Bank Accounts and Management System

Name: Venkatrao Bandaru

LINK: https://github.com/Venkatrao9999/cs567.git

Introduction:

In my code, I've crafted two essential classes: **Bank** and **BankAccount**, to embody the core functionalities of a banking system. As a BankAccount, I encapsulate key operations like deposit, withdrawal, and balance inquiry, maintaining crucial attributes such as account number, holder name, balance, and transaction history. The **Bank** class, on the other hand, acts as a manager for a collection of accounts, offering capabilities such as account creation, closure, and inter-account funds transfer. Within my **main** function, I demonstrate the practical usage of these classes by orchestrating a series of actions—creating accounts, executing transactions, fetching account details, and finalizing account closures. This structured approach not only exemplifies the essence of banking operations but also showcases the seamless interaction between myself, as the banking system, and the individual account entities.

In testing my code using DeepState, I'll orchestrate a suite of test cases that comprehensively evaluate the functionality of both the **Bank** and **BankAccount** classes. Leveraging DeepState's fuzzing capabilities, I'll generate diverse inputs to explore various execution paths and edge cases within the codebase. By employing the Universal Mutator tool, I'll systematically introduce mutations to the test inputs, ensuring robustness against potential faults and corner cases. Additionally, I'll integrate code coverage analysis tools to monitor the effectiveness of my test suite, aiming for high coverage metrics across the codebase. Through this rigorous testing regimen, I strive to enhance the reliability and resilience of my banking system implementation, thereby instilling confidence in its performance and correctness.

Code

Github:

https:// <Git Clone Command(https from github) >

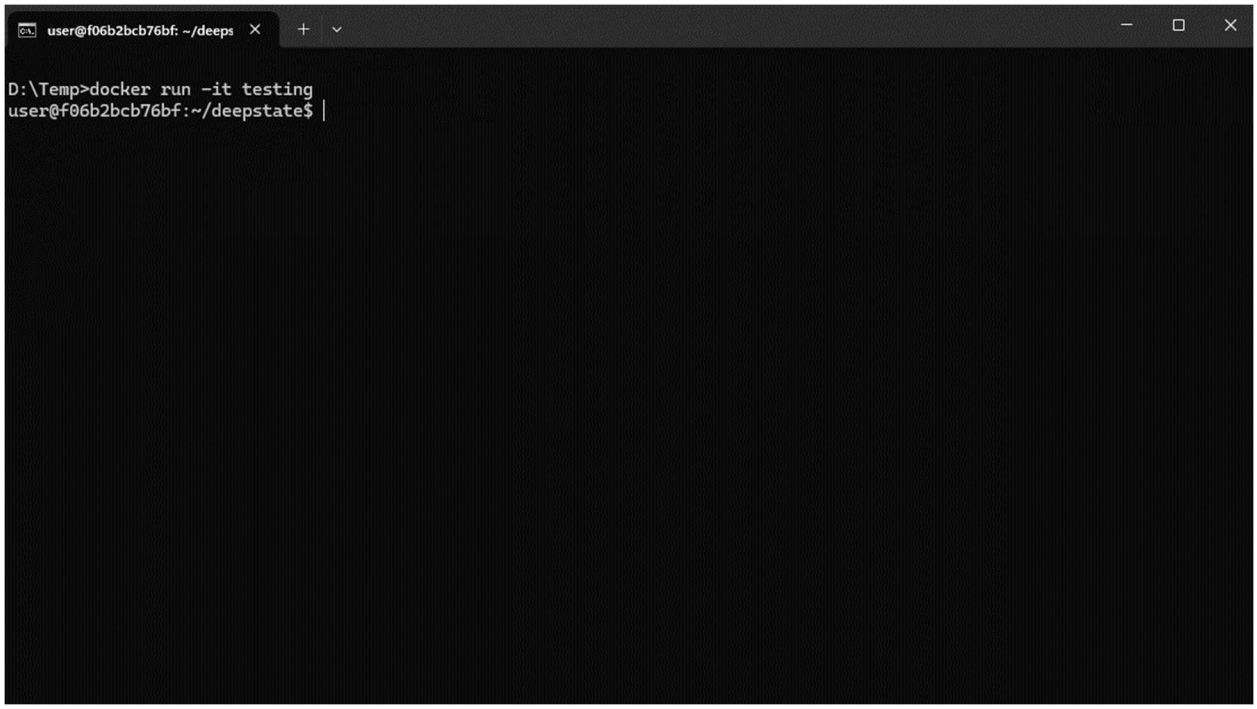
Steps used to perform Testing:

1. Setting up the docker Environment:

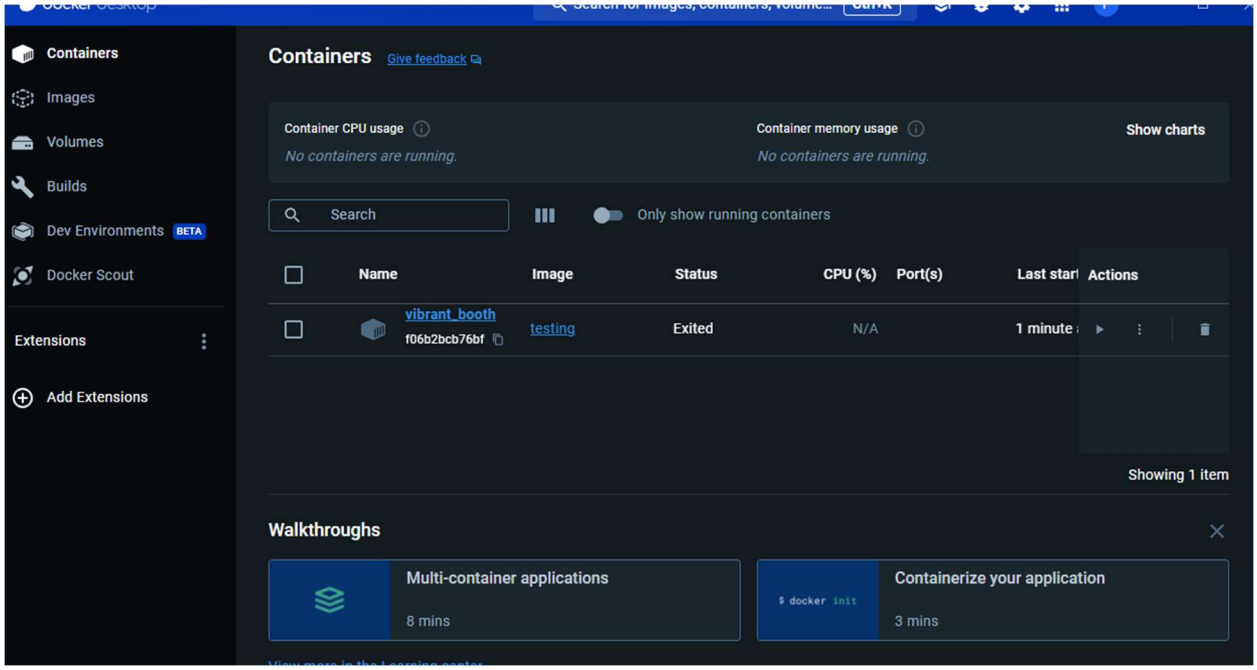
Docker run -it testing

Docker commit 19c0edd238e4 testing

By using the above commands I was able to run the docker container and commit any changes made to it, so the subsequent runs have all the changes made to it, also I need to get the container id for each instance using Docker ps.



Docker desktop:



2. Cloning the files into the container:

   For this I used github to clone the files into the container using git clone

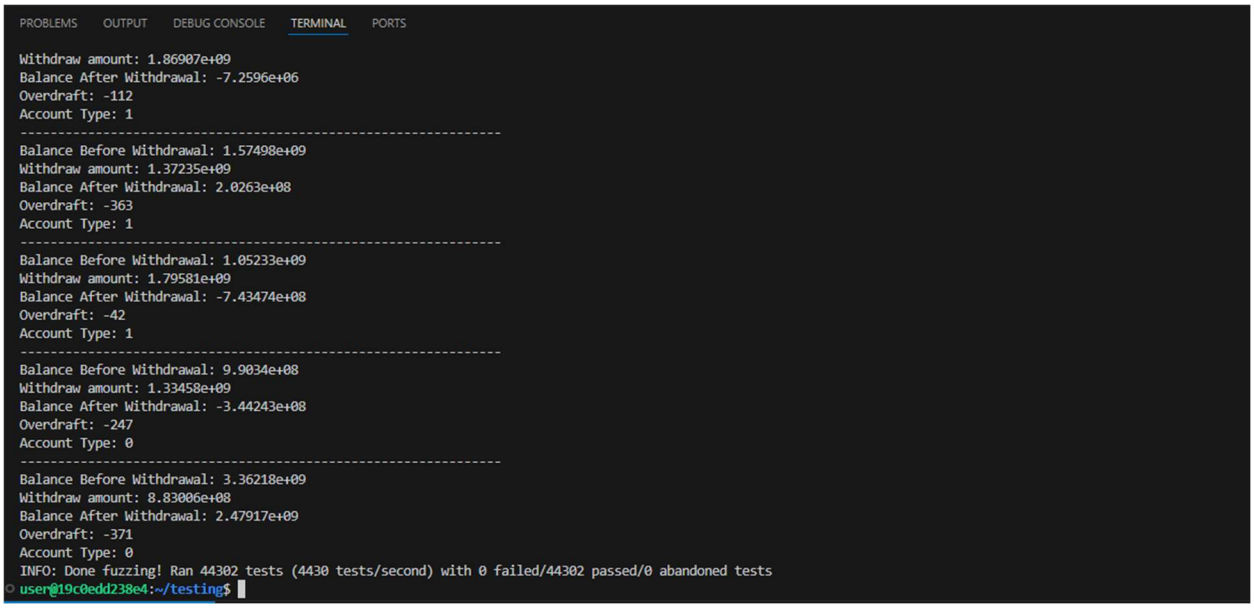
   <Git Clone Command(https from github) >

3. Compiling the code:

g++ BankAccount.cpp test.cpp -o test -ldeepstate

now to test the code with fuzz command I run

./test --fuzz --timeout 10 --output_test_dir=tests -vv

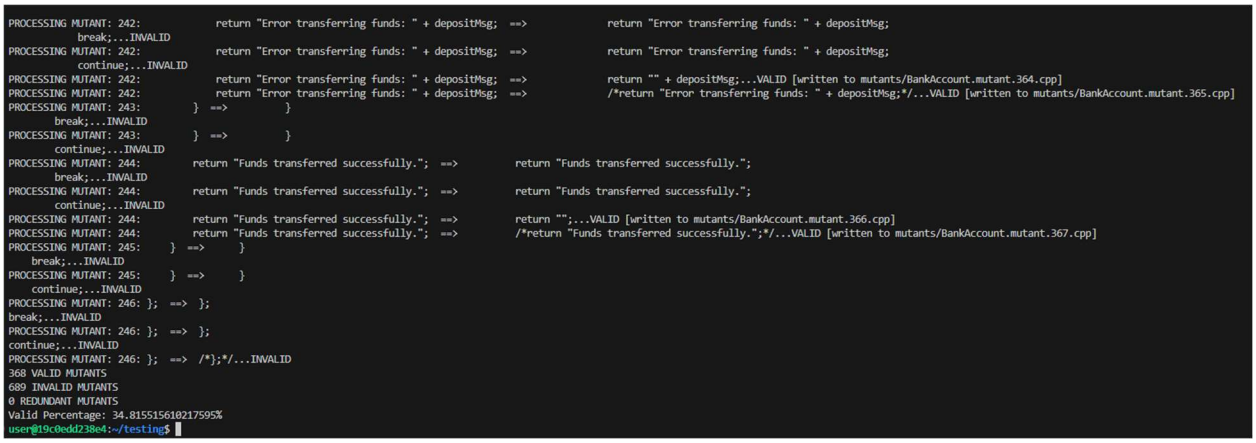


4. Mutating the code:

   By running the Universal Mutator I created and stored the mutants in the mutants directory,

   mutate BankAccount.cpp --mutantDir mutants --cmd "g++ BankAccount.cpp -c"

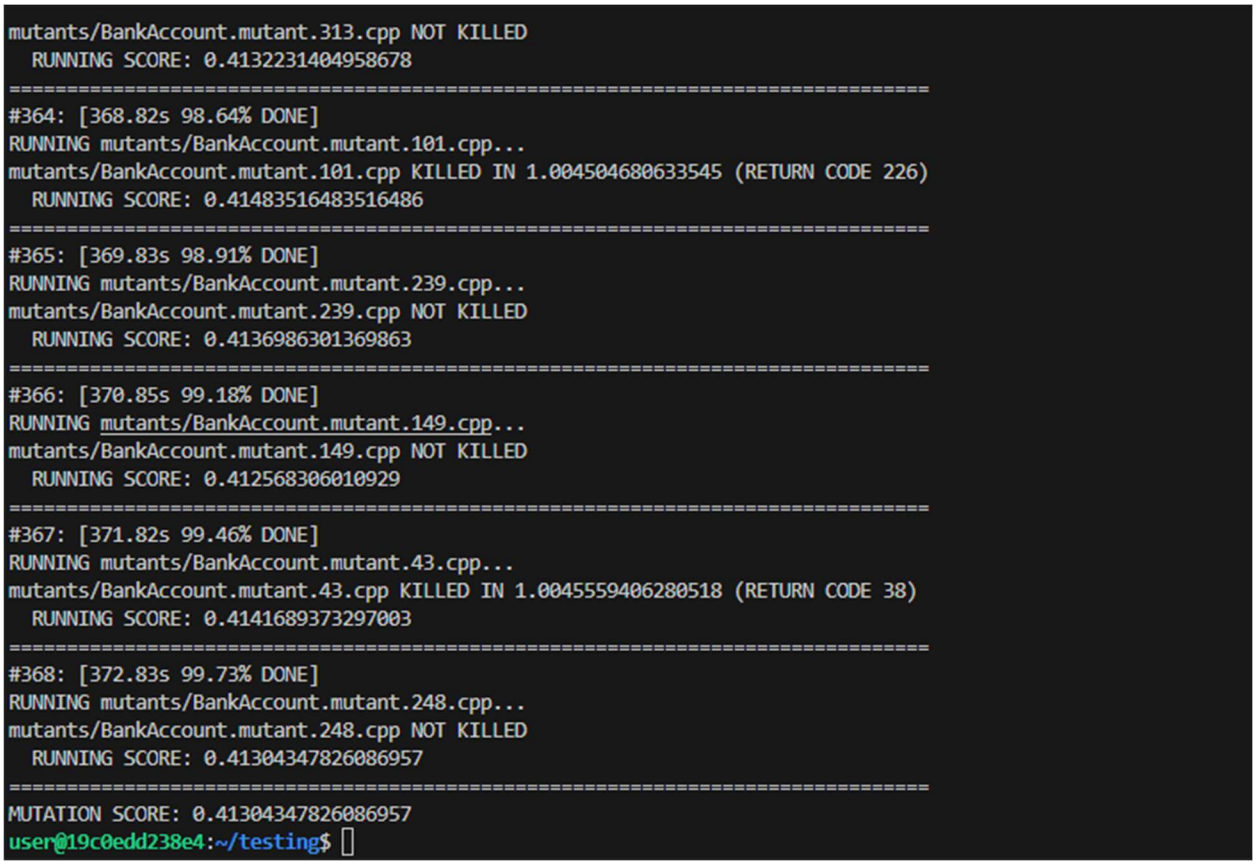
   This Not only creates mutants but also validates them and only the valid mutants are worth testing.



5. Analyzing The Mutants:

   analyze_mutants BankAccount.cpp --mutantDir mutants "g++ BankAccount.cpp test.cpp -o test -ldeepstate && ./test --fuzz --timeout 1" --timeout 5

   By running the above command we can analyze the muants and which also calculates the mutation score.



6. Checking the code Coverage:

   To check the code coverage I've used the gcov and lcov commands

   First compile with additional flags:

   g++ -O1 -fprofile-arcs -ftest-coverage test.cpp BankAccount.cpp -o test -ldeepstate -lgcov

   then run the code in fuzz mode:

   ./test --fuzz --timeout 10 --output_test_dir=coverage -vv

   After that capture the coverage using the lcov command:

   lcov --capture --directory . --output-file coverage.info

   lcov --remove coverage.info '/usr/include/*' '/usr/local/include/*' -o coverage.info

```
Reading tracefile coverage.info
Removing /usr/include/c++/7/bits/basic_string.h
Removing /usr/include/c++/7/bits/basic_string.tcc
Removing /usr/include/c++/7/bits/char_traits.h
Removing /usr/include/c++/7/bits/locale_facets.h
Removing /usr/include/c++/7/bits/move.h
Removing /usr/include/c++/7/bits/std_function.h
Removing /usr/include/c++/7/bits/stl_algobase.h
Removing /usr/include/c++/7/bits/stl_construct.h
Removing /usr/include/c++/7/bits/stl_iterator.h
Removing /usr/include/c++/7/bits/stl_iterator_base_funcs.h
Removing /usr/include/c++/7/bits/stl_uninitialized.h
Removing /usr/include/c++/7/bits/stl_vector.h
Removing /usr/include/c++/7/bits/vector.tcc
Removing /usr/include/c++/7/ext/new_allocator.h
Removing /usr/include/c++/7/ext/string_conversions.h
Removing /usr/include/c++/7/initializer_list
Removing /usr/include/c++/7/iostream
Removing /usr/include/c++/7/ostream
Removing /usr/include/x86_64-linux-gnu/bits/stdio2.h
Removing /usr/local/include/deepstate/DeepState.h
Removing /usr/local/include/deepstate/DeepState.hpp
Removing /usr/local/include/deepstate/Stream.hpp
Deleted 22 files
Writing data to coverage.info
Summary coverage rate:
  lines......: 96.3% (130 of 135 lines)
  functions..: 92.0% (23 of 25 functions)
  branches...: no data found
user@19c0edd238e4:~/testing$
```