Test Strategy for Parabank Web Application

1. Objective

The objective of this test strategy is to ensure the Parabank website functions correctly, is secure, and provides a smooth user experience. Testing will focus on validating UI elements, navigation, functional workflows, performance, and compatibility across browsers and devices.

2. Scope

In Scope:

- ATM actions (withdraw, transfer, check balance, deposits)
- Online services (bill pay, account history, transfer funds)
- Home page functionality and layout
- Navigation links (e.g., “About Us”, “Services”, “Products”, “Locations”, “Admin Page”)
- User login and registration
- Contact page accessibility
- Dynamic data loading (e.g., news, account services)
- Browser compatibility testing
- API calls triggered by user actions
- AI-driven test automation for exploratory, API and data layers.

Out of Scope:

- Backend database validation (unless explicitly accessible)

3. Test Types

a. Functional Testing

- Verify all homepage links redirect correctly.
- Validate login, registration, and logout workflows.
- Confirm all form submissions and buttons behave as expected.
- Validate account services and transfers post-login.
- Validate all online service transactions workflows
- Validate all ATM actions workflows

b. Negative Testing

- **Login and Authentication:**

        - Invalid credentials, blank fields, SQL injection strings, and long input values.
        - Verify proper error messages appear and the system prevents unauthorized access or session misuse.
    - **Registration and Form Validation:**
        - Enter invalid or incomplete data (e.g., bad email formats, weak passwords, non-numeric phone numbers).
        - Confirm validation messages display correctly and data is not accepted or stored.
    - **Account and Transaction Operations:**
        - Attempt invalid transactions such as negative or excessive fund transfers, or transfers to the same account.
        - Ensure the application rejects such actions and maintains data integrity.
    - **Navigation and Access Control:**
        - Bypass authentication by directly accessing restricted URLs.
        - Manipulate query parameters or use the browser's back button after logout to test proper session handling.
    - **Security and Input Injection:**
        - Inject JavaScript or SQL-like payloads into input fields and URLs.
        - Validate that inputs are sanitized, no code executes, and no sensitive system information is exposed.
    - **System-Level Robustness:**
        - Simulate rapid, repetitive requests to test server resilience.
        - Submit malformed or incomplete API requests and confirm the server returns safe error codes (e.g., 400 Bad Request).

d. Performance Testing

- Validate page load time (target < 3 seconds).
- Validate transactions (ATM, online services) load time (target < 1 seconds).
- Check responsiveness under simulated network latency (use Charles proxy or Chrome devtools throttling).
- Test stability under light concurrent user load (using JMeter or similar).

e. Security/Safety Checks Testing

- Only valid users can login (RBAC, etc)

- Validate input fields against SQL injection and XSS.
- Ensure session management and secure logout.
- Check for HTTPS redirection and secure cookie flags.

f. Regression Testing

- Re-run existing test cases after code updates or deployments.
- Automate high-frequency scenarios for efficiency.

g. Observability/Quality Metrics

- Logs captures key user actions, does not contain sensitive data, log levels (info, warning, error)
- Application metrics (i.e., request/response times, error rates, uptime, response codes)

4. Test Environment

- **Test URL:** https://parabank.parasoft.com/parabank/index.htm
- **Browsers:** Chrome (latest), Firefox (latest), Edge, Safari
- **Devices:** Desktop, Tablet, Mobile
- **Tools:**
    - Functional automation: Selenium / Playwright
    - API testing: Postman
    - Performance: JMeter
    - Defect tracking: JIRA
    - Test management: Xray or Zephyr
- **Data Strategy:**
    - Static data (CSV, json, db tables)
    - Dynamic data (created during automation, random data)
    - Synthetic data (use data generation tools)
    - Masked / Anonymized data (PII data should be masked or anonymized)

5. Entry Criteria

- Application build is deployed and stable.
- Test data is available and accessible.
- QA environment is configured and accessible.

6. Exit Criteria

- All planned test cases executed (95% or greater)
- No open high (P1 or P2) or critical defects.
- All regression tests passed.
- Test summary report approved.

7. Deliverables

- Test Plan
- Test Cases and Test Data
- Test Execution Report
- Defect Reports
- Test Summary Report

8. Risks and Mitigation

- **Risk:** Environment downtime → **Mitigation:** Have a backup environment.
- **Risk:** Unstable build → **Mitigation:** Run smoke tests before full cycle.
- **Risk:** Delayed defect fixes → **Mitigation:** Prioritize based on severity.

9. Automation Strategy

- Automate regression scenarios such as login, transfer funds, ATM transactions, online services and account overview.
- Web UI, use Selenium or Playwright with Page Object Model structure.
- API use Playwright or Postman
- Integrate tests into CI/CD pipeline using Jenkins or GitHub Actions.
