**Software Test Plan (STP) — Demand Forecasting System (DFS)**

**Project:** Demand Forecasting System (DFS)  
**Version:** 1.0  
**Authors:** MS Ananthesha,Kolluri Nithin,Ranjith,Karthik  
**Date:** 11/09/2025  
**Status:** Draft

**1)Introduction**

**Purpose:**This document defines the test plan for the Demand Forecasting System v1.0 (prototype). It outlines objectives, scope, strategy, resources, schedule, and responsibilities for testing.

**Scope:**Testing covers DFS prototype features including CSV ingestion & validation, preprocessing (missing-value handling, outlier flagging), feature engineering, model training and cross-validation, forecast generation and export, model registry, dashboard visualization, basic RBAC (if implemented), and basic audit logging. Enterprise DB connectors, LDAP/OAuth, distributed training, production-grade TLS and other enterprise integrations are excluded (future work).

**References:**

* DFS SRS v1.2 (trimmed MVP)
* Requirements Traceability Matrix (RTM) — Section 9 of SRS
* Project README, repo and sample datasets

**Definitions:**

* DFS — Demand Forecasting System
* SRS — Software Requirements Specification
* RTM — Requirements Traceability Matrix
* TC — Test Case
* CV — Cross-validation

1. **Test Items**

* Ingestion module (CSV upload, schema detection, preview) — DF-F-001, DF-F-003
* Preprocessing module (missing values, outlier flagging) — DF-F-004, DF-F-005
* Feature engine (date features, exogenous columns) — DF-F-006
* Modeling engine (train, CV, basic tuning) — DF-F-007, DF-F-008, DF-F-009 (optional)
* Model registry (save/load models, metadata) — DF-F-011
* Forecast engine (generate forecasts & intervals) — DF-F-010
* Dashboard & visualization (plots, metrics) — DF-F-012
* Scenario simulation UI (optional) — DF-F-013 (opt)
* Export & minimal API (CSV/XLSX) — DF-F-014
* Audit logging & backup script — DF-F-018, DF-SR-007
* Security checks: CSV sanitization, secrets/no-hardcoded-keys — DF-SR-004, DF-SR-005

1. **Features to be Tested**

Features mapped to SRS requirement IDs (primary/trimmed list):

* DF-F-001: CSV upload and preview — TC-ING-01
* DF-F-003: Schema detection & validation — TC-ING-03
* DF-F-004: Missing-value handling (interpolate/ffill/drop) — TC-PRE-01
* DF-F-005: Outlier detection & accept/reject correction — TC-PRE-02
* DF-F-006: Feature engineering (date features, exog) — TC-FE-01
* DF-F-007: Model selection & training (at least two model types) — TC-MOD-01
* DF-F-008: Time-series cross-validation & metrics (RMSE/MAPE) — TC-MOD-02
* DF-F-010: Forecast generation and prediction intervals (if supported) — TC-FC-01
* DF-F-011: Model store & versioning (local filesystem/SQLite) — TC-MOD-04
* DF-F-012: Forecast dashboard (interactive plot + toggles) — TC-UI-01
* DF-F-014: Export CSV/XLSX & (optional) REST API — TC-API-01
* DF-F-018: Audit logs (upload/train/export entries) — TC-OPS-03
* DF-SR-004: CSV sanitization (formula injection) — TC-SEC-02
* DF-SR-005: Secrets via environment variables (no secrets in repo) — TC-DEV-02

1. **Features Not to be Tested**

* Enterprise DB connectors (Postgres/MySQL) — DF-F-002 (future)
* LDAP/OAuth / enterprise single-sign-on — DF-SR-003 (future)
* Distributed or GPU training, large-scale performance testing — DF-NF-002 (future)
* External webhooks/email alerting system (optional) — DF-F-015 (defer)
* Production TLS / managed secrets (unless remote hosting is used) — DF-SR-001 (defer)

1. **Test Approach / Strategy**

**Levels:**

* Unit tests (function/module-level) — pytest for ingestion, preprocessing, model wrappers.
* Integration tests — backend endpoints and model engine interactions (use Flask/FastAPI test client).
* System tests — end-to-end: upload → preprocess → train → forecast → export (manual and automated).
* Acceptance tests (UAT) — run the demo script covering the happy path.

**Types:**

* Functional testing (core features)
* Regression testing (automated test suite)
* Performance benchmarking (light-weight: forecast runtime on dev laptop)
* Security testing (CSV sanitization, secrets scan)
* Usability testing (basic dashboard UX in Chrome/Firefox)

**Entry Criteria:**

* Stable build or commit available on dev machine.
* Test data sets present in repo.
* RTM & SRS finalized for trimmed MVP.
* Test tools installed (pytest, optional selenium/playwright).

**Exit Criteria:**

* All High-priority test cases executed and passing or documented with acceptable workarounds.
* No critical defects open.
* Demo script runs end-to-end within allotted demo time.
* Test summary report completed.

5.1 **Security Validation**

* Verify CSV sanitization: app neutralizes or rejects formula injection payloads (TC-SEC-02).
* Verify secrets management: no hard-coded secrets in repo; instructions to supply env vars in README (TC-DEV-02).
* If auth is implemented: verify password hashing (bcrypt/argon2) and session timeout (TC-SEC-01).
* Run pip-audit or safety to detect critical vulnerabilities prior to demo (TC-DEV-02).

1. **Test Environment**

**Hardware:**

* Developer laptops / local VM (Windows/Mac/Linux) — single machine demo.
* Optional cloud VM for remote demo (ensure TLS).

**Software:**

* Python 3.9+ (venv)
* Flask or FastAPI backend, SQLite for metadata
* Frontend: React or server-rendered Bootstrap pages + Plotly/matplotlib for charts
* Libraries: pandas, scikit-learn, statsmodels, SQLAlchemy, optional LightGBM/XGBoost if installable

**Tools:**

* pytest for unit/integration tests
* selenium or playwright for optional UI automation
* Postman or pytest test client for API tests
* pip-audit / safety for dependency checks
* Git/GitHub (issue tracker) or simple spreadsheet for defects

**Test Data:** (store under tests/data/)

* sample\_daily.csv — seasonal single-series
* sample\_multi.csv — multi-SKU example
* missing\_values.csv — gaps to test imputation
* outliers.csv — outlier cases
* malicious.csv — formula injection samples

1. **Test Schedule**

* Test case design: Week 1-2
* Start Date:Oct 1st
* Unit & integration test implementation: Week 3
* System & regression tests: Weeks 4–7
* Security & dependency checks: Weeks 5–7
* UAT / Demo: Week 8
* Final Test Summary Report: End of Week 8

End Date: 20th Nov

1. **Test Deliverables**

* Test Plan (this document)
* Test Cases (manual & automated) in repository tests/
* Test Scripts (pytest suites, optional UI scripts)
* Test Data sets
* Test Execution logs and screenshots / demo recording
* Defect reports (GitHub Issues or tracker)
* Final Test Summary Report

1. **Roles and Responsibilities**

| **Role** | **Name** | **Responsibility** |
| --- | --- | --- |
| QA Lead / Test Coordinator | <Name> | Maintain STP, schedule tests, coordinate execution, sign-off. |
| Test Engineer — Backend | MS Ananthesha | Implement unit & integration tests, run test suites. |
| Test Engineer — Frontend | Kolluri Nithin | Create/execute UI test scripts, verify visualizations. |
| Model Engineer | Ranjith | Validate modeling pipeline, cross-validation & explainability tests. |
| Developer (Support) | Karthik | Fix defects, assist reproduction. |
| Product Owner / Instructor | <Name> | Approve acceptance and final sign-off. |

1. **Risks and Mitigation**

| **Risk** | **Mitigation** |
| --- | --- |
| Heavy dependency install issues (Prophet/LightGBM) | Use stable libraries (statsmodels, scikit-learn). Freeze requirements.txt. |
| Dirty / inconsistent CSV input | Build robust schema validator and include representative sample data early. |
| Scope creep | Freeze MVP end of Week 2; push extras to future work. |
| Single-person blocker | Cross-train; pair-program critical tasks. |
| Spending too long polishing UI | Prioritize functionality; use UI libraries/templates. |
| Demo environment failure | Keep backup environment and recorded demo as fallback. |

1. **Assumptions & Dependencies**

* Team will provide representative datasets early.
* Local SQLite-based environment is sufficient for prototype.
* Internet access for dependency installation.
* Instructor available for UAT scheduling.

1. **Suspension & Resumption Criteria**

**Suspend testing if:**

* Build/environment unavailable for > 4 hours.
* Blocking defects prevent > 30% of test cases from running.

**Resume testing when:**

* Blocking defects resolved or rollback to stable build performed.
* Environment restored and smoke tests pass.

1. **Test Case Management & Traceability**

RTM will map each SRS requirement to one or more test cases. Test results and evidence (screenshots, logs, links to PRs) will be attached to each RTM entry.

**Example mappings:**

* DF-F-001 (CSV upload) → TC-ING-01: verify upload and preview validation.
* DF-F-004 (Missing-value handling) → TC-PRE-01: test interpolate/ffill/drop on missing\_values.csv.
* DF-F-008 (CV & metrics) → TC-MOD-02: verify rolling CV computes RMSE/MAPE per fold.
* DF-F-010 (Forecast) → TC-FC-01: forecast produced, interval plotted, CSV export correct.
* DF-SR-004 (CSV sanitization) → TC-SEC-02: malicious.csv does not produce active formulas in exports.

1. **Test Metrics & Reporting**

**Metrics:**

* % test cases executed / passed / failed
* Defect count by severity (Critical/High/Medium/Low)
* Requirement coverage (RTM coverage)
* Test automation % (automated vs manual)
* Average test execution time / tests per day

**Reports:**

* Daily execution status (during heavy test weeks)
* Weekly summary to team
* Final Test Summary Report

1. **Approvals**

| **Role** | **Name** | **Signature / Date** |
| --- | --- | --- |
| QA Lead / Test Coordinator |  |  |
| Dev Lead |  |  |
| Product Owner / Instructor |  |  |