**Dream Team**

**Smart Calendar**

**Use Case Report**

***Revision History***

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Authors** | **Description of Change** | **Sections** | **Rev** | **Date** |
|  |  |  |  |  |
|  |  |  |  |  |

**Table of Contents**

[1 Team Description 4](#_30j0zll)

[2 Project Description 4](#_1fob9te)

[2.1 Use Case Diagram 5](#_3znysh7)

[2.2 Use Case List 6](#_2et92p0)

[2.2.1 Use Case 1:](#_3dy6vkm) Inputting schedule 7

[2.2.2](#_1t3h5sf) Use Case 2: Editing schedule 7

2.2.3Use Case 3: Create Task 8

2.2.4 Use Case 4: Schedule tasks on free time slots 8

2.2.5 Use Case 5: Manage tasks 9

2.2.6 Use Case 6: Start task timer 10

2.2.7 Use Case 7: End task timer 10

2.2.8 Use Case 8: Remind user 10

# Team Description

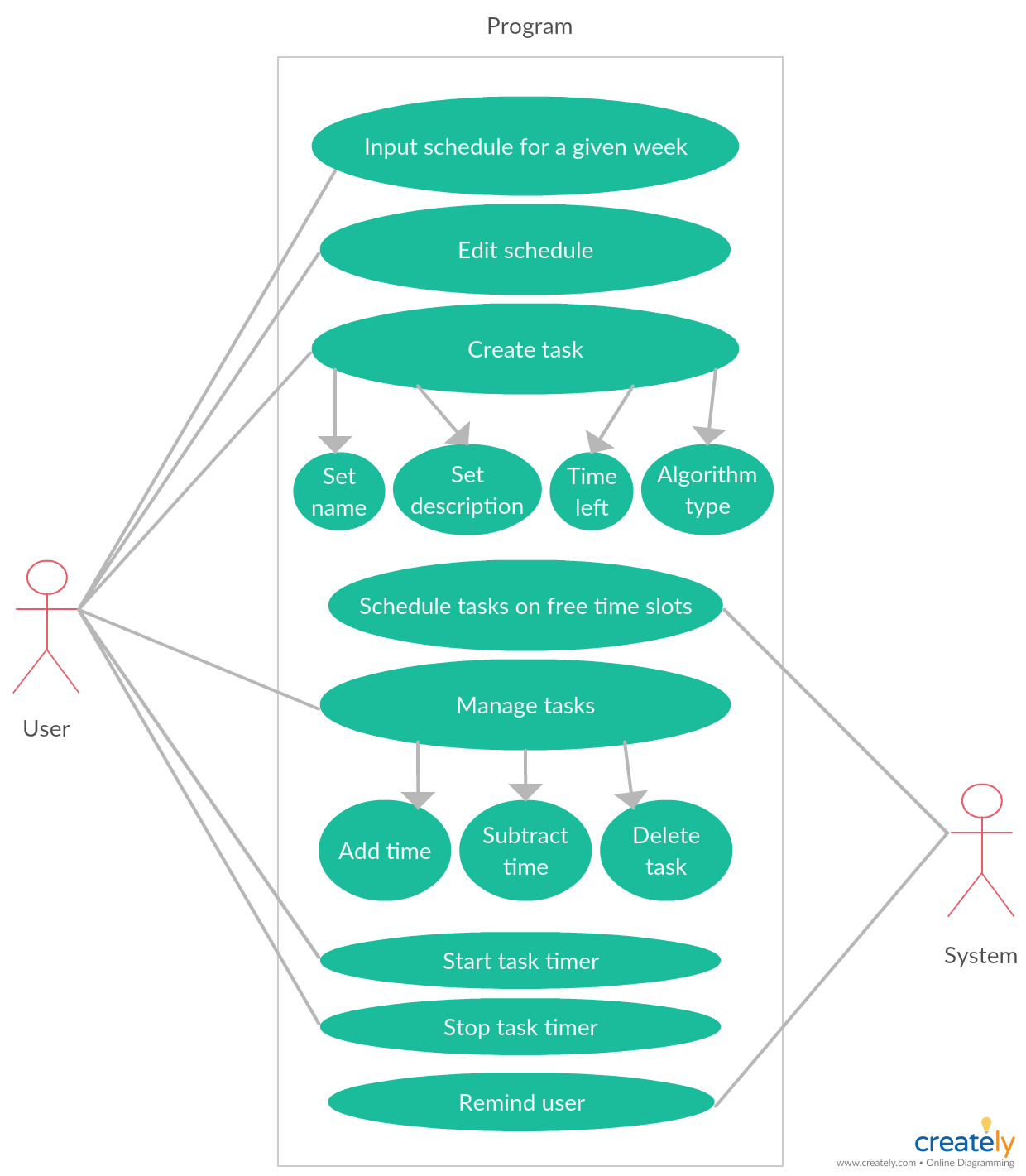
|  |  |
| --- | --- |
| **Team Member Name** | **Email Address** |
| Eric Guzman | eg70889@csu.fullerton.edu |
| Adam Weesner | klaviam@gmail.com |
| Jesus Rodriguez | jrjssrdrgz@csu.fullerton.edu |
| Michael Perez | mperez980089@yahoo.com |
| Henry Yang | hkyang@csu.fullerton.edu |
|  |  |

# Project Description

Statement of Purpose: To make it as convenient as possible for users to schedule and keep track of their progress on whatever task they input.

Detailed Description: This application will allow the user to input their schedule for the week. To do this, user’s will assign blocks of time on the UI for things such as Sleep and Classes. These blocks of time can be of any length, as long as it’s an increment of 15 minutes. After inputting a schedule, the user can then create “Tasks”, which are things such as “Study for exam”, “Prepare for interview”, “Do CPSC homework”, and so on. Tasks have a name, description, due date, amount of time the user wants to spend on it, and algorithm type. Once the task is input, the application will automatically place it in the free time on the user’s schedule depending on which algorithm is used. Two algorithm types will be available: “Slow and steady”, and “Ripping the bandaid”. The former will attempt to cut the task into segments and spread them out up until the due date (Ex: cutting a 4 hour task into four 1 hour segments), while the latter will try to keep the task in large blocks of time on your schedule (Ex: Keeping the 4 hour task as a 4 hour block on one day). Users will receive alerts when the task’s due date is approaching. Users can add or subtract times, as well as delete tasks entirely.

## Use Case Diagram

****

## Use Case List

|  |  |  |
| --- | --- | --- |
| **Use Case** | | |
| **Sequence Number** | **Actor** | **Goal** |
| 1 Inputting Schedule | User | Define the user schedule and free time |
| 2 Editing Schedule | User | Edit the time blocks in the current schedule |
| 3 Create Task | User | Create a task object; includes name, description, due date, time needed to complete the task, and algorithm preference |
| 4 Schedule Tasks | System | Assign tasks to the free time slots; based on algorithm choice |
| 5 Manage Tasks | User | Manage specific properties of the already assigned tasks |
| 6 Start Task Timer | User | Begin the timer on a specific task |
| 7 Stop Task Timer | User | End the timer for a specific task |
| 8 Remind User | System | Remind the user to start on a scheduled task |

// Ask the user if they worked on task? How much time was completed?

### Use Case 1: Inputting schedule

Primary Actor: User

Secondary Actor(s): None

Goal in Context: To define what the user’s schedule for the week is and, by nature, when their free time is

Preconditions: None

Additional Description: Users can define schedules for either the present week or any future week. Inputting a schedule will be done through a UI, where a user creates a block on the calendar UI that can be given a name. Blocks can only be a time that is an increment of 15.

### Use Case 2: Editing schedule

Primary Actor: User

Secondary Actor(s): None

Goal in Context: To edit the blocks on the user’s current schedule

Preconditions: A schedule already entered

Additional Description: Blocks can be added or deleted. Already existing blocks can have their start and end points shifted 15 minutes forward or backwards.

#### 2.2.2.1 Scenario 1: Shifting block start/end point into an already existing block

Application alerts the user that there is a scheduling conflict, and prevents the action.

**2.2.3 Use Case 3: Create Task**

Primary Actor: User

Secondary Actor(s): None

Goal in Context: To create a task with a name, description, due date, time needed to complete the task, and the algorithm.

Preconditions: None

Additional Description: The user will be prompted to enter a name, description, due date, time needed to complete the task, and the algorithm to be used when the system designates where in the user’s free time it’ll place the task.

**2.2.4 Use Case 4: Schedule tasks on free time slots**

Primary Actor: System

Secondary Actor(s): None

Goal in Context: To assign created tasks to free time slots depending on which algorithm the tasks are assigned

Preconditions: Tasks have been created

Additional Description: The user can press a button to make the system place the task somewhere in the user’s free time blocks. Depending on the algorithm chosen, the task could either be split up into small segments over the course of the week, or it could be kept as one chunk to be done on a single day.

#### 2.2.4.1 Scenario 1: Trying to schedule the tasks when the total amount of task time surpasses the total amount of free time

Application alerts the user that there isn’t enough free time to handle all of the tasks, saying that they need to manually edit their tasks or schedule to create free time.

**2.2.5 Use Case 5: Manage Tasks**

Primary Actor: User

Secondary Actor(s): None

Goal in Context: To manage specific properties about the tasks already placed in the user’s schedule

Preconditions: Tasks have been placed in the schedule

Additional Description: The user can manage already created tasks in the following ways: adding time, subtracting time, and deleting the task.

2.2.5.1 Scenario 1: Add time scenario

Adds time to a specific task, by increments of 15 that round to the nearest increment of 15

2.2.5.2 Scenario 2: Subtract time scenario

Subtracts time from a specific task, by decrements of 15 that round to the nearest increment of 15

2.2.5.3 Scenario 3: Delete task scenario

Deletes a selected task from the schedule

**2.2.6 Use Case 6: Start task timer**

Primary Actor: User

Secondary Actor: None

Goal in Context: To start the timer on the selected task

Preconditions: Task has been placed in the user’s schedule at that specific time

Additional Description: There will be a button for the user to interact with, and upon doing so, the task’s timer will begin counting down.

**2.2.7 Use Case 7: Stop task timer**

primary Actor: User

Secondary Actor: None

Goal in Context: To stop the timer on the selected task

Preconditions: Timer is currently running

Additional Description: There will be a button for the user to interact with, and upon doing so, the task’s currently running timer will stop.

**2.2.8 Use Case 8: Remind User**

Primary Actor: System

Secondary Actor: None

Goal in Context: For the system to tell the user that they should start on a scheduled task

Preconditions: A schedule has been made and is populated with at least one task

Additional Description: System will give the user a reminder to start a task that has been scheduled.