Lab08 - Exploratory, Manual and Scripted Testing (SENG 275)

Due date: Sunday July 02, at 11:59 pm.

INTRODUCTION

This lab has four main parts:

1. **Familiarization** with the system under test. This is done individually.
2. **Exploratory** (manual non-scripted) testing (you can explore the system in any manner you choose). This is done individually.
3. **Manual scripted testing**, using a predefined test suite to test the SUT. This can be done individually or in a pair.
4. **Regression testing** to re-test an updated version of the system after it has been changed (by an imaginary developer in response to a list of defect reports) This is done individually or in a pair.

SYSTEM UNDER TEST

The system under test for this lab is an ATM simulation system. Your repo contains a zip file, which in turn contains two .jar files. These jar files are two versions of the ATM system, which represent two consecutive releases of the software with bugs and bug fixes:

- ATM System - Lab 1 Version 1.0
- ATM System - Lab 1 Version 1.1

Usage of the system

To use the ATM simulation system, run the JAR file *ATM System – Lab 1 Version 1.0.jar*. This can be run with the -jar switch to java:

```
java -jar "ATM System - Lab 1 Version 1.0.jar"
```

The system should begin execution with the GUI as shown in Appendix A.

There are two valid hard-coded card numbers and PINs:

Card Number: 1	PIN: 42	Available Accounts: Checking and Savings
Card Number: 2	PIN: 1234	Available Accounts: Checking and Money Market

Note: Both of these cards access the same checking account.

The initial balances are:

Checking: $100	Savings: $1,000	Money Market: $5,000

INSTRUCTIONS

This section details the instructions for executing the lab. The Exploratory Testing activities should be performed individually , and the bugs found should be recorded in the file **exploratory_testing_report.txt**. The remainder of the lab work will be completed and submitted either individually or in pairs. If the scripted testing portion is done in pairs, one copy of **scripted_testing_report.txt** should be submitted for each pair, and both students' names recorded in that file.

The following steps should be carried out individually.

PERFORM A DEPOSIT IN THE ATM SYSTEM

1. Run the JAR file *ATM System - Lab 1 Version 1.0.jar* to show the GUI as shown in Appendix A.
2. Turn the system on using the "On" button.
3. Enter the number of $20 bills that the system is assumed to start off with, noting that this is the number of bills, not the total value of the bills. Entering a value of 10 for instance indicates that the ATM is starting with $200 (10 twenty dollar bills). Any number greater than 0 will suffice for now.
4. Click on the "Click to insert card" button which has now appeared on the main interface below the simulated ATM display.
5. The screen now changes to a prompt for the user to input the card number (since there is no actual physical card reader). Enter 1 for the card number and press Enter. Upon returning to the main screen, the display is now requesting the PIN be entered.
6. Type 42 using the simulated keypad and press Enter. The display now prompts the user to perform one of four transactions: withdraw, deposit, balance inquiry, or transfer.
7. Press 2 on the simulated keypad to perform a deposit. The display now prompts the user to indicate which account they would like to deposit to: checking, savings, or the money market account.
8. Press 2 on the simulated keypad to deposit to the savings account. The display now prompts the user to enter the deposit amount.
9. Enter a positive amount, press *Enter*. A button which represents the user inserting the deposit envelope appears.
10. Click that button to simulate inserting the envelope. The display now prompts the user whether they wish to perform another transaction or not.
11. Press 2 on the simulated keypad to indicate you do not wish to perform another transaction. The main window shows a button appearing, simulating the ejecting of the user's card.
12. Press the System Power Button once again to turn the ATM system off.

WHAT MAKES A "GOOD" BUG REPORT?

No matter what defect tracking system you are using to report defects, there are some elements that product supporters will expect. When entering your bug reports, you should be sure to include and make clear the following:

- The function being tested (e.g., Login)
- The initial state of the system (e.g., System is on and is idle, i.e., not already serving a customer)
- **Steps to reproduce the defect/bug** (e.g., Insert a card, then enter correct card number and PIN)
- What was the expected outcome? (e.g., the system successfully accepts the customer, shows the banking menu)
- What was the actual outcome?

Reporting defect reports in a simple, concise manner ensures that the developer who reads the defect report will know what the issue is, and will be more likely to fix it.

Exploratory (manual non-scripted) testing

In order to perform any testing, the requirements must first be known. Read over the requirements for the ATM simulation system as outlined in Appendix C before continuing with the rest of this section.

Before beginning testing, try to come up with a high-level exploratory test plan for how you intend to test the system. This plan could include but is not limited to, information such as: functions being targeted, the approach to be taken (test most functions a little bit, or test a few functions extensively, etc.), and how you plan to come up with test cases (test most common paths, or exceptional paths, etc.). Keep in mind that this does not need to necessarily be the best plan, as long as it is justifiable.

Reporting defects: Carry out your devised exploratory test plan for roughly half an hour. Recall that this testing phase has to be done individually. Each student needs to perform exploratory testing alone and record defects using their own copy of **exploratory_testing_report.txt.**

The following steps should be carried out with your partner.

Peer review: Once you and your lab partner have each completed the exploratory testing phase, compare the defects you found with those that your partner recorded. While reviewing, follow the guidelines outlined in Appendix D.

MANUAL SCRIPTED TESTING

This section is to be performed as a pair. One of you can 'drive' the testing (operate the computer executing the system under test), while the other keeps track of which tests have been performed, reports any defects found, and determines what order to execute tests in. Keep track of what order the tests are executed in, as it will be useful information later on. Note that it does not matter which student reports the defects, as it is a pair effort.

In Appendix E, a basic test suite has been provided for this SUT. Execute each of the test cases at least once, verifying that the actual results match the expected results for each case.

REGRESSION TESTING (VERIFICATION OF DEFECT FIXES)

This section is to be performed as a pair. The defects reported in the two previous stages of testing can be divided among the pair and can be retested individually.

Download the updated version (version 1.1) of the ATM simulation system from the website. This version of the system has been partially fixed by imaginary developers based on the defect reports previously existing.

Retest each of these defects to determine which have been fixed and which have not. To do this we must follow the defect lifecycle as shown in Appendix B. Since we do not know which defects have been fixed exactly, assume that all defects have had an attempt to fix them. If the defect has actually been fixed in the ATM system version 1.1, indicate this in the corresponding report file. If the defect has not actually been fixed in the ATM system version 1.1, indicate this, and write a comment stating "Defect still exists in version 1.1".

Execute the steps of "Manual Scripted Testing" (above) once again, looking only for new defects that have been created. If a defect is found which had previously been reported, do not report it again. When reporting these defects, ensure that you tag them as applying to version 1.1.

Deliverables and Grading

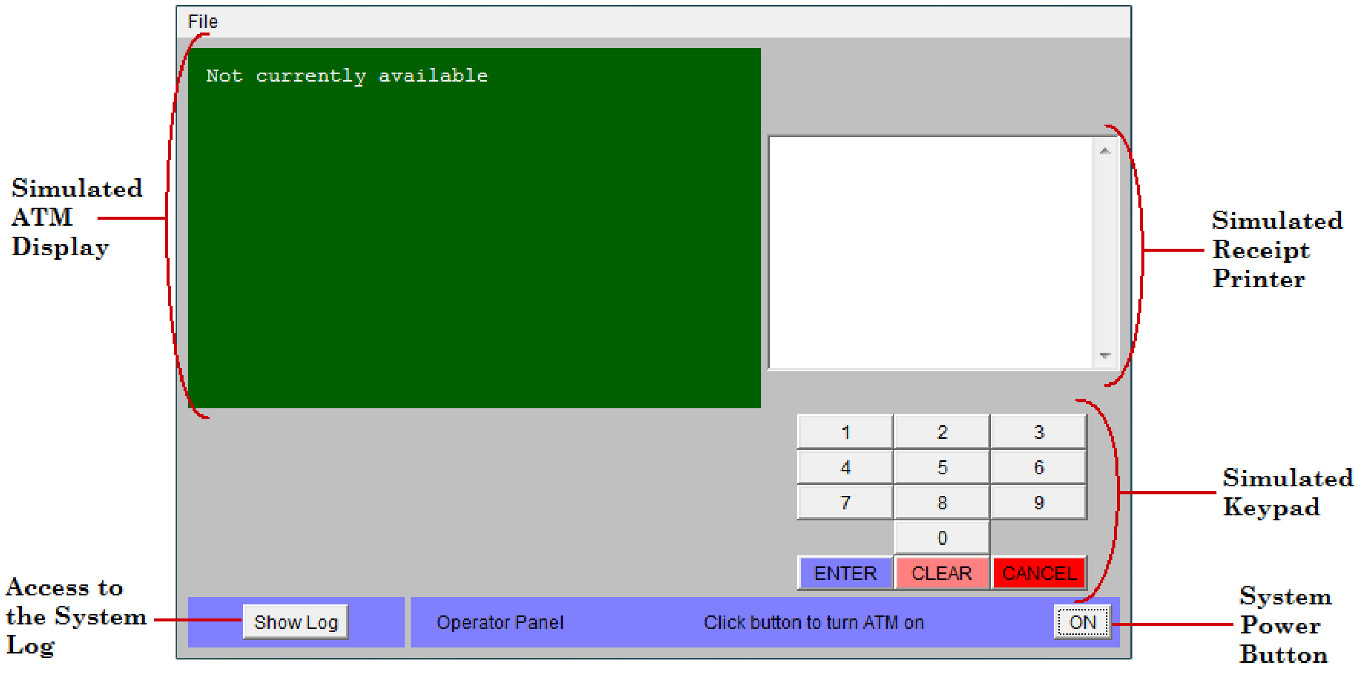
There are two files to be submitted this lab – **exploratory_testing_report.txt** and **scripted_testing_report.txt.** Use git add and git commit to stage your changes, and then push them back to gitlab.csc.uvic.ca.

Each student must submit their own copy of e**xploratory_testing_report.txt**, which records the bugs you found during your initial, solo tests of the ATM system.

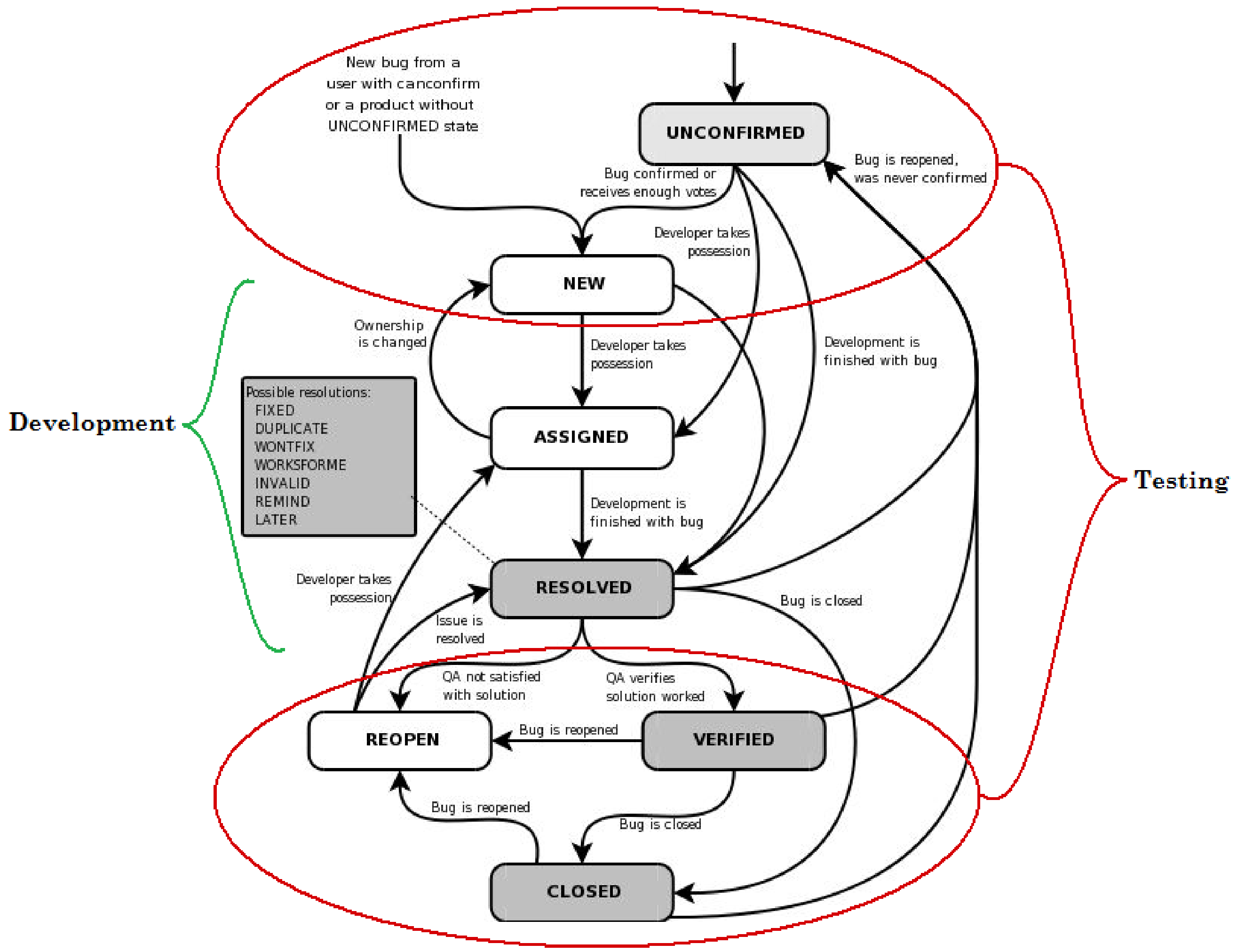
If you completed the scripted testing portion of the lab on your own, submit **scripted_testing_report.txt** as well, which records the bugs you found during the scripted portion of the lab (following Appendix E).

If you completed the scripted testing portion of the lab in a pair, one member of your pair should submit the **scripted_testing_report.txt** file – they other member of the pair can leave theirs blank. Make sure to indicate the names of both members of your pair in the **scripted_testing_report.txt** file so that both of you receive credit for the activity.

APPENDIX A – USER INTERFACE FOR ATM SIMULATION SYSTEM



APPENDIX B – DEFECT LIFECYCLE FROM INITIAL REPORT TO CLOSED


New bug from a user with canconfirm or a product without UNCONFIRMED state
UNCONFIRMED
Bug confirmed or receives enough votes
Bug is reopened, was never confirmed
Developer takes possession
NEW
Ownership is changed
Developer takes possession
Development is finished with bug
Possible resolutions:
FIXED
DUPLICATE
WONTFIX
WORKSFORME
INVALID
REMIND
LATER
Development
ASSIGNED
Development is finished with bug
Testing
Developer takes possession
RESOLVED
Issue is resolved
Bug is closed
QA not satisfied with solution
QA verifies solution worked
REOPEN
Bug is reopened
VERIFIED
Bug is reopened
Bug is closed
CLOSED

APPENDIX C – REQUIREMENTS FOR THE ATM SIMULATION SYSTEM

HIGH-LEVEL REQUIREMENTS

The software to be designed will control a simulated automated teller machine (ATM) having a magnetic stripe reader for reading an ATM card, a customer console (keyboard and display) for interaction with the customer, a slot for depositing envelopes, a dispenser for cash (in multiples of $20), a printer for printing customer receipts, and a key-operated switch to allow an operator to start or stop the machine. The ATM will communicate with the bank's computer over an appropriate communication link. (The software on the latter is not part of the requirements for this problem.)

The ATM will service one customer at a time. A customer will be required to insert an ATM card and enter a personal identification number (PIN) - both of which will be sent to the bank for validation as part of each transaction. The customer will then be able to perform one or more transactions. The card will be retained in the machine until the customer indicates that he/she desires no further transactions, at which point it will be returned - except as noted below.

The ATM must be able to provide the following services to the customer:

- A customer must be able to make a cash withdrawal from any suitable account linked to the card, in multiples of $20.00. Approval must be obtained from the bank before cash is dispensed.
- A customer must be able to make a deposit to any account linked to the card, consisting of cash and/or checks in an envelope. The customer will enter the amount of the deposit into the ATM, subject to manual verification when

the envelope is removed from the machine by an operator. Approval must be obtained from the bank before physically accepting the envelope.

- A customer must be able to make a transfer of money between any two accounts linked to the card.
- A customer must be able to make a balance inquiry of any account linked to the card.
- A customer must be able to abort a transaction in progress by pressing the Cancel key instead of responding to a request from the machine.

The ATM will communicate each transaction to the bank and obtain verification that it was allowed by the bank. Ordinarily, a transaction will be considered complete by the bank once it has been approved. In the case of a deposit, a second message will be sent to the bank indicating that the customer has deposited the envelope. (If the customer fails to deposit the envelope within the timeout period, or presses cancel instead, no second message will be sent to the bank and the deposit will not be credited to the customer.)

If the bank determines that the customer's PIN is invalid, the customer will be required to re-enter the PIN before a transaction can proceed. If the customer is unable to successfully enter the PIN after three tries, the card will be permanently retained by the machine, and the customer will have to contact the bank to get it back.

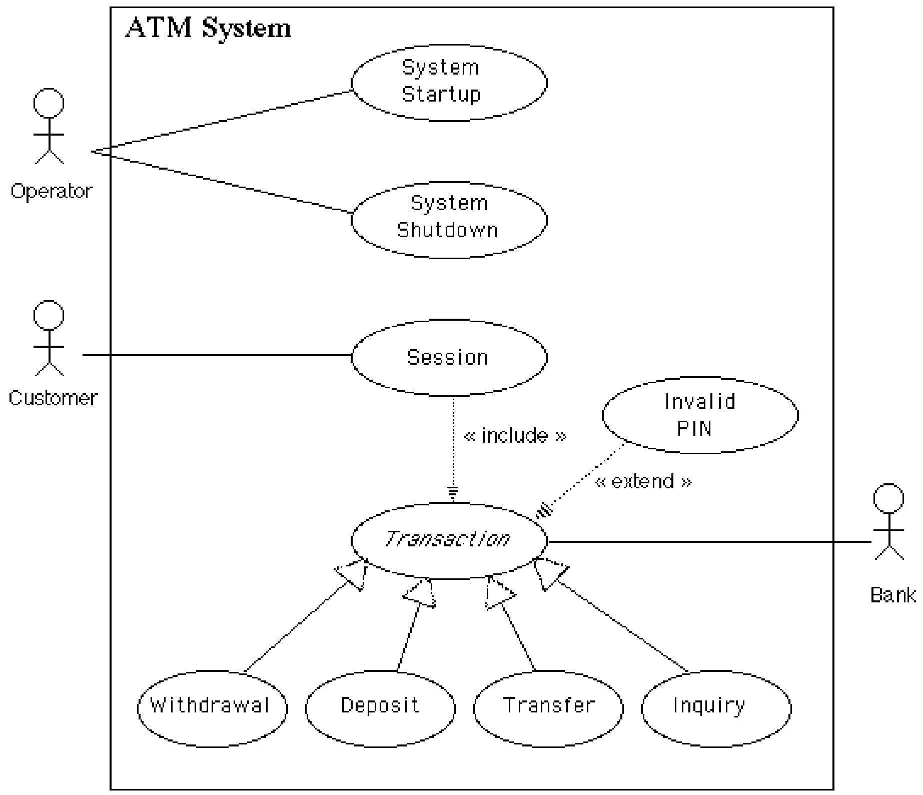
If a transaction fails for any reason other than an invalid PIN, the ATM will display an explanation of the problem, and will then ask the customer whether he/she wants to do another transaction.

The ATM will provide the customer with a printed receipt for each successful transaction, showing the date, time, machine location, type of transaction, account(s), amount, and ending and available balance(s) of the affected account ("to" account for transfers).

The ATM will have a key-operated switch that will allow an operator to start and stop the servicing of customers. After turning the switch to the "on" position, the operator will be required to verify and enter the total cash on hand. The machine can only be turned off when it is not servicing a customer. When the switch is moved to the "off" position, the machine will shut down, so that the operator may remove deposit envelopes and reload the machine with cash, blank receipts, etc.

The ATM will also maintain an internal log of transactions to facilitate resolving ambiguities arising from a hardware failure in the middle of a transaction. Entries will be made in the log when the ATM is started up and shut down, for each message sent to the Bank (along with the response back, if one is expected), for the dispensing of cash, and for the receiving of an envelope. Log entries may contain card numbers and dollar amounts, but for security will *never* contain a PIN.

USE-CASE DIAGRAM



APPENDIX D – GUIDELINES FOR REVIEWING A DEFECT REPORT

Ask at least one question within each of the four categories:

1. What are my **first impressions** of the report?
2. What happens when I attempt to **replicate the defect**?
3. What **follow-up tests** should have been done in the course of writing this report?
4. Does the report include **speculation or evaluation** by the tester, and if so, is it appropriate and useful?

Skim through this list as you read the report—don't work through every question. Your evaluation should point out the strengths of what you have read as well as the weaknesses.

FIRST IMPRESSIONS

- Is there a summary?
    - Is it short (about 50-70 characters) and descriptive?
- Can you understand the report?
    - As you read the description, do you understand what the reporter did?
    - Can you envision what the program did in response?
    - Do you understand what the failure was?
- Is it obvious where to start (what state to bring the program to) to replicate the bug?
- Is it obvious what files to use (if any)? Is it obvious what you would type?
- Is the replication sequence provided as a numbered set of steps, which tell you exactly what to do and, when useful, what you will see?
- Does the report include unnecessary information, personal opinions or anecdotes that seem out of place?
- Is the tone of the report insulting? Are any words in the report potentially insulting?

- Does the report seem too long? Too short? Does it seem to have a lot of unnecessary steps? (This is your first impression—you might be mistaken. After all, you haven't replicated it yet. But does it LOOK like there's a lot of excess in the report?)
- Does the report seem overly general ("Insert a file and you will see" – what file? What kind of file? Is there an example, like "Insert a file like blah.foo or blah2.fee"?)

REPLICATE THE REPORT

- Can you replicate the bug?
- Did you need additional information or steps?
- Did you have to guess about what to do next?
- Did you get lost or wonder whether you had done a step correctly? Would additional feedback (like: "the program will respond like this...") have helped?
- Did you have to change your configuration or environment in any way that wasn't specified in the report?
- Did some steps appear unnecessary? Were they unnecessary?
- Did the description accurately describe the failure?
- Did the summary accurately describe the failure?
- Does the description include non-factual information (such as the tester's guesses about the underlying fault) and if so, does this information seem credible and useful or not?

FOLLOW-UP TESTS

- Are there follow-up tests that you would run on this report if you had the time?
    - In follow-up testing, we vary a test that yielded a less-than-spectacular failure. We vary the operation, data, or environment, asking whether the underlying fault can yield a more serious failure or a failure under a broader range of circumstances.
    - You will probably NOT have time to run many follow-up tests yourself. For evaluation, my question is not what the results of these tests were. Rather it is, what follow-up tests should have been run—and then, what tests were run?
- What would you hope to learn from these tests?
- How important would these tests be?
- Are some tests so obviously likely to yield important information that you feel a competent reporter would have run them *and described the results?*
    - The report describes a corner case without apparently having checked non-extreme values.
    - Or the report relies on other specific values, with no indication about whether the program just fails on those or on anything in the same class (what is the class?)
    - Or the report is so general that you doubt that it is accurate ("Insert any file at this point" – *really? Any file? Any type of file? Any size? Did the tester supply* reasons for you to believe this generalization is credible? Or examples of files that actually yielded the failure?)

TESTER'S SPECULATION OR EVALUATION

Some bug reports include evaluative (rather than factual) remarks from the tester, such as hypotheses about the underlying fault or about the importance of the problem to the customer. The report need not include such information, but if it does, it should be credible, accurate, and useful.

- If the report includes such information, is it credible and useful or not?
- Does the report cite facts to support the evaluation?
- Is the evaluation based on special knowledge of the tester that might not be readily available to the programmer fixing the bug?

APPENDIX E - FUNCTIONAL TEST SUITE FOR MANUAL SCRIPTED TESTING

Test Case #	Use Case	Function Being Tested	Initial System State	Input	Expected Output
1	**System Startup**	System is started when the switch is turned "on"	System is off	Activate the "on" switch	System requests initial cash amount
2	**System Startup**	System accepts initial cash amount	System is requesting cash amount	Enter a legitimate amount	System is on
3	**System Startup**	Connection to the bank is established	System has just been turned on	Perform a legitimate inquiry transaction	System output should demonstrate that a connection has been established to the Bank
4	**System Shutdown**	System is shut down when the switch is turned "off"	System is on and not servicing a customer	Activate the "off" switch	System is off
5	**Session**	System reads a customer's ATM card	System is on and not servicing a customer	Insert a readable card	Card is accepted; System asks for entry of PIN
6	**Session**	System rejects an unreadable card	System is on and not servicing a customer	Insert an unreadable card	Card is ejected; System displays an error screen; System is ready to start a new session
7	**Session**	System accepts customer's PIN	System is asking for entry of PIN	Enter a PIN	System displays a menu of transaction types
8	**Session**	System allows customer to perform a transaction	System is displaying menu of transaction types	Perform a transaction	System asks whether customer wants another transaction
9	**Session**	System allows multiple transactions in one session	System is asking whether customer wants another transaction	Answer yes	System displays a menu of transaction types
10	**Session**	Session ends when customer chooses not to do another transaction	System is asking whether customer wants another transaction	Answer no	System ejects card and is ready to start a new session
11	**Transaction**	System handles an invalid PIN properly	A readable card has been entered	Enter an incorrect PIN and then attempt a transaction	The Invalid PIN Extension is performed
12	**Withdrawal**	System asks customer to choose an account to withdraw from	Menu of transaction types is being displayed	Choose Withdrawal transaction	System displays a menu of account types

13	**Withdrawal**	System asks customer to choose a dollar amount to withdraw	Menu of account types is being displayed	Choose checking account	System displays a menu of possible withdrawal amounts
14	**Withdrawal**	System performs a legitimate withdrawal transaction properly	System is displaying the menu of withdrawal amounts	Choose an amount that the system currently has and which is not greater than the account balance	System dispenses this amount of cash; System prints a correct receipt showing amount and correct updated balance; System records transaction correctly in the log (showing both message to the bank and approval back)
15	**Withdrawal**	System verifies that it has sufficient cash on hand to fulfill the request	System has been started up with less than the maximum withdrawal amount in cash on hand; System is requesting a withdrawal amount	Choose an amount greater than what the system currently has	System displays an appropriate message and asks customer to choose a different amount
16	**Withdrawal**	System verifies that customer's balance is sufficient to fulfill the request	System is requesting a withdrawal amount	Choose an amount that the system currently has but which is greater than the account balance	System displays an appropriate message and offers customer the option of choosing to do another transaction or not.
17	**Withdrawal**	A withdrawal transaction can be cancelled by the customer any time prior to choosing the dollar amount	System is displaying menu of account types	Press "Cancel" key	System displays an appropriate message and offers customer the option of choosing to do another transaction or not.
18	**Withdrawal**	A withdrawal transaction can be cancelled by the customer any time prior to choosing the dollar amount	System is displaying menu of dollar amounts	Press "Cancel" key	System displays an appropriate message and offers customer the option of choosing to do another transaction or not.
19	**Deposit**	System asks customer to choose an account to deposit to	Menu of transaction types is being displayed	Choose Deposit transaction	System displays a menu of account types
20	**Deposit**	System asks customer to enter a dollar amount to deposit	Menu of account types is being displayed	Choose checking account	System displays a request for the customer to type a dollar amount
21	**Deposit**	System asks customer to insert an envelope	System is displaying a request for the customer to type a dollar amount	Enter a legitimate dollar amount	System requests that customer insert an envelope
22	**Deposit**	System performs a legitimate deposit transaction properly	System is requesting that customer insert an envelope	Insert an envelope	System accepts envelope; System prints a correct receipt showing amount and correct updated balance;

					System records transaction correctly in the log (showing message to the bank, approval back, and acceptance of the envelope)
23	**Deposit**	A deposit transaction can be cancelled by the customer any time prior to inserting an envelope	System is displaying menu of account types	Press "Cancel" key	System displays an appropriate message and offers customer the option of choosing to do another transaction or not.
24	**Deposit**	A deposit transaction can be cancelled by the customer any time prior to inserting an envelope	System is requesting customer to enter a dollar amount	Press "Cancel" key	System displays an appropriate message and offers customer the option of choosing to do another transaction or not.
25	**Deposit**	A deposit transaction can be cancelled by the customer any time prior to inserting an envelope	System is requesting customer to insert an envelope	Press "Cancel" key	System displays an appropriate message and offers customer the option of choosing to do another transaction or not.
26	**Transfer**	System asks customer to choose an account to transfer from	Menu of transaction types is being displayed	Choose Transfer transaction	System displays a menu of account types specifying transfer from
27	**Transfer**	System asks customer to choose an account to transfer to	Menu of account types to transfer from is being displayed	Choose checking account	System displays a menu of account types specifying transfer to
28	**Transfer**	System asks customer to enter a dollar amount to transfer	Menu of account types to transfer to is being displayed	Choose savings account	System displays a request for the customer to type a dollar amount
29	**Transfer**	System performs a legitimate transfer transaction properly	System is displaying a request for the customer to type a dollar amount	Enter a legitimate dollar amount	System prints a correct receipt showing amount and correct updated balance; System records transaction correctly in the log (showing both message to the bank and approval back)
30	**Transfer**	A transfer transaction can be cancelled by the customer any time prior to entering dollar amount	System is displaying menu of account types specifying transfer from	Press "Cancel" key	System displays an appropriate message and offers customer the option of choosing to do another transaction or not.
31	**Transfer**	A transfer transaction can be cancelled by the customer any time prior to entering dollar amount	System is displaying menu of account types specifying transfer to	Press "Cancel" key	System displays an appropriate message and offers customer the option of choosing to do another transaction or not.
32	**Transfer**	A transfer transaction can be cancelled by the customer any time prior to entering dollar amount	System is requesting customer to enter a dollar amount	Press "Cancel" key	System displays an appropriate message and offers customer the option of choosing to do another transaction or not.

33	**Inquiry**	System asks customer to choose an account to inquire about	Menu of transaction types is being displayed	Choose Inquiry transaction	System displays a menu of account types
34	**Inquiry**	System performs a legitimate inquiry transaction properly	System is displaying menu of account types	Choose checking account	System prints a correct receipt showing correct balance; System records transaction correctly in the log (showing both message to the bank and approval back)
35	**Inquiry**	An inquiry transaction can be cancelled by the customer any time prior to choosing an account	System is displaying menu of account types	Press "Cancel" key	System displays an appropriate message and offers customer the option of choosing to do another transaction or not.
36	**Invalid PIN Extension**	Customer is asked to reenter PIN		Enter an incorrect PIN; Attempt an inquiry transaction on the customer's checking account	Customer is asked to re-enter PIN
37	**Invalid PIN Extension**	Correct re-entry of PIN is accepted	Request to re-enter PIN is being displayed	Enter correct PIN	Original transaction completes successfully
38	**Invalid PIN Extension**	Incorrect re-entry of PIN is not accepted	Request to re-enter PIN is being displayed	Enter incorrect PIN	An appropriate message is displayed and re-entry of the PIN is again requested
39	**Invalid PIN Extension**	Correct re-entry of PIN on the second try is accepted	Request to re-enter PIN is being displayed	Enter incorrect PIN the first time, then correct PIN the second time	Original transaction completes successfully
40	**Invalid PIN Extension**	Correct re-entry of PIN on the third try is accepted	Request to re-enter PIN is being displayed	Enter incorrect PIN the first time and second times, then correct PIN the third time	Original transaction completes successfully
41	**Invalid PIN Extension**	Three incorrect re-entries of PIN result in retaining card and aborting transaction	Request to re-enter PIN is being displayed	Enter incorrect PIN three times	An appropriate message is displayed; Card is retained by machine; Session is terminated

APPENDIX F -TESTING DEFINITIONS

Pair Testing

Pair Testing is a software development technique in which two team members work together at one keyboard to test the software application. One does the testing and the other analyzes or reviews the testing. This can be done between one

Tester and Developer or Business Analyst or between two testers with both participants taking turns at driving the keyboard.

Exploratory testing

Exploratory testing is a commonly used term for software testing performed without planning and documentation. Exploratory testing is performed manually and usually without any test script.

The tests are intended to be run only once, unless a defect is discovered. Exploratory testing has been criticized because it isn't structured, but this can also be a strength. By doing exploratory testing, major issues can be found quickly. It is performed with improvisation; the tester seeks to find bugs with any means that seem appropriate. It contrasts to regression testing that looks for a specific issue with detailed reproduction steps, and a clear expected result. Exploratory testing is most often used as a complement to other types of testing.

Manual scripted testing

Manual scripted testing is the oldest and one of the most rigorous types of software testing. In this particular type of testing, test cases are designed and reviewed by the team before executing it. There are many variations of this basic approach; test cases can be created at the basic functionality level or they can be created at the scenario level.

The value of scripted testing has been questioned by some experts in the field. They claim that scripted manual testing closes the minds of testers and inhibits their creativity. Also, this approach is very heavy on the documentation and requires a considerable amount of resources to create the test scripts in the first place and they often get out-dated because of the inevitable changes in the system.

Despite these drawbacks, manual scripted testing is used in many organizations of all the sizes. They make test cases repeatable and easy enough for a new person to come on board and start testing with minimum supervision. Manual scripted testing is also used in places where contractual agreement states that written specification of the software must be met for the successful implementation of the project. Scripted test cases might be useful where tests are used for the benchmarking purpose and tests have to be executed exactly in the same way, every time.

Regression testing

Regression testing is any type of software testing that seeks to uncover new errors, or regressions, in existing functionality after changes have been made to a system, such as functional enhancements, patches or configuration changes.

The intent of regression testing is to ensure that a change, such as a bug fix, did not introduce new faults. One of the main reasons for regression testing is that it's often extremely difficult for a programmer to figure out how a change in one part of the software will echo in other parts of the software. Regression testing can be done in both manual and automated testing fashion. In this lab, we will do only manual regression testing.

Issue (bug) tracking systems

There are many issue tracking systems out there, including bug (defect) tracking systems, help desk and service desk issue tracking systems. Our focus in this lab and the course are of course on bug (defect) tracking systems, and not the others.

ACKNOWLEDGEMENTS

This lab is adapted from part of a software-testing laboratory courseware available under a Creative Commons license. The laboratory courseware has been used by 50+ testing educators world-wide. More details can be found in the courseware's website:

sites.google.com/view/software-testing-labs/

The courseware has been made a reality thanks to many people, including Yuri Shewchuk, Michael Godwin, Christian Wiederseiner, Riley Kotchorek and Negar Koochakzadeh for their help in developing and testing these exercises.