Module Interface Specification for Software Engineering

Team #2, Campus Connections
Waseef Nayeem
Zihao Du
Matthew Miller
Firas Elayan
Abhiram Neelamraju
Michael Kim

January 17, 2024

1 Revision History

Date	**Version**	**Notes**
Jan 15	1.0	Add introduction and module decomposition
Date 2	1.1	Notes

2 Symbols, Abbreviations and Acronyms

See SRS Documentation at SRS

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 and Module Guide. 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

[You should describe your notation. You can use what is below as a starting point. —SS]

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 Software Engineering.

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 (-∞, ∞)

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 User Module Lecture Module Event Module User Profile Module Friend Manager Module Friend Request 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 Map Module Activity Detail View Module Pagination and Filter Module

Table 1: Module Hierarchy

6 MIS of Friend Manager Module

6.1 Module

FriendManager

6.2 Uses

User Module, Database Module, Authentication Module, Server Module

6.3 Syntax

6.3.1 Exported Constants

None

6.3.2 Exported Access Programs

Name	**In**	**Out**	**Exceptions**
GetUser	-	-	-
GetFriendList	-	-	-
DisplayFriendList	-	-	
DeleteFriend	String	-	IndexOutofBound Exception
ViewFriend	String	-	IndexOutofBound Exception
MessageFriend	String	-	IndexOutofBound Exception, Server-Connection Exception
SendRequest	String	String	-

6.4 Semantics

6.4.1 State Variables

- currentUser: FirebaseUser
- friends: set of User
- friendsContainer: set of Transform

6.4.2 Environment Variables

None

6.4.3 Assumptions

Assume friends are updated in the database immediately after the request is sent.

6.4.4 Access Routine Semantics

GetUser():

- transition: $currentUser := Authentication.CurrentUser$
- output: none
- exception: none

GetFriendList():

- transition: $friends := GetFriendsFromDB(currentUser)$
- output: none
- exception: none

DisplayFriendList():

- transition: $(\forall x : \mathbb{Z} | 0 \leq x \leq friends.length :$ $friendsContainer[i].position, friendsContainer[i].content = (0, i{*}HEIGHT), friends[i])$, then display a list using friendsContainer
- output: none
- exception: none

DeleteFriend(targetEmail):

- transition: $friends :=$ friends - {targetEmail}
- output: none
- exception: $exc := targetEmail \notin friends \Rightarrow IndexOutofBoundException$

ViewFriend(targetEmail):

- transition: Switch scene to user profile of the target user
- output: none
- exception: $exc := targetEmail \notin friends \Rightarrow IndexOutofBoundException$

MessageFriend(targetEmail):

- transition: Display Chat UI between $currentUser$ and $targetEmail$

- output: none
- exception: $exc := targetEmail \notin friends \Rightarrow IndexOutofBoundException$

SendRequest(targetEmail):

- transition: $targetEmail \notin friends \Rightarrow$ Add request in target user request list in the database
- output: $targetEmail \notin friends$
- exception: none

6.4.5 Local Functions

GetFriendsFromDB(email): Seq of User
It gets all friends under the input user email and convert them to an array of User

- transition: none
- output: $out := Database.GetValueAsync(ROOT{+}email{+}FRIENDPATH).ToArray()$
- exception: none

6.4.6 Local Constants

HEIGHT = 300 px
ROOT = Database root path
FRIENDPATH = path string for user friends list

7 MIS of Friend Request Module

7.1 Module

FriendRequest

7.2 Uses

User Module, Database Module, Authentication Module,

7.3 Syntax

7.3.1 Exported Constants

None

7.3.2 Exported Access Programs

Name	In	Out	Exceptions
GetUser	-	-	-
GetRequestList	-	-	-
DisplayRequestList	-	-	
AcceptRequest	String	-	IllegalArgument Exception
IgnoreRequest	String	-	IllegalArgument Exception

7.4 Semantics

7.4.1 State Variables

- currentUser: FirebaseUser
- requests: set of User
- requestsContainer: set of Transform
- requestNum: $\mathbb{Z}$

7.4.2 Environment Variables

None

7.4.3 Assumptions

Assume friend requests are updated in the database immediately after the request is sent.

7.4.4 Access Routine Semantics

GetUser():

- transition: $currentUser := Authentication.CurrentUser$
- output: none
- exception: none

GetRequestList():

- transition: $friends := GetRequestsFromDB(currentUser)$
- output: none
- exception: none

DisplayRequestList():

- transition: $(\forall x : \mathbb{Z} | 0 \leq x \leq requests.length :$
  $requestsContainer[i].position, requestsContainer[i].content = (0, i{*}HEIGHT), requests[i])$,
  then display a list using requestsContainer
- output: none
- exception: none

AcceptRequest(targetEmail):

- transition: targetEmail is added in currentUser friend list and currentEmail is added in targetEmail friend list in the database
  $request := request - \{targetEmail\}$
- output: none
- exception: $exc := Database.HasChild(ROOT + targetEmail) = \text{null} \Rightarrow$
  $IllegalArgumentException$

IgnoreRequest(targetEmail):

- transition: $request := request - \{targetEmail\}$
- output: none
- exception: $exc := Database.HasChild(ROOT + targetEmail) = \text{null} \Rightarrow$
  $IllegalArgumentException$

7.4.5 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

GetRequestsFromDB(email): Seq of User
It gets all friend requesters under the input user email and convert them to an array of User

- transition: none
- output: $out := Database.GetValueAsync(ROOT + email + REQUESTPATH).ToArray()$
- exception: none

7.4.6 Local Constants

HEIGHT = 150 px
ROOT = Database root path
REQUESTPATH = path string for user friend requests list

8 MIS of Activity Detail View Module

8.1 Module

ActivityDetailViewModule

8.2 Uses

Database Module, Permission Module

8.3 Syntax

8.3.1 Exported Constants

None

8.3.2 Exported Access Programs

Name	In	Out	Exceptions
ViewActivities	-	-	-
AddActivity	Activity	-	InvalidPermission Exception
EditActivity	Activity, Activity	-	IndexOutofBound Exception, InvalidPermission Exception
DeleteActivity	Activity	-	IndexOutofBound Exception, InvalidPermission Exception
PinActivity	Activity	-	-
UnpinActivity	Activity	-	IndexOutofBound Exception

8.4 Semantics

8.4.1 State Variables

- activities: set of Activity
- pinnedActivities: set of Activity
- permission: Permission

8.4.2 Environment Variables

None

8.4.3 Assumptions

8.4.4 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: $exc := targetActivity \notin activities \Rightarrow IndexOutofBoundException$, $exc := \neg permission.isAdmin \Rightarrow InvalidPermissionException$

DeleteActivity(targetActivity):

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

PinActivity(newActivity):

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

UnpinActivity(targetActivity):

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

8.4.5 Local Functions

None

8.4.6 Local Constants

None

9 MIS of Lecture Detail View Module

9.1 Module

LectureDetailViewModule

9.2 Uses

Activity Detail View Module, Lecture Module

9.3 Syntax

9.3.1 Exported Constants

None

9.3.2 Exported Access Programs

Name	In	Out	Exceptions
ViewActivities	-	-	-
AddActivity	Lecture	-	InvalidPermission Exception
EditActivity	Lecture, Lecture	-	IndexOutofBound Exception, InvalidPermission Exception
DeleteActivity	Lecture	-	IndexOutofBound Exception, InvalidPermission Exception
PinActivity	Lecture	-	-
UnpinActivity	Lecture	-	IndexOutofBound Exception

9.4 Semantics

9.4.1 State Variables

- activities: set of Lecture
- pinnedActivities: set of Lecture
- permission: Permission

9.4.2 Environment Variables

None

9.4.3 Assumptions

9.4.4 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: $exc := targetActivity \notin activities \Rightarrow IndexOutofBoundException$,
  $exc := \neg permission.isAdmin \Rightarrow InvalidPermissionException$

DeleteActivity(targetActivity):

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

PinActivity(newActivity):

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

UnpinActivity(targetActivity):

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

9.4.5 Local Functions

None

9.4.6 Local Constants

None

10 MIS of Event Detail View Module

10.1 Module

EventDetailViewModule

10.2 Uses

Activity Detail View Module, Event Module

10.3 Syntax

10.3.1 Exported Constants

None

10.3.2 Exported Access Programs

Name	In	Out	Exceptions
ViewActivities	-	-	-
AddActivity	Event	-	InvalidPermission Exception
EditActivity	Event, Event	-	IndexOutofBound Exception, InvalidPermission Exception
DeleteActivity	Event	-	IndexOutofBound Exception, InvalidPermission Exception
PinActivity	Event	-	-
UnpinActivity	Event	-	IndexOutofBound Exception

10.4 Semantics

10.4.1 State Variables

- activities: set of Event
- pinnedActivities: set of Event
- permission: Permission

10.4.2 Environment Variables

None

10.4.3 Assumptions

10.4.4 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: $exc := targetActivity \notin activities \Rightarrow IndexOutofBoundException$,
  $exc := \neg permission.isAdmin \Rightarrow InvalidPermissionException$

DeleteActivity(targetActivity):

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

PinActivity(newActivity):

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

UnpinActivity(targetActivity):

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

10.4.5 Local Functions

None

10.4.6 Local Constants

None

11 MIS of Authentication Module

11.1 Module

Authentication

11.2 Uses

User Module, Database Module, Server Module

11.3 Syntax

11.3.1 Exported Constants

None

11.3.2 Exported Access Programs

11.4 Semantics

11.4.1 State Variables

- User: FirebaseUser

11.4.2 Environment Variables

None

11.4.3 Assumptions

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

11.4.4 Access Routine Semantics

Register():

- transition: $friends := GetFriendsFromDB(currentUser)$
- output: none
- exception: none

11.4.5 Local Functions

Login(_email, _password):

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

Register():

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

11.4.6 Local Constants

- auth: FirebaseAuth
- DatabaseReference: DatabaseReference

12 MIS of Notification Module

12.1 Module

Notification

12.2 Uses

12.3 Syntax

12.3.1 Exported Constants

None

12.3.2 Exported Access Programs

Name	In	Out	Exceptions
Consent	-	String	-
MapWarning	-	String	-
DataCollectionWarning	-	-	
NoInternetNotification	-	String	-
ARCameraNotification	-	String	-

12.4 Semantics

12.4.1 State Variables

None

12.4.2 Environment Variables

None

12.4.3 Assumptions

None

12.4.4 Access Routine Semantics

Consent():

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

MapWarning():

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

DataCollectionWarning():

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

NoInternetNotification():

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

ARCameraNotification():

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

12.4.5 Local Functions

None

12.4.6 Local Constants

USERCONSENT = Text of user consent when creating account
MAP = Text of warning message show when start the map
DATACOLLECTION = Text of warning before the application collects user data
NOINTERNET = Notification message when the internet is lost
ARCAMERA = Help message for AR camera functionality

13 MIS of Database Module

13.1 Module

FirebaseDatabase
This module uses Firebase Realtime Database library. For details of all syntax and semantics of exported constants and access programs, see Firebase database documentation. documentation

13.2 Uses

13.3 Syntax

13.3.1 Exported Constants

See Firebase database documentation.

13.3.2 Exported Access Programs

The following table will show some functions the application uses most frequently, for more details, see Firebase 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

13.4 Semantics

13.4.1 State Variables

None

13.4.2 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

13.4.3 Assumptions

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

13.4.4 Access Routine Semantics

Child(pathString):

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

RemoveValueAsync(pathString):

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

13.4.5 Local Functions

None

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.

14 Appendix

[Extra information if required —SS]