Code testing:

UI Testing:  
For the requirements given above, develop an experiment to evaluate the UI.  
i. Use two testing methods: functionality and boundary.  
ii. Provide one test criterion for each method.  
iii. Design a suitable Test Requirement (TR) set for each criterion.  
iv. Provide all test cases for your TR set.  
v. Run your test cases and collect results of your UI tests using the following table:  
1. Test method, test criterion, test input values, test expected output, test actual  
output, success/fail.  
vi. Take screenshots of your test results  
vii. Analyze your results and identify any unexpected behavior or wrong results.  
viii. Provide a report interpreting your test results

**1. Overview and Testing Methods**

**A. Functionality Testing**

**Purpose:** Verify that the application works correctly for a well-defined, expected sequence of user inputs.  
**Test Criterion (FC):** *“ATM will continue to function without error during defined sequence of user inputs. The UI testing will contain a set of user inputs with the expected result of each step. An unexpected result is considered a fail.”*

**B. Boundary Testing**

**Purpose:** Validate the system’s behavior when inputs are at or just beyond defined limits.  
**Test Criterion (BC):** *“ATM should enforce the daily limits for deposit, withdraw and transfer on accounts. The system should not accept negative values or overdrafts. The UI testing will contain a set of invalid inputs as well as a set of inputs close to boundaries.*

**2. Test Requirement (TR) Sets**

**TR Set for Functionality Criterion (FC):**

* **TR-F1:** The user can properly use each part of the checking account. Deposit, withdrawal, transfer, pay bill, check balance.
* **TR-F2:** The user can properly use each part of the savings account. Deposit, transfer, check balance.
* **TR-F3:** The user can properly use each part of the utility account. Log on, payment history, next bill.

**TR Set for Boundary Criterion (BC):**

* **TR-B1:** The user should be limited by daily limits when using accounts.
* **TR-B2:** The user should be unable to use negative values or overdraft accounts.
* **TR-B3:** The user should be able to use values just up to the daily limits and overdraft limit without failure.

**3. Test Cases**

**A. Functionality Test Case (FC)**

* **Test Case FC1:** *Test checking account*
  + **Step 1:**
* **Test Case FC2:** *Test savings account*
* **Test Case FC3:** *Test utility account*

**B. Boundary Test Cases (BC)**

* **Test Case BC1:** *Test daily limits*
* **Test Case BC2:** *Test negative values and overdraft*
* **Test Case BC3:** *Test boundary values*

**5. Test Execution and Screenshots**

**Execution Process**

1. **Simulated Run:**  
   We executed each test case by providing the input values manually via the console.
2. **Result Collection:**  
   For each test case, the output on the console was recorded and compared against the expected behavior.