Second Iteration Final Report:

mycheapfriend.com

COMS 4156: Advanced Software Engineering

Team: CheapSkates

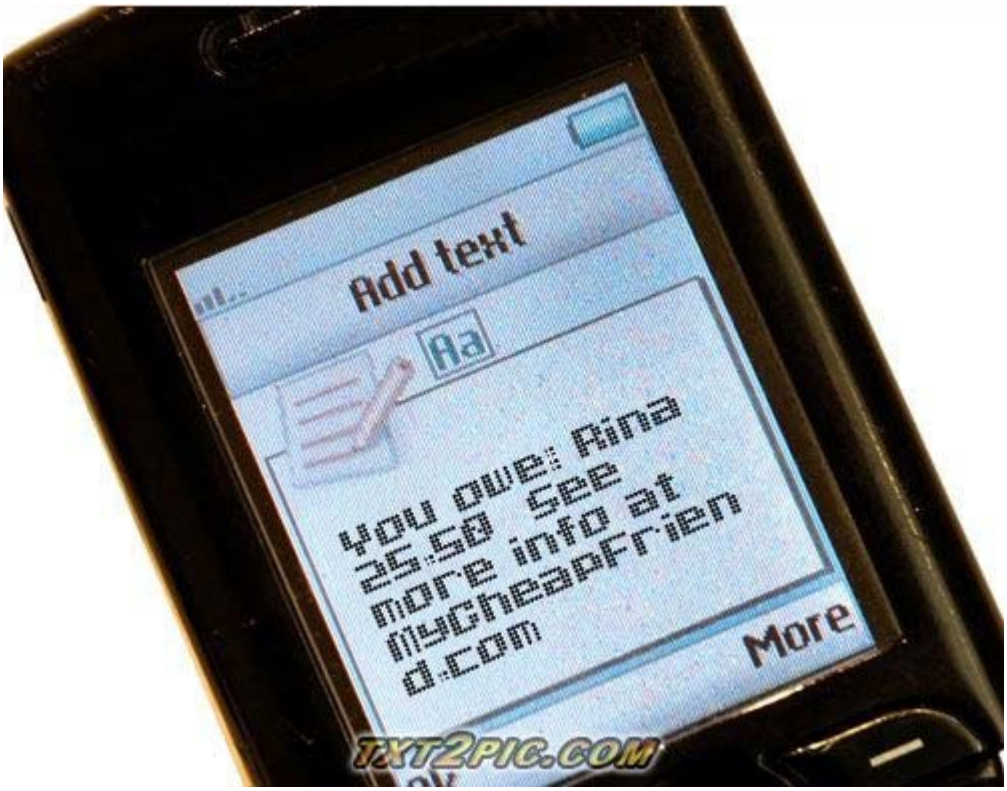
Team Members

Double Trouble: Waseem Ilahi (wki2001@columbia.edu)

Shaoqing Niu (sn2385@columbia.edu)

Dynamic Duo: Michael Glass (mgg2102@columbia.edu)

Huning "David" Dai (hd2210@columbia.edu)



The **Cheapskates** from **MyCheapFriend.com** will lend you our **Sir,** if you give us back feedback soon.

Table of Content

2. Unit Testing and Code Inspection Disposition ……. 2

3. Security and Stress Testing …….....……..... 3

4. Post Progress Report Defect Log and Disposition …… 6

5. Response to Demo Concerns ……. 6

6. Controversies …….…… 7

7. Revised Schedule (Plan Attached At the End of the Document).....7

2. Unit Testing and Code Inspection disposition

Non-logic Coding Problems:

Controller.java

(MyCheapFriend\MyCheapFriend-ejb\src\java\mycheapfriend\Controller.java)

The following defects were found in the code during the code inspection. The file name is given and the line numbers represent where the defect was in that file. The modified file has different line numbers for similar code, for there have been changes to fix the defects. So the reference is the original file, not the current file that is in the system. The original file has been attached with the submission folder with the name "code_inspection_unit_original.java".

- **Line 155, 157, 173, 184, 382, 404, 414, 490, 510, and 515:** Inconsistency in the use of true and false. Some places we use Boolean.TRUE/FALSE and others we use true/false.
  - Fixed in revision 219, committed 12/02/09 by waseemilahi
  - Everything changed to true/false. No more Bollean.TRUE/FALSE.
- **Line 73:** Very large logic expression. May be confusing to the readers.
  - Fixed in revision 219, committed by waseemilahi
  - A method was created that broke the expression into multiple if's and returned a Boolean value that was assigned to the variable. The method definition is at the line 199.
- **Line 75, 78, 83, 89, 92, 403, and 436:** Comments were placed to the right of the statements rather than above, like in other cases ion the source.
  - Fixed in revision 219, committed by waseemilahi
  - The entire inconsistent comment placement was removed and the comments were moved above each statement they document.
- **Line 612:** Left over TODO comment from earlier development.
  - Fixed in revision 219, committed by waseemilahi
  - The comment was removed.
- **Line 62 – 145:** The main method is a bunch of if – else and switch statements. TA suggested encapsulating this further by dividing those functionalities (Switch's) in to separate function calls, to avoid confusion.
  - Fixed in revision 219, committed by waseemilahi
  - Multiple methods were created, each handling a subset of features. The handle method calls these methods in succession. And the appropriate method handles the task, if it can't/doesn't the handle moves on to the next call. Thus, the control flows through the code.

Logic Errors (bugs)

EmailInfo.java

(MyCheapFriend\MyCheapFriend-ejb\src\java\mycheapfriend\EmailInfo.java)

- Error type on success wasn't always set to "no error". Fixed on revision 237.
- Didn't catch invalid boundary case "me 6462294050 bob" Fixed on rev 237.

All the problems found in code inspection and the unit testing, were fixed before the submission of second iteration progress report. After the report, no new errors were found in these two cases.

3. Security and Stress testing:

Our user interface is relatively simple, so testing each aspect of it doesn't need to be too complicated. We will split the attack into attacks on each of the interfaces.

The Web UI:

The web UI is an administrator interface, where administrators should have "root" access once they are authenticated. Here is our attack plan for the relatively simple Web UI.

1. Going through the security

   Try the overflowing the fields in the web form.

The attack: We sent requests with really looooooong username and password (1000/10000/100000 characters), and almost broke Internet Explorer. However, since we used try-catch to validate the inputs get posted, our system didn't crash.

2. Going around the security

   Try accessing administrator URLs without logging in.

The attack: We tried to get into the administrator interface by accessing its URL and skipping the login authentication. We put "localhost:8080/MyCheapfriend-war/Administrator" in the address field and succeeded to gain "root" privilege without providing proper login information. This defect is named as "**Vulnerability 1**".

3. Accessibility attack.

   If we break into the system, the root user's index page is heavy weight, as it lists all database elements. Repeatedly requesting this page (assuming A) the

database is thoroughly populated and B) we have access to the page) would be a way of attacking the system's accessibility. That being said, once a root user has been compromised, they could simply turn off the system, so it's probably not worth exploring this attack.

The attack: Since we could easily go around the authentication, we were able to stop the service in the roles of malicious users. This reveals the same vulnerability as Vulnerability 1.

The Text UI:

The Text UI receives emails from text messages and parses them into logical system objects. Given the:

1. Going through the security
   Try impersonating phone numbers with phone-number like email addresses from other hosts (impersonating SMS). ie 6462294050@gmail.com, or sending an email from our own smtp server impersonating vtext.com or att.com.

The attack: We tried to send emails from 1234567890@gmail.com (NOT from a cell phone) and succeeded to register as a legitimate user. This is name as "**Vulnerability 2**".

2. Accessibility attack:
   Sending emails with large attachments, or very large message bodies might slow down our e-mail fetcher's speed of retrieving new messages.

The attack: We sent text messages and emails (Since Vulnerability 2 is not fixed) to our server and our system slowed down tremendously. It took our server five minutes on average processing a message/email with a 5 MB attachment. This is recorded as "**Vulnerability 3**"

Stress Testing:

1. Massive users

The capacity of users in our system is actually limited by external resources other than our software system. We tried to add ten thousand users and the system was still functioning properly, however, we didn't check the upper bound.

2. Massive messages

We first sent 100/1000/10000 messages to the SMTP server and let our system to process all the messages simultaneously. The server was able to process around 30 messages in every five-second interval. Though it takes a while for our system to finish all the tasks, the system didn't crash or misbehave. As we are now only using one machine as the server, the efficiency and bandwidth are limited. However, if we have multiple servers process the messages in a distributed way, the performance will not be an issue. Assume that if we have more users, we can have more money to pursuit more servers. (We don't have a business plan to make profits yet.)

SQL injection:

1. Via user I/O interface

We composed SQL statements as user inputs and send them to our system's SMTP server. These SQL statements didn't satisfy the syntax rules of our parser and therefore were rejected.

2. Via the administrator interface

As the simplicity of our administrator interface, there are only three places possible for an attacker to inject SQL statements. The input boxes of user name and password for log in and the URL address bar. We put limitation on the digits of both the user name and password therefore blocking any possible SQL injections. As for the URL address bar, we tested it with several common SQL statements and our system rejected those requests.

4. Post Progress Report Defect log and disposition

Vulnerability 1:

We used sessions to fix this vulnerability. Corresponding code are added in Administrator,java, ListUser.java, LoginHandler.java, LoginUser.java, StartService.java, StopService.java, disable.java and enable.java. This is fixed on Rev 209.

Vulnerability 2:

In order to prohibit users registering account using emails rather than messages from cell phone, we check the domain part of the from address of each message we receive against a list of known providers. If the domain is not in the list, our system will discard the related request. Code added in EmailInfo.java. This is fixed on Rev 258.

Vulnerability 3:

As we are using Gmail as our SMTP server, we set up a filter which will discard all messages with attachments in the Gmail setting of our system's account. Moreover, since we are expecting short messages from cell phones and there is a limitation on the number of characters one can put in a short message (i.e. which is 160 if using AT&T), we don't have to worry about messages with large bodies.

5. Response to Demo Concerns

Regarding the demo, the whole class felt quite interested in our application. Professor Kaiser asked several questions about the mechanisms we used in the system as our TA Jon tried to break into our system although he failed to make it. There was no problem with our system but there were indeed two concerns. First, Professor Kaiser was confused about how a user might deal with multiple bills. So our way is just to ignore previous bills if the user receives a new one. In this case, the user won't feel confused about which bill he/she is accepting. Also, he/she doesn't need to remember the sequence in which he receives the bills. Second, Professor Kaiser asked about the scalability problem. As to this problem, the result of stress testing shows that our system can deal with messages promptly if they are not sent in a high frequency. However, if we want to deal with a large number of bills in a short period of time, we definitely need more servers. Particularly, we need to use multiple Gmail accounts to receive and send emails since Gmail policy limits the number of emails

that could be received and sent within certain minutes. We are very grateful for the advices and comments that Professor Kaiser made during the demo process.

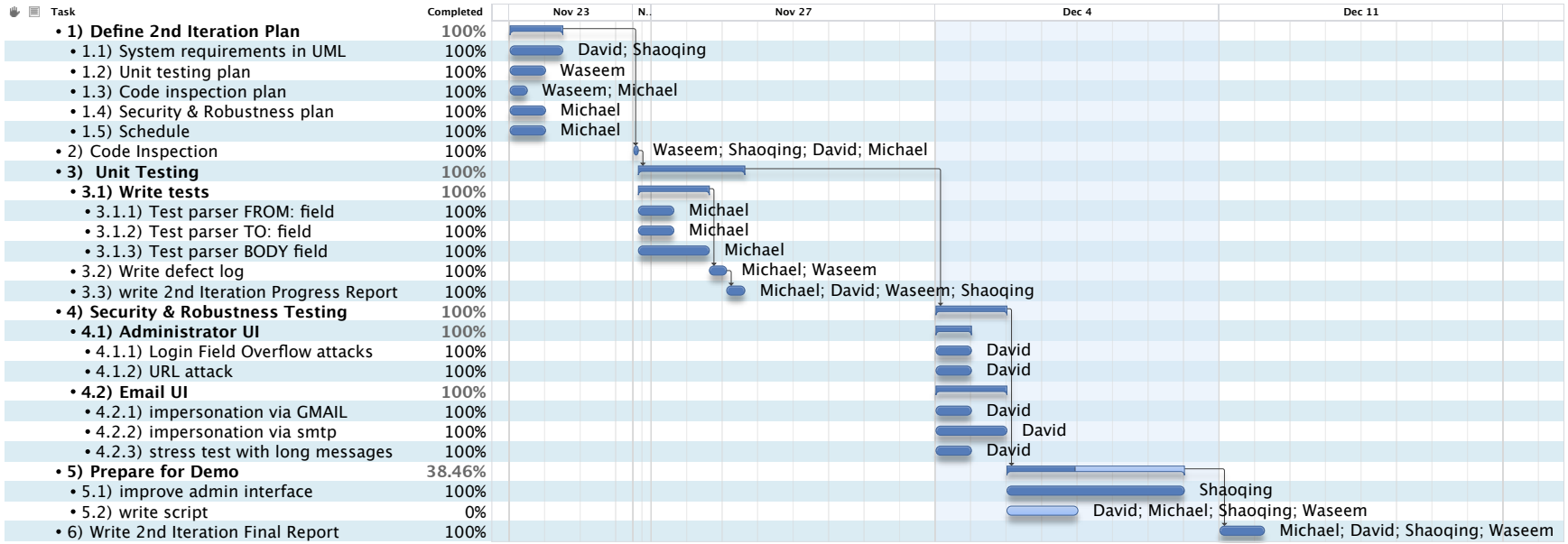
6. Controversies

There are no controversies among the team at this point.

7. Schedule

Due to the previous concern about team assignments, from the TA, we have attached the revised schedule at the end of this document.

NOTE: All changes and fixes can be browsed following this link to our code repository: http://code.google.com/p/mycheapfriend/source/list


Task
Completed
Nov 23
N..
Nov 27
Dec 4
Dec 11
• 1) Define 2nd Iteration Plan 100%
• 1.1) System requirements in UML 100% David; Shaoqing
• 1.2) Unit testing plan 100% Waseem
• 1.3) Code inspection plan 100% Waseem; Michael
• 1.4) Security & Robustness plan 100% Michael
• 1.5) Schedule 100% Michael
• 2) Code Inspection 100% Waseem; Shaoqing; David; Michael
• 3) Unit Testing 100%
• 3.1) Write tests 100%
• 3.1.1) Test parser FROM: field 100% Michael
• 3.1.2) Test parser TO: field 100% Michael
• 3.1.3) Test parser BODY field 100% Michael
• 3.2) Write defect log 100% Michael; Waseem
• 3.3) write 2nd Iteration Progress Report 100% Michael; David; Waseem; Shaoqing
• 4) Security & Robustness Testing 100%
• 4.1) Administrator UI 100%
• 4.1.1) Login Field Overflow attacks 100% David
• 4.1.2) URL attack 100% David
• 4.2) Email UI 100%
• 4.2.1) impersonation via GMAIL 100% David
• 4.2.2) impersonation via smtp 100% David
• 4.2.3) stress test with long messages 100% David
• 5) Prepare for Demo 38.46%
• 5.1) improve admin interface 100% Shaoqing
• 5.2) write script 0% David; Michael; Shaoqing; Waseem
• 6) Write 2nd Iteration Final Report 100% Michael; David; Shaoqing; Waseem