EECS3311-W – Project Report

|  |  |  |  |
| --- | --- | --- | --- |
| **Team members** | **Name** | **Prism Login** | **Signature** |
| Member 1: | Matthew MacEachern | mcmaceac |  |
| Submitted under Prism account: | | mcmaceac |  |

Contents

[1. Requirements for the Messenger Project 2](#_Toc478387569)

[2. BON class diagram overview (architecture of the design) 4](#_Toc478387570)

[3. Table of modules – responsibilities and information hiding 5](#_Toc478387571)

[4. Expanded description of design decisions 6](#_Toc478387572)

[5. Significant Contracts (Correctness) 7](#_Toc478387573)

[6. Summary of Testing Procedures 8](#_Toc478387574)

[7. Appendix (Contract view of all classes) 9](#_Toc478387575)

# Requirements for the Messenger Project

As specified in the introduction portion of the project specification file, the requirements for this project is to develop a messaging system for use in hospitals. In this messaging application, different employees at the hospital will use it for secure communication. Users can register to groups and can send messages to groups, which will be broadcast to all the users that are registered to that group. Privacy is an essential part of this project since it is important that only members of a group can read messages sent to that group, and only members can send messages to that group.

Users can be a part of 0 to many groups and groups can have 0 to many users. Messages sent to these groups must be sent out to the appropriate users that are registered for that group. A message can be either “read”, “unread”, or “unavailable” to a certain user. A message has a sender, group and content stored in it.

See *messenger.definitions.txt* for the grammar of the user interface. The acceptance tests in the student and instructor acceptance test folders show the behavior of the application at the console level.

# BON class diagram overview (architecture of the design)

# Table of modules – responsibilities and information hiding

# Expanded description of design decisions

The requirements for this project have been defined through the ETF definitions file and through the oracle program that was modified throughout the duration of development. These requirements made regression testing for the project straight forward and ensured that these requirements were completed in full.

The first and most obvious requirement for the project was to ensure that all the commands specified in the definitions worked the same as the oracle in all my tests using regression testing. The add\_group and add\_user commands are similar in terms of implementation and both add an entry into the messenger users and groups lists respectively. These two lists are lists of GROUP and USER objects. These two lists are stored in the MESSENGER class to ensure that there is a centralized class that contains important information about all the current groups and users in the system. Error messages for these two routines are used to direct the user on how to use the commands.

The next command was register\_user and was responsible for making sure a user that is currently in the system can register to a group currently in the system. To establish the link between users and groups, the USER objects have a list of the groups that the user is currently a part of. Also, each GROUP object has a list of the subscribers to the current group object. This facilitates message passing and reading. When a user is registered to a group, that group is added to the user’s group list, and the user is added to that group’s subscriber list. This change is reflected in the registrations portion of the output. This command also displays appropriate error messages to the user to ensure that they understand usage.

The send\_message command uses the above design of USER and GROUP to pass messages to the appropriate group, from the user, to ensure that the appropriate users are notified of the message. When a user sends a message, the messenger routine send\_message is called, which finds the group that the message is being sent to, and calls that groups broadcast routine. The broadcast routine essentially sends the message to all the subscribers to the group. The message which has its own class MESSAGE is then added to the messenger’s all\_messages list to keep a record of all messages sent through the messenger.

The read\_message finds the user specified and the message specified and marks that message for that message as read if it is available to the user and the message is not already read. This is reflected in the message state section of the output. Each USER object has a list of messages and a hash table that keeps track of the message status (true if read, false if not read).

After a user has read the message and under message state it is shown that the user has read the message, a user can use the delete\_message command to remove it from their personal messages (keep in mind that the message still exists in the messenger system it is just unavailable to that user now).

The set\_message\_preview command simply changes the amount of text that is shown in the all messages section of the output. This is simply stored as a variable message\_length in the MESSENGER class which is referred to in the out command of the class.

The list\_new\_messages and list\_old\_messages are similar in my design as the new and old messages are queries to the messages list which checks the message\_status hash table to see if they are read (old) or unread (new).

Finally, the list\_groups and list\_users lists the groups and users respectively in alphabetical order of their names, which is different from the way the users and groups are listed in the users and groups section of the default output, where they are sorted by their ids. To do this while using as little memory as possible I kept the users and groups both in only one list each, but kept a Boolean variable sort\_by\_id to check if I want to sort by id or by name.

As an additional requirement, the ids of the users, groups and messages, needed to be able to be in the range of 1 to 9,223,372,036,854,775,807, which is the range of a 64-bit integer in Eiffel.

My program had to match the oracle character for character, so I needed to refer to the errors.txt file provided so I could ensure that my error messages matched those that were provided initially.

# Significant Contracts (Correctness)

The module that has the most significant contracts in my design is the MESSENGER class. This class handles the commands through contracts (that are first checked using defensive programming in each of the command classes).

# Summary of Testing Procedures

|  |  |  |
| --- | --- | --- |
| **Test file** | **Description** | **Passed** |
| at1.txt | This is a general scenario where the application is tested using a variety of commands using accepted input and input where errors are produced. | ✔ |
| at2.txt | This test file specifically tests the ordering of users and groups on default output as well as registrations and messages. | ✔ |
| at3.txt | This test file tests the availability of deleted messages and the output of list\_old\_messages and read\_message once the message is deleted. | ✔ |
| at4.txt | This test file tests the output of registering users in groups after messages have been sent to certain groups. | ✔ |
| at5.txt | This test file attempts to add a group and a user both with large numbered ids. | ✔ |
| at6.txt | This is another general test file that tries to find edge cases in the program. | ✔ |

# Appendix (Contract view of all classes)

ETF\_MODEL

-- Automatic generation produced by ISE Eiffel --  
**note**  
 description: "A default business model."  
 author: "Matthew MacEachern"  
 date: "$Date$"  
 revision: "$Revision$"  
  
**class** **interface**  
 ETF\_MODEL  
  
**create** {ETF\_MODEL\_ACCESS}  
 make

**feature** -- model attributes  
 i: INTEGER\_32  
 m: MESSENGER  
 e: STRING\_8  
 -- error message  
 message: STRING\_8  
 -- warning message  
**feature** -- model operations  
 default\_update  
 -- Perform update to the model state.  
 reset  
 -- Reset model state.  
**feature** -- queries  
 out: STRING\_8  
 -- New string containing terse printable representation  
 -- of current object  
**end** -- class ETF\_MODEL  
 -- Generated by ISE Eiffel --  
 -- For more details: http://www.eiffel.com --

USER

-- Automatic generation produced by ISE Eiffel --  
**note**  
 description: "Summary description for {USER}."  
 author: ""  
 date: "$Date$"  
 revision: "$Revision$"  
  
**class** **interface**  
 USER  
  
**create**   
 make  
  
**feature** --creation  
 make (l\_name: STRING\_8; l\_id: INTEGER\_64; l\_m: MESSENGER)  
   
**feature** --attributes  
 name: STRING\_8  
 id: INTEGER\_64  
 messenger: MESSENGER  
 groups: SORTED\_TWO\_WAY\_LIST [GROUP]  
 messages: SORTED\_TWO\_WAY\_LIST [MESSAGE]  
 deleted\_messages: SORTED\_TWO\_WAY\_LIST [MESSAGE]  
 message\_status: HASH\_TABLE [BOOLEAN, INTEGER\_64]  
 --message id to read status for this user  
 old\_messages: SORTED\_TWO\_WAY\_LIST [MESSAGE]  
 new\_messages: SORTED\_TWO\_WAY\_LIST [MESSAGE]  
 registered: BOOLEAN  
   
**feature** --commands  
 add\_message (m: MESSAGE)  
 delete\_message (mid: INTEGER\_64)  
 read\_message (mid: INTEGER\_64)  
 register (g: GROUP)  
 --register this user to group g  
   
**feature** --queries  
 authorized\_to\_access\_message (mid: INTEGER\_64): BOOLEAN  
 has\_message (mid: INTEGER\_64): BOOLEAN  
 --checks if this user has a message with mid  
 no\_new\_message: BOOLEAN  
 no\_old\_message: BOOLEAN  
 out: STRING\_8  
 -- New string containing terse printable representation  
 -- of current object  
  
 list\_new\_messages: STRING\_8  
 list\_old\_messages: STRING\_8  
 list\_registrations: STRING\_8  
 number\_of\_groups: INTEGER\_64  
 --number of groups this user is a part of  
 member\_of (gid: INTEGER\_64): BOOLEAN  
 --is this user a member of group with id gid?  
 message\_id\_exists (mid: INTEGER\_64): BOOLEAN  
 message\_deleted (mid: INTEGER\_64): BOOLEAN  
   
**feature** --comparable  
 is\_less **alias** "<" (other: **like** **Current**): BOOLEAN  
 -- Is current object less than `other'?  
   
**end** -- class USER  
 -- Generated by ISE Eiffel --  
 -- For more details: http://www.eiffel.com --

GROUP

-- Automatic generation produced by ISE Eiffel --  
**note**  
 description: "Summary description for {GROUP}."  
 author: ""  
 date: "$Date$"  
 revision: "$Revision$"  
  
**class** **interface**  
 GROUP  
  
**create**   
 make  
  
**feature** --creation  
 make (l\_name: STRING\_8; l\_id: INTEGER\_64; l\_m: MESSENGER)  
   
**feature** --attributes  
 name: STRING\_8  
 id: INTEGER\_64  
 messenger: MESSENGER  
 subscribers: LINKED\_LIST [USER]  
   
**feature** --queries  
 out: STRING\_8  
 -- New string containing terse printable representation  
 -- of current object  
   
**feature** --commands  
 subscribe (u: USER)  
 --subscribe user u to this group  
 broadcast\_message (m: MESSAGE)  
 --broadcast a message sent to this group to all of the subscribers   
   
**feature** --comparable  
 is\_less **alias** "<" (other: **like** **Current**): BOOLEAN  
 -- Is current object less than `other'?  
   
**end** -- class GROUP  
 -- Generated by ISE Eiffel --  
 -- For more details: http://www.eiffel.com --

MESSENGER

-- Automatic generation produced by ISE Eiffel --  
**note**  
 description: "Summary description for {MESSENGER}."  
 author: ""  
 date: "$Date$"  
 revision: "$Revision$"  
  
**class** **interface**  
 MESSENGER  
  
**create**   
 make  
  
**feature** -- attributes  
 message\_to\_read: STRING\_8  
 message\_number: INTEGER\_64  
 message\_length: INTEGER\_64  
 sort\_by\_id: BOOLEAN  
 num\_users: INTEGER\_64  
 num\_groups: INTEGER\_64  
   
**feature** -- creation  
 make  
   
**feature** -- commands  
 add\_user (id: INTEGER\_64; name: STRING\_8)  
 **require**  
 id\_positive: id > 0  
 first\_letter\_alpha: **not** (name**.**count = 0) **and** name**.**at (1)**.**is\_alpha  
 id\_not\_in\_use: **not** user\_id\_exists (id)  
  
 add\_group (id: INTEGER\_64; name: STRING\_8)  
 **require**  
 id\_positive: id > 0  
 first\_letter\_alpha: name**.**count > 0 **and** name**.**at (1)**.**is\_alpha  
 id\_not\_in\_use: **not** group\_id\_exists (id)  
  
 delete\_message (uid: INTEGER\_64; mid: INTEGER\_64)  
 **require**  
 uid > 0 **and** mid > 0  
 user\_id\_exists (uid)  
 message\_id\_exists (uid, mid)  
 old\_message\_exists (uid, mid)  
  
 register\_user (uid: INTEGER\_64; gid: INTEGER\_64)  
 **require**  
 uid > 0 **and** gid > 0  
 user\_id\_exists (uid)  
 group\_id\_exists (gid)  
 **not** registration\_exists (uid, gid)  
  
 read\_message (uid: INTEGER\_64; mid: INTEGER\_64)  
 **require**  
 uid > 0 **and** mid > 0  
 user\_id\_exists (uid)  
 message\_id\_exists (uid, mid)  
 user\_authorized\_to\_access\_message (uid, mid)  
 **not** message\_read (uid, mid)  
  
 send\_message (uid: INTEGER\_64; gid: INTEGER\_64; txt: STRING\_8)  
 **require**  
 uid > 0 **and** gid > 0  
 user\_id\_exists (uid)  
 group\_id\_exists (gid)  
 no\_empty\_message: **not** txt**.**is\_empty  
 authorization: registration\_exists (uid, gid)  
  
 set\_message\_preview (length: INTEGER\_64)  
 --sets the preview length of all messages in the messenger to length  
 **require**  
 length > 0  
 **ensure**  
 message\_length = length  
   
**feature** -- queries  
 user\_authorized\_to\_access\_message (uid: INTEGER\_64; mid: INTEGER\_64): BOOLEAN  
 user\_id\_exists (id: INTEGER\_64): BOOLEAN  
 user\_no\_new\_message (id: INTEGER\_64): BOOLEAN  
 user\_no\_old\_message (id: INTEGER\_64): BOOLEAN  
 group\_id\_exists (id: INTEGER\_64): BOOLEAN  
 message\_id\_exists (uid: INTEGER\_64; mid: INTEGER\_64): BOOLEAN  
 --check to see if this message id exists for this user  
 list\_all\_messages: STRING\_8  
 --lists all of the messages sent  
 list\_new\_messages (uid: INTEGER\_64): STRING\_8  
 **require**  
 uid > 0  
 user\_id\_exists (uid)  
 **not** user\_no\_new\_message (uid)  
 list\_old\_messages (uid: INTEGER\_64): STRING\_8  
 **require**  
 uid > 0  
 user\_id\_exists (uid)  
 **not** user\_no\_old\_message (uid)  
 message\_status: STRING\_8  
 --lists the message status for each message and each user  
 message\_deleted (uid: INTEGER\_64; mid: INTEGER\_64): BOOLEAN  
 user\_member\_of\_group (uid: INTEGER\_64; mid: INTEGER\_64): BOOLEAN  
 message\_read (uid: INTEGER\_64; mid: INTEGER\_64): BOOLEAN  
 list\_users\_by\_id: STRING\_8  
 --lists the users in order of their id  
 list\_users: STRING\_8  
 --lists the users in order of their name  
 list\_groups\_by\_id: STRING\_8  
 --lists the groups in order of their id  
 list\_groups: STRING\_8  
 --lists the users in order of their name  
 **require**  
 num\_groups > 0  
  
 list\_registrations: STRING\_8  
 old\_message\_exists (uid: INTEGER\_64; mid: INTEGER\_64): BOOLEAN  
 registration\_exists (uid: INTEGER\_64; gid: INTEGER\_64): BOOLEAN  
   
**end** -- class MESSENGER  
 -- Generated by ISE Eiffel --  
 -- For more details: http://www.eiffel.com --

MESSAGE

-- Automatic generation produced by ISE Eiffel --  
**note**  
 description: "Summary description for {GROUP}."  
 author: ""  
 date: "$Date$"  
 revision: "$Revision$"  
  
**class** **interface**  
 GROUP  
  
**create**   
 make  
  
**feature** --creation  
 make (l\_name: STRING\_8; l\_id: INTEGER\_64; l\_m: MESSENGER)  
   
**feature** --attributes  
 name: STRING\_8  
 id: INTEGER\_64  
 messenger: MESSENGER  
 subscribers: LINKED\_LIST [USER]  
   
**feature** --queries  
 out: STRING\_8  
 -- New string containing terse printable representation  
 -- of current object  
   
**feature** --commands  
 subscribe (u: USER)  
 --subscribe user u to this group  
 broadcast\_message (m: MESSAGE)  
 --broadcast a message sent to this group to all of the subscribers   
   
**feature** --comparable  
 is\_less **alias** "<" (other: **like** **Current**): BOOLEAN  
 -- Is current object less than `other'?  
   
**end** -- class GROUP  
 -- Generated by ISE Eiffel --  
 -- For more details: http://www.eiffel.com --