Module Interface Specification for Campus Connections

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
Apr 4	1.1	Revision 1: Resolve TA feedback; Updated MIS to be consisted with design changes, implemented feedback, added new modules

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

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 $(c_1 \Rightarrow r_1|c_2 \Rightarrow r_2|...|c_n \Rightarrow r_n)$.

The following table summarizes the primitive data types used by Campus Connections.

Data Type	Notation	Description
character	char	a single symbol or digit
integer	$\mathbb{Z}$	a number without a fractional component in (-∞, ∞)
natural number	$\mathbb{N}$	a number without a fractional component in [1, ∞)
real	$\mathbb{R}$	any number in (-∞, ∞)
boolean	$\mathbb{B}$	True or False
sequence of T	<T>	a list of object with type T
asynchronous step T	Task<T>	an asynchronous result of T
activity	Activity	generic class with <T> that can be instantiated 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
server connection	Connection	A WebSocket-based connection to the backend server

The specification of Campus Connections 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, Campus Connections 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	Hardware-Hiding Module
Behaviour-Hiding	AR Interface Module
	RealTimeMap Module
	User Module
	Lecture Module
	Event Module
	DBConnector Module
	AuthConnector 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
	Notification 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	-	IndexOutofBound Exception
AddRequester	User	-	-
RemoveRequester	User	-	IndexOutofBound Exception
AddLecture	Lecture	-	-
RemoveLecture	Lecture	-	IndexOutofBound Exception
AddEvent	Event	-	-
RemoveEvent	Event	-	IndexOutofBound Exception

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: <User>, List of friends
- requesters: <User>, List of friend requester
- lectures: <Lecture>, Pinned lecture
- events: <Event>, 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	$\mathbb{N}$	-	-
AddFriend	User	-	-
DeleteFriend	User	-	IndexOutofBound Exception
AddRequest	User	-	-
DeleteRequest	User	-	IndexOutofBound Exception
PinLecture	Lecture	-	-
UnPinLecture	Lecture	-	IndexOutofBound Exception
PinEvent	Event	-	-
UnPinEvent	Event	-	IndexOutofBound Exception

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 sate 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	-	IndexOutofBound Exception
onClickViewFriend	User	-	IndexOutofBound Exception
onClickMessageFriend	User	2D seq of pixels	IndexOutofBound Exception
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* $(\forall x : \mathbb{Z} | 0 \leq x \leq Account.friends.length : friendsContainer[i].position, friendsContainer[i].content = (0, i * HEIGHT), Account.friends[i])$,
- 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 Exception
onClickIgnoreRequest	User	-	IllegalArgument Exception

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(T)

12.2 Uses

DBConnector Module, AuthConnector Module

12.3 Syntax

12.3.1 Exported Constants

none

12.3.2 Exported Type

ActivityDetailView = ?

12.3.3 Exported Access Programs

Name	In	Out	Exceptions
new ActivityDetail-View	T	-	-
ViewActivity	-	T	-
AddActivity	T	-	-
EditActivity	String, T	-	-
DeleteActivity	T	-	-
BookmarkActivity	T, String	-	-
UnbookmarkActivity	T, String	-	-

12.4 Semantics

12.4.1 State Variables

- activity: set of T
- bookmarkedActivities: set of T
- bookmarked: $\mathbb{B}$

12.4.2 Environment Variables

none

12.4.3 Assumptions

All T has an attribute ID, which stands for the identity of the element.

12.4.4 Access Routine Semantics

new ActivityDetailView(clickedEntry):

- transition: $activity, bookmarkedActivities, bookmarked := clickedEntry, Bookmarked(AuthConnector.CurrentUser.Email), activity \in bookmarkedActivities$
- output: $out := self$
- exception: none

ViewActivities():

- transition: none
- output: $out := activity$
- exception: none

AddActivity(newActivity):

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

EditActivity(targetID, editedActivity):

- transition: $\forall activityinactivities \mid activity := activity.ID = targetID \Rightarrow editedActivity \mid activity$
- output: none
- exception: none

DeleteActivity(targetActivity):

- transition: $activities := activities - \{targetActivity\}$
- output: none
- exception: none

BookmarkActivity(newActivity, targetID):

- transition: $bookmarkedActivities, bookmarked := bookmarkedActivities + \{newActivity\}$, $activity \in bookmarkedActivities$; $DBConnector.Root.Child(AuthConnector.CurrentUser.Email).Child(T).Child(targetID).$ $.setAsycValue(newActivity.ToJSON())$
- output: none
- exception: none

UnbookmarkActivity(targetActivity, targetID):

- transition: $bookmarkedActivities, bookmarked := bookmarkedActivities - \{targetActivity\}$, $activity \in bookmarkedActivities$; $DBConnector.Root.Child(AuthConnector.CurrentUser.Email).Child(T).$ $Child(targetID).setValueAsyc(null)$
- output: none
- exception: $exc := targetActivity \notin pinnedActivities \Rightarrow IndexOutofBoundException$

12.4.5 Local Functions

Bookmarked(_email):

- transition: none
- output: $out :=< new\ T(data) > |data \in DBConnector.Root.Child(_email).Child(T)$
- exception: none

12.4.6 Local Constants

None

13 MIS of Lecture Detail View Module

13.1 Module

LectureDetailView (ActivityDetailView<Lecture>)

13.2 Uses

DBConnector Module, AuthConnector Module, Lecture Module

13.3 Syntax

The rest of the sections of the module is the same as ActivityDetailView

14 MIS of Event Detail View Module

14.1 Module

EventDetailView (Activity Detail View <Event>)

14.2 Uses

Activity Detail View Module, Event Module

14.3 Syntax

The rest of the sections of the module is the same as ActivityDetailView

15 MIS of Authentication Module

15.1 Module

Authentication

15.2 External Module Documentation

This module is provided by a 3rd party library (Firebase Authentication). For details of all syntax and semantics of exported constants and access programs, refer to the Firebase Auth Unity API Documentation. documentation

15.3 Uses

Hardware-Hiding Module, Database Module

15.4 Syntax

15.4.1 Exported Constants

Please refer to the external module documentation section.

15.4.2 Exported Access Programs

Please refer to the external module documentation section.

15.5 Semantics

15.5.1 State Variables

Please refer to the external module documentation section.

15.5.2 Environment Variables

Please refer to the external module documentation section.

15.5.3 Assumptions

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

15.5.4 Access Routine Semantics

Please refer to the external module documentation section.

15.5.5 Local Functions

Please refer to the external module documentation section.

15.5.6 Local Constants

Please refer to the external module documentation section.

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: $\exists < \_email, \_password > \in FirebaseAuth \Rightarrow Login$
- output: $User = AuthResult.CurrentUser$
- exception: $exc := \neg(\exists < \_email, \_password > \in FirebaseAuth) \Rightarrow AuthFailedException$

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: FirebaseUser
- 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: $\exists < \_email, \_password > \in Authentication \rightarrow User = Auth.LoginResult$
- output: true if the credential is correct, false otherwise
- exception: $(Auth.LoginResult = false) \rightarrow InvalidLoginException$

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, PaginationNFilter<Lecture> Module, DBConnector Module, ActivityDetailView<Lecture>

19.3 Syntax

19.3.1 Exported Constants

None

19.3.2 Exported Access Programs

Name	In	Out	Exceptions
init	-	-	-
Display	<Lecture>	-	-
OnClickLecture	Lecture	-	-
nextPage	-	-	-
prevPage	-	-	-
firstPage	-	-	-
lastPage	-	-	-
AddLecture	Lecture	-	-
DeleteLecture	Lecture	-	-
FilterLecture	String, String	<Lecture>	-

19.4 Semantics

19.4.1 State Variables

- lecList: PaginationNFilter<Lecture>, displayed lectures
- lecDetailView: ActivitDetailView<Lecture>, details of the lecture the user clicks on

19.4.2 Environment Variables

None

19.4.3 Assumptions

LectureListManager.init() is called before any other access program.

19.4.4 Access Routine Semantics

init():

- transition: $lecList := newPaginationNFilter < Lecture > (allLecture(), `code', '')$
- output: none
- exception: none

Display():

- transition: Display lecture entries of the list in the current page on the screen
- output: $out := currentPage * PageCount < lecList.filteredList.length \Rightarrow lecList.filteredList[(currentPage - 1) * PageCount, (currentPage) * pageCount] \mid lecList.filteredList[(currentPage - 1) * PageCount :]$
- exception: none

OnClickLecture(targetLec):

- transition: $lecDetailView := new\ ActivityDetailView < Lecture > (targetLec)$
- output: none
- exception: none

nextPage():

- transition: $lecList.nextPage()$
- output: none
- exception: none

prevPage():

- transition: $lecList.prevPage()$
- output: none
- exception: none

firstPage():

- transition: $lecList.firstPage$

- output: none
- exception: none

lastPage():

- transition: $lecList.lastPage$
- output: none
- exception: none

AddLecture(newLec):

- transition: $lecList.Add(newLec)$;
  $DBConnector.Root.Child(Lecture).Child(newLec.code).setAsyncValue(newLec.ToJSON())$
- output: none
- exception: none

DeleteLecture(targetLec):

- transition: $lecList.Remove(newLec)$;
  $DBConnector.Root.Child(Lecture).Child(newLec.code).setAsyncValue(null))$
- output: none
- exception: none

FilterLecture(filterBy, filterString):

- transition: $lecList.filterBy, lecList.filterString := filterBy, filterString$;
  $lecList.filter(); Display()$
- output: $out := lecList.filteredlist$
- exception: none

19.4.5 Local Functions

AllLectures():

- transition: none
- output: A list of lectures retrieved from the DBConnector.Root
- exception: none

19.4.6 Local Constants

None

20 MIS of Event List Manager Module

20.1 Module

Event List Manager

20.2 Uses

Event Module, PaginationNFilter<Event> Module, DBConnector Module, ActivityDetailView<Event>

20.3 Syntax

20.3.1 Exported Constants

None

20.3.2 Exported Access Programs

Name	In	Out	Exceptions
init	-	-	-
Display	<Event>	-	-
OnClickEvent	Event	-	-
nextPage	-	-	-
prevPage	-	-	-
firstPage	-	-	-
lastPage	-	-	-
AddEvent	Event	-	-
DeleteEvent	Event	-	-
FilterEvent	String, String	<Event>	-

20.4 Semantics

20.4.1 State Variables

- eventList: PaginationNFilter<Event>, displayed events
- eventDetailView: ActivitDetailView<Event>, details of the event the user clicks on

20.4.2 Environment Variables

None

20.4.3 Assumptions

EventListManager.init() is called before any other access program.

20.4.4 Access Routine Semantics

init():

- transition: $eventList := newPaginationNFilter < Event > (allEvent(), `name', '')$
- output: none
- exception: none

Display():

- transition: Display event entries of the list in the current page on the screen
- output: $out := currentPage * PageCount < eventList.filteredList.length \Rightarrow eventList.filteredList[(currentPage - 1) * PageCount, (currentPage) * pageCount] \mid eventList.filteredList[(currentPage - 1) * PageCount :]$
- exception: none

OnClickEvent(targetEvent):

- transition: $eventDetailView := new\ ActivityDetailView < Event > (targetEvent)$
- output: none
- exception: none

nextPage():

- transition: $eventList.nextPage()$
- output: none
- exception: none

prevPage():

- transition: $eventList.prevPage()$
- output: none
- exception: none

firstPage():

- transition: $eventList.firstPage$

- output: none
- exception: none

lastPage():

- transition: $eventList.lastPage$
- output: none
- exception: none

AddEvent(newEvent):

- transition: $eventList.Add(newEvent)$;
  $DBConnector.Root.Child(Event).Child(newEvent.name).setAsyncValue(newEvent.ToJSON())$
- output: none
- exception: none

DeleteEvent(targetEvent):

- transition: $lecList.Remove(newEvent)$;
  $DBConnector.Root.Child(Event).Child(newEvent.name).setAsyncValue(null))$
- output: none
- exception: none

FilterEvent(filterBy, filterString):

- transition: $eventList.filterBy, lecList.filterString := filterBy, filterString;$
  $eventList.filter(); Display()$
- output: $out := eventList.filteredlist$
- exception: none

20.4.5 Local Functions

AllEvents():

- transition: none
- output: A list of events retrieved from the DBConnector.Root
- exception: none

20.4.6 Local Constants

None

21 MIS of Pagination and Filter Module

21.1 Module

PaginationNFilter(T)

21.2 Uses

None

21.3 Syntax

21.3.1 Exported Constants

none

21.3.2 Exported Type

PaginationNFilter = ?

21.3.3 Exported Access Programs

Name	In	Out	Exceptions
new PagniationNFilter	<T>, String, String	-	-
nextPage	-	-	-
prevPage	-	-	-
firstPage	-	-	-
lastPage	-	-	-
filter	-	<T>	-
Add	T	-	-
Remove	T	-	-

21.4 Semantics

21.4.1 State Variables

- list: <T>, displayed entries
- filteredList: <T>, filtered entries
- currentPage: $\mathbb{N}$, current page

- maxPage: $\mathbb{N}$, max page
- filterBy: String, filter option
- filterString: String, filter string

21.4.2 Environment Variables

None

21.4.3 Assumptions

T has an attribute that can be compared.

21.4.4 Access Routine Semantics

new PaginationNFilter(entries, filterBy, filterString):

- transition: $list, currentPage, filterBy, filterString := entries, 1, filterBy, filterString$; $filter()$
- output: $out := self$
- exception: none

filter():

- transition: $filteredList, maxPage, currentPage :=< entry > \mid entry \in list \wedge entry[filterBy].contains(filterString), UpdateMax(filteredList), 1$
- output: $out := filterList$
- exception: none

nextPage():

- transition: $currentPage < maxPage \Rightarrow currentPage := currentPage + 1$
- output: $out := currentPage$
- exception: none

prevPage():

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

firstPage():

- transition: $currentPage := 1$
- output: $out := currentPage$
- exception: none

lastPage():

- transition: $currentPage := maxPage$
- output: $out := currentPage$
- exception: none

Add(list, T):

- transition: $list := list + \{T\}; filter()$
- output: none
- exception: none

Remove(list, T):

- transition: $list := list - \{T\}; filter()$
- output: none
- exception: none

21.4.5 Local Functions

UpdateMaxPage(entries):

- transition: none
- output: $out := entries.length = 0 \Rightarrow 1 \mid entries.length \mod PageCount \Rightarrow entries.length/PageCount \mid \lceil entries.length/PageCount \rceil$
- exception: none

21.4.6 Local Constants

PageCount: 10

22 MIS of Notification Module

22.1 Module

Notification

22.2 Uses

None

22.3 Syntax

22.3.1 Exported Constants

none

22.3.2 Exported Access Programs

Name	In	Out	Exceptions
new Notification	String	-	-
close	-	-	-

22.4 Semantics

22.4.1 State Variables

- message: String, notification message

22.4.2 Environment Variables

None

22.4.3 Assumptions

None

22.4.4 Access Routine Semantics

new Notification(message):

- transition: $message := message$;Display the message on the screen
- output: $out := self$
- exception: none

close():

- transition: Hide the message
- output: none
- exception: none

22.4.5 Local Functions

None

22.4.6 Local Constants

None

23 MIS of Database Module

23.1 Module

FirebaseDatabase

23.2 External Module Documentation

This module is provided by a 3rd party library (Firebase Realtime Database). For details of all syntax and semantics of exported constants and access programs, refer to the Firebase Database Unity API Documentation. documentation

23.3 Uses

Hardware-Hiding Module

23.4 Syntax

23.4.1 Exported Constants

Please refer to the external module documentation section.

23.4.2 Exported Access Programs

Please refer to the external module documentation section.

23.5 Semantics

23.5.1 State Variables

Please refer to the external module documentation section.

23.5.2 Environment Variables

Please refer to the external module documentation section.

23.5.3 Assumptions

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

23.5.4 Access Routine Semantics

Please refer to the external module documentation section.

23.5.5 Local Functions

None

24 MIS of Server Module

24.1 Module

RTCServer

24.2 Uses

None

24.3 Syntax

24.3.1 Exported Constants

None

24.3.2 Exported Access Programs

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

24.4 Semantics

24.4.1 State Variables

None

24.4.2 Environment Variables

None

24.4.3 Assumptions

User identifiers are unique.

24.4.4 Access Routine Semantics

SendMessage(msg):

- transition: none
- output: out := Task; out.IsCompleted := $True$; Invokes the ReceiveMessage() function on all other connected clients.
- exception: none

SendLocation(e, lat, lon):

- transition: none
- output: out := Task; out.IsCompleted := $True$; Invokes the ReceiveMessage() function on all other connected clients.
- exception: none

24.4.5 Local Functions

None

25 MIS of AR Camera

25.1 Module

AR Camera

25.2 External Module Documentation

This module is provided by a 3rd party library (Vuforia). The API documentation can be found in the Vuforia Unity API Reference.

25.3 Uses

Hardware-Hiding Module

25.4 Syntax

25.4.1 Exported Constants

Please refer to the external module documentation section.

25.4.2 Exported Access Programs

Please refer to the external module documentation section.

25.5 Semantics

25.5.1 State Variables

Please refer to the external module documentation section.

25.5.2 Environment Variables

Please refer to the external module documentation section.

25.5.3 Assumptions

None

25.5.4 Access Routine Semantics

Please refer to the external module documentation section.

25.5.5 Local Functions

Please refer to the external module documentation section.

26 MIS of AR Interface

26.1 Module

AR Interface

26.2 Uses

AR Camera, Notification

26.3 Syntax

26.3.1 Exported Constants

None

26.3.2 Exported Access Programs

Name	In	Out	Exceptions
Initialize	(String, <GameObject>)	-	-
Display	String	-	-
OnTargetClick	-	-	-

26.4 Semantics

26.4.1 State Variables

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

26.4.2 Environment Variables

- audio: AudioPlayer

26.4.3 Assumptions

The AudioPlayer is provided as an environment variable by the Unity engine

26.4.4 Access Routine Semantics

Initialize(target, objects):

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

- exception: none

Display(target):

- transition: Displays $dictionary[target]$ objects in Unity scene
- output: none
- exception: none

OnTargetClick():

- transition: *audio*.PlaySound(); Play a sound clip when an AR Target is clicked or tapped.
- output: none
- exception: none

26.4.5 Local Types

GameObject := Data type used by the Unity engine to represent 3D AR Objects

26.4.6 Local Functions

None

27 MIS of MapBox

27.1 Module

MapBox

27.2 External Module Documentation

This module is provided by a 3rd party library (Mapbox). The API documentation can be found in the Maps SDK for Unity and Mapbox Unity API Reference.

27.3 Uses

Hardware-Hiding Module

27.4 Syntax

27.4.1 Exported Constants

Please refer to the external module documentation section.

27.4.2 Exported Access Programs

Please refer to the external module documentation section.

27.5 Semantics

27.5.1 State Variables

Please refer to the external module documentation section.

27.5.2 Environment Variables

Please refer to the external module documentation section.

27.5.3 Assumptions

None

27.5.4 Access Routine Semantics

Please refer to the external module documentation section.

27.5.5 Local Functions

Please refer to the external module documentation section.

27.5.6 Local Constants

Please refer to the external module documentation section.

28 MIS of Real-time Map

28.1 Module

Real-time Map

28.2 Uses

Map Module, Server Module

28.3 Syntax

28.3.1 Exported Constants

None

28.3.2 Exported Access Programs

Name	In	Out	Exceptions
StartConnection	String, String	-	-
SendLocation	RemoteUserLocation	-	-
ReceiveLocation	RemoteUserLocation	-	-
DisplayUserLocations	-	-	-
HandleInputBuilding	-	-	-

28.4 Semantics

28.4.1 State Variables

- connection: Connection
- buildings: List<⟨$\mathbb{R}$, $\mathbb{R}$⟩>
- usrLoc: UserLocation
- rmtUsrLoc: set of UserLocation

28.4.2 Environment Variables

- locationServer: Server
- gps: GPS

28.4.3 Assumptions

StartConnection() is called first when the module is loaded.
The user's mobile device automatically provides the gps Environment Variable.
The *gps* automatically updates *usrLoc*.

28.4.4 Access Routine Semantics

StartConnection(*url*, *end*):

- transition: *connection* := new Connection(*locationServer*, *url*, *end*)
- output: none
- exception: none

SendLocation(*l*):

- transition: $l := gps$.GetLocation(), *locationServer*.SendLocation(l)
- output: none
- exception: none

ReceiveLocation(l):

- transition: $rmtUsrLoc := rmtUsrLoc \cup \{l\}$
- output: none
- exception: none

DisplayUserLocations():

- transition: Renders *usrLoc* and $rmtUsrLoc$ on the map as visual elements.
- output: none
- exception: none

HandleInputBuilding():

- transition: Displays an interface when building $b \in building$ is tapped/clicked.
- output: none
- exception: none

28.4.5 Local Types

UserLocation = tuple of (latitude: $\mathbb{R}$, longitude: $\mathbb{R}$, email: String)

28.4.6 Local Functions

29 MIS of Friend Chat

29.1 Module

Friend Chat

29.2 Uses

Server Module

29.3 Syntax

29.3.1 Exported Constants

None

29.3.2 Exported Access Programs

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

29.4 Semantics

29.4.1 State Variables

- connection: Connection
- messages: List<String>

29.4.2 Environment Variables

- chatServer: Server

29.4.3 Assumptions

The handler parameter in StartConnection is always set to "ReceiveMessage".

29.4.4 Access Routine Semantics

StartConnection(*url*, *handler*):

- transition: *connection* := new Connection(*chatServer*, *url*, *handler*)
- output: none

- exception: none

SendMessage(m):

- transition: $chatServer$.SendMessage(m)
- output: none
- exception: none

ReceiveMessage(m):

- transition: $messages := messages$.Append(m)
- output: none
- exception: none

29.4.5 Local Functions

None

30 Appendix

30.1 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.