests:
Group Owner
- An owner of a group who would like to change the privacy setting of the group schedule.

Group Participant
- A member of the group who will be affected by the change in the privacy setting.

Precondition: Group Owner has navigated to the group schedule for a group which they own.
Postcondition: The viewability of the schedule has been updated.
Main Success Scenario:

1. Group Owner requests to edit privacy settings.

2. The system displays the privacy setting menu to Group Owner.

3. Group Owner either sets “Private”, unsets “Private”, or leaves the setting as it is.

5. Group Owner sets the viewability of the group schedule to either View/Edit, Only View, or Only View Free/Not Free.

7. Group Owner submits the changes.

8. The system updates the privacy of the group schedule.

Extensions: None

Frequency of Occurrence:
This happens frequently, since it is often used when sharing schedules.
Special Requirements:
The remote server must be available to update the setting for other users.

Information

Rank	Unspecified
ID	
Status	Unspecified
Justification	
Primary Actors	
Suppporting Actors	

Requirements

Sharing

Privacy

Edit Profile

Scope: NETime Planner Application

Level: User goal

Primary Actor: User

Stakeholders and interests:

- User: Wants to edit an existing profile.

Preconditions: User and a profile exists.

Postconditions: Profile has been edited and pertinent information has been updated server-side.

Basic Flow:

1. User indicates they want to edit their profile.

2. Profile is displayed.

3. Edit-able text fields are displayed with their current value, next to a text box where the user can enter a new value.

4. The user changes their email field.

5. The option to delete specific categories or add new ones is supplied.

6. User deletes a category.

7. User is prompted to confirm they wish to delete the category

7. User is prompted to confirm they wish to delete the category.

8. User clicks a button labeled “Yes”.

9. The category is marked to be removed *iff* Step 15 occurs.

10. User adds a category.

11. User inputs a name for the category.

12. The category is temporarily created, but not added to the profile until Step 15 occurs.

13. User clicks a button labeled “finish”.

14. User is prompted to confirm they wish to make these changes.

15. User clicks a button labeled “Yes”.

16. The changes are verified.

17. The profile window is dismissed.

18. Application reverts to default behavior.

Extensions

6a. User is prompted to confirm they wish to delete this category.

1. A User clicks a button labeled “No”.
2. Confirmation prompt is dismissed.

3. Category is not marked for deletion.

10a. User clicks a button labeled “cancel”

1. User is prompted to confirm they wish to cancel. All unsaved changes will be lost.

1a. User clicks a button labeled “Yes”

1. Profile is not changed.

Jump to step 14

1b. User clicks a button labeled “No”

1. Profile is not changed.

2. “Cancel” Prompt is dismissed.

15a. User clicks a button labeled “No”

1. Changes are not made.

2. Any categories marked for deletion/creation remain as such until step 15 occurs.

3. Prompt is dismissed.

4. User remains on the Profile page in edit mode.

5. User continues to make changes.

3. User continues to make changes.

16a. One or more text fields contain invalid characters.

1. User is directed back to the edit page as they left it.
2. Error message is displayed.
3. Offending fields are highlighted.

Special Requirements:

- While editing. Of the Editable fields; none, some, or all can be changed, and the process should complete successfully regardless.

Frequency of Occurrence:

- Could be nearly continuous across many different machines.

Information

Rank	Unspecified
ID	
Status	Unspecified
Justification	
Primary Actors	
Suppporting Actors	

Requirements

Support User Profiles

Generate To-Do List

Use case: Generate Today’s To-Do List

Scope: NETime Planner Application

Level: User goal

Primary Actor: User

Stakeholders and Interests:

- User: Wants accurate list of events and assignments with their times and pre-existing events in his/her schedule to remain untampered.

- System: Wants precise and correctly formatted data to perform time computations and apply schedule operations.

Preconditions: User has a created schedule with events emplaced upon it.

Postconditions: User is presented with List of events with their associated times, in the order in which they occur on the schedule.

Basic Flow:

1. User navigates to Profile.
2. User requests Generate Today’s To Do List.
3. System navigates to the current day within schedule.
4. System generates an empty list of events.
5. From clock-time 00:00:00 to 23:59:59 of the calculated date, System inserts each event emplaced on schedule into the List.
6. System displays the list.

Extensions:

Extensions:

a.) User wants a day other than the current day

i.) System requests specified day

ii.) User inputs day

ii.) System follows basic flow steps 3-5

Information

Rank	Unspecified
ID	
Status	Unspecified
Justification	
Primary Actors	
Suppporting Actors	

Requirements

Generating To-Do Lists

View Schedule

Works Offline

Remove Activity

Scope: NETime Planner Application
Level: User goal
Primary Actor: User
Stakeholders and Interests:

- User (User changing schedule): Wants Schedule updated for organizational/planning purposes
- Other Users (can be in the same group or not, depending on group's privacy settings): Wants Schedule updated for organization/planning purposes

Preconditions: User is authenticated and has permissions to modify Schedule.
Postconditions: Activity is removed from Schedule.
Basic Flow:

1. User A requests to delete Activity within Schedule.
2. System deletes Activity.
3. For each user "User B" the schedule is shared with...
    3.1. If User B wants to be notified when Schedule changes:
        3.1.1. System notifies User B that Schedule changed.
4. System shows User the updated schedule (see View Schedule UC).

Extensions:

1a. User or Activity doesn't exist.

1. System signals error to User.
2. Event deletion is cancelled.

1b. User is offline.

1. If Schedule is a group schedule and not User's personal schedule...
    1.1. System signals error to User.
    1.2. Event creation is cancelled.

Frequency of Occurrence: Could happen concurrently with many users.
Special Requirements:

- Must work locally for personal schedules if the remote server is unavailable.

Information

Rank	Unspecified
ID	
Status	Unspecified
Justification	
Primary Actors	
Suppporting Actors	

Requirements

Editing schedules

Works Offline

Remove Deadline

Scope: NETime Planner Application

Level: User goal

Primary Actor: User

Stakeholders and Interests:

- User (User changing schedule): Wants Schedule updated for organizational/planning purposes
- Other Users (can be in the same group or not, depending on group's privacy settings): Wants Schedule updated for organization/planning purposes

Preconditions: User is authenticated and has permissions to modify Schedule.

Postconditions: Deadline is removed from Schedule.

Basic Flow:

1. User A requests to delete Deadline within Schedule.
2. System deletes Deadline.
3. For each user "User B" the schedule is shared with...
   - 3.1. If User B wants to be notified when Schedule changes:
     - 3.1.1. System notifies User B that Schedule changed.
4. System shows User the updated schedule (see View Schedule UC).

Extensions:

1a. User or Deadline doesn't exist.

1. System signals error to User.
2. Event deletion is cancelled.

1b. User is offline.

1. If Schedule is a group schedule and not User's personal schedule...
   - 1.1. System signals error to User.
   - 1.2. Event creation is cancelled.

Frequency of Occurrence: Could happen concurrently with many users.

Special Requirements:

- Must work locally for personal schedules if the remote server is unavailable.

Information

Rank	Unspecified
ID	
Status	Unspecified
Justification	
Primary Actors	
Suppporting Actors	

Requirements

Editing schedules

Works Offline

Share Event

ID: Share Event
Scope: NETime Planner Application
Level: User Goal
Primary Actor:
Stakeholders and Interests:
Group Participant
- A person that is interested in sharing their event with another user or a group of users.

Group Owner
- The owner of a group to which Group Participant is sharing an event. If the group schedule is not editable, the owner decides whether or not to accept the event.

Event Recipient(s)
- The groups that the event will be shared with.

Precondition: Group Participant has opened the schedule containing the event that he or she intends to share.
Postcondition: The event has been added to the schedules of one or more selected groups, allowing the group members to view the event on that schedule when they use the application.
Main Success Scenario:
1. Group Participant requests to share an event.
2. The system lists each group the participant is in.
3. Group Participant selects a group to share to.
4. The system adds the event to the specified group's schedule.

Extensions:
4.a. If a group receiving the event does not have an editable schedule,
    1. The system sends a message of confirmation to the group owner.
    1.a. If the group owner confirms the request to share the event,
        1.1. The system updates the schedule with the new event from Group Participant.
    1.b. If the group owner declines the request to share the event,
        1.1. The system sends a message to Group Participant notifying that the event was declined.

Frequency of Occurrence:
This happens frequently, as this is one of the core aspects of the social element for the application.
Special Requirements:
The remote server must be available.

Information

Rank	Unspecified
ID	
Status	Unspecified
Justification	
Primary Actors	
Suppporting Actors	

Requirements

Sharing

Share Schedule

ID: Share Schedule
Scope: NETime Planner Application
Level: User Goal
Primary Actor: Group Participant
Stakeholders and Interests:
Group Participant
- A person that is interested in sharing their schedule with a group of users.
Schedule Recipient(s)
- The groups that the schedule will be shared with.

Precondition: Group Participant is a member of one or more groups
Postcondition: The schedule has been shared with one or more selected groups, allowing the group members to view the most updated version of the schedule from their computers when they open the application.
Basic Flow:

1. Group Participant requests to share their schedule.
2. The system lists each group the participant is in.
3. Group Participant selects one of the groups to share to.
4. The system gives each recipient access to the schedule created by Group Participant.
5. The system shows a success message.

Extensions: None.
Frequency of Occurrence:
This happens frequently, as this is one of the core aspects of the software’s social element.
Special Requirements:
The remote server must be available.

Information

Rank	Unspecified
ID	
Status	Unspecified
Justification	
Primary Actors	
Suppporting Actors	

Requirements

Sharing

Show Time Left for Specific Task

Scope: NETime Planner Application

Level: User goal

Primary Actor: User A

Stakeholders and Interests:

· User A (logged in user): Wants to check how much time they have left to work on a specific task based on their planner

Preconditions: User A has logged into their profile

Postconditions: User A is displayed the time left on a specific task

Basic Flow:

1. User A indicates they want to view the time left to finish a task

2. System prompts user

3. User A enters

    1. Task name

4. User A submits request to the Planner System.

5. System searches for given task.

6. System finds task and calculates the time to due date.

7. System returns data to User A.

Extensions:

1a. Invalid task name(not found in Planner System)

1. System signals this task does not exist.
2. Operation is canceled.

1a. Task has included time schedule

1. System detects task has scheduled time allocation.
2. In addition to the time to due date, the System also returns total remaining time allocated before the task ends.

5a. Alter return format based on if start date is included

Frequency of Occurrence: Heavy- Many users will check their time left on tasks as a central part of the planner.

Information

Rank	Unspecified
ID	
Status	Unspecified
Justification	
Primary Actors	
Suppporting Actors	

Requirements

Analyze Time Usage for Deadlines

Works Offline

View Schedule

Scope: NETime Planner Application

Level: User goal

Primary Actor: User

Stakeholders and interests:

- User: Wants to create an application-generated view of the upcoming week or a specific day.

Preconditions: A user with a profile exists.

Postconditions: A view of the schedule specific to the User's profile has been generated within the specified time range (week/day) and displayed.

Basic Flow:

1. User indicates they want to view their schedule

2. The schedule is displayed to the User.

3. The User views the schedule.

4. The User terminates the display and the view of the schedule is destroyed.

5. The application reverts to default behavior

Extensions

1a. For the next week.

1. A less detailed “birds-eye-view” of the upcoming week is created.

1a. For a specific day.

1. A more detailed, daily schedule organized by hourly increments is created.

3a. The User has a profile, but has not inputted any information

1. A blank view is displayed alongside a prompt to the User to enter events or add events to their schedule.

4a. The User leaves the view open.

1. The application continues performing as usual, but some functionality may be inaccessible until the view is closed.

Special Requirements:

- Whatever functionality is “covered” by the view either needs to be relocated or auxiliary as far as the schedule is concerned. Users should be able to add/remove events while the schedule is open, and the schedule should update to reflect those changes.

- Group events should exist locally in the event the user is offline. They will need to be updated when the User goes back online to reflect any changes that may have occurred.

Frequency of Occurrence: A localized event that could happen sporadically.

Information

Rank	Unspecified
ID	
Status	Unspecified
Justification	
Primary Actors	
Suppporting Actors	

Requirements

View Schedule

Works Offline

View time spent on specific task

Scope: NETime Planner Application

Level: User goal

Primary Actor: User A

Stakeholders and Interests:

· User A (logged in user): Wants to check how much time they have worked on a specific task based on their planner

Preconditions: User A has logged into their profile

Postconditions: User A is displayed the time spent on a specific task

Basic Flow:

1. User A indicates they want to view their time spent on a task

2. System prompts user

2. User A enters

    1. Task name

    2. Time frame to check over, can also select global to get entire time spent.

3. User A submits request to the Planner System.

4. System searches for given task

4. System searches for given task.

5. System finds task and calculates time spent within the given time frame.

6. System returns data to User A.

Extensions:

1a. Invalid task name(not found in Planner System)

   1. System signals this task does not exist

   2. Operation is canceled

1a. Invalid time frame(incorrectly formatted time frame entered)

   1. System signals this task does not exist

   2. Operation is canceled

1b. No time frame is entered

   1. System assumes global time frame

Frequency of Occurrence: Medium- Mostly for users that heavily use the planner as it requires for users to have logged/planned their tasks heavily for accurate results.

Information

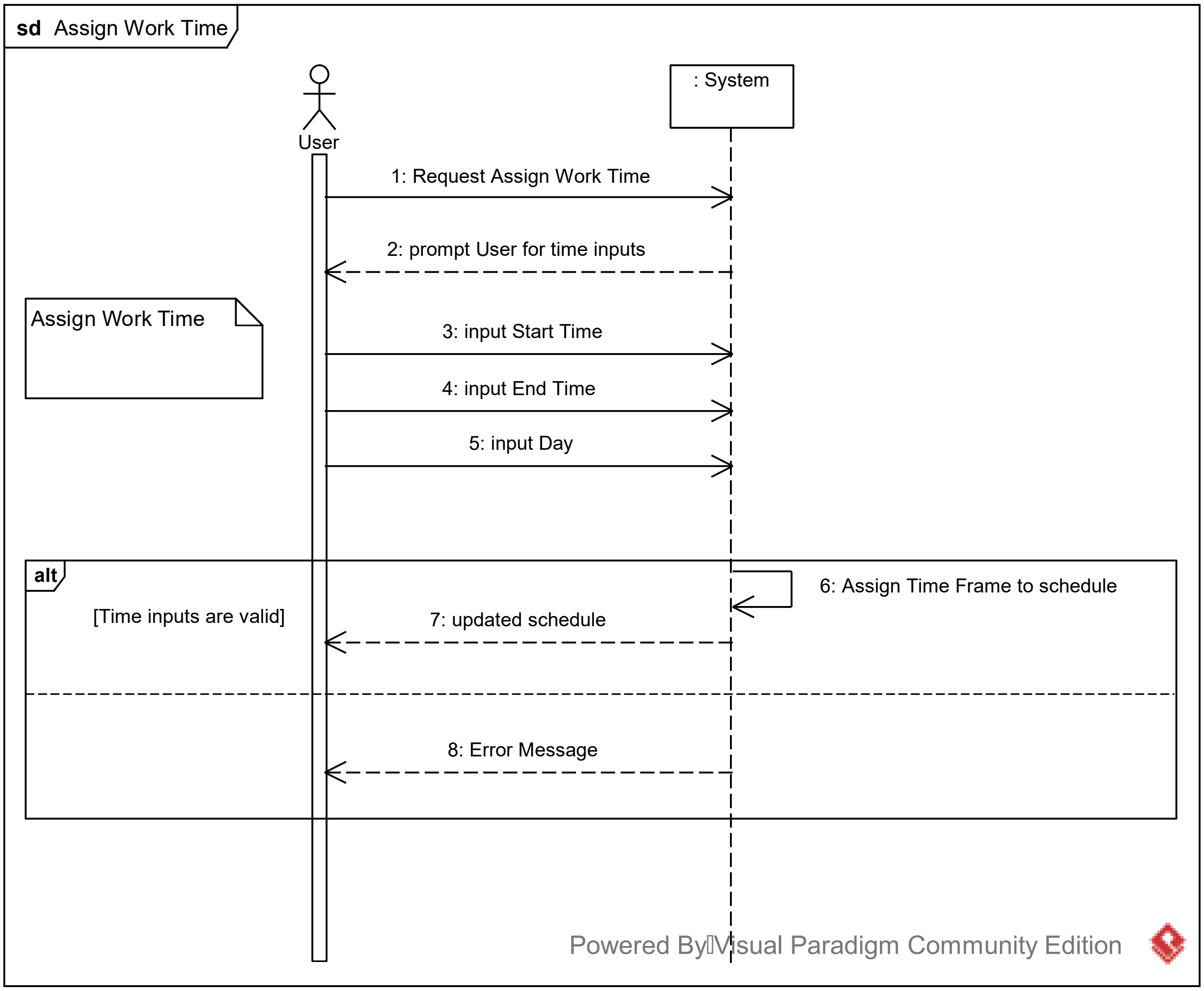
Rank	Unspecified
ID	
Status	Unspecified
Justification	
Primary Actors	
Suppporting Actors	

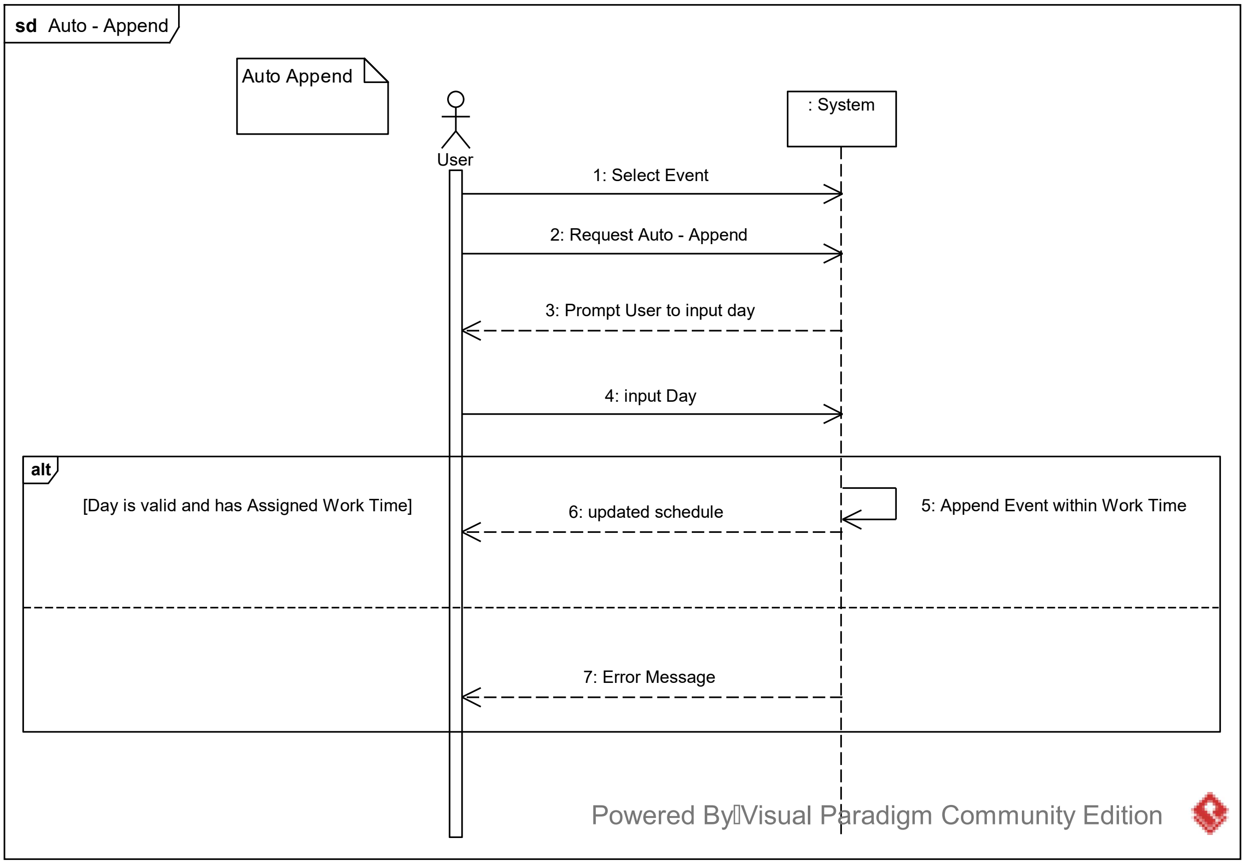
Requirements

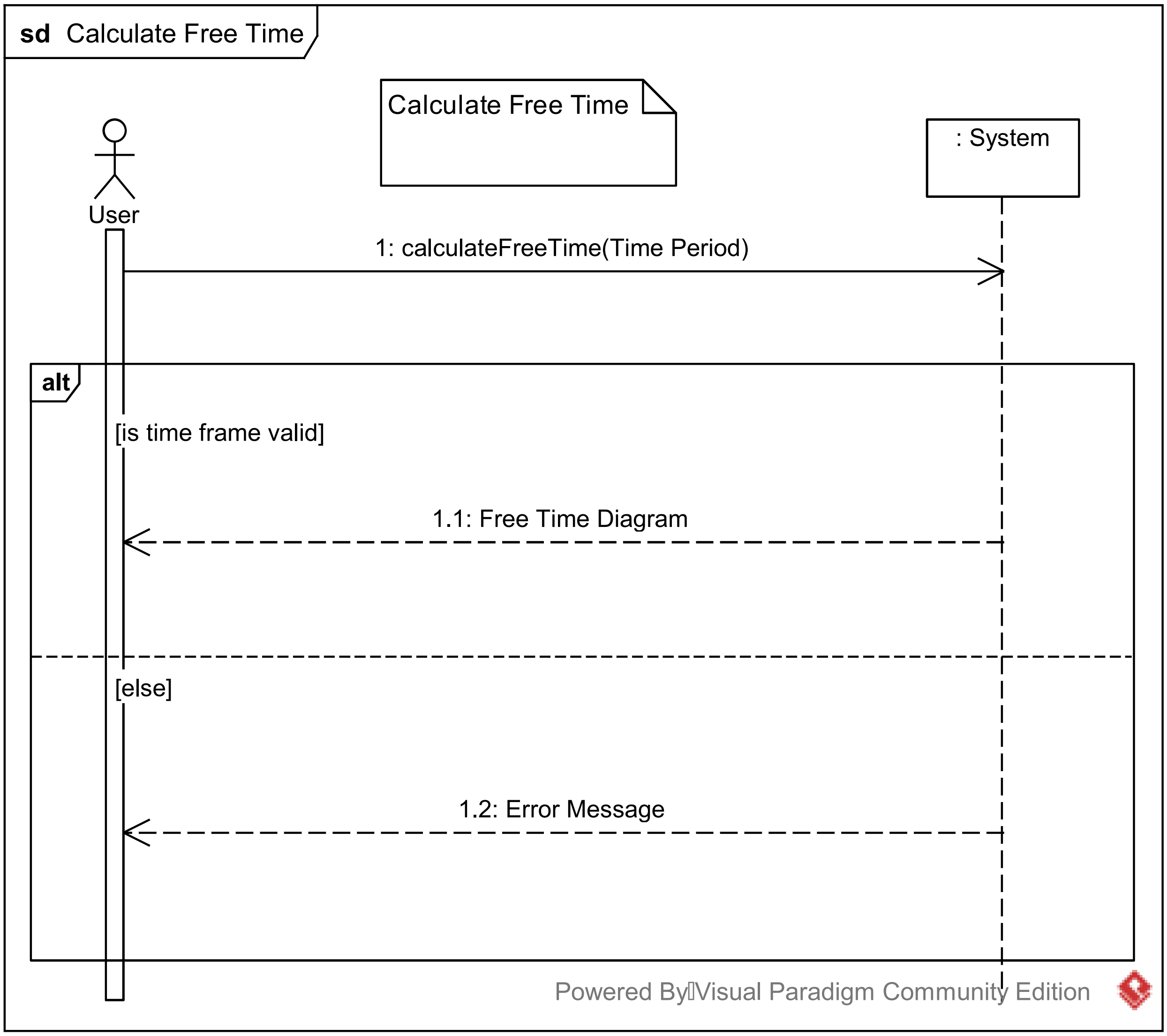
Analyze Time Usage for Deadlines

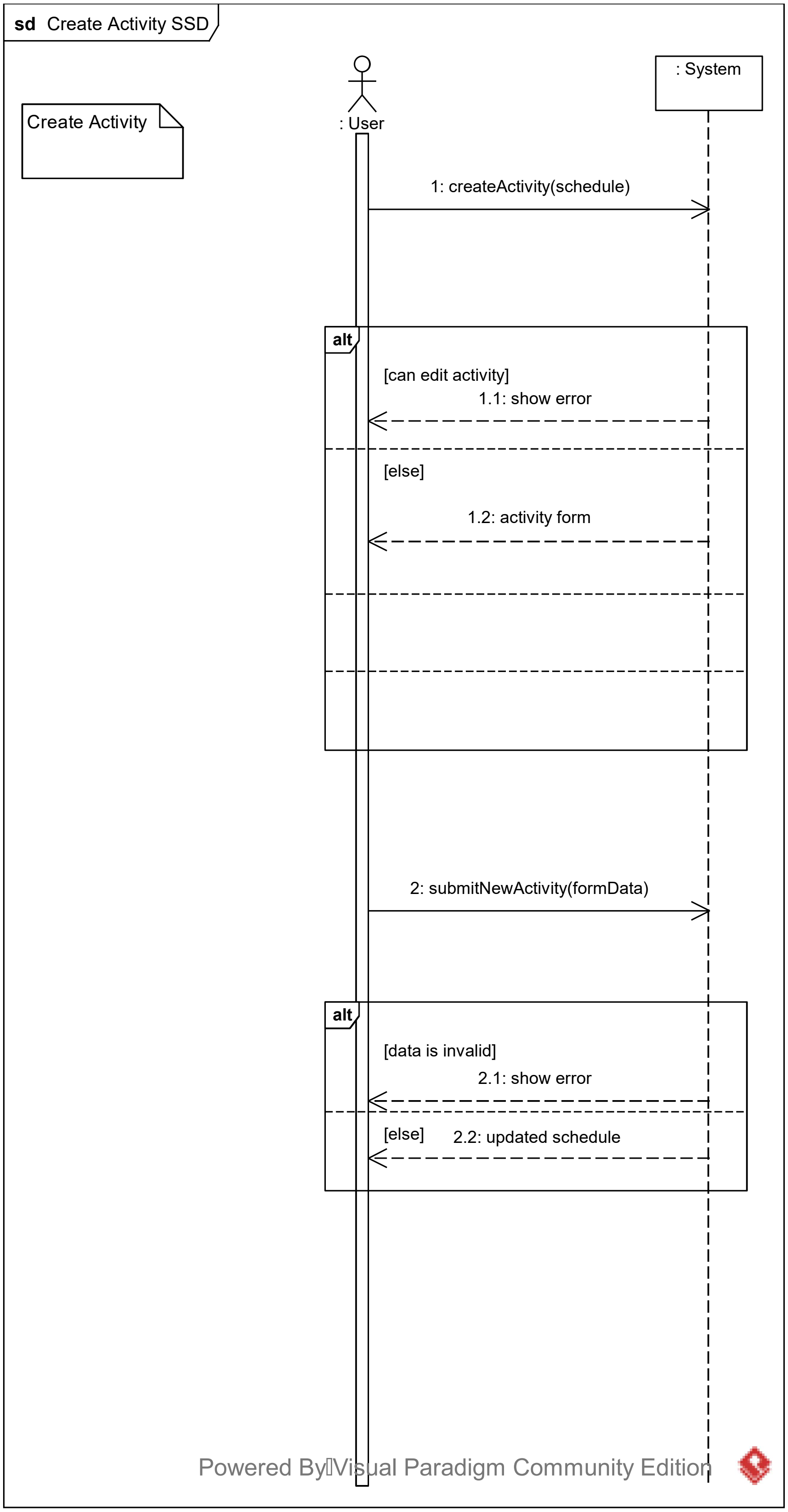
Works Offline

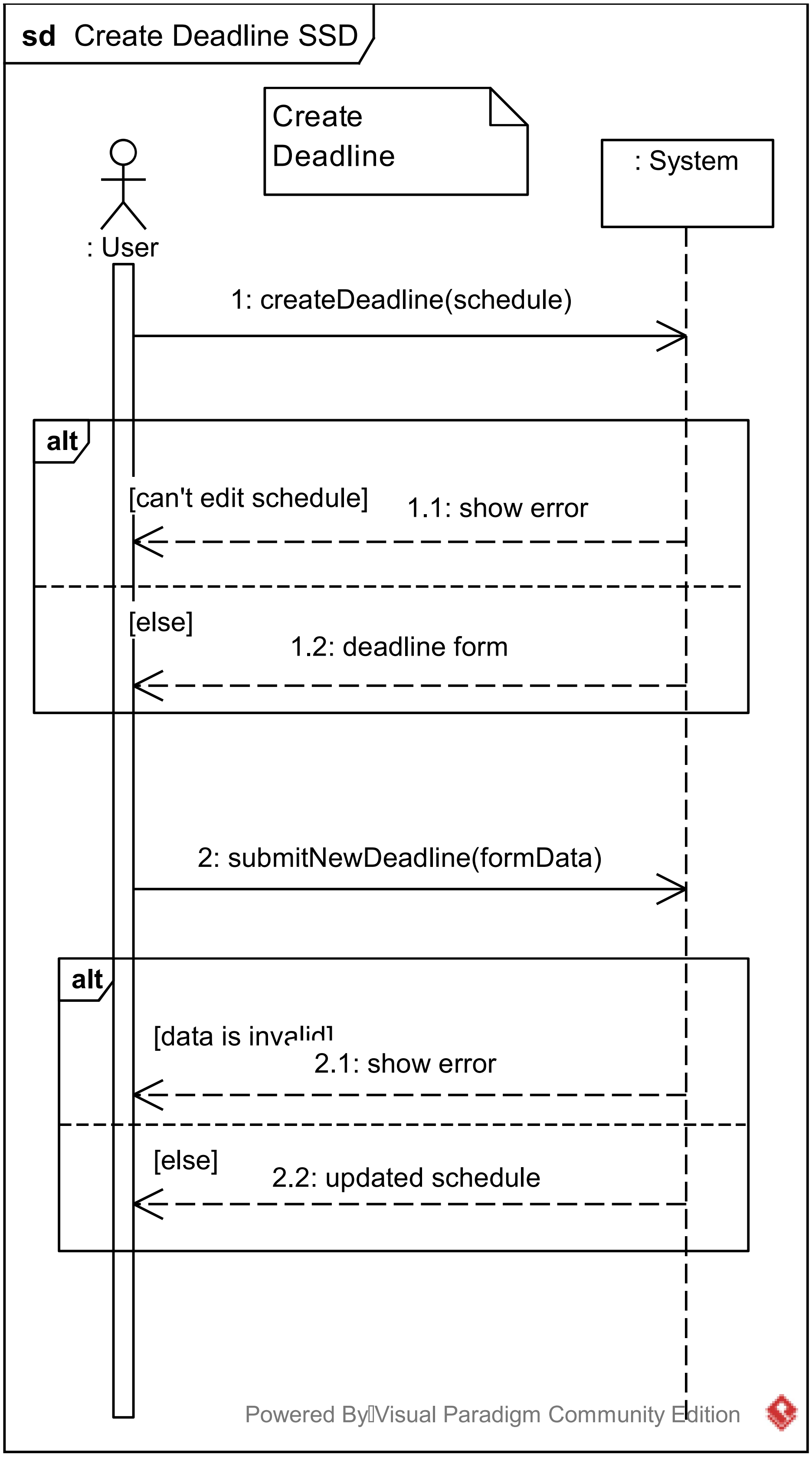
	REQ001	REQ004	REQ005	REQ006	REQ007	REQ011	REQ012	REQ014	REQ016	REQ018	REQ019	REQ025
UC01	X											X
UC02	X											X
UC03	X											X
UC04		X	X									X
UC05	X	X	X									X
UC06				X		X						X
UC07					X					X		
UC08					X					X		
UC09					X					X		
UC10					X					X	X	
UC11						X						X
UC12							X					
UC13							X					
UC14								X				X
UC15								X				X
UC16									X			X
UC17										X		
UC18										X		
UC19										X	X	
UC20	X											X
UC21	X											X
UC22	X											X

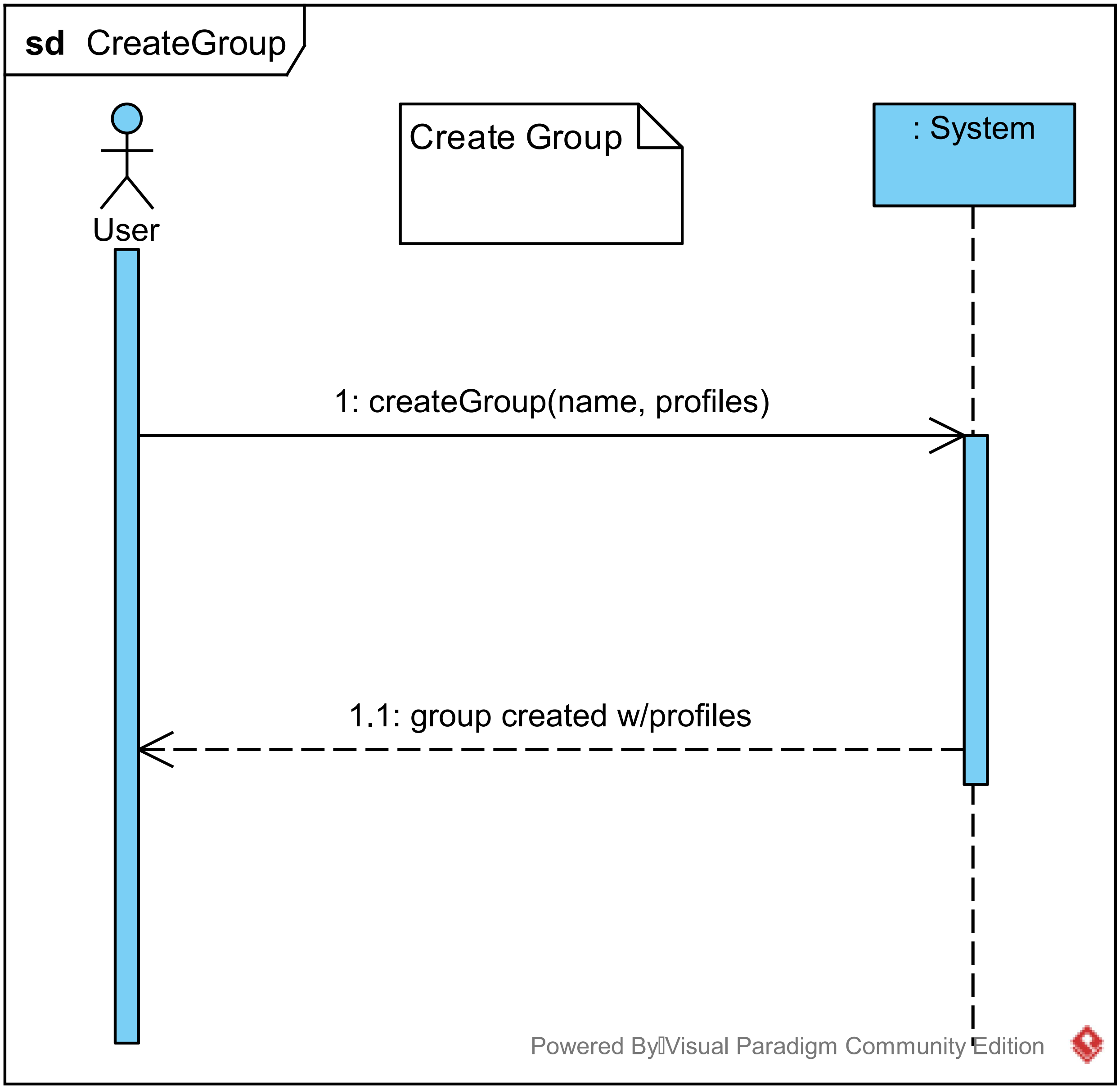

sd Assign Work Time
User
: System
1: Request Assign Work Time
2: prompt User for time inputs
Assign Work Time
3: input Start Time
4: input End Time
5: input Day
alt
[Time inputs are valid]
6: Assign Time Frame to schedule
7: updated schedule
8: Error Message
Powered By Visual Paradigm Community Edition

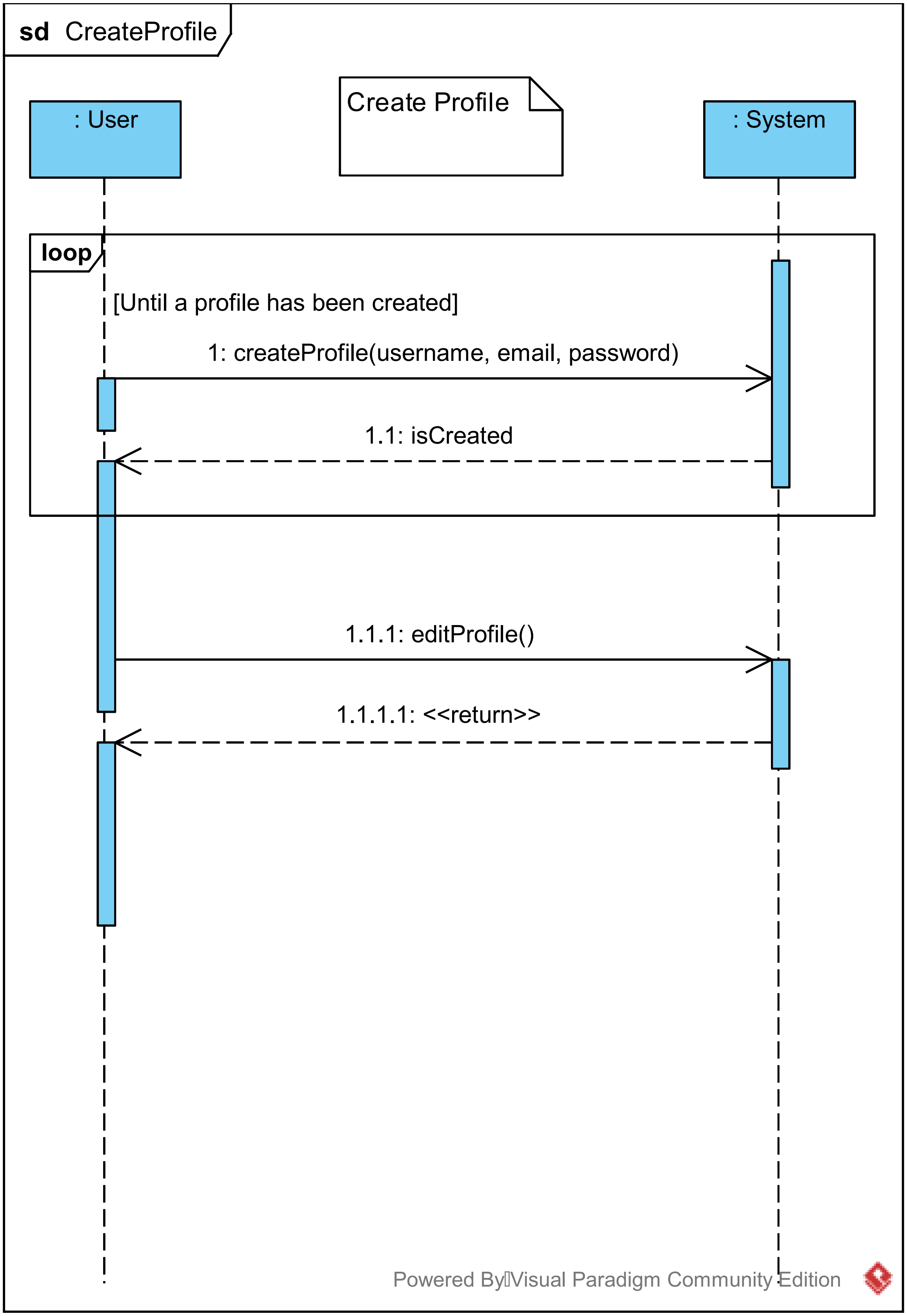

sd Auto - Append
Auto Append
User
: System
1: Select Event
2: Request Auto - Append
3: Prompt User to input day
4: input Day
alt
[Day is valid and has Assigned Work Time]
5: Append Event within Work Time
6: updated schedule
7: Error Message
Powered By Visual Paradigm Community Edition

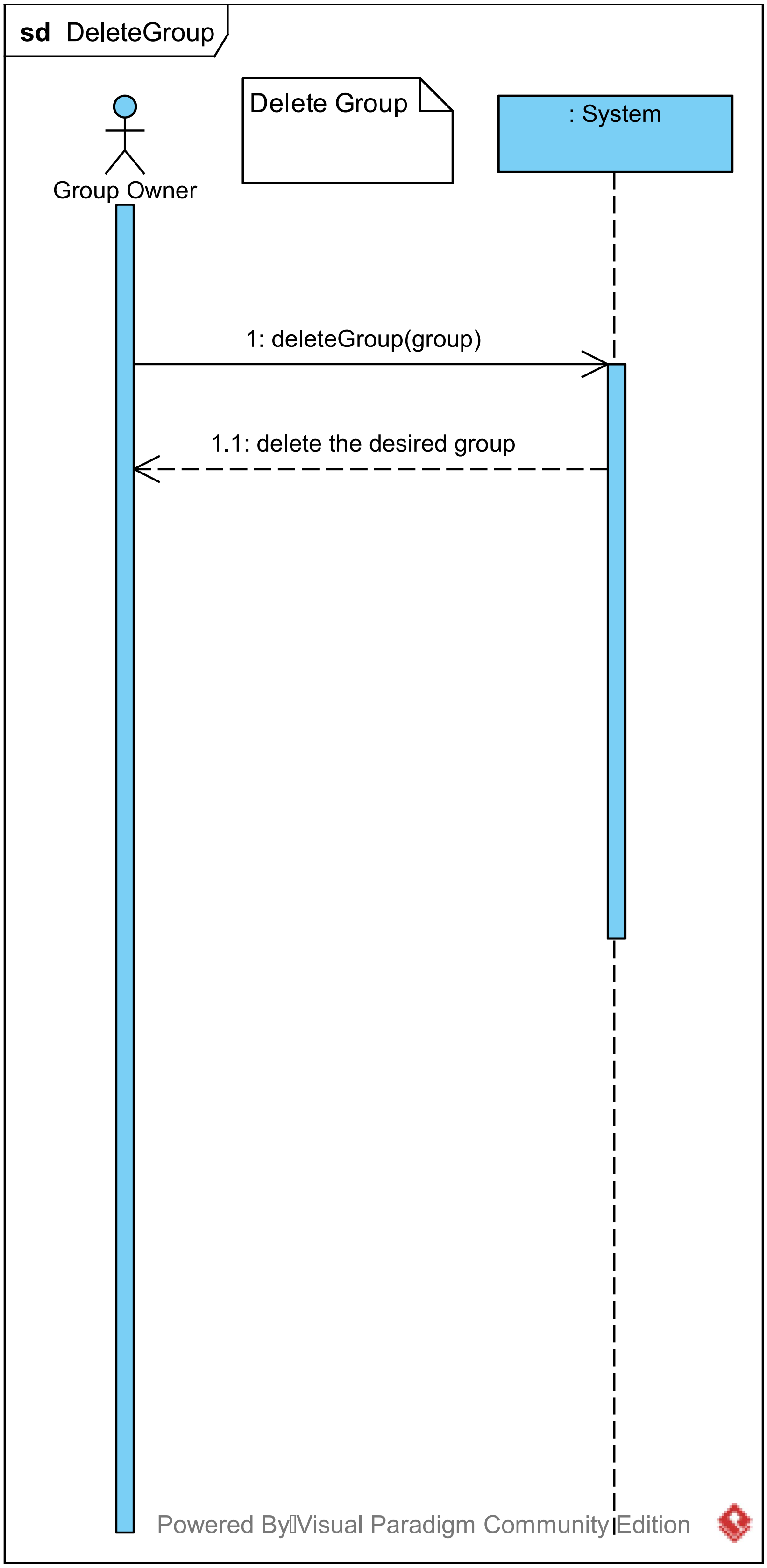

sd Calculate Free Time
Calculate Free Time
User
: System
1: calculateFreeTime(Time Period)
alt
[is time frame valid]
1.1: Free Time Diagram
[else]
1.2: Error Message
Powered By Visual Paradigm Community Edition

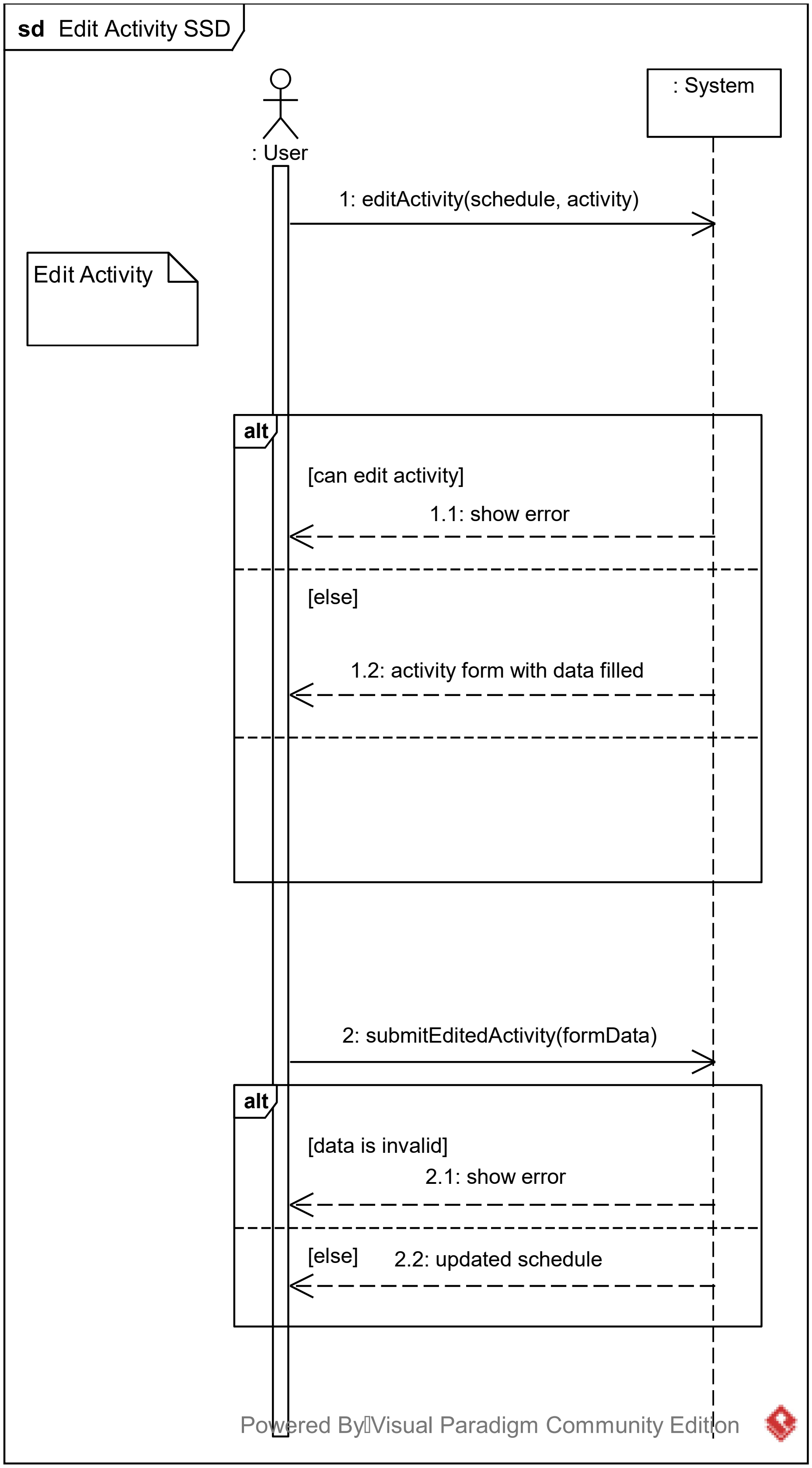

sd Create Activity SSD
Create Activity
: User
: System
1: createActivity(schedule)
alt
[can edit activity]
1.1: show error
[else]
1.2: activity form
2: submitNewActivity(formData)
alt
[data is invalid]
2.1: show error
[else]
2.2: updated schedule
Powered By Visual Paradigm Community Edition

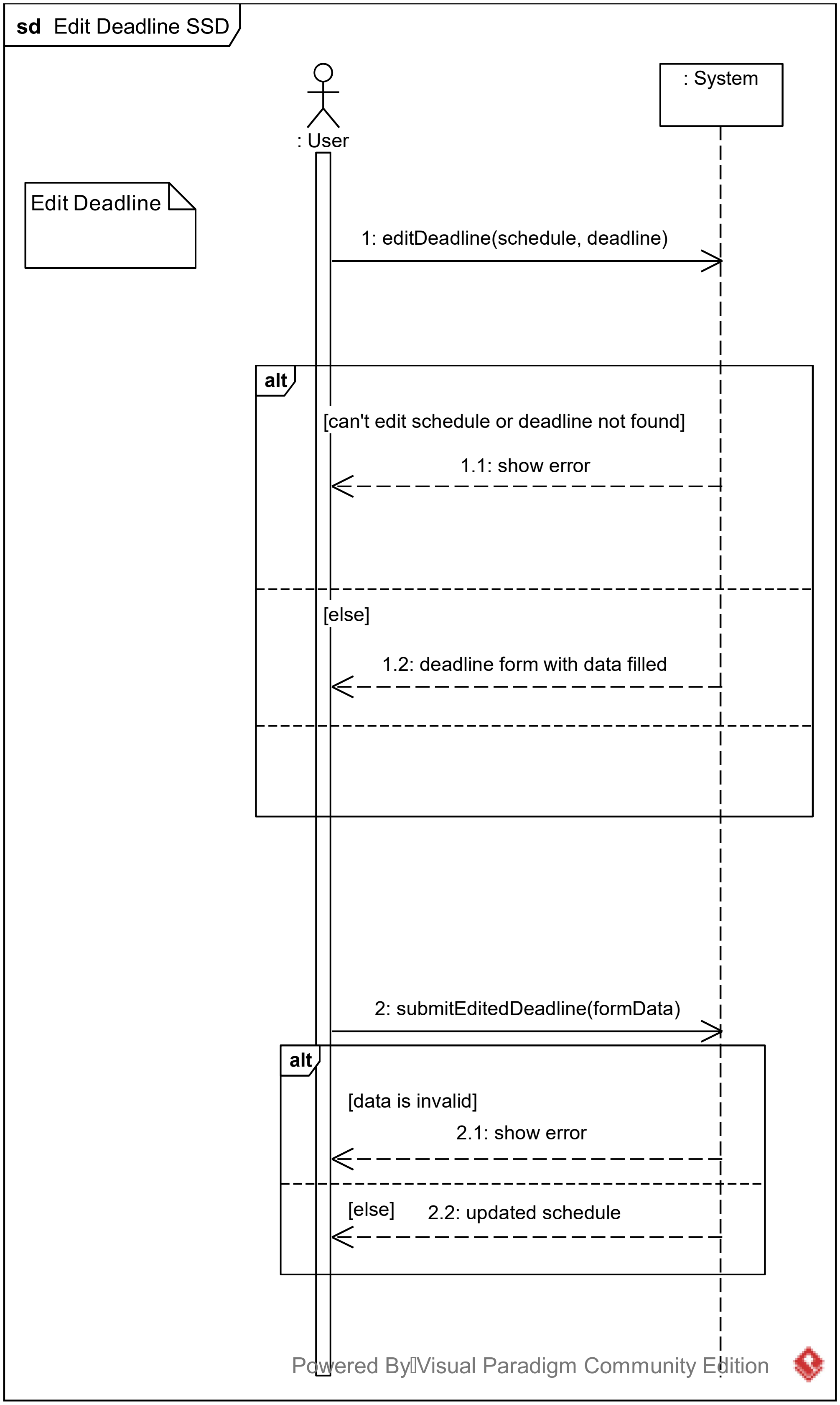

sd Create Deadline SSD
Create Deadline
: User
: System
1: createDeadline(schedule)
alt
[can't edit schedule]
1.1: show error
[else]
1.2: deadline form
2: submitNewDeadline(formData)
alt
[data is invalid]
2.1: show error
[else]
2.2: updated schedule
Powered By Visual Paradigm Community Edition

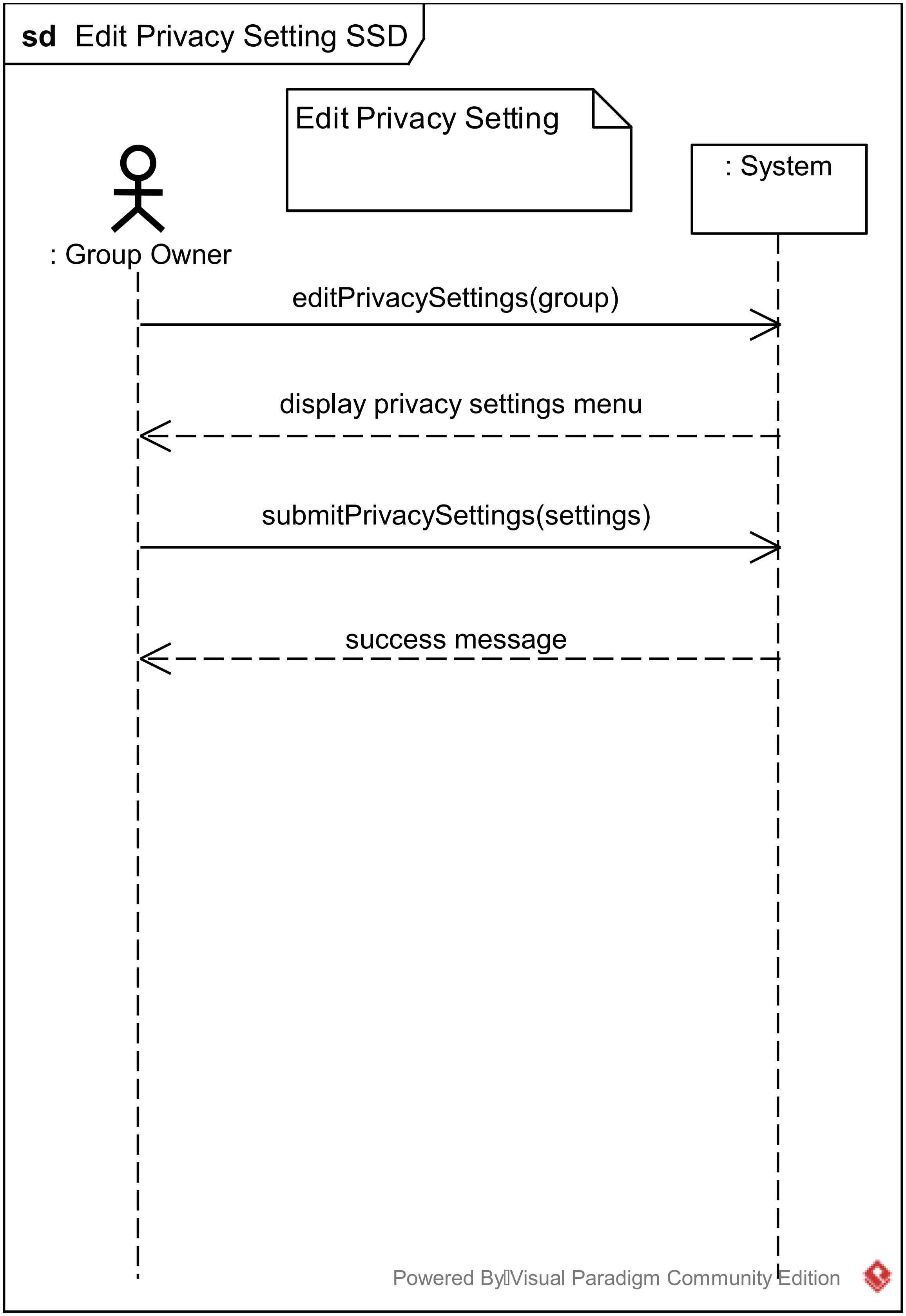

sd CreateGroup
User
Create Group
: System
1: createGroup(name, profiles)
1.1: group created w/profiles
Powered By Visual Paradigm Community Edition

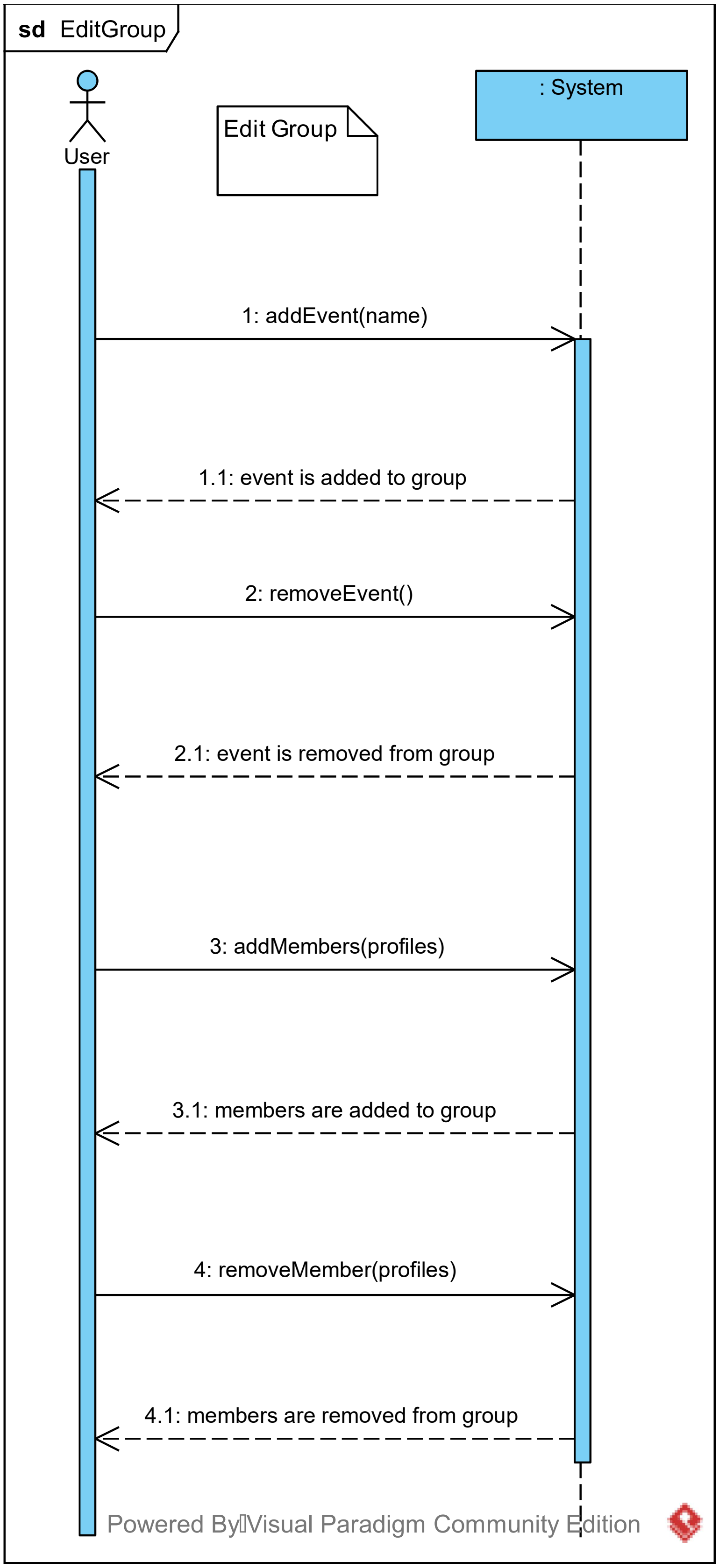

sd CreateProfile
Create Profile
: User
: System
loop
[Until a profile has been created]
1: createProfile(username, email, password)
1.1: isCreated
1.1.1: editProfile()
1.1.1.1: <<return>>
Powered By Visual Paradigm Community Edition

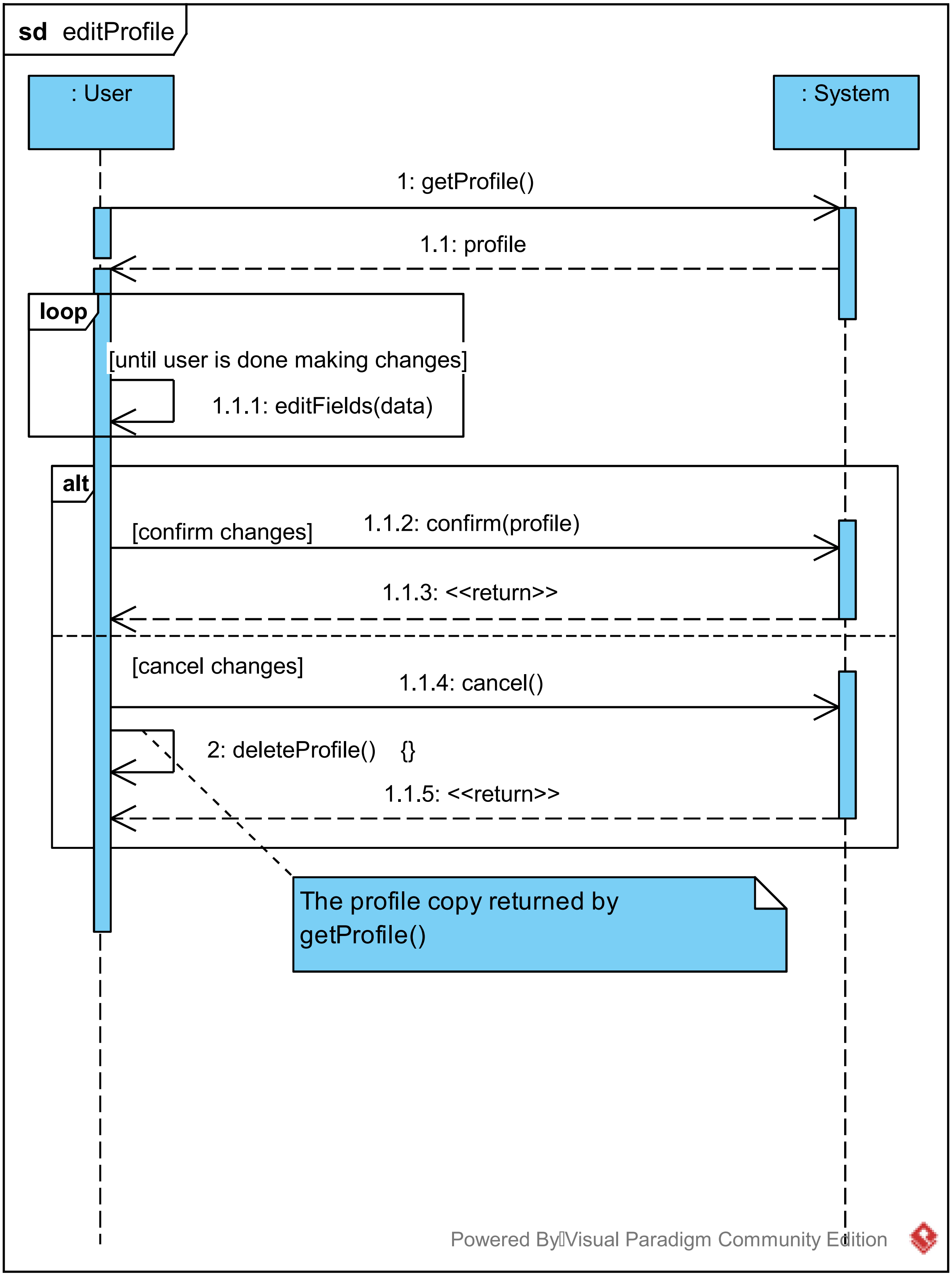

sd DeleteGroup
Delete Group
: System
Group Owner
1: deleteGroup(group)
1.1: delete the desired group
Powered By Visual Paradigm Community Edition

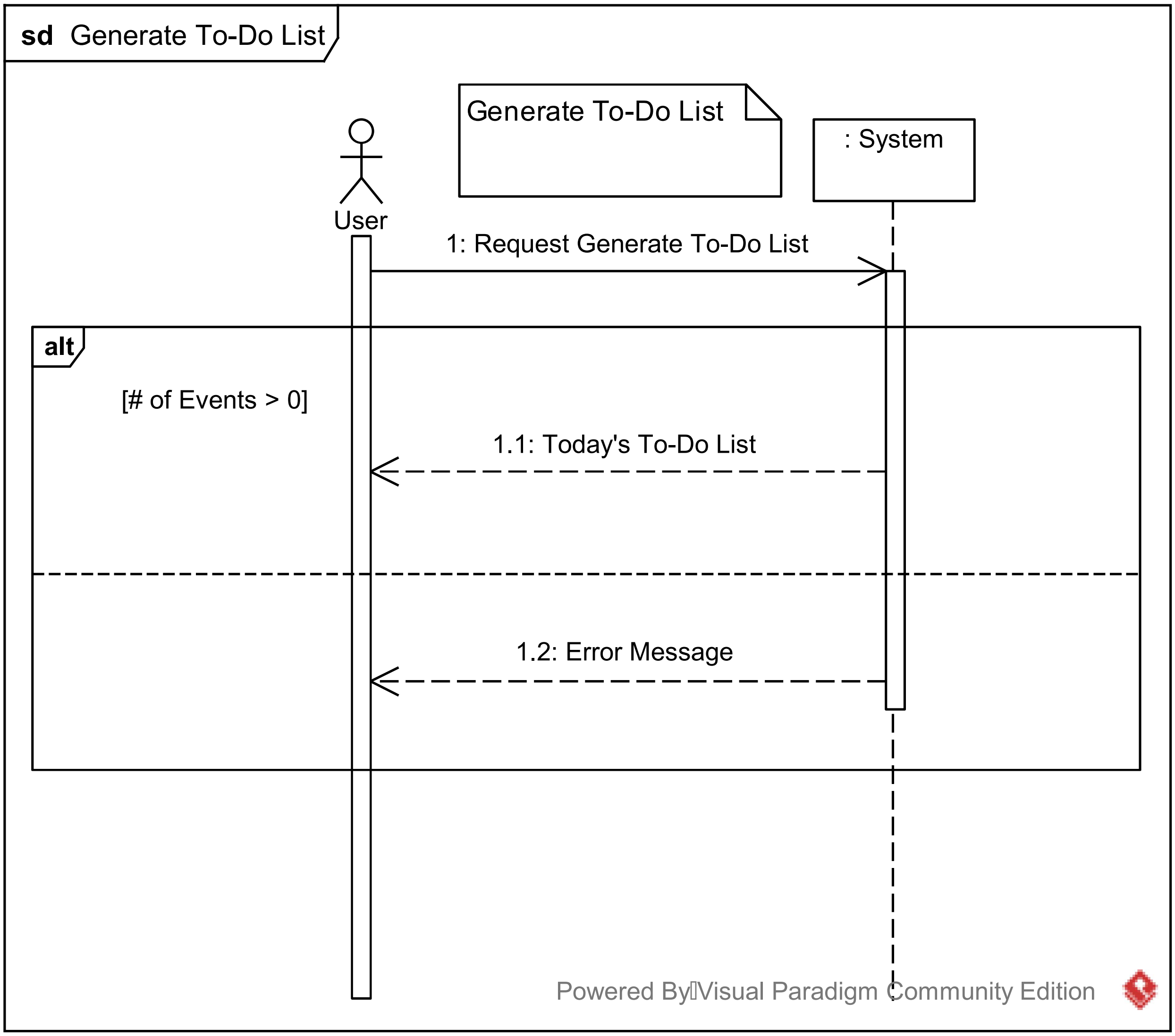

sd Edit Activity SSD
: User
: System
1: editActivity(schedule, activity)
Edit Activity
alt
[can edit activity]
1.1: show error
[else]
1.2: activity form with data filled
2: submitEditedActivity(formData)
alt
[data is invalid]
2.1: show error
[else]
2.2: updated schedule
Powered By Visual Paradigm Community Edition

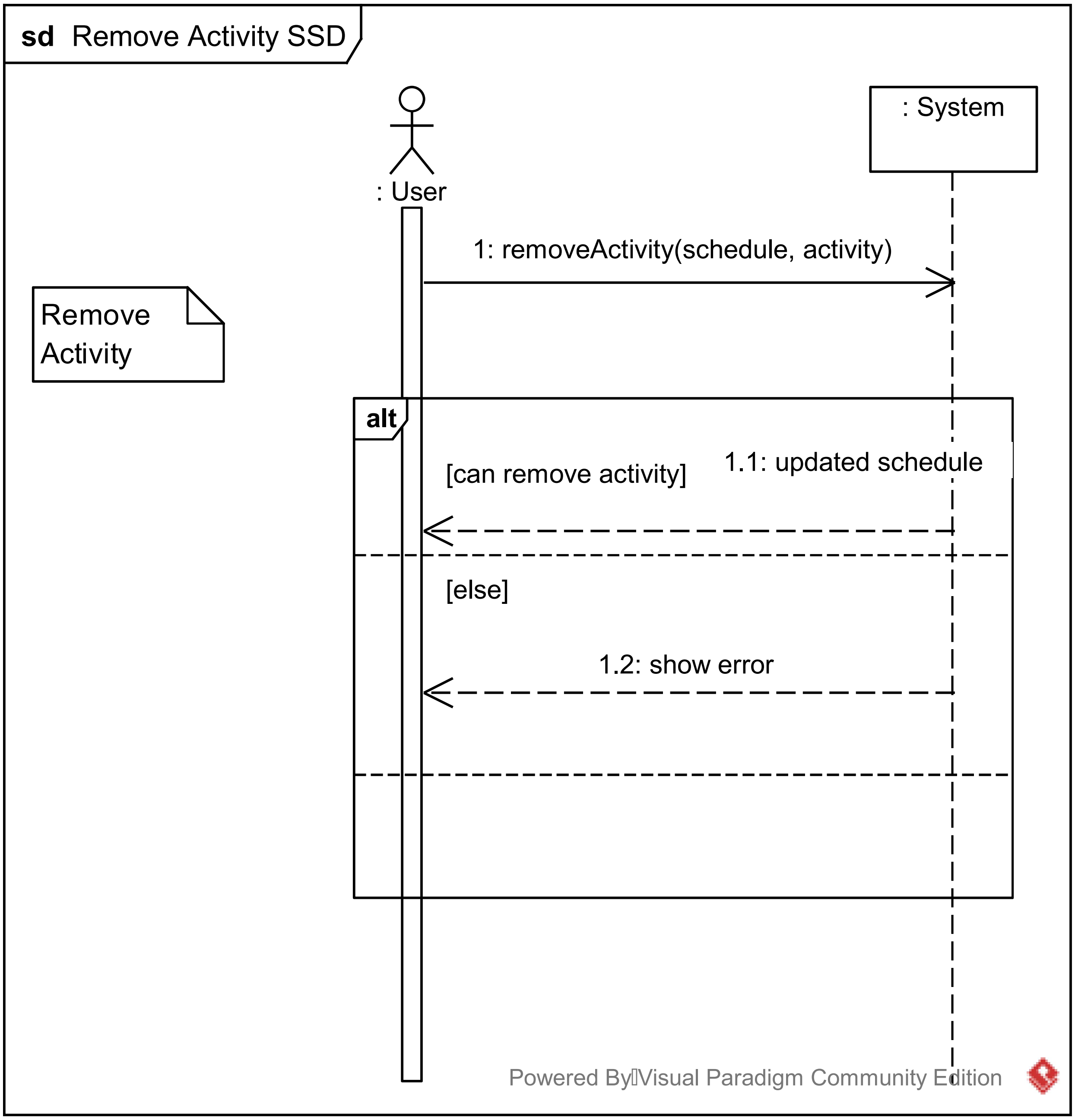

sd Edit Deadline SSD
: User
: System
Edit Deadline
1: editDeadline(schedule, deadline)
alt
[can't edit schedule or deadline not found]
1.1: show error
[else]
1.2: deadline form with data filled
2: submitEditedDeadline(formData)
alt
[data is invalid]
2.1: show error
[else]
2.2: updated schedule
Powered By Visual Paradigm Community Edition

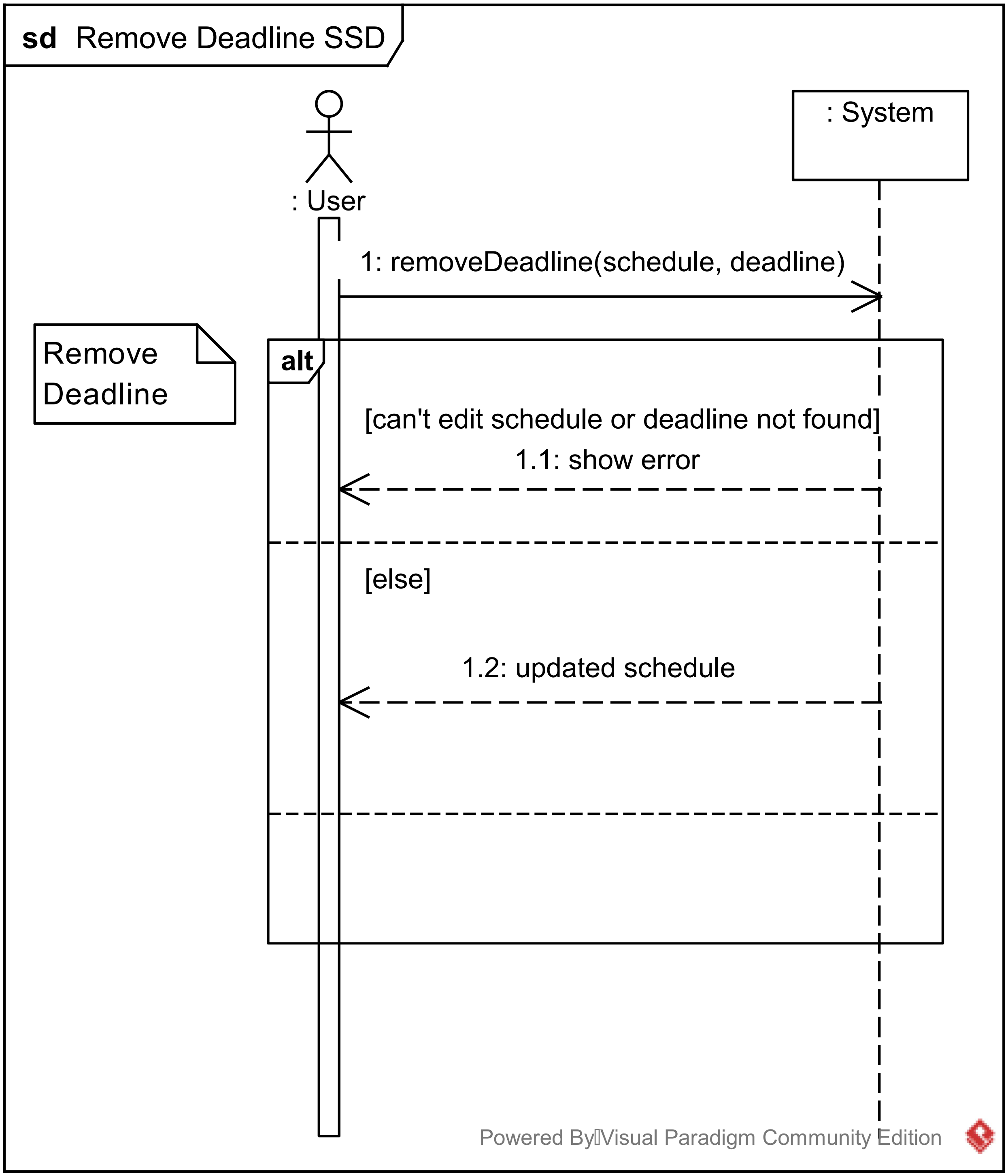

sd Edit Privacy Setting SSD
Edit Privacy Setting
: Group Owner
: System
editPrivacySettings(group)
display privacy settings menu
submitPrivacySettings(settings)
success message
Powered By Visual Paradigm Community Edition

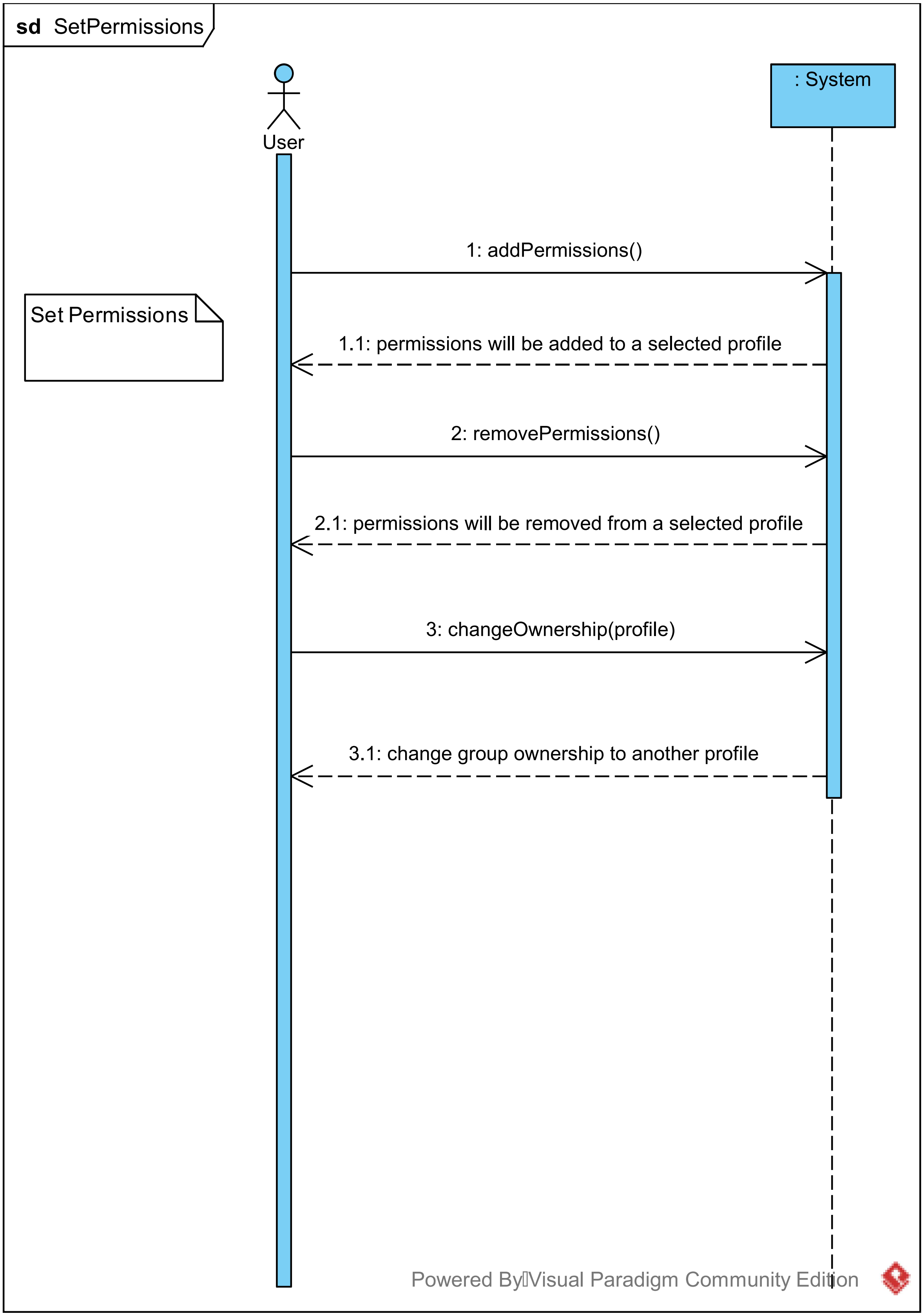

sd EditGroup
User
Edit Group
: System
1: addEvent(name)
1.1: event is added to group
2: removeEvent()
2.1: event is removed from group
3: addMembers(profiles)
3.1: members are added to group
4: removeMember(profiles)
4.1: members are removed from group
Powered By Visual Paradigm Community Edition

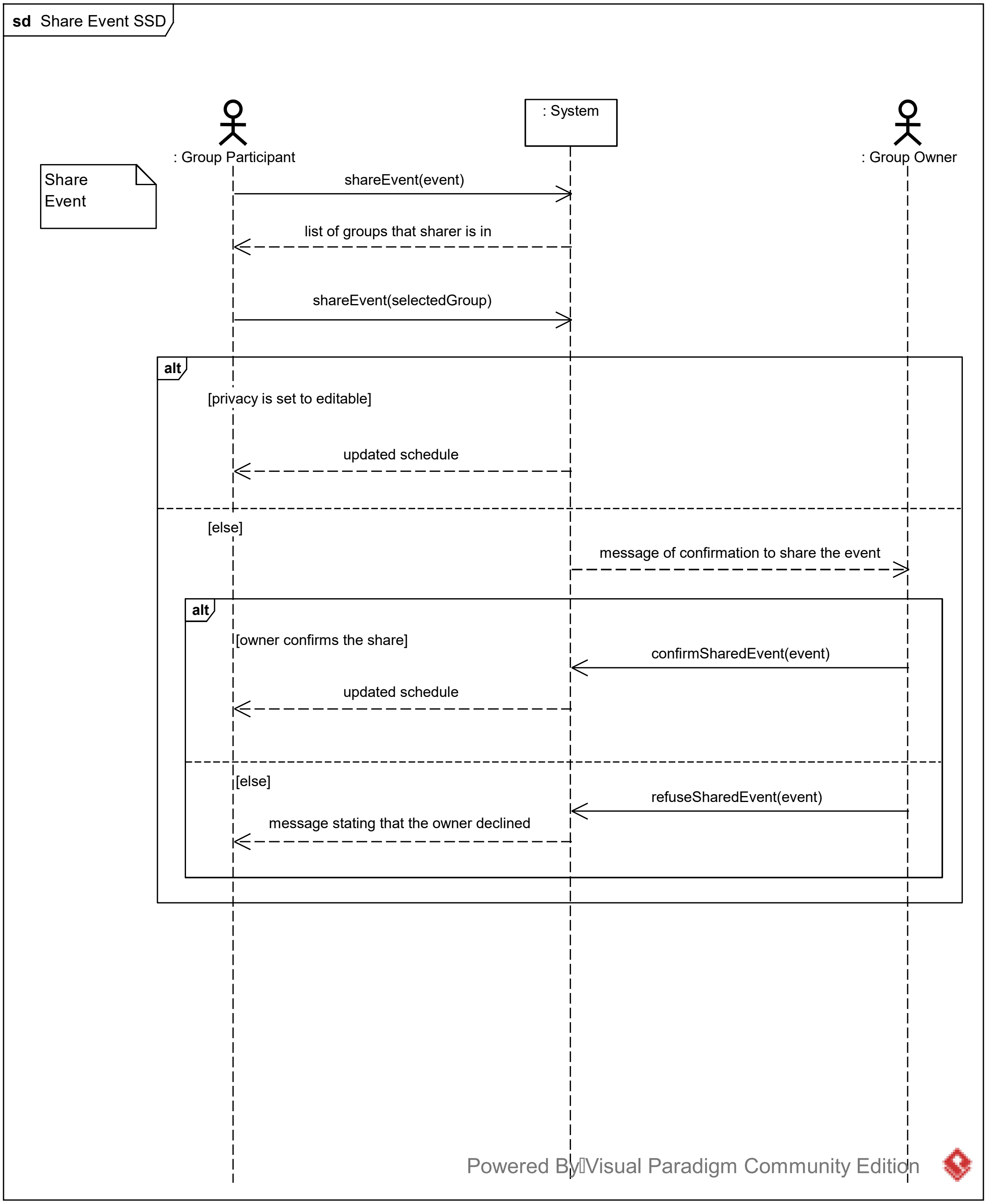

sd editProfile
: User
: System
1: getProfile()
1.1: profile
loop
[until user is done making changes]
1.1.1: editFields(data)
alt
[confirm changes]
1.1.2: confirm(profile)
1.1.3: <<return>>
[cancel changes]
1.1.4: cancel()
2: deleteProfile() {}
1.1.5: <<return>>
The profile copy returned by getProfile()
Powered By Visual Paradigm Community Edition

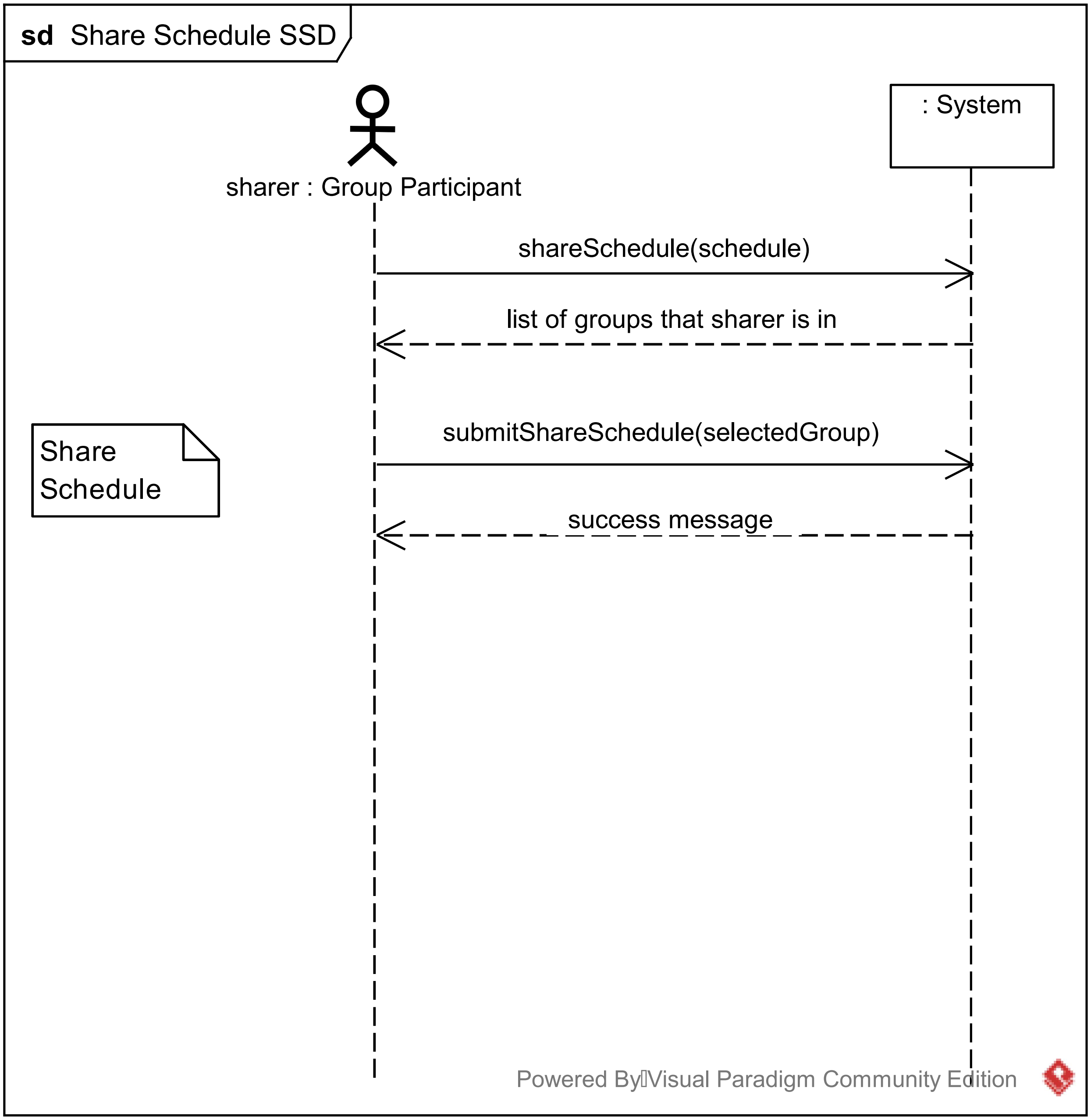

sd Generate To-Do List
Generate To-Do List
: System
User
1: Request Generate To-Do List
alt
[# of Events > 0]
1.1: Today's To-Do List
1.2: Error Message
Powered By Visual Paradigm Community Edition

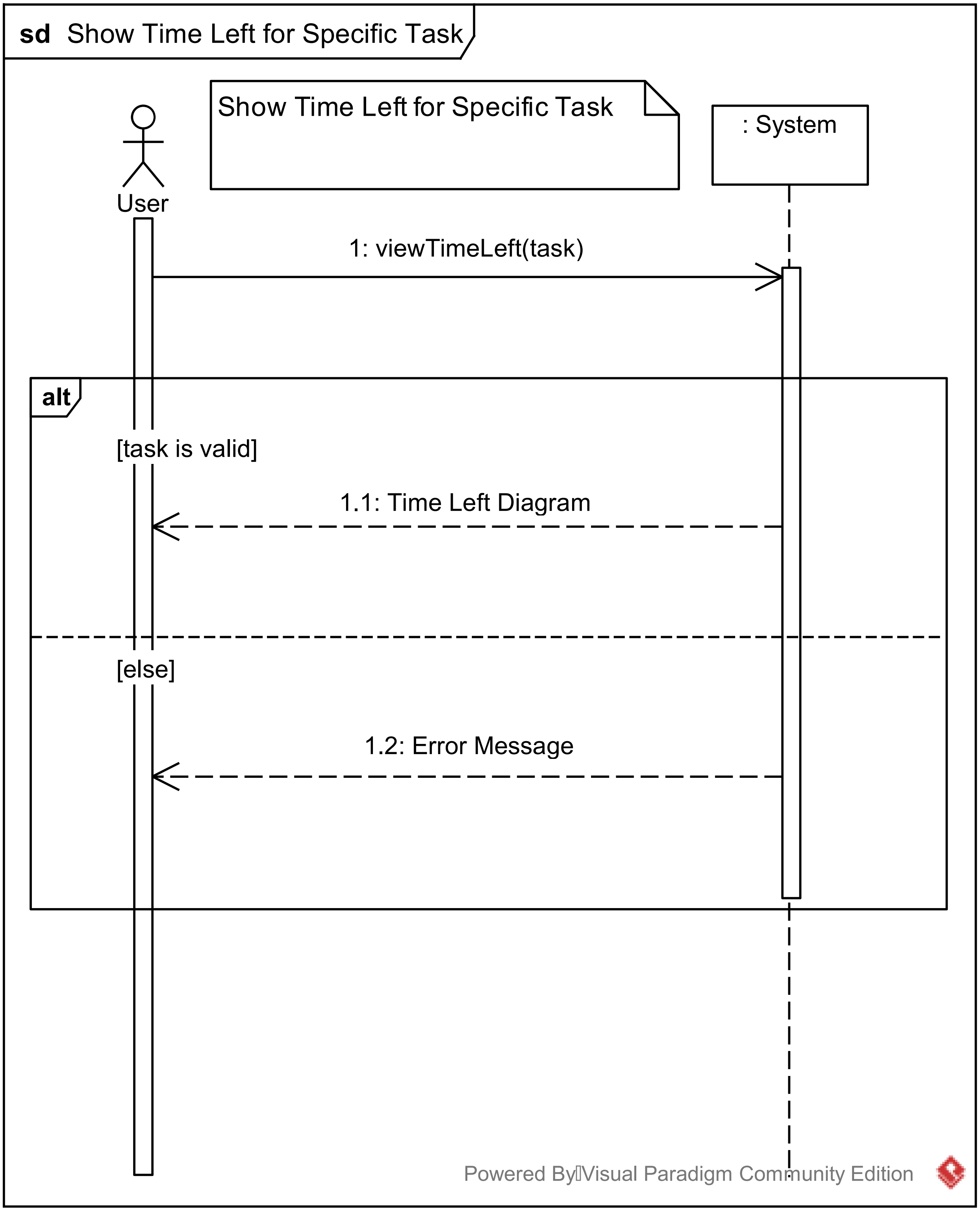

sd Remove Activity SSD
: User
: System
Remove Activity
1: removeActivity(schedule, activity)
alt
[can remove activity]
1.1: updated schedule
[else]
1.2: show error
Powered By Visual Paradigm Community Edition

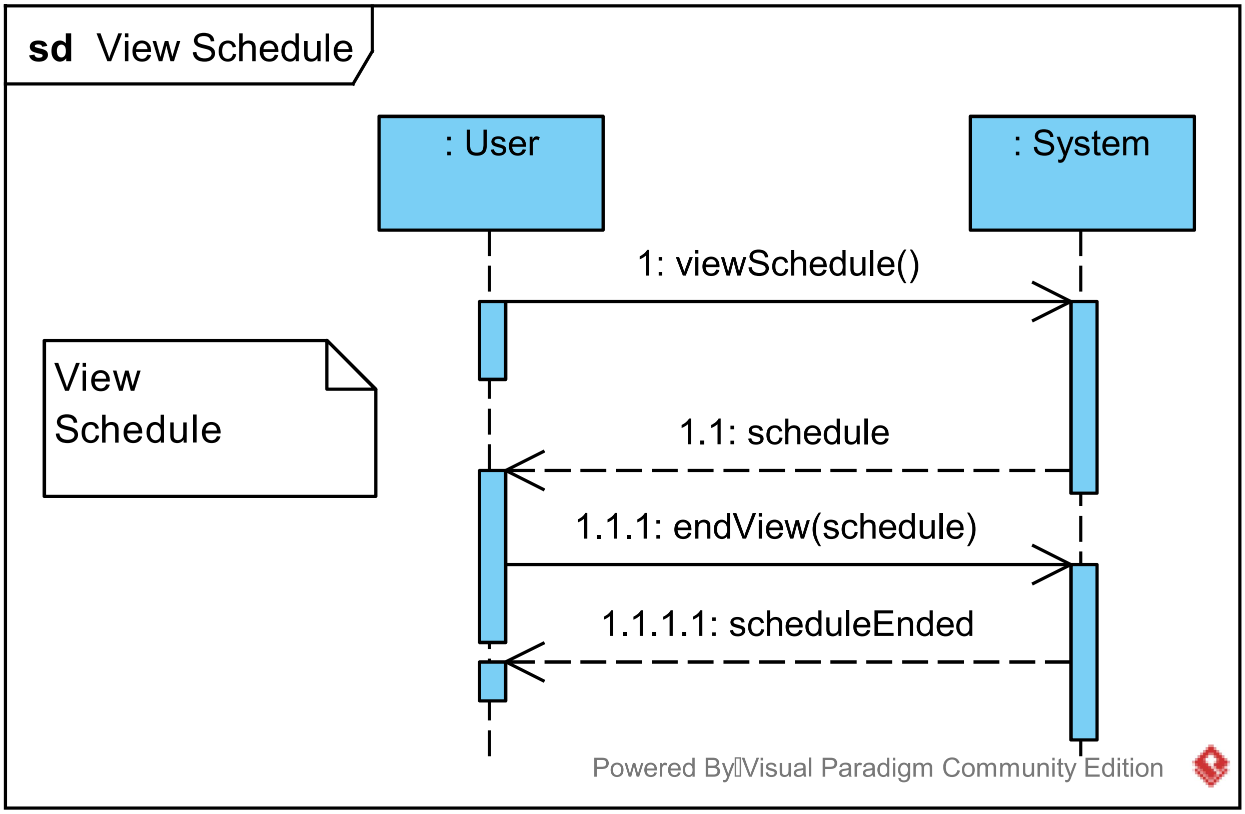

sd Remove Deadline SSD
: User
: System
1: removeDeadline(schedule, deadline)
Remove Deadline
alt
[can't edit schedule or deadline not found]
1.1: show error
[else]
1.2: updated schedule

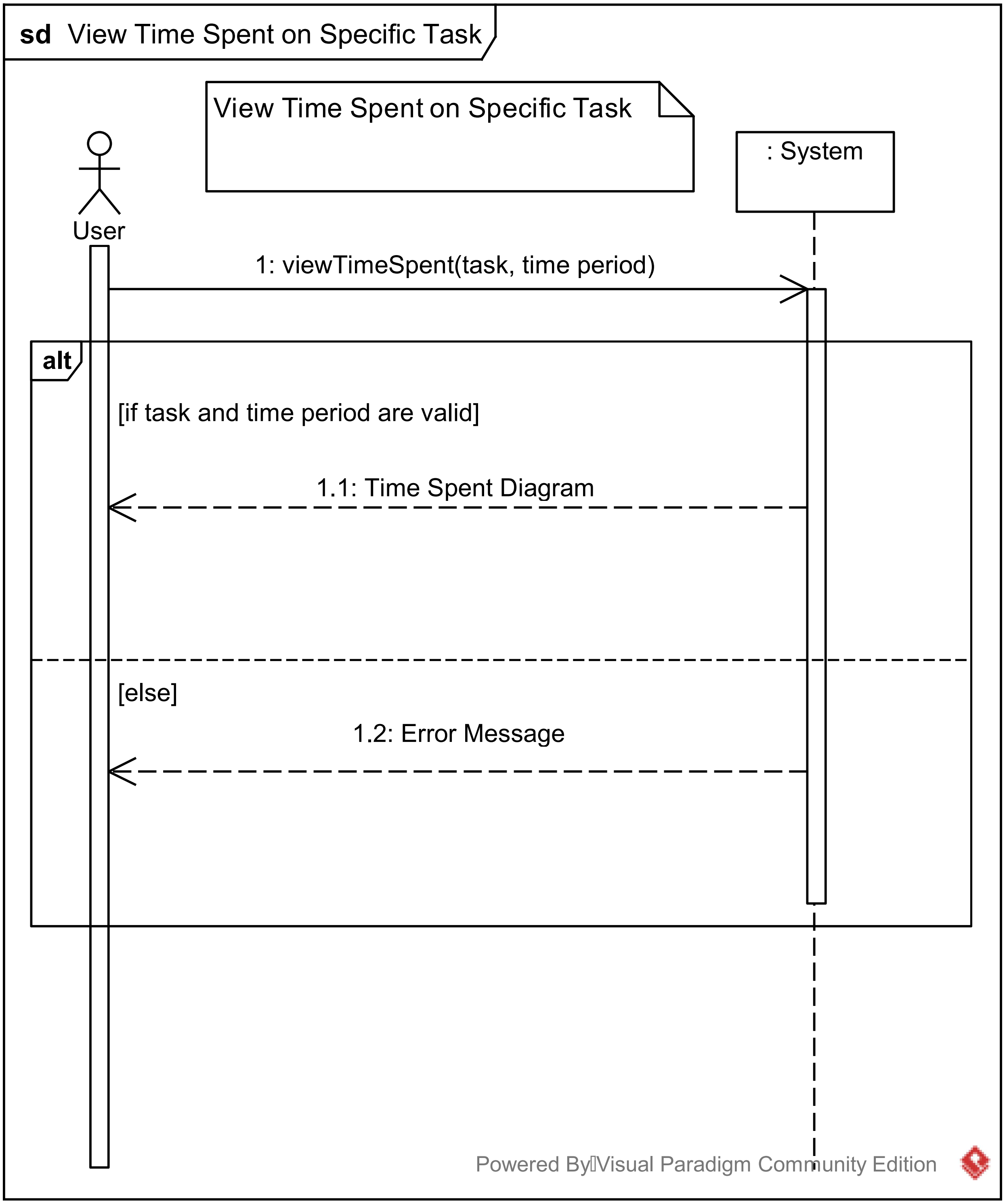

sd SetPermissions
User
: System
Set Permissions
1: addPermissions()
1.1: permissions will be added to a selected profile
2: removePermissions()
2.1: permissions will be removed from a selected profile
3: changeOwnership(profile)
3.1: change group ownership to another profile

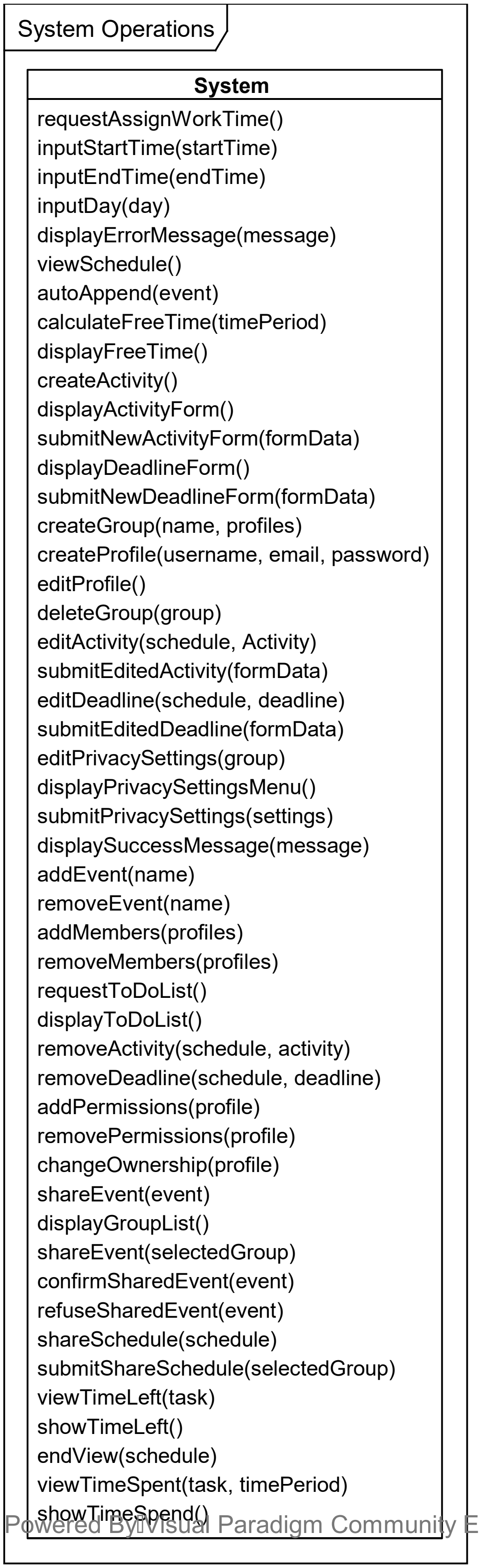

sd Share Event SSD
: Group Participant
: System
: Group Owner
Share Event
shareEvent(event)
list of groups that sharer is in
shareEvent(selectedGroup)
alt
[privacy is set to editable]
updated schedule
[else]
message of confirmation to share the event
alt
[owner confirms the share]
confirmSharedEvent(event)
updated schedule
[else]
refuseSharedEvent(event)
message stating that the owner declined


sd Share Schedule SSD
sharer : Group Participant
: System
shareSchedule(schedule)
list of groups that sharer is in
submitShareSchedule(selectedGroup)
success message
Share Schedule
Powered By Visual Paradigm Community Edition


sd Show Time Left for Specific Task
Show Time Left for Specific Task
User
: System
1: viewTimeLeft(task)
alt
[task is valid]
1.1: Time Left Diagram
[else]
1.2: Error Message
Powered By Visual Paradigm Community Edition


sd View Schedule
: User
: System
1: viewSchedule()
View Schedule
1.1: schedule
1.1.1: endView(schedule)
1.1.1.1: scheduleEnded
Powered By Visual Paradigm Community Edition

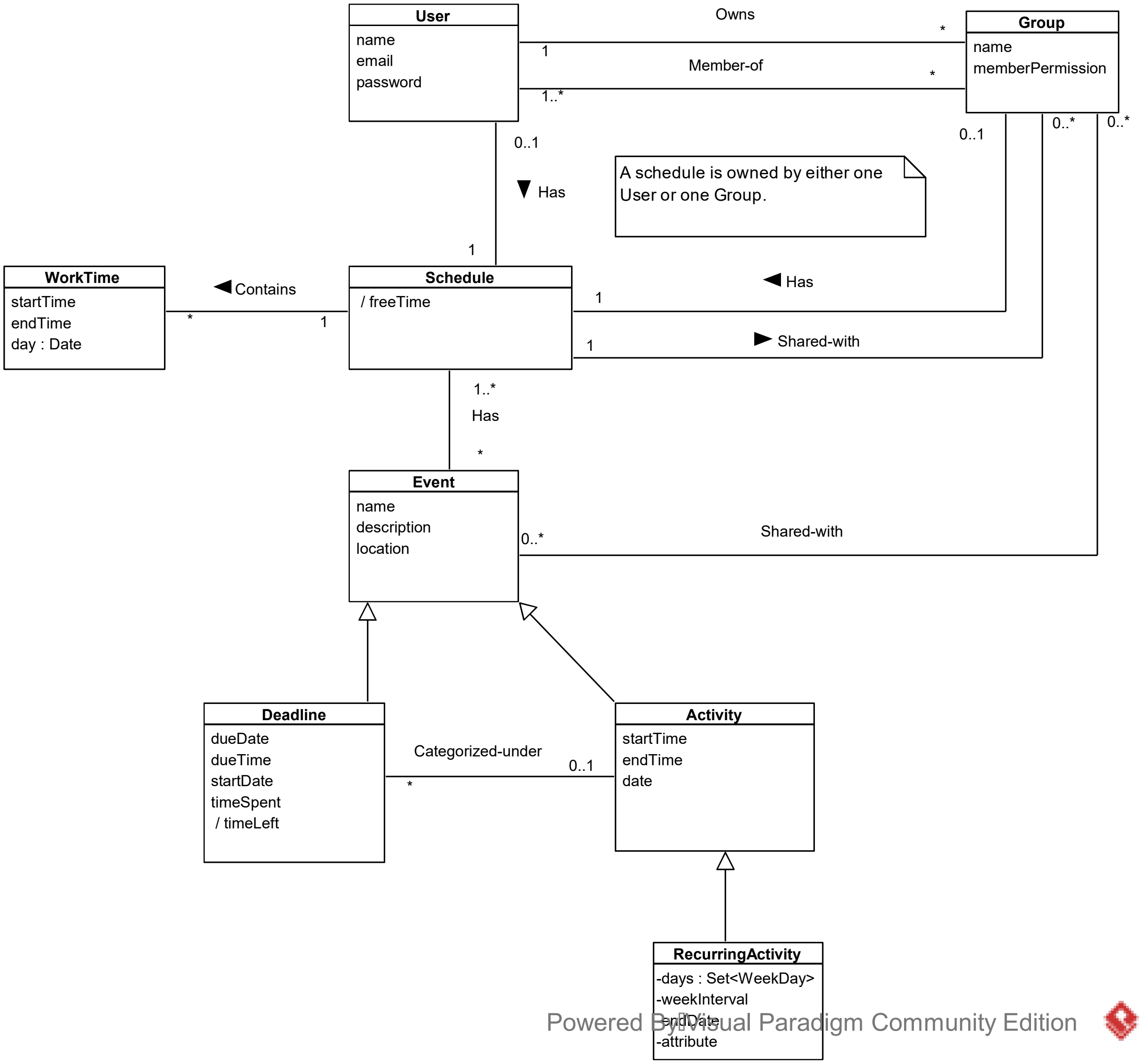

sd View Time Spent on Specific Task
View Time Spent on Specific Task
: System
User
1: viewTimeSpent(task, time period)
alt
[if task and time period are valid]
1.1: Time Spent Diagram
[else]
1.2: Error Message
Powered By Visual Paradigm Community Edition

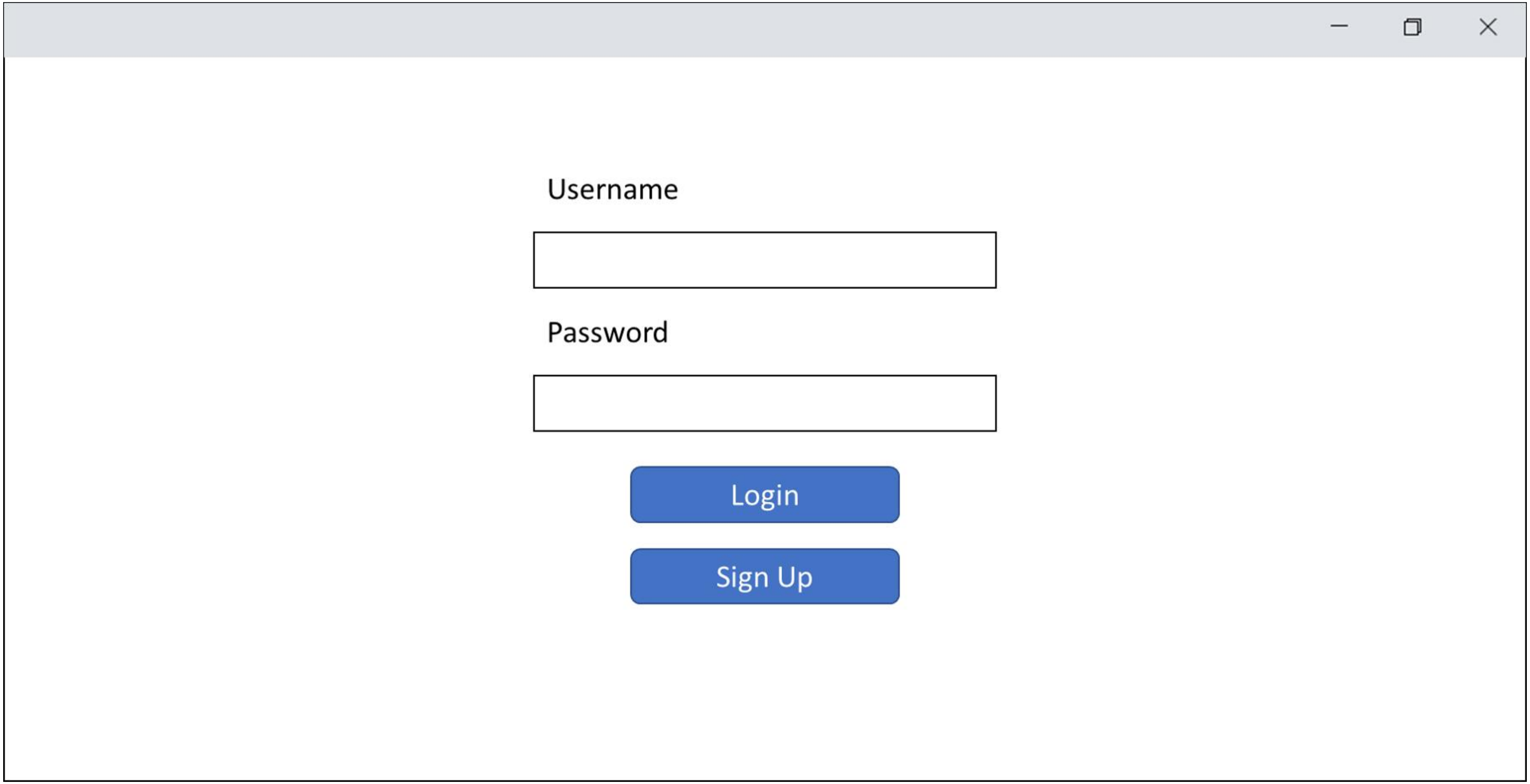


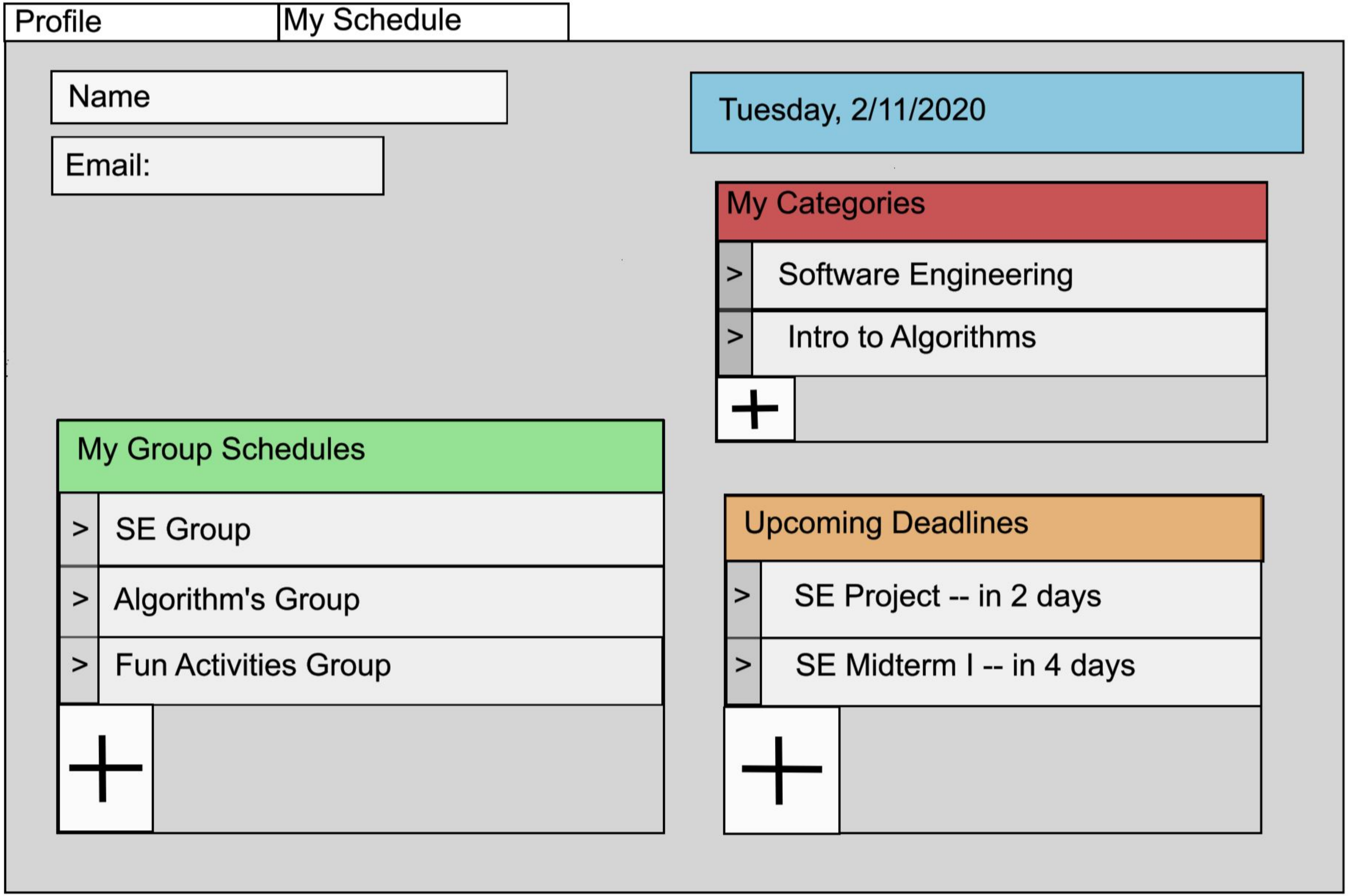
System Operations

System

requestAssignWorkTime()
inputStartTime(startTime)
inputEndTime(endTime)
inputDay(day)
displayErrorMessage(message)
viewSchedule()
autoAppend(event)
calculateFreeTime(timePeriod)
displayFreeTime()
createActivity()
displayActivityForm()
submitNewActivityForm(formData)
displayDeadlineForm()
submitNewDeadlineForm(formData)
createGroup(name, profiles)
createProfile(username, email, password)
editProfile()
deleteGroup(group)
editActivity(schedule, Activity)
submitEditedActivity(formData)
editDeadline(schedule, deadline)
submitEditedDeadline(formData)
editPrivacySettings(group)
displayPrivacySettingsMenu()
submitPrivacySettings(settings)
displaySuccessMessage(message)
addEvent(name)
removeEvent(name)
addMembers(profiles)
removeMembers(profiles)
requestToDoList()
displayToDoList()
removeActivity(schedule, activity)
removeDeadline(schedule, deadline)
addPermissions(profile)
removePermissions(profile)
changeOwnership(profile)
shareEvent(event)
displayGroupList()
shareEvent(selectedGroup)
confirmSharedEvent(event)
refuseSharedEvent(event)
shareSchedule(schedule)
submitShareSchedule(selectedGroup)
viewTimeLeft(task)
showTimeLeft()
endView(schedule)
viewTimeSpent(task, timePeriod)
showTimeSpend()


User
name
email
password
Owns
1
*
Member-of
1..*
*
Group
name
memberPermission
0..1
Has
1
0..1
0..*
0..*
A schedule is owned by either one User or one Group.
WorkTime
startTime
endTime
day : Date
Contains
*
1
Schedule
/ freeTime
Has
1
Shared-with
1
1..*
Has
*
Event
name
description
location
0..*
Shared-with
Deadline
dueDate
dueTime
startDate
timeSpent
/ timeLeft
Categorized-under
*
0..1
Activity
startTime
endTime
date
RecurringActivity
-days : Set<WeekDay>
-weekInterval
-endDate
-attribute


Username
Password
Login
Sign Up



Profile

My Schedule

Name

Email:

Tuesday, 2/11/2020

My Categories

> Software Engineering

> Intro to Algorithms

+

My Group Schedules

> SE Group

> Algorithm's Group

> Fun Activities Group

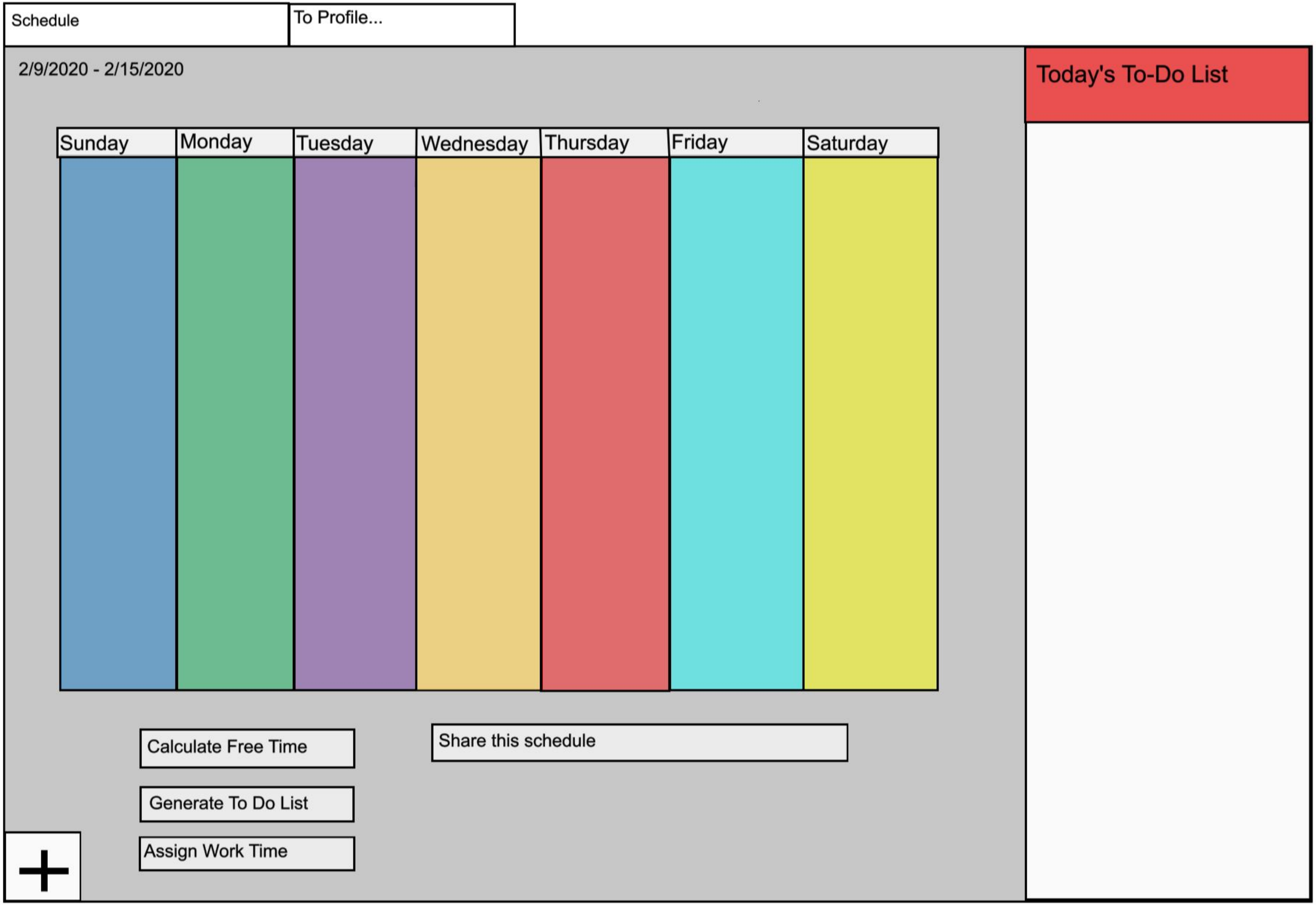
+

Upcoming Deadlines

> SE Project -- in 2 days

> SE Midterm I -- in 4 days

+


Schedule
To Profile...
2/9/2020 - 2/15/2020
Sunday
Monday
Tuesday
Wednesday
Thursday
Friday
Saturday
Calculate Free Time
Generate To Do List
Assign Work Time
Share this schedule
+
Today's To-Do List