EECS3311-W17 – Messenger Project

26 September 2013. *Google Gchat Privacy Error: Hangouts Sent Private Messages To Wrong People.* If you're using Google's Gchat today, chat carefully. Google Inc. is still investigating an apparent Gchat glitch that alarmed Google Hangout users in the wee hours of the night. Late Wednesday, users began reporting that the instant-message service was malfunctioning by sending chat messages to the wrong people. Private messages meant for certain recipients went to someone else or, in some cases, multiple people. Some Gchat users, mystified by the mix-up, posted about the security glitch on Twitter. … It's unclear what caused the malfunction, which is particularly troubling in light of ongoing privacy concerns regarding Google products and other cloud-based services. Some Twitter users joked about the glitch in the context of the NSA surveillance scandal.[1]

Not for distribution. By retrieving this Project and using this document you affirm that you are registered in EECS3311 at York University this term. Registered EECS3311 students may download this Prokect for their private use, but may not communicate it to anyone else, even other students in the class. Each student (or team) must submit their own work thus complying with York academic integrity guidelines. See course wiki for details of academic integrity principles.

Table of Contents

[1] http://www.ibtimes.com/gchat-privacy-error-google-inc-goog-hangouts-sent-private-messages-wrong-people-1411440

1 Introduction

Management has decided to develop a messaging system for use in hospitals. Physicians, nurses, and administrators shall be able to securely use it for communication. Users can be subscribed to groups and can send a message to the group. It is essential for privacy concerns that only those users registered in a group may read messages directed at the group.

This is mission critical application

- in the sense that you must design the software system to guarantee patient privacy;
- You must also implement and document your design to show that your design is correct and satisfies the specified safety behavior;
- You validate the safety of your system using contracts, unit tests and acceptance tests.

Please ensure that you follow all submission instructions carefully. Submissions that do not comply with the instructions will not receive a grade.

2 Requirements

Requirements elicitation has resulted in an abstract grammar for the user interface.[2] Eventually, there will be a desktop application, a webapp and mobile app for users (physicians, nurses, administrators and patients). However, the concrete GUIs will be developed at a later stage and are beyond the scope of this project. The concrete GUIs will support all the operations in the abstract grammar.

Also, one Use Case has been developed so far. This Use Case has been refined into an Acceptance test called at1.txt.[3]

When an operation can be performed, the system responds with "ok". However, when some operation cannot be performed, the acceptance test currently responds with a meaningful error message.[4] You are required to perform further requirements elicitation to obtain the precise error messages.

Instead of a complete specification, you are also provided with an Oracle.[5] You will thus need to develop your own additional acceptance test at2.txt, at3.txt etc. It is interesting to note that these acceptance test (being structured text files) are independent of the programming language used to develop the system. These acceptance tests can thus be written by users who are not familiar with programming. Such acceptance tests can thus be developed before the system is ever implemented. A good set of acceptance tests is thus part of the eHealth specification.

2.1 Additional specifications

You must develop a program that can be tested from the console. For example, executing

```
messenger.exe -b at1.txt
```

[2] In the course directory, this is: /cs/course/3311/labs/project/messenger.definitions.txt.

[3] Also in the above course directory.

[4] The error messages are described in /cs/course/3311/labs/project/errors.txt. Note that where there is difference between this file and the oracle, the oracle governs.

[5] /cs/course/3311/labs/project/messenger.exe

shall produce the output *at1.expected.txt* (see course directory), with the proper agreed upon error messages. The “-b” switch stands for batch mode. Both relative and absolute paths using Linux conventions shall be supported.

Integer identifiers (e.g. for a use id) are in the range 1 to 9,223,372,036,854,775,807.[6] This is the maximum number of identifiers that we need to support.

In all cases the behavior of the Oracle is definitive. Your program must match the Oracle character-by-character.

3 Directory structure of your project

Your project shall reside in a directory **messenger-project** with the following structure:

```
messenger-project/
├── docs
│   ├── messenger.pdf
│   └── team.txt
├── messenger.ecf
├── root
│   └── root.e
└── tests
    ├── acceptance
    │   ├── instructor
    │   │   └── at1.txt
    │   └── student
    │       ├── at1.txt
    │       ├── at2.txt
    │       ├── at3.txt
    │       └── at4.txt
    └── unit
        ├── instructor
        └── student
```

You must submit a **superset** of the above directory structure (i.e. you may add additional folders and files) so that we can compile your project. Note that you will be submitting additional directories such as *messenger* (with your model, etc.), *generated_code*, etc.

In the *docs* folder you must submit

- *messenger.pdf*: this is the report of your team, professionally prepared
- *team.txt*: is the Prism logins of the members of your team

The *team.txt* file looks like this:

```
cse99783
cse67999
```

The Prism login on the first line of the *team.txt* file shall also the login for the team submission. If team.txt is not precisely as specified, you will not receive a grade. You may

[6] 64 bit integers are in the range −9,223,372,036,854,775,808 to +9,223,372,036,854,775,807, or from −(2^63) to 2^63 − 1

work on your own (in which case there is only one login) or in a team of at most two members.

You shall write at least four acceptance tests of your own, which appear in the *student* directory. (The test in the *instructor* directory is the one that is given to you by the instructor). It is advisable that you also write unit tests to test your code, but no lower limit is supplied.

You shall ensure that there are no EIFGENs directories in your submission, i.e. you must eclean before you submit.

4 How to submit your report and project

There is

- (a) an electronic submission
- and (b) you must also submit your report *messenger.pdf* in the course Dropbox (via Moodle) by the due date.

Electronic submission:

```
submit  -l  3311  project  messenger-project
```

5 The structure of your report

Your report must follow the 3311 SDD structure template documented at https://wiki.eecs.yorku.ca/project/sel-students/p:tutorials:sdd:start

Please read all the details at the above URL carefully.

6 Modes of failure

We grade your project by

- Testing your code submission for correctness
- Evaluating your design in the report that you provide

We test your project for correctness by running our own suite of acceptance tests. The behavior of your system must be **character-by-character** the same as the Oracle specification. The following problems are serious and will impact on your ability to obtain a passing grade:

- An exception is generated when running one of our acceptance tests
- Your system does not terminate when running one of our acceptance tests
- Your system does not provide the correct status messages when running one of our acceptance tests
- The messages are correct but the business logic is incorrect (i.e. the output is wrong either in a detail or in the macro sense)

Given that this is a mission critical application, you must ensure the correctness of your system and demonstrate that you have taken sufficient precautions to ensure its correctness.