

# Module Interface Specification for Software Engineering

Team #2, Campus Connections

Waseef Nayeem

Zihao Du

Matthew Miller

Firas Elayan

Abhiram Neelamraju

Michael Kim

April 4, 2024

# 1 Revision History

Date	Version	Notes
Jan 15	1.0	Add introduction and module decomposition
Jan 17	1.0	Revision 0

## 2 Symbols, Abbreviations and Acronyms

See SRS Documentation at [SRS](#)

### 2.1 Abbreviations and Acronyms

symbol	description
MIS	Module Interface Specification
MG	Module Guide
SRS	Software Requirement Specification
AR	Augmented Reality

# Contents

<b>1</b>	<b>Revision History</b>	<b>i</b>
<b>2</b>	<b>Symbols, Abbreviations and Acronyms</b>	<b>ii</b>
2.1	Abbreviations and Acronyms . . . . .	ii
<b>3</b>	<b>Introduction</b>	<b>1</b>
<b>4</b>	<b>Notation</b>	<b>1</b>
<b>5</b>	<b>Module Decomposition</b>	<b>2</b>
<b>6</b>	<b>MIS of User Module</b>	<b>3</b>
6.1	Module . . . . .	3
6.2	Uses . . . . .	4
6.3	Syntax . . . . .	4
6.3.1	Exported Constants . . . . .	4
6.3.2	Exported Access Programs . . . . .	4
6.4	Semantics . . . . .	5
6.4.1	State Variables . . . . .	5
6.4.2	Environment Variables . . . . .	5
6.4.3	Assumptions . . . . .	5
6.4.4	Access Routine Semantics . . . . .	5
6.4.5	Local Functions . . . . .	7
<b>7</b>	<b>MIS of Lecture Module</b>	<b>8</b>
7.1	Module . . . . .	8
7.2	Uses . . . . .	8
7.3	Syntax . . . . .	8
7.3.1	Exported Constants . . . . .	8
7.3.2	Exported Access Programs . . . . .	8
7.4	Semantics . . . . .	8
7.4.1	State Variables . . . . .	8
7.4.2	Environment Variables . . . . .	9
7.4.3	Assumptions . . . . .	9
7.4.4	Access Routine Semantics . . . . .	9
7.4.5	Local Functions . . . . .	9
<b>8</b>	<b>MIS of Event Module</b>	<b>10</b>
8.1	Module . . . . .	10
8.2	Uses . . . . .	10
8.3	Syntax . . . . .	10
8.3.1	Exported Constants . . . . .	10

8.3.2	Exported Access Programs . . . . .	10
8.4	Semantics . . . . .	10
8.4.1	State Variables . . . . .	10
8.4.2	Environment Variables . . . . .	11
8.4.3	Assumptions . . . . .	11
8.4.4	Access Routine Semantics . . . . .	11
8.4.5	Local Functions . . . . .	12
<b>9</b>	<b>MIS of Account Module</b>	<b>13</b>
9.1	Module . . . . .	13
9.2	Uses . . . . .	13
9.3	Syntax . . . . .	13
9.3.1	Exported Constants . . . . .	13
9.3.2	Exported Access Programs . . . . .	13
9.4	Semantics . . . . .	13
9.4.1	State Variables . . . . .	13
9.4.2	Environment Variables . . . . .	14
9.4.3	Assumptions . . . . .	14
9.4.4	Access Routine Semantics . . . . .	14
9.4.5	Local Functions . . . . .	15
<b>10</b>	<b>MIS of Friend Manager Module</b>	<b>16</b>
10.1	Module . . . . .	16
10.2	Uses . . . . .	16
10.3	Syntax . . . . .	16
10.3.1	Exported Constants . . . . .	16
10.3.2	Exported Access Programs . . . . .	16
10.4	Semantics . . . . .	16
10.4.1	State Variables . . . . .	16
10.4.2	Environment Variables . . . . .	16
10.4.3	Assumptions . . . . .	16
10.4.4	Access Routine Semantics . . . . .	17
10.4.5	Local Functions . . . . .	17
10.4.6	Local Constants . . . . .	17
<b>11</b>	<b>MIS of Friend Request Module</b>	<b>18</b>
11.1	Module . . . . .	18
11.2	Uses . . . . .	18
11.3	Syntax . . . . .	18
11.3.1	Exported Constants . . . . .	18
11.3.2	Exported Access Programs . . . . .	18
11.4	Semantics . . . . .	18
11.4.1	State Variables . . . . .	18

11.4.2	Environment Variables . . . . .	18
11.4.3	Assumptions . . . . .	18
11.4.4	Access Routine Semantics . . . . .	18
11.4.5	Local Functions . . . . .	19
11.4.6	Local Constants . . . . .	19
<b>12</b>	<b>MIS of Activity Detail View Module</b>	<b>20</b>
12.1	Module . . . . .	20
12.2	Uses . . . . .	20
12.3	Syntax . . . . .	20
12.3.1	Exported Constants . . . . .	20
12.3.2	Exported Access Programs . . . . .	20
12.4	Semantics . . . . .	20
12.4.1	State Variables . . . . .	20
12.4.2	Environment Variables . . . . .	20
12.4.3	Assumptions . . . . .	21
12.4.4	Access Routine Semantics . . . . .	21
12.4.5	Local Functions . . . . .	22
12.4.6	Local Constants . . . . .	22
<b>13</b>	<b>MIS of Lecture Detail View Module</b>	<b>23</b>
13.1	Module . . . . .	23
13.2	Uses . . . . .	23
13.3	Syntax . . . . .	23
13.3.1	Exported Constants . . . . .	23
13.3.2	Exported Access Programs . . . . .	23
13.4	Semantics . . . . .	23
13.4.1	State Variables . . . . .	23
13.4.2	Environment Variables . . . . .	23
13.4.3	Assumptions . . . . .	24
13.4.4	Access Routine Semantics . . . . .	24
13.4.5	Local Functions . . . . .	25
13.4.6	Local Constants . . . . .	25
<b>14</b>	<b>MIS of Event Detail View Module</b>	<b>26</b>
14.1	Module . . . . .	26
14.2	Uses . . . . .	26
14.3	Syntax . . . . .	26
14.3.1	Exported Constants . . . . .	26
14.3.2	Exported Access Programs . . . . .	26
14.4	Semantics . . . . .	26
14.4.1	State Variables . . . . .	26
14.4.2	Environment Variables . . . . .	26

14.4.3	Assumptions . . . . .	27
14.4.4	Access Routine Semantics . . . . .	27
14.4.5	Local Functions . . . . .	28
14.4.6	Local Constants . . . . .	28
<b>15</b>	<b>MIS of Authentication Module</b>	<b>29</b>
15.1	Module . . . . .	29
15.2	Uses . . . . .	29
15.3	Syntax . . . . .	29
15.3.1	Exported Constants . . . . .	29
15.3.2	Exported Access Programs . . . . .	29
15.4	Semantics . . . . .	29
15.4.1	State Variables . . . . .	29
15.4.2	Environment Variables . . . . .	29
15.4.3	Assumptions . . . . .	29
15.4.4	Access Routine Semantics . . . . .	29
15.4.5	Local Functions . . . . .	30
15.4.6	Local Constants . . . . .	30
<b>16</b>	<b>MIS of Permission Module</b>	<b>31</b>
16.1	Module . . . . .	31
16.2	Uses . . . . .	31
16.3	Syntax . . . . .	31
16.3.1	Exported Constants . . . . .	31
16.3.2	Exported Access Programs . . . . .	31
16.4	Semantics . . . . .	31
16.4.1	State Variables . . . . .	31
16.4.2	Environment Variables . . . . .	31
16.4.3	Assumptions . . . . .	31
16.4.4	Access Routine Semantics . . . . .	31
16.4.5	Local Functions . . . . .	32
16.4.6	Local Constants . . . . .	32
<b>17</b>	<b>MIS of User Profile Module</b>	<b>33</b>
17.1	Module . . . . .	33
17.2	Uses . . . . .	33
17.3	Syntax . . . . .	33
17.3.1	Exported Constants . . . . .	33
17.3.2	Exported Access Programs . . . . .	33
17.4	Semantics . . . . .	33
17.4.1	State Variables . . . . .	33
17.4.2	Environment Variables . . . . .	33
17.4.3	Assumptions . . . . .	33

17.4.4	Access Routine Semantics	33
17.4.5	Local Functions	34
17.4.6	Local Constants	34
<b>18</b>	<b>MIS of User Login Module</b>	<b>35</b>
18.1	Module	35
18.2	Uses	35
18.3	Syntax	35
18.3.1	Exported Constants	35
18.3.2	Exported Access Programs	35
18.4	Semantics	35
18.4.1	State Variables	35
18.4.2	Environment Variables	35
18.4.3	Assumptions	35
18.4.4	Access Routine Semantics	35
18.4.5	Local Functions	36
18.4.6	Local Constants	36
<b>19</b>	<b>MIS of Lecture List Manager Module</b>	<b>37</b>
19.1	Module	37
19.2	Uses	37
19.3	Syntax	37
19.3.1	Exported Constants	37
19.3.2	Exported Access Programs	37
19.4	Semantics	37
19.4.1	State Variables	37
19.4.2	Environment Variables	37
19.4.3	Assumptions	38
19.4.4	Access Routine Semantics	38
19.4.5	Local Functions	39
19.4.6	Local Constants	39
<b>20</b>	<b>MIS of Event List Manager Module</b>	<b>40</b>
20.1	Module	40
20.2	Uses	40
20.3	Syntax	40
20.3.1	Exported Constants	40
20.3.2	Exported Access Programs	40
20.4	Semantics	40
20.4.1	State Variables	40
20.4.2	Environment Variables	40
20.4.3	Assumptions	41
20.4.4	Access Routine Semantics	41



20.4.5	Local Functions . . . . .	42
20.4.6	Local Constants . . . . .	42
<b>21</b>	<b>MIS of Pagination and Filter Module</b>	<b>43</b>
21.1	Module . . . . .	43
21.2	Uses . . . . .	43
21.3	Syntax . . . . .	43
21.3.1	Exported Constants . . . . .	43
21.3.2	Exported Access Programs . . . . .	43
21.4	Semantics . . . . .	43
21.4.1	State Variables . . . . .	43
21.4.2	Environment Variables . . . . .	43
21.4.3	Assumptions . . . . .	44
21.4.4	Access Routine Semantics . . . . .	44
21.4.5	Local Functions . . . . .	45
21.4.6	Local Constants . . . . .	45
<b>22</b>	<b>MIS of Database Module</b>	<b>46</b>
22.1	Module . . . . .	46
22.2	Uses . . . . .	46
22.3	Syntax . . . . .	46
22.3.1	Exported Constants . . . . .	46
22.3.2	Exported Access Programs . . . . .	46
22.4	Semantics . . . . .	47
22.4.1	State Variables . . . . .	47
22.4.2	Environment Variables . . . . .	47
22.4.3	Assumptions . . . . .	48
22.4.4	Access Routine Semantics . . . . .	48
22.4.5	Local Functions . . . . .	49
<b>23</b>	<b>MIS of Server Module</b>	<b>50</b>
23.1	Module . . . . .	50
23.2	Uses . . . . .	50
23.3	Syntax . . . . .	50
23.3.1	Exported Constants . . . . .	50
23.3.2	Exported Access Programs . . . . .	50
23.4	Semantics . . . . .	50
23.4.1	State Variables . . . . .	50
23.4.2	Environment Variables . . . . .	50
23.4.3	Assumptions . . . . .	50
23.4.4	Access Routine Semantics . . . . .	50
23.4.5	Local Functions . . . . .	51

<b>24 MIS of AR Camera</b>	<b>52</b>
24.1 Module . . . . .	52
24.2 Uses . . . . .	52
24.3 Syntax . . . . .	52
24.3.1 Exported Constants . . . . .	52
24.3.2 Exported Access Programs . . . . .	52
24.4 Semantics . . . . .	52
24.4.1 State Variables . . . . .	52
24.4.2 Environment Variables . . . . .	52
24.4.3 Assumptions . . . . .	52
24.4.4 Access Routine Semantics . . . . .	53
24.4.5 Local Functions . . . . .	53
<b>25 MIS of AR Interface</b>	<b>54</b>
25.1 Module . . . . .	54
25.2 Uses . . . . .	54
25.3 Syntax . . . . .	54
25.3.1 Exported Constants . . . . .	54
25.3.2 Exported Access Programs . . . . .	54
25.4 Semantics . . . . .	54
25.4.1 State Variables . . . . .	54
25.4.2 Environment Variables . . . . .	54
25.4.3 Assumptions . . . . .	54
25.4.4 Access Routine Semantics . . . . .	54
25.4.5 Local Functions . . . . .	55
<b>26 MIS of MapBox</b>	<b>56</b>
26.1 Module . . . . .	56
26.2 Uses . . . . .	56
26.3 Syntax . . . . .	56
26.3.1 Exported Constants . . . . .	56
26.3.2 Exported Access Programs . . . . .	56
26.4 Semantics . . . . .	56
26.4.1 State Variables . . . . .	56
26.4.2 Environment Variables . . . . .	56
26.4.3 Assumptions . . . . .	56
26.4.4 Access Routine Semantics . . . . .	57
26.4.5 Local Functions . . . . .	57
26.4.6 Local Constants . . . . .	57

<b>27 MIS of Real-time Map</b>	<b>58</b>
27.1 Module . . . . .	58
27.2 Uses . . . . .	58
27.3 Syntax . . . . .	58
27.3.1 Exported Constants . . . . .	58
27.3.2 Exported Access Programs . . . . .	58
27.4 Semantics . . . . .	58
27.4.1 State Variables . . . . .	58
27.4.2 Environment Variables . . . . .	58
27.4.3 Assumptions . . . . .	58
27.4.4 Access Routine Semantics . . . . .	58
27.4.5 Local Functions . . . . .	59
<b>28 MIS of Friend Chat</b>	<b>60</b>
28.1 Module . . . . .	60
28.2 Uses . . . . .	60
28.3 Syntax . . . . .	60
28.3.1 Exported Constants . . . . .	60
28.3.2 Exported Access Programs . . . . .	60
28.4 Semantics . . . . .	60
28.4.1 State Variables . . . . .	60
28.4.2 Environment Variables . . . . .	60
28.4.3 Assumptions . . . . .	60
28.4.4 Access Routine Semantics . . . . .	60
28.4.5 Local Functions . . . . .	61
<b>29 Appendix</b>	<b>62</b>
29.1 Database Tables . . . . .	62

### 3 Introduction

The following document details the Module Interface Specifications for CampusConnections. CampusConnections is a social media application with impressive AR camera and real time location map features that allows McMaster University students and visitors have an immersive user experience and expand their social networking. This application allows users to make new friends online and also encourage users to strengthen the friendship by in-person meet-ups with a on-campus location-sharing feature. It also provides heat maps of events and users, which allows students to join the most popular activities on campus. Besides, the application maintainers will share up-to-date events and lectures information for the community. The MIS will detail specifications for the project described above.

Complementary documents include the System Requirement Specifications (SRS) and Module Guide. (MG) The full documentation and implementation can be found at <https://github.com/beatlepie/4G06CapstoneProjectTeam2/blob/main/docs/SRS-Volere/SRS.pdf> and <https://github.com/beatlepie/4G06CapstoneProjectTeam2/blob/main/docs/Design/SoftArchitecture/MG.pdf>

### 4 Notation

The structure of the MIS for modules comes from Hoffman and Strooper (1995), with the addition that template modules have been adapted from Ghezzi et al. (2003). The mathematical notation comes from Chapter 3 of Hoffman and Strooper (1995). For instance, the symbol  $:=$  is used for a multiple assignment statement and conditional rules follow the form  $(c_1 \Rightarrow r_1 | c_2 \Rightarrow r_2 | \dots | c_n \Rightarrow r_n)$ .

The following table summarizes the primitive data types used by Software Engineering.

Data Type	Notation	Description
character	char	a single symbol or digit
integer	$\mathbb{Z}$	a number without a fractional component in $(-\infty, \infty)$
natural number	$\mathbb{N}$	a number without a fractional component in $[1, \infty)$
real	$\mathbb{R}$	any number in $(-\infty, \infty)$
boolean	$\mathbb{B}$	True or False
sequence of T	$\langle T \rangle$	a list of object with type T
asynchronous step T	Task $\langle T \rangle$	an asynchronous result of T
activity	Activity	generic class with $\langle T \rangle$ that can be in- stantiated with type Lecture or Event
lecture	Lecture	see MIS of Lecture Module
event	Event	see MIS of Lecture Module
uniform resource identifier	Uri	C# Class that provides easy access to a link (URI)
date and time	DateTime	provides a specific date and time
user	User	see MIS of User Module
scene	Scene	a user interface created in Unity
database reference	DatabaseReference	reference to the root location of a database

The specification of Software Engineering uses some derived data types: sequences, strings, and tuples. Sequences are lists filled with elements of the same data type. Strings are sequences of characters. Tuples contain a list of values, potentially of different types. In addition, Software Engineering uses functions, which are defined by the data types of their inputs and outputs. Local functions are described by giving their type signature followed by their specification.

## 5 Module Decomposition

The following table is taken directly from the Module Guide document for this project.

Level 1	Level 2
Hardware-Hiding	
Behaviour-Hiding	AR Interface Module
	Map Interface Module
	User Module
	Lecture Module
	Event Module
	Account Module
	Permission Module
	User Profile Module
	User Login Module
	Friend Manager Module
	Friend Request Module
	Friend Chat Module
	Lecture Detail View Module
	Event Detail View Module
	Lecture List Manager Module
	Event List Manager Module
Software Decision	Database Module
	Server Module
	Authentication Module
	AR Camera Module
	Mapbox Module
	Activity Detail View Module
	Pagination and Filter Module

Table 1: Module Hierarchy

## 6 MIS of User Module

### 6.1 Module

User

## 6.2 Uses

Lecture Module, Event Module

## 6.3 Syntax

### 6.3.1 Exported Constants

None

### 6.3.2 Exported Access Programs

Name	In	Out	Exceptions
User	String, String, Uri, String, $\mathbb{R}$ , <User>, <User>, <Lecture>, <Event>	User	-
SetNickName	String	-	-
SetPhotoUri	Uri	-	-
SetProgram	String	-	-
SetLevel	$\mathbb{R}$	-	-
AddFriend	User	-	-
RemoveFriend	User	-	IndexOutOfBounds Ex- ception
AddRequester	User	-	-
RemoveRequester	User	-	IndexOutOfBounds Ex- ception
AddLecture	Lecture	-	-
RemoveLecture	Lecture	-	IndexOutOfBounds Ex- ception
AddEvent	Event	-	-
RemoveEvent	Event	-	IndexOutOfBounds Ex- ception

## 6.4 Semantics

### 6.4.1 State Variables

- email: String, User email
- nickName: String, User nickName
- photoUri: Uri, User avatar
- program: String, User program
- level:  $\mathbb{R}$ , User program level
- friends:  $\langle \text{User} \rangle$ , List of friends
- requesters:  $\langle \text{User} \rangle$ , List of friend requester
- lectures:  $\langle \text{Lecture} \rangle$ , Pinned lecture
- events:  $\langle \text{Event} \rangle$ , Pinned event

### 6.4.2 Environment Variables

None

### 6.4.3 Assumptions

Strings passed as input are of valid format, all the state variables of the object are directly accessible so getter is not needed.

### 6.4.4 Access Routine Semantics

User(email, nickName, photoUri, program, level, friends, requesters, lectures,events):

- transition: *email, nickName, photoUri, program, level, friends, requesters, lectures, events := email, nickName, photoUri, program, level, friends, requesters, lectures, events*
- output: *out := self*
- exception: none

SetNickName(newName):

- transition: *nickName := newName*
- output: none
- exception: none



SetPhotot(newUri):

- transition:  $photoUri := newUri$
- output: none
- exception: none

SetProgram(newProgram):

- transition:  $program := newProgram$
- output: none
- exception: none

SetLevel(newLevel):

- transition:  $level := newLevel$
- output: none
- exception: none

AddFriend(newFriend):

- transition:  $friends := friends + \{newFriend\}$
- output: none
- exception: none

RemoveFriend(targetFriend):

- transition:  $friends := friends - \{targetFriend\}$
- output: none
- exception:  $exc := targetFriend \notin friends \Rightarrow IndexOutOfBoundException$

AddRequester(newRequester):

- transition:  $requesters := requesters + \{newRequester\}$
- output: none
- exception: none

RemoveRequester(targetRequester):

- transition:  $requesters := requesters - \{targetRequester\}$

- output: none
- exception:  $exc := targetRequester \notin requesters \Rightarrow IndexOutOfBoundException$

AddLecture(newLec):

- transition:  $lectures := lectures + \{newLec\}$
- output: none
- exception: none

RemoveLecture(targetLecture):

- transition:  $lectures := lectures - \{targetLecture\}$
- output: none
- exception:  $exc := targetLecture \notin lectures \Rightarrow IndexOutOfBoundException$

AddEvent(newEvent):

- transition:  $events := events + \{newEvent\}$
- output: none
- exception: none

RemoveEvent(targetEvent):

- transition:  $events := events - \{targetEvent\}$
- output: none
- exception:  $exc := targetEvent \notin events \Rightarrow IndexOutOfBoundException$

#### 6.4.5 Local Functions

None

## 7 MIS of Lecture Module

### 7.1 Module

Lecture

### 7.2 Uses

None

### 7.3 Syntax

#### 7.3.1 Exported Constants

None

#### 7.3.2 Exported Access Programs

Name	In	Out	Exceptions
Lecture	String, String, String, String, String	Lecture	-
SetName	String	-	-
SetInstructor	String	-	-
SetTime	String	-	-
SetLocation	String	-	-

### 7.4 Semantics

#### 7.4.1 State Variables

- code: String, Lecture code
- name: String, Lecture name
- instructor: String, Lecture instructor
- time: String, Lecture time
- location: String, Lecture location

### 7.4.2 Environment Variables

None

### 7.4.3 Assumptions

Strings passed as input are of valid format, all the state variables of the object are directly accessible so getter is not needed.

### 7.4.4 Access Routine Semantics

Lecture(lecCode, lecName, lecInstructor, lecTime, lecLocation):

- transition: *code, name, instructor, time, location := lecCode, lecName, lecInstructor, lecTime, lecLocation*
- output: *out := self*
- exception: none

SetName(newName):

- transition: *name := newName*
- output: none
- exception: none

SetInstructor(newInstructor):

- transition: *instructor := newInstructor*
- output: none
- exception: none

SetTime(newTime):

- transition: *time := newTime*
- output: none
- exception: none

SetLocation(newLocation):

- transition: *location := newLocation*
- output: none
- exception: none

### 7.4.5 Local Functions

None

## 8 MIS of Event Module

### 8.1 Module

Event

### 8.2 Uses

None

### 8.3 Syntax

#### 8.3.1 Exported Constants

None

#### 8.3.2 Exported Access Programs

Name	In	Out	Exceptions
Event	String, String, String,DateTime, $\mathbb{R}$ , String, $\mathbb{B}$	Event	-
setDescription	String	-	-
setOrganizer	String	-	-
setStartTime	DateTime	-	-
setDuration	$\mathbb{R}$	-	-
setLocation	String	-	-
setPublic	$\mathbb{B}$	-	-

### 8.4 Semantics

#### 8.4.1 State Variables

- name: String, Event name
- description: String, Event description
- organizer: String, Event hosted by
- startTime: DateTime, Event start date and time
- duration:  $\mathbb{R}$ , Event duration (in minutes)

- location: String, Event location (room and building)
- public :  $\mathbb{B}$ , is event public

#### 8.4.2 Environment Variables

None

#### 8.4.3 Assumptions

Strings passed as input are of valid format, all the state variables of the object are directly accessible so getter is not needed.

#### 8.4.4 Access Routine Semantics

Event(name, description, organizer, startTime, duration, location, public):

- transition: *name, description, organizer, startTime, duration, location, public := name, description, organizer, startTime, duration, location, public*
- output: *out := self*
- exception: none

SetDescription(newDescription):

- transition: *description := newDescription*
- output: none
- exception: none

SetOrganizer(newOrganizer):

- transition: *organizer := newOrganizer*
- output: none
- exception: none

SetStartTime(newTime):

- transition: *startTime := newTime*
- output: none
- exception: none

SetDuration(newDuration):

- transition: *duration := newDuration*
- output: none
- exception: none

SetLocation(newLocation):

- transition: *location := newLocation*
- output: none
- exception: none

SetPublic(newPublicity):

- transition: *public := newPublicity*
- output: none
- exception: none

#### **8.4.5 Local Functions**

None

## 9 MIS of Account Module

### 9.1 Module

Account

### 9.2 Uses

Database Module, User Module, Authentication Module

### 9.3 Syntax

#### 9.3.1 Exported Constants

None

#### 9.3.2 Exported Access Programs

Name	In	Out	Exceptions
UpdateNickName	String	-	-
UpdateProgram	String	-	-
UpdateLevel	N	-	-
AddFriend	User	-	-
DeleteFriend	User	-	IndexOutOfBounds Ex- ception
AddRequest	User	-	-
DeleteRequest	User	-	IndexOutOfBounds Ex- ception
PinLecture	Lecture	-	-
UnPinLecture	Lecture	-	IndexOutOfBounds Ex- ception
PinEvent	Event	-	-
UnPinEvent	Event	-	IndexOutOfBounds Ex- ception

### 9.4 Semantics

#### 9.4.1 State Variables

- User: User User of the account



### 9.4.2 Environment Variables

None

### 9.4.3 Assumptions

All the state variables of User is accessible directly so there is no getters in the module.

### 9.4.4 Access Routine Semantics

UpdateNickName(newName):

- transition:  $User.SetNickName(newName)$
- output: none
- exception: none

UpdateProgram(newProgram):

- transition:  $User.SetProgram(newProgram)$
- output: none
- exception: none

UpdateLevel(newLevel):

- transition:  $User.SetLevel(newLevel)$
- output: none
- exception: none

AddFriend(newFriend):

- transition:  $User.AddFriend(newFriend)$
- output: none
- exception: none

DeleteFriend(targetFriend):

- transition:  $User.RemoveFriend(targetFriend)$
- output: none
- exception:  $exc := targetFriend \notin User.friends \Rightarrow IndexOutOfBoundException$

AddRequest(newFriend):

- transition:  $User.AddRequester(newFriend)$
- output: none
- exception: none

DeleteRequest(targetFriend):

- transition:  $User.RemoveRequester(targetFriend)$
- output: none
- exception:  $exc := targetFriend \notin User.friendRequests \Rightarrow IndexOutOfBoundException$

PinLecture(newLec):

- transition:  $User.AddLecture(newLec)$
- output: none
- exception: none

UnpinLecture(targetLec):

- transition:  $User.RemoveLecture(targetLec)$
- output: none
- exception:  $exc := targetLec \notin User.lectures \Rightarrow IndexOutOfBoundException$

PinEvent(newEvent):

- transition:  $User.AddEvent(newEvent)$
- output: none
- exception: none

UnpinLecture(targetEvent):

- transition:  $User.RemoveEvent(targetEvent)$
- output: none
- exception:  $exc := targetEvent \notin User.events \Rightarrow IndexOutOfBoundException$

#### 9.4.5 Local Functions

None

## 10 MIS of Friend Manager Module

### 10.1 Module

FriendManager

### 10.2 Uses

Account Module, Chat Module, Unity Transform Type

### 10.3 Syntax

#### 10.3.1 Exported Constants

None

#### 10.3.2 Exported Access Programs

Name	In	Out	Exceptions
DisplayFriendList	-	<Tranform>	-
onClickDeleteFriend	User	-	IndexOutOfBounds Ex- ception
onClickViewFriend	User	-	IndexOutOfBounds Ex- ception
onClickMessageFriend	User	2D seq of pixels	IndexOutOfBounds Ex- ception
onClickSendRequest	User	$\mathbb{B}$	-

### 10.4 Semantics

#### 10.4.1 State Variables

None

#### 10.4.2 Environment Variables

None

#### 10.4.3 Assumptions

Assume the singleton Account is accessible from this module.

#### 10.4.4 Access Routine Semantics

DisplayFriendList():

- transition: none
- output:  $out := friendContainer$  where  $(\forall x : \mathbb{Z} | 0 \leq x \leq Account.friends.length : friendsContainer[x].position, friendsContainer[x].content = (0, i * HEIGHT), Account.friends[i]),$
- exception: none

onClickDeleteFriend(targetUser):

- transition:  $Account.DeleteFriend(targetUser)$
- output: none
- exception:  $exc := targetUser.email \notin Account.User.friends \Rightarrow IndexOutOfBoundException$

onClickViewFriend(targetUser):

- transition: Switch scene to user profile where  $User = targetUser$
- output: none
- exception:  $exc := targetUser.email \notin Account.User.friends \Rightarrow IndexOutOfBoundException$

onClickMessageFriend(targetUser):

- transition: Call Chat Module to establish a connection
- output: UI of friend chat between  $Account.User$  and  $targetUser$
- exception:  $exc := targetUser.email \notin Account.User.friends \Rightarrow IndexOutOfBoundException$

onClickSendRequest(targetUser):

- transition:  $targetUser.AddRequest(Account1.User.email)$  if the current user has not send a request yet
- output:  $Account1.User.email \notin targetUser.friendRequest$
- exception: none

#### 10.4.5 Local Functions

None

#### 10.4.6 Local Constants

HEIGHT = 300 px

## 11 MIS of Friend Request Module

### 11.1 Module

FriendRequest

### 11.2 Uses

Account Module, Unity Transform Type

### 11.3 Syntax

#### 11.3.1 Exported Constants

None

#### 11.3.2 Exported Access Programs

Name	In	Out	Exceptions
DisplayRequestList	-	<Transform>	-
onClickAcceptRequest	User	-	IllegalArgument Ex-ception
onClickIgnoreRequest	User	-	IllegalArgument Ex-ception

### 11.4 Semantics

#### 11.4.1 State Variables

None

#### 11.4.2 Environment Variables

None

#### 11.4.3 Assumptions

Assume the singleton Account is accessible from this module.

#### 11.4.4 Access Routine Semantics

DisplayRequestList():

- transition: none

- output:  $out := requestContainer$  where  $(\forall x : \mathbb{Z} | 0 \leq x \leq Account.friendRequests.length :$

$requestContainer[i].position, requestContainer[i].content =$   
 $(0, i * HEIGHT), Account.friendRequests[i]),$

- exception: none

onClickAcceptRequest(targetUser):

- transition:  $targetUser.friends := targetUser.friends + Account.User.email$   
 $Account.User.AddFriend(targetUser)$   
 $Account.User.DeleteRequest(targetUser)$
- output: none
- exception:  $exc := targetUser \notin Account.User.friendRequests \Rightarrow$   
 $IllegalArgumentException$

onClickIgnoreRequest(targetUser):

- transition:  $Account.User.DeleteRequest(targetUser)$
- output: none
- exception:  $exc := targetUser \notin Account.User.friendRequests \Rightarrow$   
 $IllegalArgumentException$

#### 11.4.5 Local Functions

UpdateBadge(): String

It returns the content of friend request badge given the request number

- transition: none
- output:  $out := requestNum = 0 \Rightarrow emptystring$   
 $0 < requestNum < 100 \Rightarrow requestNum$   
 $100 \leq requestNum \Rightarrow 99+$
- exception: none

#### 11.4.6 Local Constants

HEIGHT = 150 px

## 12 MIS of Activity Detail View Module

### 12.1 Module

ActivityDetailView

### 12.2 Uses

Database Module, Permission Module

### 12.3 Syntax

#### 12.3.1 Exported Constants

None

#### 12.3.2 Exported Access Programs

Name	In	Out	Exceptions
ViewActivities	-	-	-
AddActivity	Activity	-	InvalidPermission Exception
EditActivity	Activity, Activity	-	IndexOutOfBounds Exception, Invalid- Permission Exception
DeleteActivity	Activity	-	IndexOutOfBounds Exception, Invalid- Permission Exception
PinActivity	Activity	-	-
UnpinActivity	Activity	-	IndexOutOfBounds Ex- ception

### 12.4 Semantics

#### 12.4.1 State Variables

- activities: set of Activity
- pinnedActivities: set of Activity

#### 12.4.2 Environment Variables

None

### 12.4.3 Assumptions

The singleton module `Permission` is accessible from this module.

### 12.4.4 Access Routine Semantics

`ViewActivities()`:

- transition: Display activities
- output: none
- exception: none

`AddActivity(newActivity)`:

- transition:  $activities := activities + \{newActivity\}$
- output: none
- exception:  $exc := \neg Permission.isAdmin \Rightarrow InvalidPermissionException$

`EditActivity(targetActivity, editedActivity)`:

- transition:  $activities := activities - \{targetActivity\} + \{editedActivity\}$
- output: none
- exception:  $exc := targetActivity \notin activities \Rightarrow IndexOutOfBounds Exception,$   
 $exc := \neg Permission.isAdmin \Rightarrow InvalidPermissionException$

`DeleteActivity(targetActivity)`:

- transition:  $activities := activities - \{targetActivity\}$
- output: none
- exception:  $exc := targetActivity \notin activities \Rightarrow IndexOutOfBounds Exception,$   
 $exc := \neg Permission.isAdmin \Rightarrow InvalidPermissionException$

`PinActivity(newActivity)`:

- transition:  $pinnedActivities := pinnedActivities + \{newActivity\}$
- output: none
- exception: none

`UnpinActivity(targetActivity)`:

- transition:  $pinnedActivities := pinnedActivities - \{targetActivity\}$
- output: none
- exception:  $exc := targetActivity \notin pinnedActivities \Rightarrow IndexOutOfBounds Exception$



#### **12.4.5 Local Functions**

None

#### **12.4.6 Local Constants**

None

## 13 MIS of Lecture Detail View Module

### 13.1 Module

LectureDetailView

Inherit Activity Detail View Module (Activity Detail View <Lecture>)

### 13.2 Uses

Activity Detail View Module, Lecture Module

### 13.3 Syntax

#### 13.3.1 Exported Constants

None

#### 13.3.2 Exported Access Programs

Name	In	Out	Exceptions
ViewActivities	-	-	-
AddActivity	Lecture	-	InvalidPermission Ex- ception
EditActivity	Lecture, Lecture	-	IndexOutOfBounds Exception, Invalid- Permission Exception
DeleteActivity	Lecture	-	IndexOutOfBounds Exception, Invalid- Permission Exception
PinActivity	Lecture	-	-
UnpinActivity	Lecture	-	IndexOutOfBounds Ex- ception

### 13.4 Semantics

#### 13.4.1 State Variables

- activities: set of Lecture
- pinnedActivities: set of Lecture

#### 13.4.2 Environment Variables

None

### 13.4.3 Assumptions

The singleton module `Permission` is accessible from this module.

### 13.4.4 Access Routine Semantics

`ViewActivities()`:

- transition: `Display lectures`
- output: `none`
- exception: `none`

`AddActivity(newActivity)`:

- transition:  $activities := activities + \{newActivity\}$
- output: `none`
- exception:  $exc := \neg Permission.isAdmin \Rightarrow InvalidPermissionException$

`EditActivity(targetActivity, editedActivity)`:

- transition:  $activities := activities - \{targetActivity\} + \{editedActivity\}$
- output: `none`
- exception:  $exc := targetActivity \notin activities \Rightarrow IndexOutOfBoundsOutOfRangeException,$   
 $exc := \neg Permission.isAdmin \Rightarrow InvalidPermissionException$

`DeleteActivity(targetActivity)`:

- transition:  $activities := activities - \{targetActivity\}$
- output: `none`
- exception:  $exc := targetActivity \notin activities \Rightarrow IndexOutOfBoundsOutOfRangeException,$   
 $exc := \neg Permission.isAdmin \Rightarrow InvalidPermissionException$

`PinActivity(newActivity)`:

- transition:  $pinnedActivities := pinnedActivities + \{newActivity\}$
- output: `none`
- exception: `none`

`UnpinActivity(targetActivity)`:

- transition:  $pinnedActivities := pinnedActivities - \{targetActivity\}$
- output: `none`
- exception:  $exc := targetActivity \notin pinnedActivities \Rightarrow IndexOutOfBoundsOutOfRangeException$

#### **13.4.5 Local Functions**

None

#### **13.4.6 Local Constants**

None

## 14 MIS of Event Detail View Module

### 14.1 Module

EventDetailView

Inherit Activity Detail View Module (Activity Detail View <Event>)

### 14.2 Uses

Activity Detail View Module, Event Module

### 14.3 Syntax

#### 14.3.1 Exported Constants

None

#### 14.3.2 Exported Access Programs

Name	In	Out	Exceptions
ViewActivities	-	-	-
AddActivity	Event	-	InvalidPermission Exception
EditActivity	Event, Event	-	IndexOutOfBounds Exception, Invalid- Permission Exception
DeleteActivity	Event	-	IndexOutOfBounds Exception, Invalid- Permission Exception
PinActivity	Event	-	-
UnpinActivity	Event	-	IndexOutOfBounds Ex- ception

### 14.4 Semantics

#### 14.4.1 State Variables

- activities: set of Event
- pinnedActivities: set of Event

#### 14.4.2 Environment Variables

None

### 14.4.3 Assumptions

The singleton module `Permission` is accessible from this module.

### 14.4.4 Access Routine Semantics

`ViewActivities()`:

- transition: Display events
- output: none
- exception: none

`AddActivity(newActivity)`:

- transition:  $activities := activities + \{newActivity\}$
- output: none
- exception:  $exc := \neg Permission.isAdmin \Rightarrow InvalidPermissionException$

`EditActivity(targetActivity, editedActivity)`:

- transition:  $activities := activities - \{targetActivity\} + \{editedActivity\}$
- output: none
- exception:  $exc := targetActivity \notin activities \Rightarrow IndexOutOfBounds Exception,$   
 $exc := \neg Permission.isAdmin \Rightarrow InvalidPermissionException$

`DeleteActivity(targetActivity)`:

- transition:  $activities := activities - \{targetActivity\}$
- output: none
- exception:  $exc := targetActivity \notin activities \Rightarrow IndexOutOfBounds Exception,$   
 $exc := \neg Permission.isAdmin \Rightarrow InvalidPermissionException$

`PinActivity(newActivity)`:

- transition:  $pinnedActivities := pinnedActivities + \{newActivity\}$
- output: none
- exception: none

`UnpinActivity(targetActivity)`:

- transition:  $pinnedActivities := pinnedActivities - \{targetActivity\}$
- output: none
- exception:  $exc := targetActivity \notin pinnedActivities \Rightarrow IndexOutOfBounds Exception$

#### **14.4.5 Local Functions**

None

#### **14.4.6 Local Constants**

None

## 15 MIS of Authentication Module

### 15.1 Module

Authentication

### 15.2 Uses

Database Module

### 15.3 Syntax

#### 15.3.1 Exported Constants

None

#### 15.3.2 Exported Access Programs

Name	In	Out	Exceptions
User	-	FirebaseUser	TokenExpiredException

### 15.4 Semantics

#### 15.4.1 State Variables

- User: FirebaseUser

#### 15.4.2 Environment Variables

None

#### 15.4.3 Assumptions

The user will have a unique account and only has access to that account.

#### 15.4.4 Access Routine Semantics

User():

- transition:  $(Auth.CurrentUser.valid = true) \rightarrow User = Auth.CurrentUser$
- output:  $User := Auth.CurrentUser$
- exception:  $(Auth.CurrentUser.valid = false) \rightarrow TokenExpiredException$



### 15.4.5 Local Functions

Login(*\_email*, *\_password*):

- transition:  $\exists \langle \_email, \_password \rangle \in \text{FirebaseAuth} \Rightarrow \text{Login}$
- output:  $\text{User} = \text{AuthResult.CurrentUser}$
- exception:  $\text{exc} := \neg(\exists \langle \_email, \_password \rangle \in \text{FirebaseAuth}) \Rightarrow \text{AuthFailedException}$

Register():

- transition:  $\neg(\exists \_email \in \text{FirebaseAuth}) \rightarrow \text{FirebaseAuth.add}(\text{User}) \wedge \text{FirebaseDatabase.add}(\text{User})$
- output:  $\text{User} \in \text{FirebaseAuth} \wedge \text{User} \in \text{FirebaseDatabase}$
- exception:  $\exists \_email \in \text{FirebaseAuth} \rightarrow \text{IllegalDatabaseOperationException}$

### 15.4.6 Local Constants

- auth: FirebaseAuth
- DatabaseReference: DatabaseReference

## 16 MIS of Permission Module

### 16.1 Module

Permission

### 16.2 Uses

Authentication Module

### 16.3 Syntax

#### 16.3.1 Exported Constants

None

#### 16.3.2 Exported Access Programs

Name	In	Out	Exceptions
isAdmin	-	Boolean	-
ChangePermission	User	-	InvalidPermissionException

### 16.4 Semantics

#### 16.4.1 State Variables

- User: `FirebaseUser`

#### 16.4.2 Environment Variables

None

#### 16.4.3 Assumptions

The user is logged in already.

#### 16.4.4 Access Routine Semantics

`isAdmin()`:

- transition:  $(Auth.CurrentUser.valid = true) \rightarrow User = Auth.CurrentUser$
- output:  $(Auth.CurrentUser.admin = true \rightarrow true) \vee (Auth.CurrentUser.admin = false \rightarrow false)$
- exception:  $(Auth.CurrentUser.valid = false) \rightarrow TokenExpiredException$

### 16.4.5 Local Functions

RefreshToken(user):

- transition:  $\exists \langle \_email, \_password \rangle \in \text{FirebaseAuth} \Rightarrow \text{Login}$
- output:  $User = \text{AuthResult.CurrentUser}$
- exception:  $exc := \neg(\exists \langle \_email, \_password \rangle \in \text{FirebaseAuth}) \Rightarrow \text{AuthFailedException}$

### 16.4.6 Local Constants

None

## 17 MIS of User Profile Module

### 17.1 Module

User Profile

### 17.2 Uses

Authentication Module, Database Module, User Module

### 17.3 Syntax

#### 17.3.1 Exported Constants

None

#### 17.3.2 Exported Access Programs

None

### 17.4 Semantics

#### 17.4.1 State Variables

- User: FirebaseAuth
- CurrentUser: Boolean

#### 17.4.2 Environment Variables

None

#### 17.4.3 Assumptions

The user exists and the current user is logged in already.

#### 17.4.4 Access Routine Semantics

Name	In	Out	Exceptions
UpdateDisplay	String	Scene	-

### 17.4.5 Local Functions

UpdateDisplay(Message):

- transition:  $StatusMessage = Message$
- output: Scene
- exception:  $(Auth.LoginResult = false) \rightarrow InvalidLoginException$

GetUserData(user):

- transition:  $\exists\_email \in Database \Rightarrow Database.UserData$
- output:  $User = UserData$
- exception: None

### 17.4.6 Local Constants

- Placeholder: set of Strings
- Scene: Unity Scene that contains the default UI page

## 18 MIS of User Login Module

### 18.1 Module

User Login

### 18.2 Uses

Authentication Module

### 18.3 Syntax

#### 18.3.1 Exported Constants

None

#### 18.3.2 Exported Access Programs

Name	In	Out	Exceptions
UpdateDisplay	String	Scene	-

### 18.4 Semantics

#### 18.4.1 State Variables

- User: `FirebaseUser`

#### 18.4.2 Environment Variables

None

#### 18.4.3 Assumptions

The user is logged in already.

#### 18.4.4 Access Routine Semantics

`UpdateDisplay(Message)`:

- transition:  $StatusMessage = Message$
- output: `Scene`
- exception:  $(Auth.LoginResult = false) \rightarrow InvalidLoginException$

### 18.4.5 Local Functions

Login(\_email, \_password):

- transition:  $\exists \langle \_email, \_password \rangle \in \textit{Authentication} \rightarrow \textit{User} = \textit{Auth.LoginResult}$
- output: true if the credential is correct, false otherwise
- exception:  $(\textit{Auth.LoginResult} = \textit{false}) \rightarrow \textit{InvalidLoginException}$

### 18.4.6 Local Constants

Scene: Unity Scene that contains the default UI page

## 19 MIS of Lecture List Manager Module

### 19.1 Module

Lecture List Manager

### 19.2 Uses

Lecture Module, Pagination and Filter Module

### 19.3 Syntax

#### 19.3.1 Exported Constants

None

#### 19.3.2 Exported Access Programs

Name	In	Out	Exceptions
LectureList	-	-	-
Display	-	-	-
OnClickLecture	Lecture	-	IllegalArgumentException    Ex- ception
nextPage	-	-	-
prevPage	-	-	-
firstPage	-	-	-
lastPage	-	-	-
filter	String	-	-
AddLecture	Lecture	-	-

### 19.4 Semantics

#### 19.4.1 State Variables

- lecList: <Lecture>, displayed lectures
- pageNum:  $\mathbb{N}$ , current page
- keyword: String, filter keyword

#### 19.4.2 Environment Variables

None



### 19.4.3 Assumptions

None

### 19.4.4 Access Routine Semantics

LectureList():

- transition:  $lecList, pageNum, keyword := \text{all lectures in the database, 1, null}$
- output: none
- exception: none

Display():

- transition: Display a list of lecture displayedLecs, where  $displayedLecs := lecList.filter(keyword)[(pageNum-1)*PAGECOUNT, pageNum*PAGECOUNT]$
- output: none
- exception: none

OnClickLecture(targetLec):

- transition: Switch to Lecture Detail view with targetLec
- output: none
- exception:  $exc := targetLec \notin lecList \Rightarrow IllegalArgumentException$

nextPage():

- transition:  $pageNum*PAGECOUNT < lecList.length \Rightarrow pageNum := pageNum + 1$
- output: none
- exception: none

prevPage():

- transition:  $pageNum > 1 \Rightarrow pageNum := pageNum - 1$
- output: none
- exception: none

firstPage():

- transition:  $pageNum := 1$

- output: none
- exception: none

lastPage():

- transition:  $pageNum := int(lecList / PAGECOUNT) + 1$
- output: none
- exception: none

AddLecture(newLec):

- transition:  $lecList := PagniationandFilter.Add(lecList, newLec)$
- output: none
- exception: none

#### 19.4.5 Local Functions

None

#### 19.4.6 Local Constants

PAGECOUNT = 10

## 20 MIS of Event List Manager Module

### 20.1 Module

Event List Manager

### 20.2 Uses

Event Module, Pagination and Filter Module

### 20.3 Syntax

#### 20.3.1 Exported Constants

None

#### 20.3.2 Exported Access Programs

Name	In	Out	Exceptions
EventList	-	-	-
Display	-	-	-
OnClickEvent	Lecture	-	IllegalArgumentException    Ex- ception
nextPage	-	-	-
prevPage	-	-	-
firstPage	-	-	-
lastPage	-	-	-
filter	String	-	-
AddEvent	Event	-	-

### 20.4 Semantics

#### 20.4.1 State Variables

- eventList: <Event>, displayed events
- pageNum:  $\mathbb{N}$ , current page
- keyword: String, filter keyword

#### 20.4.2 Environment Variables

None

### 20.4.3 Assumptions

None

### 20.4.4 Access Routine Semantics

EventList():

- transition:  $eventList, pageNum, keyword := \text{all events in the database, 1, null}$
- output: none
- exception: none

Display():

- transition: Display a list of event displayedEvents, where  $displayedEvents := eventList.filter(keyword)[(pageNum-1)*PAGECOUNT, pageNum*PAGECOUNT]$
- output: none
- exception: none

OnClickEvent(targetEvent):

- transition: Switch to Event Detail view with targetEvent
- output: none
- exception:  $exc := targetEvent \notin eventList \Rightarrow IllegalArgumentException$

nextPage():

- transition:  $pageNum*PAGECOUNT < eventList.length \Rightarrow pageNum := pageNum + 1$
- output: none
- exception: none

prevPage():

- transition:  $pageNum > 1 \Rightarrow pageNum := pageNum - 1$
- output: none
- exception: none

firstPage():

- transition:  $pageNum := 1$

- output: none
- exception: none

lastPage():

- transition:  $pageNum := int(eventList / PAGECOUNT) + 1$
- output: none
- exception: none

AddEvent(newEvent):

- transition:  $eventList := PagniationandFilter.Add(eventList, newEvent)$
- output: none
- exception: none

#### **20.4.5 Local Functions**

None

#### **20.4.6 Local Constants**

PAGECOUNT = 10

## 21 MIS of Pagination and Filter Module

### 21.1 Module

Pagination and Filter

### 21.2 Uses

Database Module, Permission Module

### 21.3 Syntax

#### 21.3.1 Exported Constants

None

#### 21.3.2 Exported Access Programs

Name	In	Out	Exceptions
Initialize	-	-	-
Display	-	-	-
nextPage	-	-	-
prevPage	-	-	-
firstPage	-	-	-
lastPage	-	-	-
filter	String	-	-
Add	<T>, Activity	-	-

### 21.4 Semantics

#### 21.4.1 State Variables

- list: <T>, displayed entries
- pageNum:  $\mathbb{N}$ , current page
- keyword: String, filter keyword

#### 21.4.2 Environment Variables

None

### 21.4.3 Assumptions

Activity is a generic type  $\langle T \rangle$  and it can be instantiated with type Lecture and Event. The singleton module Permission is accessible from this module.

### 21.4.4 Access Routine Semantics

Initialize():

- transition:  $list, pageNum, keyword :=$  all type T entries in the database, 1, null
- output: none
- exception: none

Display():

- transition: Display a list of event T, where  $T := list.filter(keyword)[(pageNum - 1) * PAGECOUNT, pageNum * PAGECOUNT]$
- output: none
- exception: none

nextPage():

- transition:  $pageNum * PAGECOUNT < list.length \Rightarrow pageNum := pageNum + 1$
- output: none
- exception: none

prevPage():

- transition:  $pageNum > 1 \Rightarrow pageNum := pageNum - 1$
- output: none
- exception: none

firstPage():

- transition:  $pageNum := 1$
- output: none
- exception: none

lastPage():

- transition:  $pageNum := int(list/PAGECOUNT) + 1$

- output: none
- exception: none

Add(list, T):

- transition: none
- output:  $out := Permission.isAdmin \Rightarrow list := list + \{T\}$  and update database
- exception: none

#### **21.4.5 Local Functions**

None

#### **21.4.6 Local Constants**

PAGECOUNT: number of entries shown in one page



## 22 MIS of Database Module

### 22.1 Module

FirestoreDatabase

This module uses Firestore Realtime Database library. For details of all syntax and semantics of exported constants and access programs, see [Firestore database documentation](#).

### 22.2 Uses

None

### 22.3 Syntax

#### 22.3.1 Exported Constants

See [Firestore database documentation](#).

#### 22.3.2 Exported Access Programs

The following table will show some functions the application uses most frequently, for more details, see [Firestore database documentation](#).

Name	In	Out	Exceptions
Child	String	DatabaseReference	PermissionDenied, NetworkError, ExpiredToken
HasChild	String	$\mathbb{B}$	PermissionDenied, NetworkError, ExpiredToken
RemoveValueAsync	String	Task< $\mathbb{B}$ >	PermissionDenied, NetworkError, ExpiredToken
SetValueAsync	String, String	Task< $\mathbb{B}$ >	PermissionDenied, NetworkError, ExpiredToken
GetValueAsync	String	Task<DataSnapshot>	PermissionDenied, NetworkError, ExpiredToken
GoOffline	-	-	PermissionDenied, NetworkError, ExpiredToken
GoOnline	-	-	PermissionDenied, NetworkError, ExpiredToken

## 22.4 Semantics

### 22.4.1 State Variables

None

### 22.4.2 Environment Variables

- DBreference: `Firebase.Database.DatabaseReference`  
A reference to the root location of this database
- User: `Firebase.Auth.FirebaseUser`  
The current user that operates this database
- PermittedUsers: set of String  
The list of user emails that are allowed to read the database content
- Admins: set of String  
The list of user emails that are allowed to edit the database content

### 22.4.3 Assumptions

Assume the database connection is stable and it will not disconnect unless the user disconnect it manually.

### 22.4.4 Access Routine Semantics

Child(pathString):

- transition: none
- output:  $out := \text{DatabaseReference to pathString relative to the root}$
- exception:  $exc := NoInternet \Rightarrow NetworkError \mid TokenExpired \Rightarrow ExpiredToken \mid User.email \notin PermittedUsers \Rightarrow PermissionDenied$

HasChild(pathString):

- transition: none
- output:  $out := DBreference.Child(pathString) = \text{null}$
- exception:  $exc := NoInternet \Rightarrow NetworkError \mid TokenExpired \Rightarrow ExpiredToken \mid User.email \notin PermittedUsers \Rightarrow PermissionDenied$

RemoveValueAsync(pathString):

- transition:  $DBreference.Child(pathString) := \text{null}$
- output:  $out := DBreference.HasChild(pathString)$
- exception:  $exc := NoInternet \Rightarrow NetworkError \mid TokenExpired \Rightarrow ExpiredToken \mid User.email \notin Admins \Rightarrow PermissionDenied$

SetValueAsync(pathString, value):

- transition:  $DBreference.Child(pathString) := value$
- output:  $out := DBreference.Child(pathString) = value$
- exception:  $exc := NoInternet \Rightarrow NetworkError \mid TokenExpired \Rightarrow ExpiredToken \mid User.email \notin Admins \Rightarrow PermissionDenied$

GetValueAsync(pathString):

- transition: none
- output:  $out := \text{Snapshot of } DBreference.Child(pathString)$

- exception:  $exc := NoInternet \Rightarrow NetworkError \mid TokenExpired \Rightarrow ExpiredToken \mid User.email \notin PermittedUsers \Rightarrow PermissionDenied$

GoOffline():

- transition: Manually disconnect the FirebaseDatabase client from the server and disable automatic reconnection.
- output: none
- exception:  $exc := NoInternet \Rightarrow NetworkError \mid TokenExpired \Rightarrow ExpiredToken \mid User.email \notin Admins \Rightarrow PermissionDenied$

GoOnline():

- transition: Manually reestablish a connection to the FirebaseDatabase server and enable automatic reconnection.
- output: none
- exception:  $exc := NoInternet \Rightarrow NetworkError \mid TokenExpired \Rightarrow ExpiredToken \mid User.email \notin Admins \Rightarrow PermissionDenied$

#### 22.4.5 Local Functions

None

## 23 MIS of Server Module

### 23.1 Module

RTCTServer

### 23.2 Uses

None

### 23.3 Syntax

#### 23.3.1 Exported Constants

None

#### 23.3.2 Exported Access Programs

Name	In	Out	Exceptions
SendMessage	User, String	Task	-
SendLocationGroup	$\mathbb{R}$ , $\mathbb{R}$	Task	-

### 23.4 Semantics

#### 23.4.1 State Variables

None

#### 23.4.2 Environment Variables

None

#### 23.4.3 Assumptions

User identifiers are unique.

#### 23.4.4 Access Routine Semantics

SendMessage(recipient, msg):

- transition: none
- output:  $out := \text{Task}$ ;  $out.IsCompleted := \text{True}$
- exception: none

SendLocation(friendGroup, lat, lon):

- transition: none
- output:  $out := \text{Task}; out.IsCompleted := \text{True}$
- exception: none

#### **23.4.5 Local Functions**

None

## 24 MIS of AR Camera

### 24.1 Module

AR Camera

### 24.2 Uses

None

### 24.3 Syntax

#### 24.3.1 Exported Constants

None

#### 24.3.2 Exported Access Programs

Name	In	Out	Exceptions
DetectTarget	-	-	-

### 24.4 Semantics

#### 24.4.1 State Variables

None

#### 24.4.2 Environment Variables

- cameraFeed: 2D array of pixels
- sceneCamera: Camera
- imageTargets: list of Target
- scanTargets: list of Target

#### 24.4.3 Assumptions

None

#### **24.4.4 Access Routine Semantics**

DetectTarget():

- transition: Implicitly invokes the AR Interface when a valid target is detected.
- output: none
- exception: none

#### **24.4.5 Local Functions**

None



## 25 MIS of AR Interface

### 25.1 Module

AR Interface

### 25.2 Uses

AR Camera

### 25.3 Syntax

#### 25.3.1 Exported Constants

None

#### 25.3.2 Exported Access Programs

Name	In	Out	Exceptions
Initialize	(String, <3D Objects>)	-	IllegalArgument Exception
Display	String	-	IllegalArgument Exception

### 25.4 Semantics

#### 25.4.1 State Variables

- dictionary: Dictionary<String, <3D Objects>>, the dictionary of target name and corresponding AR objects

#### 25.4.2 Environment Variables

None

#### 25.4.3 Assumptions

None

#### 25.4.4 Access Routine Semantics

Initialize(target, objects):

- transition:  $dictionary[target] := objects$
- output: none

- exception:  $target \notin dictionary.keys \Rightarrow IllegalArgumentException$

Display(target):

- transition: Displays *dictionary[target]* objects in Unity scene
- output: none
- exception:  $target \notin dictionary.keys \Rightarrow IllegalArgumentException$

#### **25.4.5 Local Functions**

None

## 26 MIS of MapBox

### 26.1 Module

MapBox

Third party library Mapbox

### 26.2 Uses

None

### 26.3 Syntax

#### 26.3.1 Exported Constants

None

#### 26.3.2 Exported Access Programs

Name	In	Out	Exceptions
Map	-	-	IllegalArgument Exception
Display	-	-	-
ChangeStyle	String	-	IllegalArgument Exception
Pan	$(\mathbb{R}, \mathbb{R})$	-	
Zoom	$\mathbb{N}$	-	

### 26.4 Semantics

#### 26.4.1 State Variables

- MapStyle: String
- MapCenter:  $(\mathbb{R}, \mathbb{R})$
- Zoom:  $\mathbb{N}$

#### 26.4.2 Environment Variables

- APIKey: String

#### 26.4.3 Assumptions

None

#### 26.4.4 Access Routine Semantics

Map():

- transition: Check APIKey and initialize the map
- output: none
- exception:  $exc := APIKeyexpires \Rightarrow IllegalArgumentException$

Display():

- transition: Displays a map with default state variables
- output: none
- exception: none

ChangeStyle(style):

- transition:  $MapStyle := style$
- output: none
- exception:  $exc := style \notin MAPSTYLES \Rightarrow IllegalArgumentException$

Pan((long, lat)):

- transition:  $MapCenter := (long, lat)$
- output: none
- exception: none

Zoom(scale):

- transition:  $Zoom := scale$
- output: none
- exception: none

#### 26.4.5 Local Functions

None

#### 26.4.6 Local Constants

MAPSTYLES = a sequences of map styles with type String

## 27 MIS of Real-time Map

### 27.1 Module

Real-time Map

### 27.2 Uses

Map Module, Server Module

### 27.3 Syntax

#### 27.3.1 Exported Constants

None

#### 27.3.2 Exported Access Programs

Name	In	Out	Exceptions
HandleInputBuilding	-	-	-
DisplayAvatar	Uri, ( $\mathbb{R}$ , $\mathbb{R}$ )	-	-
ConnectToServer	-	-	-
DisplayEventHeatMap	-	-	-

### 27.4 Semantics

#### 27.4.1 State Variables

- building: list of BuildingLocation

#### 27.4.2 Environment Variables

- camera: Camera

#### 27.4.3 Assumptions

ConnectToServer() is called when the module is loaded.

#### 27.4.4 Access Routine Semantics

HandleInputBuilding():

- transition: Opens user interface when a building marker is tapped
- output: none

- exception: none

DisplayAvatar(photoUri, (long, lat)):

- transition: Displays the corresponding avatar on the map at (long, lat)
- output: none
- exception: none

DisplayUserHeatMap():

- transition: Retrieves collected user location data from the database and plot them on the map
- output: none
- exception: none

DisplayEventHeatMap():

- transition: Retrieves collected event location data from the database and plot them on the map
- output: none
- exception: none

#### **27.4.5 Local Functions**

None

## 28 MIS of Friend Chat

### 28.1 Module

Friend Chat

### 28.2 Uses

Server Module

### 28.3 Syntax

#### 28.3.1 Exported Constants

None

#### 28.3.2 Exported Access Programs

Name	In	Out	Exceptions
StartConnection	String, String	-	-
SendMessage	User, String	-	-
ReceiveMessage	User, String	-	-

### 28.4 Semantics

#### 28.4.1 State Variables

- connection: HubConnection
- onMessageReceived: Action

#### 28.4.2 Environment Variables

None

#### 28.4.3 Assumptions

None

#### 28.4.4 Access Routine Semantics

StartConnection(url, handler):

- transition: Creates a new HubConnection and stores it in connection. Connects the given handler to the server endpoint.

- output: none
- exception: none

SendMessage(recipient, message):

- transition: Sends a message to the recipient through the server hub connection.
- output: none
- exception: none

ReceiveMessage(sender, message):

- transition: Receives a message from the server hub connection. The sender's id is received as well.
- output: none
- exception: none

#### **28.4.5 Local Functions**

None



## 29 Appendix

### 29.1 Database Tables

#### User

Column Name	Type	Description
email	String	ID of a user
nickName	(Optional) String	Nickname/display name of a user
photoUri	(Optional) Uri	Visual Avatar
program	(Optional) String	Study field
level	(Optional) int	Level of program
friends	(Optional) <User>	List of friends
friendRequests	(Optional) <User>	List of requesters
lectures	(Optional) <Lecture>	List of pinned lecture
events	(Optional) <Event>	List of pinned event

#### Lecture

Column Name	Type	Description
code	String	ID of a course, course code
name	(Optional) String	formal name of a course
instructor	(Optional) String	name of the instructor
time	(Optional) String	Includes start and end time in a weekly schedule
location	(Optional) String	Building and room

#### Event

Column Name	Type	Description
name	String	ID of an event
description	(Optional) String	event description
organizer	(Optional) String	organizer of the event
startTime	(Optional) DateTime	when it starts
duration	(Optional) int	how long is the event (in minutes)
location	(Optional) String	Building and room
isPublic	$\mathbb{B}$	If it is a public event

## References

- Carlo Ghezzi, Mehdi Jazayeri, and Dino Mandrioli. *Fundamentals of Software Engineering*. Prentice Hall, Upper Saddle River, NJ, USA, 2nd edition, 2003.
- Daniel M. Hoffman and Paul A. Strooper. *Software Design, Automated Testing, and Maintenance: A Practical Approach*. International Thomson Computer Press, New York, NY, USA, 1995. URL <http://citeseer.ist.psu.edu/428727.html>.