Test Data Management Plan

ParaBank Demo Application

Document Information

Field	Value
Document ID	TDMP-PARABANK-001
Version	1.0
Date	August 30, 2025
Author	Test Data Manager
Reviewed by	Test Lead
Approved by	Test Manager
Status	Active

Document Changes

Version	Date	Author	Changes
1.0	2025-08-30	Test Data Manager	Initial data plan for 200 test cases

---

1. What is Test Data Management?

1.1 Purpose

This document explains how we create, store, update, and protect test data for ParaBank testing. Good test data management helps us run our 200 test cases efficiently and safely.

1.2 Why We Need This Plan

- ✅ **Consistent testing** - same data gives same results
- ✅ **Faster testing** - ready data means faster test runs
- ✅ **Safe testing** - no real customer data used
- ✅ **Easy maintenance** - clear process to update data
- ✅ **Team coordination** - everyone uses same data

1.3 Related Documents

- **Test Case Specification**: TCS-PARABANK-001
- **Master Test Plan**: MTP-PARABANK-002
- **Traceability Matrix**: TM-PARABANK-001

---

2. Types of Test Data We Need

2.1 User Account Data

2.1.1 Standard Test Users (10 users)

Username	Password	Purpose	Account Types	Balance
john	demo	Basic functionality testing	Checking, Savings	$500, $1,000
testuser1	TestPass123!	Registration testing	None (new user)	$0
richuser	RichPass456!	High balance testing	Checking, Savings, Loan	$10,000, $5,000, -$2,000
pooruser	PoorPass789!	Low balance testing	Checking	$5.00
multiuser	MultiPass000!	Multiple accounts	3x Checking, 2x Savings	$100 each
loanuser	LoanPass111!	Loan testing	Checking, Savings, Loan	$1,000, $2,000, -$5,000
billuser	BillPass222!	Bill payment testing	Checking	$2,000
transferuser	TransPass333!	Transfer testing	2x Checking	$500, $300
newuser	NewPass444!	Fresh account testing	Will be created	TBD
adminuser	AdminPass555!	Admin functionality	All types	Varies

2.1.2 User Personal Information

Field	Sample Data	Validation Rules	Usage
First Name	John, Jane, Mike, Sarah	1-50 characters, letters only	Registration tests
Last Name	Smith, Doe, Johnson, Wilson	1-50 characters, letters only	Registration tests
Address	123 Main St, 456 Oak Ave	1-100 characters	Address validation
City	New York, Los Angeles, Chicago	1-50 characters	City validation
State	NY, CA, IL	2 character state code	State validation
Zip Code	12345, 12345-6789	5 or 9 digit format	Zip validation
Phone	555-123-4567, (555) 123-4567	Various phone formats	Phone validation
SSN	123-45-6789	XXX-XX-XXXX format	SSN validation

2.2 Account Data

2.2.1 Bank Accounts

Account Type	Account Number	Customer	Balance	Status
Checking	12345	john	$500.00	Active
Savings	54321	john	$1,000.00	Active
Checking	11111	richuser	$10,000.00	Active
Savings	22222	richuser	$5,000.00	Active
Loan	33333	richuser	-$2,000.00	Active
Checking	99999	pooruser	$5.00	Active

2.2.2 Transaction History

Transaction ID	Type	From Account	To Account	Amount	Date	Status
TXN_001	Transfer	12345	54321	$100.00	2025-08-29	Completed
TXN_002	Bill Payment	12345	Electric Co	$75.50	2025-08-28	Completed
TXN_003	Deposit	-	12345	$200.00	2025-08-27	Completed

2.3 Bill Payment Data

2.3.1 Payee Information

Payee Name	Account Number	Address	Phone	Type
Electric Company	12345-67890	100 Power St, NY 10001	555-POWER	Utility
Water Department	WTR-98765	200 Water Ave, NY 10002	555-WATER	Government
Gas Company	GAS-11111	300 Gas Blvd, NY 10003	555-GAS99	Utility
Credit Card Co	CC-22222	400 Credit Way, NY 10004	555-CREDIT	Financial
Phone Company	PHN-33333	500 Phone Rd, NY 10005	555-PHONE	Telecom

2.3.2 Bill Payment Scenarios

Scenario	Payee	Amount	From Account	Purpose
Standard Bill	Electric Company	$85.50	Checking	Normal payment
Large Bill	Credit Card Co	$500.00	Checking	High amount payment
Small Bill	Water Department	$25.00	Checking	Low amount payment
Insufficient Funds	Gas Company	$1,000.00	$5 account	Error testing

2.4 Loan Data

2.4.1 Loan Applications

Loan Type	Amount	Down Payment	Customer	Purpose
Auto Loan	$15,000	$3,000	loanuser	Car purchase
Personal Loan	$5,000	$0	john	Personal use
Home Loan	$200,000	$40,000	richuser	House purchase

---

3. Data Creation Strategy

3.1 How We Create Test Data

3.1.1 Manual Data Creation

When to use:

- Small amounts of data (under 10 records)
- Special test scenarios
- One-time test cases
- Complex business scenarios

Process:

1. **Plan data needs** - what data does test case need?
2. **Create manually** - enter data through UI or database
3. **Verify data** - check data is correct
4. **Document data** - record what was created
5. **Share with team** - make data available to others

3.1.2 Automated Data Generation

When to use:

- Large amounts of data (over 10 records)
- Regular data refresh
- Performance testing data
- Bulk operations testing

Tools we use:

- **Faker.js** - generate realistic fake data
- **Database scripts** - create data directly in database
- **API calls** - create data through ParaBank API
- **CSV import** - bulk data upload

3.1.3 Data Templates

User Registration Template:

```
json

{
  "firstName": "{{firstName}}",
  "lastName": "{{lastName}}",
  "address": "{{streetAddress}}",
  "city": "{{city}}",
  "state": "{{state}}",
  "zipCode": "{{zipCode}}",
  "phoneNumber": "{{phoneNumber}}",
  "ssn": "{{ssn}}",
  "username": "{{username}}",
  "password": "{{password}}"
}
```

Account Template:

```
json

{
  "accountType": "{{accountType}}",
  "customerId": "{{customerId}}",
  "initialDeposit": "{{amount}}"
}
```

3.2 Data Generation Rules

3.2.1 Realistic Data Rules

- **Names**: Use common first/last names
- **Addresses**: Real street formats but fake addresses
- **Phone numbers**: 555-xxx-xxxx format (fake numbers)
- **SSN**: 123-xx-xxxx format (test numbers only)
- **Amounts**: Realistic banking amounts ($0.01 to $100,000)

3.2.2 Data Consistency Rules

- **Same user** always has same personal information
- **Account numbers** are unique across all accounts
- **Transaction IDs** are unique and sequential
- **Dates** are logical (no future dates for past transactions)
- **Balances** match transaction history

---

4. Data Storage and Organization

4.1 Where We Store Test Data

4.1.1 Database Storage

ParaBank Test Database:

- **Location**: hsqldb://localhost/parabank
- **Access**: Read/write for test team
- **Backup**: Daily automated backup
- **Reset**: Weekly full reset to baseline

4.1.2 File Storage

JSON Data Files:

- **users.json** - user account information
- **accounts.json** - bank account details
- **transactions.json** - transaction history

- **payees.json** - bill payment recipients

CSV Data Files:

- **bulk_users.csv** - large user datasets
- **performance_data.csv** - performance testing data
- **import_accounts.csv** - account bulk import

4.1.3 Configuration Files

Environment configs:

- **dev-data.json** - development environment data
- **test-data.json** - testing environment data
- **staging-data.json** - staging environment data

4.2 Data Organization Structure

```
test-data/
├── users/
│   ├── standard-users.json
│   ├── admin-users.json
│   └── test-users.csv
├── accounts/
│   ├── checking-accounts.json
│   ├── savings-accounts.json
│   └── loan-accounts.json
├── transactions/
│   ├── transfer-history.json
│   ├── bill-payments.json
│   └── loan-payments.json
├── payees/
│   ├── utility-companies.json
│   ├── credit-cards.json
│   └── other-payees.json
└── scripts/
    ├── data-generator.js
    ├── data-cleanup.js
    └── data-validator.js
```

---

5. Data Security and Privacy

5.1 Data Protection Rules

5.1.1 What Data We NEVER Use

- ❌ **Real customer information** - no actual bank data
- ❌ **Real social security numbers** - only fake test SSNs
- ❌ **Real phone numbers** - use 555-xxx-xxxx format
- ❌ **Real addresses** - fictional addresses only
- ❌ **Production database** - never connect to live systems

5.1.2 Test Data Security

- 🔒 **Encrypt sensitive data** in storage
- 🔒 **Limit access** to test team only
- 🔒 **Use VPN** for remote access
- 🔒 **Regular cleanup** of old test data
- 🔒 **Audit data access** - log who uses what data

5.1.3 Data Masking Rules

When we need realistic data:

- **SSN**: 123-45-xxxx (first 5 digits always 123-45)
- **Credit Cards**: 4111-1111-1111-xxxx (test card numbers)
- **Phone**: 555-xxx-xxxx (555 prefix for fake numbers)
- **Email**: testuser@example.com (example.com domain)

5.2 Compliance Requirements

- ✅ **GDPR compliance** - no personal data from EU citizens
- ✅ **Banking regulations** - follow test data guidelines
- ✅ **Company policies** - match internal data policies
- ✅ **Audit requirements** - maintain data trail

---

6. Data Lifecycle Management

6.1 Data Creation Process

6.1.1 New Test Data Creation

Step-by-step process:

1. **Identify need** - which test cases need new data?
2. **Design data** - what fields and values needed?
3. **Create data** - generate using tools or manually
4. **Validate data** - check data meets requirements
5. **Store data** - save in proper location with documentation
6. **Share data** - inform team about new data availability

6.1.2 Data Creation Schedule

Data Type	Creation Frequency	Who Creates	When
Basic Users	Once per project	Test Data Manager	Project start
Performance Data	Weekly	Performance Tester	Before perf tests
New Scenarios	As needed	QA Engineers	When writing new tests
Refresh Data	Weekly	Automated script	Sunday nights

6.2 Data Maintenance

6.2.1 Regular Maintenance Tasks

Daily tasks:

- ✅ **Check data integrity** - verify no corruption
- ✅ **Monitor data usage** - see which data is being used
- ✅ **Clean temporary data** - remove test artifacts

Weekly tasks:

- ✅ **Full data refresh** - reset to baseline state
- ✅ **Backup current data** - save before refresh
- ✅ **Update test data** - add any new requirements
- ✅ **Validate all datasets** - ensure data quality

Monthly tasks:

- ✅ **Review data usage** - remove unused datasets
- ✅ **Update documentation** - keep data docs current
- ✅ **Audit data access** - review who accessed what
- ✅ **Performance analysis** - optimize data storage

6.2.2 Data Refresh Strategy

Why we refresh data:

- Tests change account balances
- New transactions are created
- User accounts get modified
- System state becomes messy

How we refresh:

1. **Stop all testing** - coordinate with team
2. **Backup current state** - save before reset
3. **Run reset script** - restore baseline data
4. **Validate reset** - check all data is correct
5. **Notify team** - inform refresh is complete

---

7. Test Data for Each Test Area

7.1 Authentication Test Data (TC_001-035)

7.1.1 Valid Login Credentials

Username	Password	First Name	Last Name	Purpose
john	demo	John	Smith	Standard login testing
jane	password	Jane	Doe	Alternative valid user
mike	testpass	Mike	Johnson	Third valid user

7.1.2 Invalid Login Data

Username	Password	Expected Error	Test Case
wronguser	demo	Invalid credentials	TC_004
john	wrongpass	Invalid credentials	TC_005
empty	empty	Required field error	TC_007
' OR '1'='1	' OR '1'='1	Security error	TC_016
<script>alert('xss')</script>	normalpass	XSS prevention	TC_019

7.1.3 Security Test Data

Data Type	Test Values	Purpose
SQL Injection	', ", UNION, SELECT, DROP	Test input sanitization
XSS Payloads	<script>, javascript:, <img onerror>	Test script injection prevention
Long Strings	1000+ character strings	Test buffer overflow protection
Special Characters	!@#$%^&*()_+	Test character handling

7.2 Registration Test Data (TC_036-060)

7.2.1 Valid Registration Data Sets

Data Set	First Name	Last Name	Address	City	State	Zip	Phone	SSN
Set 1	Alice	Brown	789 Pine St	Boston	MA	02101	555-111-2222	111-22-3333
Set 2	Bob	Wilson	321 Elm Ave	Austin	TX	73301	555-444-5555	444-55-6666
Set 3	Carol	Davis	654 Oak Blvd	Seattle	WA	98101	555-777-8888	777-88-9999

7.2.2 Invalid Registration Data

Field	Invalid Values	Expected Error	Test Cases
First Name	(empty), 123Numbers, @#$%	Required/Invalid format	TC_039, TC_041
Phone	123, phone, 555-PHONE	Invalid format	TC_049, TC_050
SSN	123, 123456789, ABC-DE-FGHI	Invalid format	TC_052, TC_053
Zip Code	1234, ABCDE, 123456	Invalid format	TC_047, TC_048

7.3 Transaction Test Data (TC_101-155)

7.3.1 Transfer Amounts

Amount	Purpose	Expected Result
$0.01	Minimum amount testing	Should work
$10.00	Standard amount	Should work
$100.00	Normal transfer	Should work
$1,000.00	Large amount	Should work
$999,999.99	Maximum amount	Should work or show limit
$0.00	Zero amount	Should fail
-$50.00	Negative amount	Should fail

7.3.2 Account Combinations

From Account Type	To Account Type	Should Work	Test Cases
Checking	Savings	✅ Yes	TC_101, TC_102
Savings	Checking	✅ Yes	TC_103, TC_104
Checking	Checking	✅ Yes	TC_105
Same Account	Same Account	❌ No	TC_109

7.4 API Test Data (TC_176-200)

7.4.1 API Request Data

Endpoint	Method	Sample Request	Expected Response
/customers/{id}	GET	/customers/12212	Customer JSON object
/accounts	GET	/accounts?customerId=12212	Array of accounts
/transfer	POST	{"fromAccountId":12345,"toAccountId":54321,"amount":50}	Transfer confirmation

7.4.2 API Error Test Data

Scenario	Request	Expected Status	Expected Error
Invalid Customer	/customers/99999	404 Not Found	"Customer not found"
Invalid Account	/accounts/99999	404 Not Found	"Account not found"
Invalid Transfer	Invalid JSON	400 Bad Request	"Invalid request format"

---

8. Data Environment Management

8.1 Environment-Specific Data

8.1.1 Development Environment

- **Purpose**: Initial testing and debugging
- **Data Size**: Small dataset (5 users, 10 accounts)
- **Refresh**: Manual as needed
- **Access**: Development team + QA

8.1.2 Testing Environment

- **Purpose**: Main test execution
- **Data Size**: Full dataset (20 users, 50 accounts)
- **Refresh**: Automated weekly
- **Access**: QA team only

8.1.3 Staging Environment

- **Purpose**: Final validation before production
- **Data Size**: Production-like dataset
- **Refresh**: Manual before major releases
- **Access**: QA team + Business stakeholders

8.2 Data Synchronization

8.2.1 Between Environments

Process:

1. **Export data** from source environment
2. **Transform data** if needed for target environment
3. **Import data** into target environment
4. **Validate data** - check everything copied correctly
5. **Test data** - run smoke tests to verify

8.2.2 Sync Schedule

From Environment	To Environment	Frequency	Who Does It
Development	Testing	Weekly	DevOps Engineer
Testing	Staging	Before releases	Test Data Manager
Baseline	All environments	Monthly	Automated script

---

9. Data Backup and Recovery

9.1 Backup Strategy

9.1.1 What We Backup

- ✅ **User accounts** - all test user information
- ✅ **Bank accounts** - account details and balances
- ✅ **Transaction history** - all test transactions
- ✅ **Configuration data** - app settings and parameters
- ✅ **Test scripts** - data generation and cleanup scripts

9.1.2 Backup Schedule

Backup Type	Frequency	Retention	Storage Location
Daily Backup	Every night at 2 AM	7 days	Local server
Weekly Backup	Sunday at midnight	4 weeks	Cloud storage
Monthly Backup	1st of month	6 months	Archive storage
Release Backup	Before each release	1 year	Long-term archive

9.2 Recovery Procedures

9.2.1 Quick Recovery (Minor Issues)

When to use: Small data corruption, test mistakes **Process**:

1. **Stop current testing** - prevent further issues
2. **Identify problem** - what data is affected?
3. **Restore from daily backup** - get clean data
4. **Validate restoration** - check data is correct
5. **Resume testing** - inform team recovery complete

Time needed: 30 minutes

9.2.2 Full Recovery (Major Issues)

When to use: Database corruption, major system failure **Process**:

1. **Emergency stop** - halt all testing immediately
2. **Assess damage** - understand scope of problem
3. **Contact DevOps** - get technical support
4. **Restore from weekly backup** - clean baseline
5. **Rebuild current data** - recreate recent changes
6. **Full validation** - test all functionality
7. **Team notification** - update schedule if needed

Time needed: 2-4 hours

---

10. Data Quality Assurance

10.1 Data Validation Rules

10.1.1 Format Validation

Data Field	Validation Rule	Error Action
Username	3-50 chars, alphanumeric	Reject and log error
Password	6+ chars, mixed case + numbers	Reject and log error
Phone	(xxx) xxx-xxxx or xxx-xxx-xxxx	Accept both formats
SSN	xxx-xx-xxxx	Reject other formats
Amount	Positive number, 2 decimal places	Round to cents

10.1.2 Business Rule Validation

Business Rule	Check Method	Frequency
Account balances = transaction sum	Automated script	Daily
Unique account numbers	Database constraint	Real-time
Valid customer-account relationships	SQL query	Weekly
Loan balances are negative	Data validation script	Daily

10.2 Data Quality Metrics

10.2.1 Quality Measurements

Metric	Target	Current	How We Measure
Data Accuracy	99.9%	TBD	Compare with expected values
Data Completeness	95%	TBD	Check for missing required fields
Data Consistency	99%	TBD	Cross-check related data
Data Freshness	< 1 week old	TBD	Check creation/update dates

10.2.2 Quality Monitoring

Daily checks:

- Run data validation scripts
- Check for data corruption
- Verify backup completion
- Monitor data access logs

Weekly reports:

- Data quality metrics summary
- Issues found and resolved
- Data usage statistics
- Recommendations for improvements

---

11. Data Access Management

11.1 Who Can Access What

11.1.1 Access Levels

Role	Read Access	Write Access	Delete Access	Admin Access
Test Manager	All data	All data	All data	Yes
Senior QA Engineer	All data	Test data	Test data	Limited
QA Engineer	Assigned data	Assigned data	No	No
Performance Tester	Performance data	Performance data	Performance data	No
Developer	Read only	No	No	No

11.1.2 Access Control Process

Getting access:

1. **Request access** - fill out access request form
2. **Manager approval** - get approval from Test Manager
3. **Account creation** - IT creates data access account
4. **Training** - complete data handling training
5. **Agreement signed** - sign data use agreement

Removing access:

- **Project end** - remove access when project finishes
- **Role change** - update access when role changes
- **Security incident** - immediate access removal
- **Regular review** - quarterly access audit

11.2 Data Usage Tracking

11.2.1 What We Track

- **Who** accessed data (user ID)
- **What** data was accessed (dataset name)
- **When** data was accessed (timestamp)
- **How** data was used (read/write/delete)
- **Why** data was needed (test case ID)

11.2.2 Usage Reports

Weekly usage report:

- Most accessed datasets

- Users with highest data usage
- New data created this week
- Issues or problems found

---

12. Automation and Tools

12.1 Data Generation Tools

12.1.1 Automated Tools

Tool	Purpose	Usage	Maintenance
Faker.js	Generate realistic fake data	User creation, addresses	Update monthly
Custom Scripts	ParaBank-specific data	Account creation, transactions	Update as needed
Database Scripts	Direct database population	Bulk data operations	Review quarterly
API Scripts	Create data through API	Realistic data creation	Update with API changes

12.1.2 Data Generation Scripts

User Generator Script:

```javascript
// Generate test user data
function generateUser() {
  return {
    firstName: faker.name.firstName(),
    lastName: faker.name.lastName(),
    address: faker.address.streetAddress(),
    city: faker.address.city(),
    state: faker.address.stateAbbr(),
    zipCode: faker.address.zipCode(),
    phone: '555-' + faker.phone.phoneNumber('###-####'),
    ssn: '123-45-' + faker.random.number({min: 1000, max: 9999}),
    username: faker.internet.userName(),
    password: faker.internet.password()
  };
}
```

12.2 Data Management Automation

12.2.1 Automated Processes

Daily automation:

- **Data backup** at 2:00 AM
- **Data validation** at 6:00 AM
- **Usage report** at 8:00 AM
- **Cleanup temporary files** at 10:00 PM

Weekly automation:

- **Full data refresh** Sunday at midnight
- **Data quality report** Monday at 9:00 AM
- **Usage analysis** Friday at 5:00 PM

12.2.2 Alert System

Automated alerts for:

- ⚠️ **Data corruption** detected
- ⚠️ **Backup failure** occurred
- ⚠️ **Disk space low** (under 10% free)
- ⚠️ **Data access violation** - unauthorized access attempt
- ⚠️ **Data generation failure** - scripts failed

---

13. Data Requirements by Test Type

13.1 Functional Test Data (160 test cases)

13.1.1 User Account Data Needs

Test Category	Users Needed	Accounts Needed	Special Requirements
Authentication	5 users	Existing accounts	Valid/invalid credentials
Registration	New users	No accounts	Fresh usernames
Account Management	3 users	2-5 accounts each	Different balances
Transactions	4 users	Multiple accounts	Various balance levels
Search	2 users	Accounts with history	Existing transactions

13.1.2 Transaction Data Needs

Test Scenario	Data Required	Volume
Transfer Testing	Account pairs with funds	20 combinations
Bill Pay Testing	Payee list, payment history	10 payees, 50 payments
Loan Testing	Loan applications, approval data	5 loan scenarios
Search Testing	Transaction history	100+ historical transactions

13.2 Performance Test Data (10 test cases)

13.2.1 Load Testing Data

Scenario	Users	Accounts	Transactions	Purpose
Light Load	10 users	20 accounts	100 transactions	Baseline performance
Normal Load	50 users	100 accounts	500 transactions	Typical usage
Heavy Load	100 users	200 accounts	1000 transactions	Peak usage

13.2.2 Stress Testing Data

Test Type	Data Volume	Purpose
Large Reports	10,000 transactions	Report generation speed
Bulk Transfers	100 simultaneous	Concurrent processing
High Frequency	1000 requests/minute	Rate limiting testing

13.3 Security Test Data (25 test cases)

13.3.1 Malicious Input Data

Attack Type	Test Payloads	Target Fields
SQL Injection	', '', UNION SELECT, DROP TABLE	All input fields
XSS	<script>, javascript:, <img onerror>	Text fields
Buffer Overflow	10,000 character strings	All fields
Command Injection	; ls, && dir, | whoami	Command-like fields

---

14. Data Cleanup and Maintenance

14.1 Cleanup Procedures

14.1.1 After Each Test Run

Automated cleanup:

1. **Remove temporary users** created during testing
2. **Reset account balances** to baseline values
3. **Clear session data** from previous runs
4. **Delete uploaded files** from tests
5. **Clean browser cache** for next run

14.1.2 Weekly Deep Cleanup

Manual cleanup:

1. **Review all test data** for inconsistencies
2. **Remove orphaned records** (accounts without users)
3. **Fix data relationships** (customer-account links)
4. **Update baseline data** with new requirements
5. **Archive old data** that's no longer needed

14.2 Data Monitoring

14.2.1 Health Checks

Daily monitoring:

- ✅ **Database connectivity** - can we access data?
- ✅ **Data integrity** - are relationships correct?

- ✅ **Performance** - how fast are data operations?
- ✅ **Storage space** - enough room for new data?

14.2.2 Quality Metrics Dashboard

Metric	Green Zone	Yellow Zone	Red Zone	Action
Data Errors	0-2 per day	3-5 per day	6+ per day	Investigate immediately
Response Time	< 1 second	1-3 seconds	> 3 seconds	Optimize queries
Storage Usage	< 70%	70-85%	> 85%	Clean up or expand
Backup Success	100%	95-99%	< 95%	Fix backup system

---

15. Troubleshooting Guide

15.1 Common Data Issues

15.1.1 Data Not Found

Problem: Test case can't find expected data **Solution steps:**

1. Check if data exists in database
2. Verify data creation scripts ran successfully
3. Check data file paths are correct
4. Confirm test is looking in right place
5. Recreate data if missing

15.1.2 Data Corruption

Problem: Data has wrong values or format **Solution steps:**

1. Stop using corrupted data immediately
2. Identify scope of corruption
3. Restore from most recent clean backup
4. Find root cause of corruption
5. Fix underlying issue
6. Implement prevention measures

15.1.3 Performance Issues

Problem: Data operations are too slow **Solution steps:**

1. Check database performance
2. Analyze slow queries
3. Optimize data indexes
4. Reduce data volume if possible
5. Consider data archiving

15.2 Emergency Procedures

15.2.1 Complete Data Loss

If all test data is lost:

1. **Don't panic** - we have backups
2. **Stop all testing** - prevent further issues
3. **Contact DevOps** - get technical help
4. **Restore from backup** - use most recent backup
5. **Validate restoration** - check everything works
6. **Rebuild recent data** - recreate any new data
7. **Resume testing** - careful restart
8. **Document incident** - learn from what happened

Recovery time: 2-4 hours maximum

---

16. Roles and Responsibilities

16.1 Data Management Team

Role	Person	Responsibilities	Time Allocation
Test Data Manager	Sarah Chen	Overall data strategy, quality, compliance	50%
Senior QA Engineer	Mike Rodriguez	Data creation scripts, automation	25%
QA Engineers	Lisa, Tom, Alex	Daily data usage, issue reporting	10% each
DevOps Engineer	Emma Davis	Infrastructure, backups, security	20%

16.2 Daily Responsibilities

Test Data Manager:

- Monitor data quality metrics
- Approve new data requests
- Resolve data issues
- Update data documentation

QA Engineers:

- Use data properly in tests
- Report data problems immediately
- Follow data security guidelines
- Clean up after testing

DevOps Engineer:

- Maintain data infrastructure
- Monitor backup systems
- Handle security and access
- Support data recovery if needed

Document Approval

Role	Name	Signature	Date
Test Data Manager	Sarah Chen	_____	_____
Test Lead	Test Lead	_____	_____
Test Manager	Test Manager	_____	_____

This plan will be updated when new data requirements are identified or when data management processes change.