Task 5 . [Test design] Describe how would you test the following 2 scenarios: a. How would you test an ATM? What functions of the ATM would you check and from what perspectives? b. How would you test an application performing the four basic functions: addition, subtraction, multiplication, division?

1. For the ATM scenario, this is how I would go about *functional testing*:

**1. Start with card & PIN functionality**

* Insert various cards and check if the ATM reads valid cards and rejects invalid/expired ones.
* Enter the correct/incorrect PIN and verify if the responses are appropriate
* Test the maximum allowed PIN attempts before the card gets blocked (typically 3 or wait for a while until the limit is reset)

**2. Continue with checking transaction types**

* **Withdrawal**: Check for valid amounts, insufficient balance, denomination handling (bills of 10,5,100 etc.), daily withdrawal limits, large amount handling (have this confirmed with a bank teller)
* **Deposit**: Cash and check deposits — validate amount, recognition, and updates to balance.
* **Balance Inquiry**: Ensure it reflects the accurate balance (maybe cross-check with a bank employee and the banking app)
* **Fund Transfer**: Between funds belonging to the same bank / different bank accounts.
* **Statements**: Print/display recent transactions and see if this works and if the transactions are accurate

**3. Account Handling**

* Test for different account types (savings, checking, etc.).
* Test for joint account access and validation (for example, spouses who have access to the same account - this can be done via simulation)
* Test foreign card acceptance, if it’s the case

**4. Receipts**

* Verify that the details on the printed receipt match the banking operation
* Optional receipt operations: for example, test what happens if a user chooses not to print.

**5. Session Handling**

* Test that the session times out after inactivity.
* Test what happens if the card is ejected before/in-between transactions (if it works as expected)

Then there are some *non-functional* test aspects we could go about :

**1. Security**

* **Physical Security**: Test the card skimming resistance, camera placement (some may try to place small cameras on the ATM to get a user’s PIN and use clone cards), secure cash compartment
* **Data Security**: PIN encryption, secure communication with bank servers, session timeouts (not leaving potential breaches)
* **Fraud Prevention**: Suspicious activity detection (remote places, large amounts), card retention for stolen cards

**2. Usability**

* **Interface Design**: Screen readability, button responsiveness, accessibility features for disabled users
* **User Flow**: Intuitive navigation, clear error messages, multilingual support
* **Physical Ergonomics**: Screen height, keypad accessibility, privacy screen angles

**3. Performance (I think this can be done via simulation)**

* **Response Time**: Transaction processing speed, network connectivity handling
* **Concurrent Users**: Multiple simultaneous transactions
* **Load Testing**: Peak usage periods, system stability under stress

**4. Integration**

* Backend banking system, poor connectivity handling and network failures, offline mode capabilities

**5. Hardware Testing:**

* Weather resistance (dust, humidity), vandalism protection (through mechanical testing), power failure recovery

1. For the basic calculator application, this is how I would go about *functional testing*:

* **Basic Operations**: Verify that each operation (addition, subtraction, division, multiplication) produces correct results with positive numbers, negative numbers, decimals, and zero
* **Edge Cases**: Division by zero handling, very large numbers, very small numbers, maximum input length
* **Input Validation**: Test non-numeric input rejection, try using special characters, test for empty inputs

**Boundary Testing:**

* **Numerical Limits**: Test the maximum and minimum values that the system can handle
* **Precision Testing**: Check decimal place accuracy, rounding behavior
* **Overflow/Underflow**: System behavior when results exceed limits

And this is how I would go about *non-functional testing*:

**User Interface Testing:**

* **Input Methods**: Keyboard entry, button clicks, test the copy-paste functionality
* **Display**: Result formatting, scientific notation for large numbers, error message clarity
* **User Experience**: Clear operation indicators, history tracking(if it’s the case), memory functions

**Error Handling:**

* **Invalid Operations**: Malformed expressions, incomplete calculations
* **System Errors**: Memory allocation issues, unexpected shutdowns
* **Recovery**: Application state after errors, data persistence

**Performance Testing:**

* **Calculation Speed**: Response time for complex operations
* **Memory Usage**: Efficient resource utilization (especially for large numbers)
* **Stress Testing**: Rapid successive calculations, very long calculation chains

**Compatibility Testing:**

* **Platform Testing**: Different operating systems (Windows, Linux), browsers (if web-based), mobile devices (Iphones, Samsung Galaxy etc.) (This can also be done via simulation software)
* **Integration**: Compatibility with other applications (if it’s the case), data export/import capabilities (.txt, .xls files perhaps ?)