University of Phoenix®

Sample Software Test Plan

Project name: Kentech Banking Application

Purpose of project: To design and deploy a scalable, secure, and high-performing financial web platform that supports modern banking services, ensures secure transactions, and complies with financial industry regulations.

Features To Be Tested/Not To Be Tested

- User registration and authentication (including multi-factor authentication)
- Account creation and management (Checking, Savings, Loans)
- Transaction processing (Deposits, Withdrawals, Payments, Transfers)
- Card management (Activation, Block, Unblock)
- Loan management (Application, Approval, Payments)
- Investment account management
- Insurance policy handling
- Compliance and auditing mechanisms
- Data encryption in transit and at rest
- Performance under load
- Responsive UI and accessibility
- Security mechanisms (e.g., RBAC, audit logging)
- Third-party integration validation: Test connectivity, data exchange, and error handling for critical third-party services (e.g., payment gateways, KYC APIs).
- Disaster recovery and backup restoration (e.g., failover to secondary regions, data restoration from backups)

Features Not To Be Tested:

None. All critical third-party integrations (e.g., payment gateways, external APIs) will be tested in collaboration with vendors to ensure compatibility and reliability. Dedicated integration tests will validate end-to-end functionality.

Reason: This change ensures third-party integrations are included in the testing scope, addressing the risk of untested dependencies.

Testing Pass/Fail Criteria

- Functional Tests: Pass if features operate according to specifications without critical errors
- Security Tests: Pass if no vulnerabilities are detected
- Performance Tests: Pass if the system meets defined performance benchmarks under load
- Compliance Tests: Pass if the system meets regulatory compliance requirements

Testing Approach

Testing will include automated and manual methods covering unit, integration, system, and acceptance tests. The testing strategy employs agile practices with iterative testing cycles, continuous integration and delivery, and regular regression testing.

Testing Cases

1. User registration validation
2. User login/logout (including MFA)
3. Checking account creation
4. Funds transfer between accounts
5. Credit card application and activation
6. Loan application submission and approval
7. Investment portfolio view and update
8. Insurance policy creation and premium calculation
9. Data encryption verification
10. Load test for 50,000 concurrent users, including auto-scaling validation for ECS Fargate
11. Security breach simulation
12. Role-based access validation
13. Audit log integrity verification
14. Transaction processing under heavy load
15. UI responsiveness and accessibility test: Validate WCAG 2.1 compliance using automated tools (Axe, WAVE) and user testing with diverse groups (e.g., users with visual/motor disabilities).
16. Disaster recovery validation: Test backup restoration, failover to secondary AWS region, and recovery time objectives (RTO/RPO).

Testing Materials (Hardware/Software Requirements)

- Hardware: Test servers, workstations, mobile devices, sandbox environments
- Software: Testing suites (e.g., Selenium, JMeter, Postman), database clients, performance testing tools
- Tools: Jira for issue tracking, GitHub for code management, Jenkins for CI/CD, security testing tools (nmap, wireshark, nessus)

Testing Schedule

Testing Activity	Duration	Resource	Comments
Requirement Analysis	3 days	QA Lead	

Testing Activity	Duration	Resource	Comments
Environment Setup	4 days	Infrastructure team	Setup testing infrastructure
Initial Unit Testing	15 days	Development Team	
Comprehensive Integration Testing	20 days	QA Engineers	
Functional and System Testing	12 days	QA team	
Security and Compliance Testing	14 days	Security Analysts	Includes penetration testing, compliance audits (PCI DSS, SOC 2), and continuous vulnerability scanning with tools like OWASP ZAP
Performance StressTesting	10 days	QA Engineers	Test for 50,000 concurrent users, validate auto-scaling, and benchmark response times against industry standards
Accessibility Testing	5 days	End Users / Product Owners	
Final Regression Testing	6 days	End Users / Product Owners	
Disaster Recovery Testing	7 days	QA Engineers, Infrastructure Team	Test backup restoration and regional failover using AWS Backup and multi-region setups
Accessibility Testing	7 days	QA Engineers End Users	Use automated tools (e.g., Axe, WAVE) and validate against WCAG 2.1 standards, include diverse user testing

Risks and Contingencies Matrix

Risk	Probability	Risk Type	Owner	Contingencies/Mitigation Approach
Resource Availability	30%	Project Resources	QA Manager	Testing schedule will be Flexible allocation of testers and timely resource forecasting

Risk	Probability	Risk Type	Owner	Contingencies/Mitigation Approach
Technical Issues with Test Environments	20%	Infrastructure	Infrastructure Lead	Provision backup environments and allocate additional setup time
Insufficient Skills in Testing Team	15%	Project Resources	QA Manager	Conduct targeted training sessions, allocate tasks based on skills