

# SE 3XA3: Module Interface Specification

## ScrumBot

Team 304, ScrumBot  
Arkin Modi, modia1  
Leon So, sol4  
Timothy Choy, choyt2

Last Updated: March 13, 2020

Table 1: Revision History

<b>Date</b>	<b>Developer(s)</b>	<b>Change</b>
March 9, 2020	Timothy Choy	Create template, ScrumBot Module
March 10, 2020	Leon So	MeetingList module
March 11, 2020	Leon So	Meeting, MeetingTypes Modules
March 11, 2020	Leon So	Project, ProjectList, Meeting, MeetingTypes, Generic Dict Modules
March 11, 2020	Timothy Choy	Scrumbot Module
March 12, 2020	Leon So	Rename Meeting Module to Generic Meeting Module
March 12, 2020	Timothy Choy	Generic Meeting Module, Fixed Formatting
March 12, 2020	Arkin Modi	Fix Formatting
March 12, 2020	Leon So	Dict, Meeting, Project, Task, TaskList, Sprint Modules
March 12, 2020	Timothy Choy	Project, Meeting Modules
March 12, 2020	Timothy Choy	Completed ScrumBot module
March 13, 2020	Arkin Modi	Updated ScrumBot Module

# ScrumBot Module

## Module

AdminCog, ProjectCog, MeetingCog, MemberCog, Scrumbot, SprintCog, TaskCog

## Uses

discord  
discord.ext.commands  
MeetingTypes  
Generic Dictionary  
MeetingList  
Meeting  
TaskList  
Task  
Sprint  
ProjectList  
Project

## Syntax

### Exported Constants

None

### Exported Types

None

### Exported Access Programs

Routine Name	In	Out	Exceptions
on_ready			

Table 2: ScrumBot Programs

Routine Name	In	Out	Exceptions
load	commands.Cogs	String	MissingRole, Exception
reload	commands.Cogs	String	MissingRole, Exception
unload	commands.Cogs	String	MissingRole, Exception

Table 3: Admin Programs

<b>Routine Name</b>	<b>In</b>	<b>Out</b>	<b>Exceptions</b>
get_roles		discord.embed	
get_roles	String	discord.embed	

Table 4: Member Programs

<b>Routine Name</b>	<b>In</b>	<b>Out</b>	<b>Exceptions</b>
add_meeting	$\mathbb{N}$ , String, Date, Time, MeetingT	String	MissingRole
add_meeting	$\mathbb{N}$ , String, Date, Time, MeetingT, String	String	MissingRole
add_project	String	String	MissingRole
add_project	String, String	String	MissingRole
add_rqe	$\mathbb{N}$ , String	String	MissingRole
add_sprint	$\mathbb{N}$	String	MissingRole
get_project_desc	$\mathbb{N}$	discord.embed	
get_rqes	$\mathbb{N}$	discord.embed	
get_sprints	$\mathbb{N}$	discord.embed	
list_meetings	$\mathbb{N}$	discord.embed	
list_projects		discord.embed	
rm_last_sprint	$\mathbb{N}$	String	MissingRole
rm_meeting	$\mathbb{N}$ , $\mathbb{N}$	String	MissingRole
rm_project	$\mathbb{N}$	String	MissingRole
rm_rqe	$\mathbb{N}$	String	MissingRole
set_project_desc	$\mathbb{N}$ , String	String	MissingRole

Table 5: Project Programs

<b>Routine Name</b>	<b>In</b>	<b>Out</b>	<b>Exceptions</b>
get_meeting_desc	$\mathbb{N}$ , $\mathbb{N}$	discord.embed	
set_meeting_desc	$\mathbb{N}$ , $\mathbb{N}$ , String	String	MissingRole

Table 6: Meeting Programs

<b>Routine Name</b>	<b>In</b>	<b>Out</b>	<b>Exceptions</b>
add_task	$\mathbb{N}$ , $\mathbb{N}$ , String, Date, Time	String	MissingRole
add_task	$\mathbb{N}$ , $\mathbb{N}$ , String, Date, Time, String	String	MissingRole
list_tasks	$\mathbb{N}$ , $\mathbb{N}$	discord.embed	
rm_task	$\mathbb{N}$ , $\mathbb{N}$ , $\mathbb{N}$	String	MissingRole

Table 7: Sprint Programs

<b>Routine Name</b>	<b>In</b>	<b>Out</b>	<b>Exceptions</b>
add_feedback	N, N, N, String	String	MissingRole
get_details	N, N, N	discord.embed	
list_feedback	N, N, N	discord.embed	
rm_feedback	N, N, N, N	String	MissingRole
set_details	N, N, String	String	MissingRole

Table 8: Task Programs

## Semantics

### Environment Variables

TOKEN: A string constant containing the information linking ScrumBot to Discord's API

### State Variables

startup\_extensions: A list of command cogs.

### State Invariant

None

### Assumptions

This module is the interface between the users and the code. The input taken for the export access programs are text-based commands in Discord, and the output is also in text channels as text output. The methods also keep an output in the terminal as a log of commands used.

The exception to this assumption is the access routine on\_ready as the output for the method is found in the terminal only. Due to this, there is no output value in the exported access programs table.

Though it is comprised of multiple files with many different purposes, the main purpose for these files are the same. For this reason, they have been grouped together into one module, with the secret being communication between the client from Discord and the system.

It is assumed that author refers to the user that initiated the command.

### Access Routine Semantics

#### ScrumBot Programs

on\_ready()

- transition: waits for ScrumBot to connect to Discord’s API servers using the TOKEN, and upon successful connection, will print out the bot’s name and id to the terminal. It is a check for successful connection to the server.

## Admin Programs

load(*cog*)

- transition: attempts to load a Cog into ScrumBot through the use of the command `load_extension(cog)`
- output:  $out := (exc \Rightarrow \text{“Error: failed to load”} \parallel cog \mid \neg exc \Rightarrow cog \parallel \text{“loaded”})$
- exception:

	$exc :=$
author_role $\neq$ admin	MissingRole
any issue with loading cog	Exception

reload(*cog*)

- transition: attempts to reload the cog in ScrumBot through unloading the cog then loading the cog
- output:  $out := (exc \Rightarrow \text{“Error: failed to reload”} \parallel cog \mid \neg exc \Rightarrow cog \parallel \text{“reloaded”})$
- exception:

	$exc :=$
author_role $\neq$ admin	MissingRole
any issue with unloading or loading cog	Exception

unload(*cog*)

- transition: attempts to unload a cog from ScrumBot
- output:  $out := (exc \Rightarrow \text{“Error: failed to unload”} \parallel cog \mid \neg exc \Rightarrow cog \parallel \text{“unloaded”})$
- exception:

	$exc :=$
author_role $\neq$ admin	MissingRole
any issue with unloading cog	Exception

## Member Programs

get\_roles()

- transition: finds all roles that are assigned to the author
- output:  $out :=$  a discord.embed such that it contains the chosen author’s name, icon, and a list of their roles

`get_roles(name)`

- transition: takes in the name of a Discord user and finds all roles that are assigned to that user
- output:  $out :=$  a discord.embed such that it contains the chosen user's name, icon, and a list of their roles

### Project Programs

`add_meeting( $n, name, date, time, meeting$ )`

- transition: new Meeting( $name, date, time, meeting$ ), add it to project  $n$  using add\_meeting
- output:  $out := (exc \Rightarrow \text{"MissingRole: You have insufficient permissions"} \mid \neg exc \Rightarrow \text{"Successfully added meeting"})$
- exception:  $exc := \text{author\_role} \neq \text{Scrum Master} \Rightarrow \text{MissingRole}$

`add_meeting( $n, name, date, time, meeting, desc$ )`

- transition: new Meeting( $name, date, time, meeting, desc$ ), add it to project  $n$  using add\_meeting
- output:  $out := (exc \Rightarrow \text{"MissingRole: You have insufficient permissions"} \mid \neg exc \Rightarrow \text{"Successfully added meeting"})$
- exception:  $exc := \text{author\_role} \neq \text{Scrum Master} \Rightarrow \text{MissingRole}$

`add_project(name)`

- transition: new Project( $name$ )
- output:  $out := (exc \Rightarrow \text{"MissingRole: You have insufficient permissions"} \mid \neg exc \Rightarrow \text{"Successfully added project"})$
- exception:  $exc := \text{author\_role} \neq \text{admin} \Rightarrow \text{MissingRole}$

`add_project(name, desc)`

- transition: new Project( $name, desc$ )
- output:  $out := (exc \Rightarrow \text{"MissingRole: You have insufficient permissions"} \mid \neg exc \Rightarrow \text{"Successfully added project"})$
- exception:  $exc := \text{author\_role} \neq \text{admin} \Rightarrow \text{MissingRole}$

`add_rqe( $n, s$ )`

- transition: add\_rqe( $s$ ) in project  $n$

- output:  $out := (exc \Rightarrow \text{“MissingRole: You have insufficient permissions”} \mid \neg exc \Rightarrow \text{“Successfully added requirements”})$
- exception:  $exc := \text{author\_role} \neq \text{Business Analyst} \Rightarrow \text{MissingRole}$

`add_sprint( $n$ )`

- transition: new Sprint(), then add the sprint to project  $n$  using `add_sprint(sprint)`
- output:  $out := (exc \Rightarrow \text{“MissingRole: You have insufficient permissions”} \mid \neg exc \Rightarrow \text{“Successfully added a sprint”})$
- exception:  $exc := \text{author\_role} \neq \text{Scrum Master} \Rightarrow \text{MissingRole}$

`get_project_desc( $n$ )`

- output:  $out :=$  A discord.embed such that it contains the project name and description of project  $n$

`get_rqes( $n$ )`

- output:  $out :=$  A discord.embed such that it contains the project name and list of requirements of project  $n$

`get_sprints( $n$ )`

- output:  $out :=$  A discord.embed such that it contains the project name and list of sprints of project  $n$

`list_meetings( $n$ )`

- output:  $out :=$  A discord.embed such that it contains the project name and list of meetings of project  $n$ , as well as show the times for all the meetings

`list_projects()`

- output:  $out :=$  A discord.embed containing all the created projects, identified by their name

`rm_last_sprint( $n$ )`

- transition: removes the last sprint of project  $n$  using `rm_sprint()`
- output:  $out := (exc \Rightarrow \text{“MissingRole: You have insufficient permissions”} \mid \neg exc \Rightarrow \text{“Successfully removed a sprint”})$
- exception:  $exc := \text{author\_role} \neq \text{Scrum Master} \Rightarrow \text{MissingRole}$

`rm_meeting( $n, m$ )`



- transition: removes a meeting from project  $n$  using `rm_meeting( $m$ )`
- output:  $out := (exc \Rightarrow \text{"MissingRole: You have insufficient permissions"} \mid \neg exc \Rightarrow \text{"Successfully removed a meeting"})$
- exception:  $exc := \text{author\_role} \neq \text{Scrum Master} \Rightarrow \text{MissingRole}$

`rm_project( $n$ )`

- transition: removes a project by their project id using `remove( $n$ )`
- output:  $out := (exc \Rightarrow \text{"MissingRole: You have insufficient permissions"} \mid \neg exc \Rightarrow \text{"Successfully removed a project"})$
- exception:  $exc := \text{author\_role} \neq \text{admin} \Rightarrow \text{MissingRole}$

`rm_rqe( $n, m$ )`

- transition: remove requirement  $m$  from project  $n$  using `rm_rqe( $m$ )`
- output:  $out := (exc \Rightarrow \text{"MissingRole: You have insufficient permissions"} \mid \neg exc \Rightarrow \text{"Successfully removed a requirement"})$
- exception:  $exc := \text{author\_role} \neq \text{Business Analyst} \Rightarrow \text{MissingRole}$

`set_project_desc( $n, s$ )`

- transition: `set_desc( $s$ )` in project  $n$
- output:  $out := (exc \Rightarrow \text{"MissingRole: You have insufficient permissions"} \mid \neg exc \Rightarrow \text{"Successfully set description"})$
- exception:  $exc := \text{author\_role} \neq \text{Business Analyst} \wedge \text{author\_role} \neq \text{admin} \Rightarrow \text{MissingRole}$

## Meeting Programs

`get_meeting_desc( $n, m$ )`

- output:  $out :=$  A discord.embed such that it contains the meeting description of meeting  $m$  in project  $n$ , showing the meeting name and project name as well

`set_meeting_desc( $n, m, s$ )`

- transition: `set_desc( $s$ )` for meeting  $m$  in project  $n$
- output:  $out := (exc \Rightarrow \text{"MissingRole: You have insufficient permissions"} \mid \neg exc \Rightarrow \text{"Successfully set description"})$
- exception:  $exc := \text{author\_role} \neq \text{Scrum Master} \Rightarrow \text{MissingRole}$

## Sprint Programs

`add_task( $n, m, name, date, time$ )`

- transition: `new Task( $name, date, time$ )` then `add_task(task)` in sprint  $m$ , project  $n$
- output:  $out := (exc \Rightarrow \text{"MissingRole: You have insufficient permissions"} \mid \neg exc \Rightarrow \text{"Successfully added task"})$
- exception:  $exc := \text{author\_role} \neq \text{Scrum Master} \Rightarrow \text{MissingRole}$

`add_task( $n, m, name, date, time, detail$ )`

- transition: `new Task( $name, date, time, detail$ )` then `add_task(task)` in sprint  $m$ , project  $n$
- output:  $out := (exc \Rightarrow \text{"MissingRole: You have insufficient permissions"} \mid \neg exc \Rightarrow \text{"Successfully added task"})$
- exception:  $exc := \text{author\_role} \neq \text{Scrum Master} \wedge \text{author\_role} \neq \text{Business Analyst} \Rightarrow \text{MissingRole}$

`list_tasks( $n, m$ )`

- output:  $out :=$  A discord.embed such that it lists all the tasks of sprint  $m$  in project  $n$ , as well as their deadlines

`rm_task( $n, m, k$ )`

- transition: `rm_task( $k$ )` in sprint  $m$  in project  $n$
- output:  $out := (exc \Rightarrow \text{"MissingRole: You have insufficient permissions"} \mid \neg exc \Rightarrow \text{"Successfully removed task"})$
- exception:  $exc := \text{author\_role} \neq \text{Scrum Master} \wedge \text{author\_role} \neq \text{Business Analyst} \Rightarrow \text{MissingRole}$

## Task Programs

`add_feedback( $n, m, k, s$ )`

- transition: `add_feedback( $s$ )` in task  $k$ , sprint  $m$ , project  $n$
- output:  $out := (exc \Rightarrow \text{"MissingRole: You have insufficient permissions"} \mid \neg exc \Rightarrow \text{"Successfully added feedback"})$
- exception:  $exc := \text{author\_role} \neq \text{Scrum Master} \Rightarrow \text{MissingRole}$

`get_details( $n, m, k$ )`

- output:  $out :=$  A discord.embed such that it shows the details of task  $k$ , in sprint  $m$ , in project  $n$ , using `get_details()`

`list_feedback( $n, m, k$ )`

- output:  $out :=$  A discord.embed such that it lists all the feedback of task  $k$ , in sprint  $m$ , in project  $n$ , using `get_feedback()`

`rm_feedback( $n, m, k, x$ )`

- transition: `rm_feedback( $x$ )` for task  $k$  in sprint  $m$  in project  $n$
- output:  $out := (exc \Rightarrow \text{"MissingRole: You have insufficient permissions"} \mid \neg exc \Rightarrow \text{"Successfully removed feedback"})$
- exception:  $exc := \text{author\_role} \neq \text{Scrum Master} \Rightarrow \text{MissingRole}$

`set_details( $n, m, s$ )`

- transition: `set_details( $s$ )` for task  $k$  in sprint  $m$  in project  $n$
- output:  $out := (exc \Rightarrow \text{"MissingRole: You have insufficient permissions"} \mid \neg exc \Rightarrow \text{"Successfully set details"})$
- exception:  $exc := \text{author\_role} \neq \text{Scrum Master} \wedge \text{author\_role} \neq \text{Business Analyst} \Rightarrow \text{MissingRole}$

# MeetingTypes Module

## Module

MeetingTypes

## Uses

N/A

## Syntax

### Exported Constants

N/A

### Exported Types

MeetingT = {Grooming, StandUp, Retrospective, SprintPlanning}

### Exported Access Programs

None

## Semantics

### State Variables

None

### State Invariant

None

# Generic Dictionary Module

## Generic Template Module

Dict(T)

### Uses

N/A

### Syntax

#### Exported Types

Dict = ?

#### Exported Constants

None

#### Exported Access Programs

Routine Name	In	Out	Exceptions
new Dict(T)		Dict(T)	
add	T		
remove	$\mathbb{N}$		KeyError
to_seq		seq of ( $\mathbb{N}$ ,T)	

### Semantics

#### State Variables

d: seq of ( $\mathbb{N}$ ,T)

c:  $\mathbb{N}$

#### State Invariant

$|d| \geq 0$

$c \geq 0$

### Assumptions & Design Decisions

The Dict(T) constructor is called for each object instance before any other access routine is called for that object.

It is assumed that the first term in Dict is referred to as the "key" and the second term is the "value".

## Access Routine Semantics

new Dict():

- transition:  $d, c := \langle \rangle, 0$
- output:  $out := self$

add( $e$ ):

- transition:  $d := d \parallel \langle c, e \rangle, c := c + 1$

remove( $id$ ):

- transition:  $d := d - \langle id, e \rangle$
- exception:  $exc := \langle id, e \rangle \notin d \Rightarrow \text{KeyError}$

to\_seq():

- output:  $out := d$  such that  $(\forall i \in \mathbb{N} \mid 0 \leq i < |d| \cdot d[i].key() \leq d[i+1].key())$

## MeetingList Module

### Template Module

MeetingList is a Dict(Meeting)

# Meeting Module

## Module

Meeting

## Uses

MeetingTypes

## Syntax

### Exported Constants

None

### Exported Types

Meeting = ?

### Exported Access Programs

Routine Name	In	Out	Exceptions
new Meeting	String, Date, Time, MeetingT	Meeting	
new Meeting	String, Date, Time, MeetingT, String	Meeting	
get_name		String	
get_date		Date	
get_time		Time	
get_type		MeetingT	
get_desc		String	
set_desc	String		

## Semantics

### State Variables

*name*: String

*date*: Date

*time*: Time

*type*: MeetingT

*desc*: String

### State Invariant

None



## Assumptions

- The Meeting constructor is called for each object instance before any other access routine is called for that object.

## Access Routine Semantics

new Meeting( $n, d, t, type$ )

- transition:  $name, time, date, type, desc := n, d, t, type, None$
- output:  $out := self$

new Meeting( $n, d, t, type, desc$ )

- transition:  $name, time, date, type, desc := n, d, t, type, desc$
- output:  $out := self$

get\_name()

- output:  $out := name$

get\_date()

- output:  $out := date$

get\_time()

- output:  $out := time$

get\_type()

- output:  $out := type$

get\_desc()

- output:  $out := desc$

set\_desc( $s$ )

- transition:  $desc := s$

## TaskList Module

### Template Module

TaskList is a Dict(Task)

# Task Module

## Module

Task

## Uses

None

## Syntax

### Exported Constants

None

### Exported Types

Task = ?

### Exported Access Programs

Routine Name	In	Out	Exceptions
new Task	String, Date, Time	Task	
new Task	String, Date, Time, Details	Task	
get_deadline		(Date, Time)	
get_details		String	
get_feedback		seq of String	
add_feedback	String		
rm_feedback	$\mathbb{N}$		
set_details	String		

## Semantics

### State Variables

*name*: String

*deadline*: (Date, Time)

*details*: String

*feedback*: seq of String

### State Invariant

None

## Assumptions

- The Task constructor is called for each object instance before any other access routine is called for that object.

## Access Routine Semantics

new Task( $s, d, t$ )

- transition:  $name, deadline, details := s, (d, t), \text{None}$
- output:  $out := \text{self}$

new Task( $s, d, t, details$ )

- transition:  $name, deadline, details := s, (d, t), details$
- output:  $out := \text{self}$

get\_deadline()

- output:  $out := deadline$

get\_details()

- output:  $out := (details = \text{None} \Rightarrow \text{“No details”} \mid details)$

get\_feedback()

- output:  $out := feedback$

add\_feedback( $s$ )

- transition:  $feedback := feedback \parallel s$

rm\_feedback( $s$ )

- transition:  $feedback := feedback - s$

set\_details( $s$ )

- transition:  $details := s$

# Sprint Module

## Module

Sprint

## Uses

TaskList, Task

## Syntax

### Exported Constants

None

### Exported Types

Sprint = ?

### Exported Access Programs

Routine Name	In	Out	Exceptions
new Sprint		Sprint	
get_tasks		seq of Task	
add_task	Task		
rm_task	$\mathbb{N}$		

## Semantics

### State Variables

*tasks*: TaskList

### State Invariant

None

### Assumptions

- The Sprint constructor is called for each object instance before any other access routine is called for that object.

## Access Routine Semantics

new Sprint( $n$ )

- transition:  $tasks := TaskList()$
- output:  $out := self$

get\_task( $n$ )

- output:  $out := tasks.to\_seq()$

add\_task( $task$ )

- transition:  $tasks := tasks.add(task)$

rm\_task( $n$ )

- transition:  $tasks := tasks.remove(n)$

## ProjectList Module

### Template Module

ProjectList is a Dict(Project)



# Project Module

## Module

Project

## Uses

Sprint, Tasklist, Task

## Syntax

### Exported Constants

None

### Exported Types

Project = ?

### Exported Access Programs

Routine Name	In	Out	Exceptions
new Project	String	Project	
new Project	String, String	Project	
get_desc		String	
get_meetings		seq of Meeting	
get_rqs		seq of String	
get_sprints		seq of Sprint	
set_desc	String		
add_meeting	Meeting		
add_rqe	String		
add_sprint	Sprint		
rm_meeting	$\mathbb{N}$		
rm_rqe	$\mathbb{N}$		IndexError
rm_sprint			IndexError

## Semantics

### State Variables

*name*: String

*desc*: String

*meetings*: MeetingList

*rqs*: seq of String

*sprints*: seq of Sprint

*c*:  $\mathbb{N}$

### State Invariant

$$c = |sprints| \wedge c \geq 0$$

### Assumptions

- The Project constructor is called for each object instance before any other access routine is called for that object.

### Access Routine Semantics

new Project(*n*)

- transition:  $name, desc, rques, sprints := n, \text{None}, [], []$
- output:  $out := \text{self}$

new Project(*n*, *d*)

- transition:  $name, desc, rques, sprints := n, d, [], []$
- output:  $out := \text{self}$

get\_desc()

- output:  $out := (desc = \text{None} \Rightarrow \text{"No description"} \mid desc)$

get\_meetings()

- output:  $out := meetings.to\_seq()$

get\_rques()

- output:  $out := rques$

get\_sprints()

- output:  $out := sprints$

set\_desc(*s*)

- transition:  $desc := s$

add\_meeting(*meeting*)

- transition:  $meetings := meetings.add(meeting)$

`add_rqe(s)`

- transition:  $rqs := rqs \parallel s$

`add_sprint(sprint)`

- transition:  $sprints, c := sprints \parallel sprint, c + 1$

`rm_meeting(n)`

- transition:  $meetings := meetings.remove(n)$

`rm_rqe(n)`

- transition:  $rqs := rqs - rqs[n]$

`rm_sprint()`

- transition:  $sprints, c := sprints[0 : |sprints| - 2], c - 1$
- exception:  $exc := c = 0 \Rightarrow \text{IndexError}$