Module Interface Specification for Flick Picker

Team 7, 7eam
Talha Asif - asift
Jarrod Colwell - colwellj
Madhi Nagarajan - nagarajm
Andrew Carvalino - carvalia
Ali Tabar - sahraeia

February 8, 2023

1 Revision History

Date	**Version**	**Notes**
January 18	1.0	Added title, Module Decomposition table from Module Guide
January 18	2.0	Updated section 2, 3, 4

2 Symbols, Abbreviations and Acronyms

See SRS Documentation at https://github.com/Flick-Picker/full-stack/blob/develop/docs/SRS/SRS.pdf

Term	**Abbreviation**
True	T
False	F

Contents

3 Introduction

The following document details the Module Interface Specifications for Flick Picker. Flick Picker is a web application that lets people select their preferences for watchable media and find recommendations of things to watch, being able to take multiple sets of preferences into account to find new media. Users can find new things to watch by themselves using only their preferences, or join user groups that find media based on what everyone likes in the group.

Complementary documents include the System Requirement Specifications and Module Guide. The full documentation and implementation can be found at https://github.com/Flick-Picker/full-stack.

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 Flick Picker.

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	bool	either T or F

The specification of Flick Picker uses some common derived data types: sequences and strings. Sequences are lists filled with elements of the same data type. Strings are sequences of characters. Flick Picker also uses enumerated types, which are data types that hold a static set of constant values. In addition, Flick Picker 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.

Flick Picker also uses its own custom data types, summarized in the following table.

Data Type	**Notation**	**Description**
Preferences	Preferences	a class that stores multiple boolean and enumerated values to keep track of user preferences such as genre or runtime
User	User	a class that stores user information such as an ID, name, and Preferences
Group	Group	a class that stores information about a user group, including an ID, owner, and user list.
Authentication	Authentication	a class that stores information about a user's login, connecting a user ID to the appropriate password

5 Module Decomposition

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

Level 1	**Level 2**
Hardware-Hiding Module	
Behaviour-Hiding Module	Native Login Module Friends Module Groups Module Profile Module
Software Decision Module	Matching Algorithm Module OAuth Login Module API Module

Table 1: Module Hierarchy

6 MIS of M3

6.1 Module

Native Login Module

6.2 Uses

Types: User and Authentication

6.3 Syntax

6.3.1 Exported Constants

N/A

6.3.2 Exported Access Programs

Name	In	Out	Exceptions
getProfile()	String email, String password	-	Unregistered email, Incorrect password
signOut()	-	-	-

6.4 Semantics

6.4.1 State Variables

int profileId, String profileName, String profileEmail, String profilePassword

6.4.2 Environment Variables

N/A

6.4.3 Assumptions

User has a profile with the OAuth service they use to sign-in, with the provider of that service being able to handle invalid sign-in attempts.

6.4.4 Access Routine Semantics

getProfile():

- transition: User.id, User.name, User.email, Authentication.id, Authentication.password := profileId, profileName, profileEmail, profileId, profilePassword

- output: N/A
- exception: N/A

signOut():

- transition: Authentication := Null
- output: N/A
- exception: N/A

6.4.5 Local Functions

N/A

7 MIS of M4

7.1 Module

Friends Module

7.2 Uses

Types: User

7.3 Syntax

7.3.1 Exported Constants

N/A

7.3.2 Exported Access Programs

Name	In	Out	Exceptions
findFriend()	String searchName	List$\langle$User$\rangle$users	No user with that name
addFriend()	-	-	-
deleteFriend()	-	-	-

7.4 Semantics

7.4.1 State Variables

int friendId, String searchName

7.4.2 Environment Variables

N/A

7.4.3 Assumptions

The selected friend's account won't be deleted during the process of adding them

7.4.4 Access Routine Semantics

findFriend():

- transition: N/A
- output: List⟨User⟩→ User.name == searchName
- exception: No user with the entered name

addFriend():

- transition: User.friends.append(friendId)
- output: N/A
- exception: N/A

deleteFriend():

- transition: User.friends.remove(friend)
- output: N/A
- exception: N/A

7.4.5 Local Functions

N/A

8 MIS of M5

8.1 Module

Groups Module

8.2 Uses

Types: User, Group

8.3 Syntax

8.3.1 Exported Constants

N/A

8.3.2 Exported Access Programs

Name	**In**	**Out**	**Exceptions**
createGroup()	List⟨User⟩selectedUsers, String groupName	Group newGroup	-
deleteGroup()	Group selectedGroup	-	-
joinGroup()	-	Group newGroup	-
leaveGroup()	-	-	-
inviteToGroup()	User name	-	-

8.4 Semantics

8.4.1 State Variables

Group newGroup, int groupId, List⟨int⟩groupIds, Group invitedGroup

8.4.2 Environment Variables

N/A

8.4.3 Assumptions

N/A

8.4.4 Access Routine Semantics

createGroup():

- transition: groupIds := selectedUsers.id
- output: Group newGroup := groupId, User.id, groupIds
- exception: N/A

deleteGroup():

- transition: selectedGroup := Null
- output: N/A
- exception: N/A

joinGroup():

- transition: N/A
- output: newGroup
- exception: N/A

leaveGroup():

- transition: deletes user id from old group, but changes no state variable in the module
- output: N/A
- exception: N/A

inviteToGroup():

- transition: N/A
- output: sends group info (name, id, user list, etc.) to the selected user
- exception: N/A

8.4.5 Local Functions

N/A

9 MIS of M6

9.1 Module

Profile Module

9.2 Uses

Types: User, Authentication, Preferences
Modules: M4

9.3 Syntax

9.3.1 Exported Constants

N/A

9.3.2 Exported Access Programs

Name	In	Out	Exceptions
editName()	String newName	-	Invalid name (swearing, length, etc.)
editEmail()	String newEmail	-	Invalid email
editPassword()	String newPassword	-	User does not authenticate password change
editFriends()	-	User.friends	-
editPreferences()	Preferences newPreferences	newPreferences	-

9.4 Semantics

9.4.1 State Variables

User.name, User.email, Authentication.password, User.friends, User.preferences

9.4.2 Environment Variables

N/A

9.4.3 Assumptions

The user is the one making changes, and not some other party

9.4.4 Access Routine Semantics

editName():

- transition: User.name := newName
- output: N/A
- exception: N/A

editEmail():

- transition: User.email := newEmail
- output: N/A
- exception: N/A

editPassword():

- transition: Authentication.password := newPassword
- output: N/A
- exception: N/A

editFriends():

- transition: uses Friends Module
- output: User.friends
- exception: N/A

editPreferences():

- transition: User.preferences := newPreferences
- output: newPreferences
- exception: N/A

9.4.5 Local Functions

editPreferences() will rely on functions that display and allow the user to chose the values for different Preferences keys

10 MIS of M8

10.1 Module

Matching Algorithm Module

10.2 Uses

Types: Preferences

10.3 Syntax

10.3.1 Exported Constants

N/A

10.3.2 Exported Access Programs

Name	In	Out	Exceptions
recommendGroup()	Preferences group.preferences	List⟨String⟩shows	no matching results
recommendUser()	Preferences user.preferences	List⟨String⟩shows	no matching results

10.4 Semantics

10.4.1 State Variables

N/A

10.4.2 Environment Variables

N/A

10.4.3 Assumptions

N/A

10.4.4 Access Routine Semantics

recommendGroup():

- transition: N/A
- output: List⟨String⟩shows
- exception: N/A

recommendUser():

- transition: N/A
- output: List⟨String⟩shows
- exception: N/A

10.4.5 Local Functions

N/A

11 MIS of M9

11.1 Module

OAuth Login Module

11.2 Uses

Types: User and Authentication

11.3 Syntax

11.3.1 Exported Constants

N/A

11.3.2 Exported Access Programs

Name	In	Out	Exceptions
getProfile()	String email, password	-	-
signOut()	-	-	-

11.4 Semantics

11.4.1 State Variables

int profileId, String profileName, String profileEmail

11.4.2 Environment Variables

N/A

11.4.3 Assumptions

User has a profile with the OAuth service they use to sign-in, with the provider of that service being able to handle invalid sign-in attempts.

11.4.4 Access Routine Semantics

getProfile():

- transition: User.id, User.name, User.email := profileId, profileName, profileEmail
- output: N/A
- exception: N/A

signOut():

- transition: Authentication := Null
- output: N/A
- exception: N/A

11.4.5 Local Functions

N/A as they are implemented within the OAuth (Google, Meta, or Apple)

12 MIS of M10

12.1 Module

Matching Algorithm Module

12.2 Uses

Types: Preferences Modules: M8

12.3 Syntax

12.3.1 Exported Constants

N/A

12.3.2 Exported Access Programs

Name	**In**	**Out**	**Exceptions**
groupData()	M8.recommendGroup	List⟨String⟩media	no matching results
userData()	M8.recommendUser	List⟨String⟩media	no matching results

12.4 Semantics

12.4.1 State Variables

N/A

12.4.2 Environment Variables

N/A

12.4.3 Assumptions

N/A

12.4.4 Access Routine Semantics

groupData():

- transition: N/A
- output: List⟨String⟩media
- exception: N/A

userData():

- transition: N/A
- output: List⟨String⟩media
- exception: N/A

12.4.5 Local Functions

N/A

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.

13 Appendix

[Extra information if required —SS]