IEEE Standard 829-1983 Software Test Documentation

BiasXplorer-Mini Testing Plan

Test Plan Identifier

Document ID: BXM-TEST-PLAN-2025-001
Version: 1.0
Date: November 18, 2025
Project: BiasXplorer-Mini
Repository: MadhuSudhanAV03/BiasXplorer-Mini
Branch: main

1. Introduction

1.1 Summary of Items and Features to be Tested

BiasXplorer-Mini is a full-stack web application designed to detect and correct bias in datasets. The system consists of:

Backend Components (Flask API):

- File upload and validation system
- Data preprocessing engine (NaN/duplicate removal)
- Column type classification (categorical vs continuous)
- Bias detection service (categorical imbalance)
- Skewness detection service (continuous distribution analysis)
- Bias correction service (SMOTE, over/undersampling, reweighting)
- Skewness correction service (log, sqrt, Box-Cox, Yeo-Johnson, quantile transformations)
- Visualization generation (Plotly charts)
- PDF report generation (ReportLab)
- File management and download services

Frontend Components (React/Vite):

- File upload interface
- Dataset preview component
- Data preprocessing interface
- Column selector (target column selection)
- Feature selector (column type classification)
- Bias detection display
- Unified bias fix interface
- Skewness fix sandbox
- Visualization display (Plotly integration)

- Report generation and download
- Multi-step dashboard workflow
- Persisted state management

Key Features:

1. CSV/Excel file upload (max 10MB)
2. Automated data cleaning
3. Categorical bias detection with severity levels (Low/Moderate/Severe)
4. Continuous skewness detection with interpretation
5. Multiple correction algorithms
6. Before/after visualizations
7. Comprehensive PDF reporting
8. Downloadable corrected datasets

1.2 Need for and History of Each Item

Backend Testing Need: The backend handles critical data transformations, statistical calculations, and file operations. Errors in bias detection or correction could lead to invalid scientific conclusions.

Frontend Testing Need: The frontend orchestrates a 7-step workflow with persistent state. Navigation errors or state corruption could lose user work.

Development History:

- October 1, 2025: Initial signup/login functionality completed
- October 1, 2025: Column classification feature completed
- November 18, 2025: Full bias detection and correction system operational

1.3 References to Related Documents

- **Project Documentation:** `PROJECT_EXPLANATION.md`
- **API Mapping:** `FRONTEND_BACKEND_API_MAPPING.md`
- **Component Datasheet:** `APPENDIX_A_COMPONENT_DATASHEET.md`
- **Backend Calculation Migration:** `BACKEND_CALCULATION_MIGRATION.md`
- **README:** `README.md`
- **Requirements Specification:** `backend/requirements.txt`, `frontend/package.json`
- **Design Specifications:**
  - Backend: `backend/app.py` (Flask application factory)
  - Frontend: `frontend/src/App.jsx` (Router configuration)

1.4 References to Lower Level Test Plans

- **Unit Test Plan:** BXM-UNIT-TEST-2025-001 (to be created)
- **Integration Test Plan:** BXM-INT-TEST-2025-001 (to be created)
- **API Test Plan:** BXM-API-TEST-2025-001 (to be created)
- **UI/UX Test Plan:** BXM-UI-TEST-2025-001 (to be created)

---

2. Test Items

2.1 Backend Test Items

Version: Current (main branch, November 18, 2025)

Items:

1. **app.py** (v1.0)
    - Application factory pattern
    - Blueprint registration
    - CORS configuration
    - Custom JSON provider for NaN handling
    - Health check endpoints
2. **resources/upload_routes.py**
    - File upload endpoint `/api/upload`
    - File validation (extension, size limits)
    - File storage to uploads directory
3. **resources/preprocess_routes.py**
    - Data cleaning endpoint `/api/preprocess`
    - NaN removal
    - Duplicate removal
    - Dataset shape reporting
4. **resources/select_routes.py**
    - Column type storage endpoint `/api/column-types`
    - Feature selection endpoint `/api/features`
    - Dataset preview endpoint `/api/preview`
5. **resources/bias_routes.py**
    - Bias detection endpoint `/api/bias/detect`
    - Bias correction endpoint `/api/bias/fix`
    - Skewness detection endpoint `/api/skewness/detect`
    - Skewness correction endpoint `/api/skewness/fix`
6. **resources/report_routes.py**
    - Visualization generation endpoints
    - PDF report generation endpoint
    - File download endpoints
7. **services/bias_detection_service.py**
    - Class imbalance detection algorithm
    - Severity classification (Low/Moderate/Severe)
    - Distribution calculation

8. **services/bias_correction_service.py**
    - SMOTE implementation
    - Random oversampling
    - Random undersampling
    - Class reweighting
9. **services/skewness_detection_service.py**
    - Skewness calculation (scipy.stats)
    - Skewness interpretation
10. **services/skewness_correction_service.py**
    - Log transformation
    - Square root transformation
    - Power transformations
    - Yeo-Johnson transformation
    - Quantile transformer
11. **services/visualization_service.py**
    - Plotly chart generation
    - Before/after comparison charts
12. **services/file_service.py**
    - CSV/Excel reading
    - Dataset writing
    - File path validation
13. **utils/data_stats.py**
    - Statistical calculations
    - Skewness computation
14. **utils/validators/** (file_validator.py, path_validator.py)
    - Input validation
    - Security checks
15. **utils/transformers/** (categorical.py, continuous.py)
    - Transformation implementations
    - Algorithm parameters

Transmittal Media: Git repository (GitHub)

2.2 Frontend Test Items

Version: Current (main branch, November 18, 2025)

Items:

1. **src/main.jsx** - React entry point
2. **src/App.jsx** - Router configuration
3. **src/pages/Dashboard.jsx** - 7-step workflow orchestrator
4. **src/pages/ReportPage.jsx** - Final report display
5. **src/pages/Home.jsx** - Landing page
6. **src/components/FileUpload.jsx** - File upload interface
7. **src/components/DatasetPreview.jsx** - Data table display
8. **src/components/Preprocess.jsx** - Cleaning interface
9. **src/components/ColumnSelector.jsx** - Target column selection
10. **src/components/FeatureSelector.jsx** - Column type classification
11. **src/components/BiasDetection.jsx** - Bias results display
12. **src/components/UnifiedBiasFix.jsx** - Combined fix interface
13. **src/components/BiasFixSandbox.jsx** - Categorical correction UI
14. **src/components/SkewnessFixSandbox.jsx** - Continuous correction UI
15. **src/components/Visualization.jsx** - Chart display
16. **src/hooks/usePersistedState.js** - LocalStorage state management

Transmittal Media: Git repository (GitHub)

2.3 Related Bug Reports

- To be tracked in GitHub Issues: `https://github.com/MadhuSudhanAV03/BiasXplorer-Mini/issues`

2.4 Features Not to Be Tested

The following items are explicitly excluded from this test plan:

1. **Authentication/Authorization** - Login/signup functionality exists but is not integrated with main workflow
2. **Database Operations** - No persistent database (uses in-memory store)
3. **Email Notifications** - Not implemented
4. **Real-time Collaboration** - Single-user application
5. **Mobile Responsive Design** - Desktop-only application
6. **Internationalization (i18n)** - English only
7. **Accessibility Features** - Not currently prioritized
8. **Browser Compatibility** - Testing limited to Chrome/Edge (modern browsers)

---

3. Features to Be Tested

3.1 Core Functionality Features

Feature ID	Feature Name	Component	Test Design Spec
F-001	File Upload	FileUpload.jsx, upload_routes.py	TD-001
F-002	File Validation	upload_routes.py, file_validator.py	TD-002
F-003	Dataset Preview	DatasetPreview.jsx, select_routes.py	TD-003

Feature ID	Feature Name	Component	Test Design Spec
F-004	Data Preprocessing	Preprocess.jsx, preprocess_routes.py	TD-004
F-005	Column Type Classification	FeatureSelector.jsx, select_routes.py	TD-005
F-006	Categorical Bias Detection	BiasDetection.jsx, bias_detection_service.py	TD-006
F-007	Skewness Detection	BiasDetection.jsx, skewness_detection_service.py	TD-007
F-008	SMOTE Correction	BiasFixSandbox.jsx, categorical.py	TD-008
F-009	Oversampling Correction	BiasFixSandbox.jsx, categorical.py	TD-009
F-010	Undersampling Correction	BiasFixSandbox.jsx, categorical.py	TD-010
F-011	Log Transformation	SkewnessFixSandbox.jsx, continuous.py	TD-011
F-012	Square Root Transformation	SkewnessFixSandbox.jsx, continuous.py	TD-012
F-013	Yeo-Johnson Transformation	SkewnessFixSandbox.jsx, continuous.py	TD-013
F-014	Quantile Transformation	SkewnessFixSandbox.jsx, continuous.py	TD-014
F-015	Visualization Generation	Visualization.jsx, visualization_service.py	TD-015
F-016	PDF Report Generation	ReportPage.jsx, report_routes.py	TD-016
F-017	File Download	ReportPage.jsx, report_routes.py	TD-017
F-018	State Persistence	usePersistedState.js	TD-018
F-019	Multi-Step Navigation	Dashboard.jsx	TD-019

3.2 API Endpoint Features

Endpoint	Method	Feature	Test Spec
/api/upload	POST	File upload	TD-020
/api/preview	POST	Dataset preview	TD-021
/api/preprocess	POST	Data cleaning	TD-022
/api/column-types	POST	Save column types	TD-023
/api/features	POST	Feature selection	TD-024
/api/bias/detect	POST	Detect categorical bias	TD-025
/api/bias/fix	POST	Fix categorical bias	TD-026

Endpoint	Method	Feature	Test Spec
/api/skewness/detect	POST	Detect skewness	TD-027
/api/skewness/fix	POST	Fix skewness	TD-028
/api/bias/visualize	POST	Generate bias charts	TD-029
/api/skewness/visualize	POST	Generate skewness charts	TD-030
/api/reports/generate	POST	Generate PDF report	TD-031
/api/corrected/download/	GET	Download corrected file	TD-032

3.3 Integration Features

Integration	Components	Test Spec
Frontend-Backend API	All components	TD-033
File Upload to Processing	FileUpload → Preprocess	TD-034
Detection to Correction	BiasDetection → UnifiedBiasFix	TD-035
Correction to Visualization	UnifiedBiasFix → Visualization	TD-036
State Persistence	Dashboard ↔ LocalStorage	TD-037

4. Features Not to Be Tested

4.1 Excluded Features

1. **User Authentication System**
    - Reason: Not integrated with main workflow; exists as separate module
2. **Database Persistence**
    - Reason: Application uses file-based storage and in-memory state
3. **Concurrent User Sessions**
    - Reason: Single-user application design
4. **Real-time Updates**
    - Reason: Synchronous processing model
5. **Mobile/Tablet Interfaces**
    - Reason: Desktop-only target platform
6. **Legacy Browser Support (IE11, older Safari)**
    - Reason: Modern browser requirement (ES6+)

7. **Network Failure Recovery**
    - Reason: Local development environment assumption
8. **Large Dataset Performance (>100MB)**
    - Reason: 10MB file size limit enforced
9. **Multi-language Support**
    - Reason: English-only interface
10. **Advanced Statistical Validation**
    - Reason: Assumes user has domain knowledge for result interpretation

---

5. Approach

5.1 Overall Testing Strategy

The testing approach follows a **multi-layered strategy**:

1. **Unit Testing:** Individual function/method testing
2. **Integration Testing:** API endpoint and component interaction testing
3. **System Testing:** End-to-end workflow testing
4. **Acceptance Testing:** User scenario validation

5.2 Testing by Feature Groups

5.2.1 File Management Testing

Components: upload_routes.py, file_service.py, file_validator.py, FileUpload.jsx

Approach:

- **Techniques:** Boundary value analysis, equivalence partitioning
- **Tools:** pytest, pytest-flask, Jest/React Testing Library
- **Activities:**
    - Test valid file uploads (CSV, Excel)
    - Test invalid file types (PDF, ZIP, TXT)
    - Test file size limits (0 bytes, 5MB, 10MB, 15MB)
    - Test malformed files
    - Test path traversal attacks
    - Test concurrent upload attempts

Comprehensiveness:

- All supported file types tested
- All validation rules verified
- Security boundary testing completed

Traceability:

- Maps to requirements in `FRONTEND_BACKEND_API_MAPPING.md` sections 1-2

5.2.2 Data Processing Testing

Components: preprocess_routes.py, select_routes.py, data_stats.py, Preprocess.jsx

Approach:

- **Techniques:** Decision table testing, state transition testing
- **Tools:** pytest, pandas.testing, Jest
- **Activities:**
  - Test NaN removal (all NaN, partial NaN, no NaN)
  - Test duplicate removal
  - Test empty dataset handling
  - Test single-column datasets
  - Test mixed data types
  - Test Unicode/special characters

Comprehensiveness:

- All preprocessing operations tested
- Edge cases (empty, single row, all duplicates) covered

Traceability:

- Maps to requirements in `PROJECT_EXPLANATION.md` Section 5.2

5.2.3 Bias Detection Testing

Components: bias_detection_service.py, skewness_detection_service.py, BiasDetection.jsx

Approach:

- **Techniques:** Statistical validation, classification testing
- **Tools:** pytest, scipy.stats verification, Jest
- **Activities:**
  - Test categorical imbalance detection (50/50, 90/10, 99/1 ratios)
  - Test severity classification (Low/Moderate/Severe)
  - Test skewness calculation accuracy
  - Test multi-class imbalance
  - Test single-class datasets
  - Test missing value handling

Statistical Validation:

- Verify imbalance ratio calculations against manual calculations
- Verify skewness values against scipy.stats.skew()
- Verify severity thresholds (0.5 for Low, 0.2 for Moderate)

Comprehensiveness:

- All severity levels tested
- All skewness ranges tested (-5 to +5)
- Boundary conditions verified

Traceability:

- Maps to `bias_detection_service.py` lines 10-65
- Maps to `skewness_detection_service.py` lines 10-85

5.2.4 Bias Correction Testing

Components: bias_correction_service.py, skewness_correction_service.py, categorical.py, continuous.py, UnifiedBiasFix.jsx

Approach:

- **Techniques:** Algorithm validation, statistical hypothesis testing
- **Tools:** pytest, scikit-learn verification, imbalanced-learn validation
- **Activities:**
  - Test SMOTE synthetic sample generation
  - Test SMOTE-NC for mixed data types
  - Test oversampling sample counts
  - Test undersampling sample counts
  - Test all transformation methods (log, sqrt, Yeo-Johnson, etc.)
  - Test transformation reversibility where applicable
  - Validate before/after distributions

Algorithm Validation:

- Verify SMOTE generates K-nearest neighbor interpolated samples
- Verify class balance after correction
- Verify skewness reduction after transformation

Comprehensiveness:

- All correction methods tested
- All transformation types tested
- Edge cases (already balanced, extreme skew) tested

Traceability:

- Maps to `categorical.py` lines 1-189
- Maps to `continuous.py` lines 1-103

5.2.5 Visualization Testing

Components: visualization_service.py, Visualization.jsx, ReportPage.jsx

Approach:

- **Techniques:** Visual regression testing, chart validation
- **Tools:** pytest, Plotly testing utilities, Jest, Puppeteer
- **Activities:**
  - Test chart generation for all data types
  - Test before/after comparison charts
  - Test chart data accuracy
  - Test chart rendering in browser
  - Test interactive features (zoom, pan, hover)
  - Test export functionality

Comprehensiveness:

- All chart types tested (bar, histogram, box plot)
- All data scenarios tested

Traceability:

- Maps to `visualization_service.py`

5.2.6 Workflow Integration Testing

Components: Dashboard.jsx, usePersistedState.js

Approach:

- **Techniques:** State transition testing, scenario-based testing
- **Tools:** Jest, React Testing Library, Cypress (E2E)
- **Activities:**
  - Test 7-step workflow completion
  - Test forward/backward navigation
  - Test state persistence across steps
  - Test browser refresh recovery
  - Test LocalStorage quota handling
  - Test workflow interruption recovery

Comprehensiveness:

- All navigation paths tested
- All state transitions verified
- Error recovery scenarios tested

Traceability:

- Maps to `Dashboard.jsx` lines 1-877

5.3 Minimum Degree of Comprehensiveness

Code Coverage Targets:

- Unit tests: 80% line coverage, 70% branch coverage
- Integration tests: 90% API endpoint coverage

- E2E tests: 100% critical path coverage

Comprehensiveness Measurement:

- **Techniques:**
  - pytest-cov for Python coverage
  - Istanbul/nyc for JavaScript coverage
  - Manual feature checklist verification

Completion Criteria:

- All test cases executed
- All critical defects resolved
- Coverage targets achieved
- Performance benchmarks met

5.4 Requirement Traceability

All features traced to:

- `PROJECT_EXPLANATION.md` requirements
- `FRONTEND_BACKEND_API_MAPPING.md` API specifications
- User stories (to be documented)

Traceability Matrix: See Appendix A

5.5 Testing Constraints

1. **Test Item Availability:**
   - Datasets available: Sample_Dataset.csv, bias_continuous_dataset.csv
   - Test datasets to be created for edge cases
2. **Resource Availability:**
   - Testing environment: Local development (localhost:5000, localhost:5173)
   - CI/CD: GitHub Actions (to be configured)
3. **Deadlines:**
   - Unit tests: Week 1
   - Integration tests: Week 2
   - System tests: Week 3
   - Acceptance tests: Week 4
4. **Technical Constraints:**
   - Python version: 3.9+
   - Node.js version: 18+
   - Browser: Chrome 90+, Edge 90+

---

6. Item Pass/Fail Criteria

6.1 Unit Test Pass Criteria

A unit test passes if:

- Function returns expected output for given input
- All assertions pass
- No exceptions raised (unless testing error handling)
- Execution time < 1 second per test

A unit test fails if:

- Assertion fails
- Unexpected exception raised
- Timeout exceeded
- Memory leak detected

6.2 Integration Test Pass Criteria

An integration test passes if:

- API returns expected HTTP status code
- Response JSON matches schema
- Database/file state correctly updated
- No server errors logged
- Response time < 5 seconds

An integration test fails if:

- Wrong HTTP status returned
- Response schema mismatch
- Data corruption occurs
- Server crash/500 error
- Timeout exceeded

6.3 System Test Pass Criteria

A system test passes if:

- End-to-end workflow completes successfully
- All UI elements render correctly
- Data flows correctly through all components
- Final output matches expected results
- No console errors

A system test fails if:

- Workflow cannot complete
- UI breaks/freezes
- Data loss occurs

- Incorrect final output
- Critical console errors

6.4 Specific Feature Pass Criteria

Feature	Pass Criteria
File Upload	File saved to uploads/ directory with correct name
Bias Detection	Severity level correctly assigned based on imbalance ratio
SMOTE Correction	Minority class count increases, balanced within 5% of majority
Skewness Correction	Absolute skewness reduced by at least 50%
Visualization	Chart renders with correct data points, no rendering errors
PDF Generation	PDF file created, all sections present, no corruption
State Persistence	State restored correctly after page refresh

6.5 Performance Pass Criteria

Operation	Maximum Time
File upload (10MB)	5 seconds
Bias detection	3 seconds
SMOTE correction (10K rows)	10 seconds
Visualization generation	2 seconds
PDF generation	8 seconds
Page load	2 seconds

6.6 Security Pass Criteria

- No path traversal vulnerabilities
- No SQL injection vulnerabilities (N/A - no database)
- No XSS vulnerabilities
- File type validation enforced
- File size limits enforced
- CORS properly configured

---

7. Suspension Criteria and Resumption Requirements

7.1 Suspension Criteria

Testing will be **suspended** if any of the following occur:

1. **Critical Defect Discovered:**

- Application crashes on startup
- Data corruption in core operations
- Complete loss of functionality in critical path
- Security vulnerability (path traversal, XSS)

2. **Environment Failure:**

   - Test environment unavailable for >4 hours
   - Database/file system corruption
   - Network connectivity lost

3. **Build Failure:**

   - Code does not compile/build
   - Dependencies cannot be installed
   - Version conflicts unresolvable

4. **Resource Unavailability:**

   - Test data not available
   - Required test tools not accessible
   - Key personnel unavailable (>2 days)

5. **Requirement Changes:**

   - Major requirement changes requiring test redesign
   - Feature scope significantly altered

7.2 Resumption Requirements

Testing can **resume** when:

1. **After Critical Defect:**

   - Defect fixed and verified
   - Code review completed
   - Regression test suite executed
   - All previously passing tests re-run

2. **After Environment Failure:**

   - Environment restored and verified
   - Test data restored/recreated
   - Connectivity verified
   - Smoke tests passed

3. **After Build Failure:**

   - Build successful
   - Dependencies resolved
   - Smoke tests passed

4. **After Resource Unavailability:**
    - Resources available
    - Test readiness verified
5. **After Requirement Changes:**
    - Test cases updated
    - Test plan reviewed and approved
    - Impact analysis completed

7.3 Re-execution Requirements

Activities to redo after suspension:

- All failed tests
- All tests dependent on fixed components
- Regression test suite (subset based on impact analysis)
- Sanity test suite (full)

---

8. Test Deliverables

8.1 Planning Documents

- ☑ **Test Plan** (this document)
- ☐ **Test Design Specifications** (TD-001 through TD-037)
- ☐ **Test Case Specifications** (TC-001 through TC-XXX)
- ☐ **Test Procedure Specifications** (TP-001 through TP-XXX)

8.2 Execution Documents

- ☐ **Test Item Transmittal Reports** (as features are delivered)
- ☐ **Test Logs** (automated test run logs)
- ☐ **Test Incident Reports** (bug reports linked to GitHub Issues)
- ☐ **Test Summary Reports** (daily/weekly status)

8.3 Code and Data

- ☐ **Test Scripts:**
    - Unit test suite (pytest files)
    - Integration test suite (pytest-flask files)
    - Frontend test suite (Jest/React Testing Library)
    - E2E test suite (Cypress/Playwright)
- ☐ **Test Data:**
    - `test_data/valid_datasets/` (CSV/Excel samples)
    - `test_data/invalid_datasets/` (malformed files)
    - `test_data/edge_cases/` (empty, single-row, all-NaN files)

    - `test_data/expected_outputs/` (golden results)

- ☐ **Test Input Data:**
    - API request payloads (JSON files)
    - File upload samples
    - Mock data generators
- ☐ **Test Output Data:**
    - Expected API responses (JSON files)
    - Expected visualizations (image baselines)
    - Expected PDF reports (PDF samples)

8.4 Test Tools

- ☐ **Automated Test Framework:**
    - pytest (backend)
    - pytest-flask (API testing)
    - pytest-cov (coverage)
    - Jest (frontend)
    - React Testing Library (component testing)
    - Cypress or Playwright (E2E)
- ☐ **Performance Testing:**
    - Locust or Apache JMeter (load testing)
- ☐ **Code Quality:**
    - pylint, flake8 (Python linting)
    - ESLint (JavaScript linting)
    - SonarQube (code quality metrics)
- ☐ **CI/CD Integration:**
    - GitHub Actions workflows
    - Test result dashboards

8.5 Reports

- ☐ **Daily Test Execution Reports**
- ☐ **Weekly Test Summary Reports**
- ☐ **Final Test Summary Report**
- ☐ **Code Coverage Reports**
- ☐ **Performance Test Reports**
- ☐ **Defect Density Reports**

---

9. Testing Tasks

9.1 Planning Phase

Task ID	Task	Dependencies	Owner	Duration
T-001	Review requirements	None	QA Lead	2 days
T-002	Create test plan	T-001	QA Lead	3 days
T-003	Review test plan	T-002	Stakeholders	1 day
T-004	Approve test plan	T-003	PM	1 day

9.2 Design Phase

Task ID	Task	Dependencies	Owner	Duration
T-005	Design unit test cases	T-004	QA Engineers	5 days
T-006	Design integration test cases	T-004	QA Engineers	5 days
T-007	Design system test cases	T-005, T-006	QA Engineers	3 days
T-008	Create test data	T-004	QA Engineers	3 days
T-009	Review test designs	T-007	QA Lead	2 days

9.3 Implementation Phase

Task ID	Task	Dependencies	Owner	Duration
T-010	Set up test environment	T-004	DevOps	2 days
T-011	Install test tools	T-010	QA Engineers	1 day
T-012	Implement unit tests	T-005, T-011	QA Engineers	10 days
T-013	Implement integration tests	T-006, T-011	QA Engineers	8 days
T-014	Implement system tests	T-007, T-011	QA Engineers	5 days
T-015	Configure CI/CD pipeline	T-010	DevOps	3 days

9.4 Execution Phase

Task ID	Task	Dependencies	Owner	Duration
T-016	Execute unit tests	T-012	QA Engineers	3 days
T-017	Execute integration tests	T-013	QA Engineers	3 days
T-018	Execute system tests	T-014	QA Engineers	5 days
T-019	Execute performance tests	T-018	QA Engineers	2 days
T-020	Execute security tests	T-018	Security Tester	2 days
T-021	Log defects	T-016-T-020	QA Engineers	Ongoing

9.5 Evaluation Phase

Task ID	Task	Dependencies	Owner	Duration
T-022	Analyze test results	T-016-T-020	QA Lead	2 days
T-023	Generate coverage reports	T-022	QA Engineers	1 day
T-024	Verify defect fixes	T-021	QA Engineers	3 days
T-025	Re-test after fixes	T-024	QA Engineers	2 days
T-026	Create final test report	T-025	QA Lead	2 days
T-027	Test sign-off	T-026	PM, Stakeholders	1 day

9.6 Special Skills Required

Task	Skill Required	Justification
T-012	Python testing (pytest)	Backend unit tests
T-013	API testing (REST)	Integration tests
T-014	React testing (Jest)	Frontend component tests
T-018	E2E testing (Cypress)	System tests
T-019	Performance testing	Load testing
T-020	Security testing	Vulnerability assessment
T-008	Statistical knowledge	Bias/skewness test data creation
T-022	Data analysis	Test metrics analysis

10. Environmental Needs

10.1 Development/Test Environment

Hardware:

- CPU: 4+ cores (Intel i5/AMD Ryzen 5 or better)
- RAM: 8GB minimum, 16GB recommended
- Disk: 20GB free space (SSD preferred)
- Network: Broadband internet (for dependency installation)

Software:

- **Operating System:** Windows 10/11, macOS 11+, or Linux (Ubuntu 20.04+)
- **Python:** 3.9 or higher
- **Node.js:** 18.x or higher
- **npm:** 9.x or higher
- **Git:** 2.30 or higher

Backend Environment:

- Flask 3.x
- Virtual environment (venv or conda)
- Python dependencies: `backend/requirements.txt`
  - flask-smorest
  - pandas
  - numpy
  - scikit-learn
  - imbalanced-learn
  - scipy
  - plotly
  - reportlab

Frontend Environment:

- React 19.1
- Vite 7.1
- Node modules: `frontend/package.json`
  - axios
  - react-router-dom
  - plotly.js-dist
  - tailwindcss

Browser Requirements:

- Chrome 90+ or Edge 90+ (primary)
- Firefox 88+ (secondary)
- Safari 14+ (secondary)

10.2 Test Environment Configuration

Directories:

```
BiasXplorer-Mini/
├── backend/
│   ├── uploads/        # Test upload directory
│   ├── corrected/      # Test output directory
│   └── test_reports/   # Test PDF reports
├── frontend/
│   └── coverage/       # Frontend coverage reports
└── test_data/
    ├── valid_datasets/
    ├── invalid_datasets/
    ├── edge_cases/
    └── expected_outputs/
```

Environment Variables:

- `FLASK_APP=backend/app.py`
- `FLASK_ENV=development`
- `FRONTEND_URL=http://localhost:5173`
- `BACKEND_URL=http://localhost:5000`

Network Configuration:

- Backend server: `http://localhost:5000`
- Frontend dev server: `http://localhost:5173`
- CORS enabled for localhost:5173

10.3 Security Requirements

Level of Security: Development/Test environment (low security)

Security Measures:

- File upload validation (extension, size)
- Path traversal prevention
- Input sanitization for API endpoints
- CORS restricted to localhost

Not Required in Test Environment:

- HTTPS/SSL
- User authentication (unless testing auth module)
- Database encryption
- API rate limiting

10.4 Special Test Tools

Tool	Purpose	Source
pytest	Backend unit testing	`pip install pytest`
pytest-flask	Flask API testing	`pip install pytest-flask`
pytest-cov	Code coverage	`pip install pytest-cov`
Jest	Frontend unit testing	npm (included in package.json)
React Testing Library	Component testing	npm (included in package.json)
Cypress	E2E testing	`npm install -D cypress`
Locust	Load testing	`pip install locust`
pandas.testing	DataFrame assertions	Included with pandas
unittest.mock	Mocking	Python standard library

10.5 Other Testing Needs

Test Data Generation:

- Python script to generate synthetic datasets with known bias
- CSV files with controlled skewness values
- Malformed file samples (corrupted CSV, wrong encoding)

Visualization Baseline Images:

- Pre-generated Plotly charts for visual regression testing
- Screenshot automation for chart comparison

Performance Monitoring:

- CPU/memory profiling tools (cProfile, memory_profiler)
- Network request timing (Chrome DevTools)

10.6 Source for Needs Not Currently Available

Need	Current Status	Source	ETA
CI/CD Pipeline	Not configured	GitHub Actions setup	Week 2
E2E Test Suite	Not implemented	Cypress installation	Week 3
Load Testing Setup	Not available	Locust configuration	Week 4
Visual Regression Tool	Not available	Percy.io or BackstopJS	Week 5
Test Data Repository	Partial	Create test_data/ directory	Week 1

---

11. Responsibilities

11.1 Test Management

QA Lead:

- Overall test plan coordination
- Test strategy definition
- Resource allocation
- Progress tracking
- Stakeholder communication
- Final test report sign-off

11.2 Test Design

Senior QA Engineer:

- Test case design (unit, integration)
- Test data design
- Test automation framework setup
- Code review of test scripts

QA Engineer:

- Test case implementation
- Test script development
- Test execution
- Defect logging

11.3 Test Execution

Backend QA Engineer:

- Execute backend unit tests
- Execute API integration tests
- Execute performance tests
- Monitor backend test coverage

Frontend QA Engineer:

- Execute frontend component tests
- Execute E2E tests
- Execute UI/UX tests
- Monitor frontend test coverage

11.4 Environment Setup

DevOps Engineer:

- Test environment setup
- CI/CD pipeline configuration
- Test tool installation
- Infrastructure monitoring

11.5 Defect Management

QA Engineers:

- Log defects in GitHub Issues
- Verify defect fixes
- Execute regression tests
- Update defect status

Development Team:

- Fix defects
- Provide root cause analysis
- Update documentation

11.6 Test Item Provision

Development Team:

- Provide code for testing
- Provide API documentation

- Provide unit test coverage
- Address test blockers

Data Team:

- Provide sample datasets
- Validate statistical correctness
- Review bias detection logic

11.7 Environmental Needs Provision

DevOps Engineer:

- Provide test servers (if cloud-based)
- Configure network access
- Install required software

QA Lead:

- Procure test tools (if licensed)
- Manage test data storage

11.8 Sign-off and Approval

Project Manager:

- Approve test plan
- Approve test schedule
- Approve resource allocation
- Final project sign-off

Product Owner:

- Approve acceptance criteria
- Review test results
- Accept final product

QA Lead:

- Certify test completion
- Certify quality metrics met

---

12. Staffing and Training Needs

12.1 Staffing Requirements

Role	Skill Level	Count	Availability
QA Lead	Senior (5+ years)	1	Full-time
Senior QA Engineer	Senior (3+ years)	1	Full-time

Role	Skill Level	Count	Availability
Backend QA Engineer	Mid-level (2+ years)	2	Full-time
Frontend QA Engineer	Mid-level (2+ years)	2	Full-time
Performance Tester	Mid-level (2+ years)	1	Part-time
Security Tester	Senior (3+ years)	1	Part-time
DevOps Engineer	Mid-level (2+ years)	1	Part-time

Total: 9 resources (6 full-time, 3 part-time)

12.2 Skill Requirements by Role

QA Lead:

- Test planning and strategy
- Risk management
- Stakeholder communication
- Team leadership
- Agile/Scrum methodologies

Senior QA Engineer:

- Advanced test automation (Python, JavaScript)
- API testing (REST, JSON)
- Performance testing
- Test framework design
- Mentoring junior testers

Backend QA Engineer:

- Python testing (pytest, unittest)
- API testing (Postman, pytest-flask)
- SQL/database testing (if applicable)
- Statistical knowledge (bias, skewness concepts)
- Linux/command line proficiency

Frontend QA Engineer:

- JavaScript testing (Jest, React Testing Library)
- E2E testing (Cypress, Playwright)
- Browser DevTools proficiency
- UI/UX testing principles
- CSS/HTML knowledge

Performance Tester:

- Load testing tools (Locust, JMeter)
- Performance metrics analysis

- Profiling tools (cProfile, Chrome DevTools)
- System monitoring

Security Tester:

- Web application security (OWASP Top 10)
- Penetration testing techniques
- Security scanning tools
- Vulnerability assessment

DevOps Engineer:

- CI/CD setup (GitHub Actions)
- Docker/containerization
- Build automation
- Environment configuration

12.3 Training Needs

12.3.1 Domain Training

Statistical Bias and Skewness:

- **Target:** All QA Engineers
- **Duration:** 4 hours
- **Content:**
  - Understanding class imbalance
  - Skewness interpretation
  - SMOTE algorithm overview
  - Transformation methods (log, Box-Cox, etc.)
- **Source:** Internal data scientist or online course (Coursera, Udemy)

BiasXplorer Application Walkthrough:

- **Target:** All QA team
- **Duration:** 2 hours
- **Content:**
  - Application architecture overview
  - 7-step workflow demonstration
  - API endpoint documentation
  - File structure and code organization
- **Source:** Development team presentation

12.3.2 Tool Training

pytest and pytest-flask:

- **Target:** Backend QA Engineers (if not proficient)
- **Duration:** 8 hours (1 day)
- **Content:**

    - Writing unit tests
    - Fixtures and mocking
    - API testing with pytest-flask
    - Coverage reporting
- **Source:** Online tutorial (RealPython, Udemy)

Jest and React Testing Library:

- **Target:** Frontend QA Engineers (if not proficient)
- **Duration:** 8 hours (1 day)
- **Content:**
    - Component testing
    - Mocking API calls
    - Testing hooks
    - Snapshot testing
- **Source:** Official documentation, Egghead.io

Cypress E2E Testing:

- **Target:** Frontend QA Engineers
- **Duration:** 8 hours (1 day)
- **Content:**
    - Cypress basics
    - Writing E2E tests
    - Assertions and commands
    - CI integration
- **Source:** Cypress documentation, Udemy course

Locust Load Testing:

- **Target:** Performance Tester
- **Duration:** 4 hours
- **Content:**
    - Writing load test scenarios
    - Running distributed tests
    - Analyzing results
- **Source:** Locust documentation

12.3.3 Process Training

GitHub Actions CI/CD:

- **Target:** QA Lead, DevOps Engineer
- **Duration:** 4 hours
- **Content:**
    - Workflow creation
    - Test automation integration
    - Artifact management
- **Source:** GitHub Learning Lab

Defect Management in GitHub Issues:

- **Target:** All QA team
- **Duration:** 2 hours
- **Content:**
  - Issue templates
  - Labeling system
  - Linking to pull requests
  - Tracking workflow
- **Source:** Internal QA Lead training

12.4 Training Schedule

Week	Training	Target	Duration
0 (Pre-project)	Statistical Bias/Skewness	All QA	4 hours
1	Application Walkthrough	All QA	2 hours
1	pytest Training	Backend QA	8 hours
1	Jest/RTL Training	Frontend QA	8 hours
2	Cypress Training	Frontend QA	8 hours
2	Locust Training	Performance Tester	4 hours
2	GitHub Actions	QA Lead, DevOps	4 hours
3	Defect Management	All QA	2 hours

13. Schedule

13.1 Test Milestones

Milestone	Description	Target Date	Dependencies
M1	Test Plan Approved	Week 0, Day 5	Stakeholder review
M2	Test Environment Ready	Week 1, Day 2	DevOps setup
M3	Test Data Created	Week 1, Day 5	QA team
M4	Unit Tests Implemented	Week 2, Day 5	Test design complete
M5	Integration Tests Implemented	Week 3, Day 3	API stable
M6	System Tests Implemented	Week 3, Day 5	Full workflow testable
M7	First Test Cycle Complete	Week 4, Day 2	All tests executed
M8	Regression Testing Complete	Week 4, Day 4	Defects fixed
M9	Final Test Report	Week 5, Day 1	All testing complete

Milestone	Description	Target Date	Dependencies
M10	Test Sign-off	Week 5, Day 3	Report approved

13.2 Item Transmittal Events

Event	Item Transmitted	From	To	Target Date
E1	Backend code (initial)	Dev Team	QA Team	Week 1, Day 1
E2	Frontend code (initial)	Dev Team	QA Team	Week 1, Day 1
E3	API documentation	Dev Team	QA Team	Week 1, Day 2
E4	Sample datasets	Dev Team	QA Team	Week 1, Day 3
E5	Unit tests (backend)	QA Team	Dev Team	Week 2, Day 5
E6	Unit tests (frontend)	QA Team	Dev Team	Week 2, Day 5
E7	Integration tests	QA Team	Dev Team	Week 3, Day 3
E8	Defect reports (batch 1)	QA Team	Dev Team	Week 4, Day 1
E9	Fixed code	Dev Team	QA Team	Week 4, Day 3
E10	Final test report	QA Team	PM/Stakeholders	Week 5, Day 1

13.3 Task Schedule (Gantt Chart Summary)

Week 0 (Pre-project):

- Days 1-3: Test plan creation
- Day 4: Test plan review
- Day 5: Test plan approval ✅ **M1**

Week 1:

- Days 1-2: Test environment setup ✅ **M2**
- Days 1-5: Test design (unit, integration, system)
- Days 3-5: Test data creation ✅ **M3**
- Days 4-5: Training (pytest, Jest)

Week 2:

- Days 1-5: Unit test implementation (backend, frontend)
- Days 3-5: CI/CD setup
- Day 5: Unit tests complete ✅ **M4**

Week 3:

- Days 1-3: Integration test implementation ✅ **M5**
- Days 4-5: System test implementation ✅ **M6**
- Days 1-5: Cypress training and E2E test creation

Week 4:

- Days 1-2: Execute all tests (first cycle) ✅ **M7**
- Day 1: Log defects
- Days 2-3: Defect fixes by dev team
- Days 3-4: Regression testing ✅ **M8**

Week 5:

- Day 1: Create final test report ✅ **M9**
- Days 2-3: Review and approval ✅ **M10**

13.4 Time Estimates for Testing Tasks

Task	Estimated Time	Notes
Write 1 unit test	30 minutes	Average complexity
Write 1 integration test	1 hour	Includes API setup
Write 1 system test	2 hours	E2E scenario
Execute unit test suite	5 minutes	Automated
Execute integration test suite	15 minutes	Automated
Execute system test suite	30 minutes	Automated (Cypress)
Create 1 test dataset	1 hour	Manual validation
Log 1 defect	15 minutes	Includes screenshots, steps
Verify 1 defect fix	30 minutes	Re-test and regression check
Generate coverage report	5 minutes	Automated
Generate test summary report	2 hours	Manual analysis

Total Estimated Testing Effort:

- Unit tests: 100 tests × 30 min = 50 hours
- Integration tests: 50 tests × 1 hour = 50 hours
- System tests: 20 tests × 2 hours = 40 hours
- Test data creation: 20 datasets × 1 hour = 20 hours
- Execution and reporting: 30 hours
- **Total:** 190 hours (~5 weeks with 2 engineers)

13.5 Resource Periods of Use

Resource	Period	Allocation
QA Lead	Weeks 0-5	100% (full-time)
Senior QA Engineer	Weeks 1-5	100% (full-time)

Resource	Period	Allocation
Backend QA Engineer (×2)	Weeks 1-5	100% each (full-time)
Frontend QA Engineer (×2)	Weeks 1-5	100% each (full-time)
Performance Tester	Week 4	50% (part-time)
Security Tester	Week 4	50% (part-time)
DevOps Engineer	Weeks 1-2	50% (part-time)

Peak Resource Usage: Week 4 (all resources active)

---

14. Risks and Contingencies

14.1 High-Risk Assumptions

Risk ID	Assumption	Probability	Impact	Risk Level
R-001	Test environment will be stable throughout testing	Medium	High	**HIGH**
R-002	All required datasets will be available by Week 1	Low	High	**MEDIUM**
R-003	Code changes will not be frequent during testing	Medium	High	**HIGH**
R-004	QA team has sufficient statistical knowledge	Medium	Medium	**MEDIUM**
R-005	Third-party libraries (SMOTE, Plotly) are stable	Low	Medium	**LOW**
R-006	No major requirement changes during testing	High	High	**CRITICAL**
R-007	CI/CD pipeline will be operational by Week 2	Medium	Medium	**MEDIUM**
R-008	Test data generation scripts will work correctly	Low	Medium	**LOW**
R-009	Browser compatibility issues will be minimal	Low	Low	**LOW**
R-010	Performance targets are achievable with current architecture	Medium	High	**HIGH**

14.2 Contingency Plans

R-001: Test Environment Instability

Contingency:

- Maintain backup test environment (Docker containers)
- Document environment setup in Dockerfile
- Use environment snapshot/restore capability
- Allocate 1-day buffer for environment recovery

Trigger: Environment down for >2 hours

R-002: Dataset Unavailability

Contingency:

- Use synthetic data generation scripts
- Leverage public datasets (Kaggle, UCI ML Repository)
- Create minimal test datasets manually
- Prioritize testing with available data

Trigger: Dataset not delivered by Week 1, Day 3

R-003: Frequent Code Changes

Contingency:

- Implement feature freeze after Week 2
- Use feature branches for non-critical changes
- Automate regression testing
- Increase regression testing frequency

Trigger: >10 code commits per day during testing

R-004: Insufficient Statistical Knowledge

Contingency:

- Provide additional training (see Section 12.3.1)
- Assign data scientist as consultant
- Create statistical testing cheat sheet
- Pair junior testers with senior testers

Trigger: Test cases with incorrect statistical validation

R-005: Third-Party Library Issues

Contingency:

- Pin library versions in requirements.txt
- Test with multiple library versions
- Maintain library upgrade documentation
- Have fallback implementations for critical algorithms

Trigger: Library bug discovered or version conflict

R-006: Major Requirement Changes

Contingency:

- Suspend testing (see Section 7.1)
- Re-evaluate test plan scope
- Obtain stakeholder approval for schedule extension

- Prioritize testing of unchanged features

Trigger: >20% of requirements changed

Recovery Plan:

- 1-week buffer for test plan updates
- 2-week buffer for test re-implementation
- Phased testing approach (critical features first)

R-007: CI/CD Pipeline Delays

Contingency:

- Execute tests manually until pipeline ready
- Use local test runners
- Delay integration of performance tests
- Allocate DevOps resource full-time if needed

Trigger: Pipeline not operational by Week 2, Day 3

R-008: Test Data Generation Failures

Contingency:

- Use manual test data creation
- Leverage existing sample datasets
- Simplify test data requirements
- Create reusable test data templates

Trigger: Generation script fails validation

R-009: Browser Compatibility Issues

Contingency:

- Focus on Chrome for initial testing
- Use Browserstack for cross-browser testing
- Document known browser issues
- Deprioritize older browser support

Trigger: Critical failures on secondary browsers

R-010: Performance Target Failures

Contingency:

- Profile code to identify bottlenecks
- Optimize critical paths
- Re-negotiate performance targets with stakeholders
- Implement caching or async processing

Trigger: >30% of performance tests fail

14.3 Risk Monitoring

Weekly Risk Review:

- QA Lead to review risk status
- Update risk probability and impact
- Activate contingency plans as needed
- Communicate risk status to PM

Risk Escalation:

- **MEDIUM** risks: Notify PM
- **HIGH** risks: Notify PM and stakeholders
- **CRITICAL** risks: Emergency meeting with all stakeholders

14.4 Budget Contingency

Contingency Reserve: 20% of testing budget

- Allocated for risk mitigation
- Covers additional resources, tools, or time

Authorization Required: PM approval for contingency use

---

15. Approvals

This test plan requires approval from the following stakeholders:

15.1 Prepared By

Name: _____
Title: QA Lead
Signature: _____
Date: _____

15.2 Technical Review

Name: _____
Title: Senior QA Engineer
Signature: _____
Date: _____

Name: _____
Title: Development Lead
Signature: _____
Date: _____

15.3 Management Approval

Name: _____
Title: Project Manager
Signature: _____
Date: _____

Name: _____
Title: Product Owner
Signature: _____
Date: _____

15.4 Quality Assurance Approval

Name: _____
Title: QA Director
Signature: _____
Date: _____

15.5 Final Approval

Name: _____
Title: Engineering Manager
Signature: _____
Date: _____

Appendix A: Requirements Traceability Matrix

Requirement ID	Requirement	Test Cases	Priority
REQ-001	File upload (CSV, Excel)	TC-001, TC-002	High
REQ-002	File size validation (<10MB)	TC-003, TC-004	High
REQ-003	Dataset preview	TC-010, TC-011	Medium
REQ-004	NaN removal	TC-020, TC-021, TC-022	High
REQ-005	Duplicate removal	TC-023, TC-024	High
REQ-006	Column type classification	TC-030, TC-031	High
REQ-007	Categorical bias detection	TC-040, TC-041, TC-042	High
REQ-008	Skewness detection	TC-050, TC-051, TC-052	High
REQ-009	SMOTE correction	TC-060, TC-061	High
REQ-010	Oversample correction	TC-062, TC-063	Medium
REQ-011	Undersample correction	TC-064, TC-065	Medium
REQ-012	Log transformation	TC-070, TC-071	High
REQ-013	Yeo-Johnson transformation	TC-075, TC-076	Medium

Requirement ID	Requirement	Test Cases	Priority
REQ-014	Visualization generation	TC-080, TC-081, TC-082	High
REQ-015	PDF report generation	TC-090, TC-091	Medium
REQ-016	File download	TC-095, TC-096	Medium
REQ-017	State persistence	TC-100, TC-101	High
REQ-018	Multi-step navigation	TC-105, TC-106, TC-107	High

Appendix B: Test Environment Setup Guide

B.1 Backend Setup

```
# Clone repository
git clone https://github.com/MadhuSudhanAV03/BiasXplorer-Mini.git
cd BiasXplorer-Mini/backend

# Create virtual environment
python -m venv venv
.\venv\Scripts\activate  # Windows
# source venv/bin/activate  # macOS/Linux

# Install dependencies
pip install -r requirements.txt

# Run backend
flask run --port 5000
```

B.2 Frontend Setup

```
# Navigate to frontend
cd ../frontend

# Install dependencies
npm install

# Run frontend
npm run dev
```

B.3 Test Framework Setup

```
# Backend testing
cd backend
pip install pytest pytest-flask pytest-cov
```

```
# Frontend testing
cd ../frontend
npm install -D @testing-library/react @testing-library/jest-dom cypress
```

Appendix C: Test Data Samples

C.1 Valid CSV Sample

File: `test_data/valid_datasets/balanced_gender.csv`

```
age,gender,income,label
25,Male,50000,0
30,Female,55000,1
35,Male,60000,0
28,Female,52000,1
```

C.2 Imbalanced Dataset Sample

File: `test_data/valid_datasets/imbalanced_gender.csv`

```
age,gender,income,label
25,Male,50000,0
30,Male,55000,0
35,Male,60000,0
28,Male,52000,0
40,Male,70000,0
45,Male,75000,0
50,Male,80000,0
55,Male,85000,0
60,Male,90000,0
33,Female,58000,1
```

(90% Male, 10% Female - Severe imbalance)

C.3 Skewed Dataset Sample

File: `test_data/valid_datasets/right_skewed_income.csv`

```
id,income
1,30000
2,35000
3,32000
4,38000
5,40000
6,45000
7,50000
```

```
8,55000
9,120000
10,200000
```

(Right-skewed distribution)

Appendix D: Defect Severity Definitions

Severity	Definition	Example	Response Time
Critical	System crash, data loss, security breach	Application won't start	4 hours
High	Major feature broken, no workaround	SMOTE correction fails	1 day
Medium	Feature partially broken, workaround exists	Chart rendering slow	3 days
Low	Minor issue, cosmetic	Button alignment off	1 week

Appendix E: Test Metrics to Track

Metric	Formula	Target
Test Coverage	(Lines Tested / Total Lines) × 100	80%
Defect Density	Defects Found / KLOC	<5
Test Pass Rate	(Passed Tests / Total Tests) × 100	>95%
Defect Removal Efficiency	(Defects Found in Testing / Total Defects) × 100	>90%
Mean Time to Detect	Average time from defect introduction to detection	<2 days
Mean Time to Resolve	Average time from defect detection to resolution	❤️ days

---

Document History

Version	Date	Author	Changes
1.0	November 18, 2025	GitHub Copilot	Initial IEEE 829-1983 compliant test plan

---

END OF DOCUMENT