

Testing checklist: mobile banking app

Source: <https://howtotestaproduct.com/how-to-test/mobile-banking-app/>

Generated: 2026-10-07. Licence: CC BY 4.0.

☐ 1. release smoke tests

Done when: On every store build, a tester installs the app fresh, logs in to a test customer, sees only that customer's accounts and moves money between two of them.

Risks: store-rejection, data-breach. Tool class: mobile automation. Usually run by: QA engineer.

☐ 2. integration tests

Done when: A test debit card authorisation and its later settlement create one ledger entry, and no overdraft fee appears when the balance covered the amount at authorisation.

Risks: money-loss, dependency-failure. Tool class: API client. Usually run by: developer.

☐ 3. end-to-end functional tests

Done when: A tester adds a payee, sees the name check result, sends a first transfer, and the receiving test account shows the exact amount once.

Risks: fraud-abuse, money-loss. Tool class: mobile automation. Usually run by: QA engineer.

☐ 4. penetration testing

Done when: An external tester logged in as customer A cannot open customer B's transactions, statements or payees by changing account identifiers in app requests.

Risks: data-breach, fraud-abuse. Tool class: proxy. Usually run by: security specialist.

☐ 5. data integrity and migration testing

Done when: After the overnight batch, balances, interest and statement lines for a seeded set of test accounts match the values worked out by hand in the reconciliation sheet.

Risks: data-corruption, money-loss. Tool class: data validation. Usually run by: QA engineer.

☐ 6. compliance checks

Done when: A transfer to a new payee shows the scam warning, the customer's confirmation appears in the audit log, and the reimbursement claim opens from the transaction screen.

Risks: fraud-abuse. Tool class: test management. Usually run by: product manager.

☐ 7. resilience and failure testing

Done when: With the core banking test service switched off, the app shows a service message, and a transfer submitted during the outage posts once after recovery.

Risks: dependency-failure, regulatory-penalty. Tool class: chaos tool. Usually run by: developer.

☐ 8. accessibility testing

Done when: A screen reader user logs in, hears the balance of each account and completes a transfer to a saved payee without sighted help.

Risks: accessibility-liability. Tool class: accessibility checker. Usually run by: accessibility specialist.

☐ 9. regression suite

Done when: Every app update replays the account switch, transaction list and statement download cases, and no screen shows a transaction belonging to another test customer.

Risks: data-breach, data-corruption. Tool class: mobile automation. Usually run by: QA engineer.

☐ 10. security scanning

Done when: Each release build is scanned and no open high finding remains on session storage, certificate checks or logged account numbers.

Risks: data-breach. Tool class: SAST scanner. Usually run by: security specialist.

☐ 11. device and browser compatibility

Done when: The release build installs, logs in and shows the account list on the oldest iOS and Android versions named on the store listing.

Risks: store-rejection. Tool class: device cloud. Usually run by: QA engineer.