# **Software Requirements Specification (SRS) — Demand Forecasting Software**

**Project name:** Demand Forecasting System (DFS)  
**Version:** 1.0  
**Authors:** Team of 4 undergraduate students  
**Project duration:** 2 months (college project)  
**Document purpose:** Define functional and non-functional requirements, interfaces, constraints, quality attributes and acceptance criteria for the Demand Forecasting Software.

## **1. Introduction**

### **1.1 Purpose**

This SRS describes the requirements for the Demand Forecasting System (DFS). DFS ingests historical sales and related data, trains forecasting models, produces actionable demand forecasts for multiple horizons, and provides visualization, export, and integration endpoints suitable for planners and analysts.

### **1.2 Scope**

DFS is a modular software product that will:

* Ingest data from CSV, database, or REST sources.
* Preprocess, clean, and feature-engineer time series.
* Train one or more forecasting models (classical and ML-based).
* Produce forecasts for configurable horizons (daily/weekly/monthly).
* Evaluate model performance and support model selection.
* Provide a dashboard for visualization, scenario simulation, and export.
* Expose REST APIs for programmatic access and integration.  
   Project excludes: hardware provisioning, enterprise authentication systems (can integrate via connectors), and production-scale deployment orchestration (only a prototype/demo delivered).

### **1.3 Audience**

* Instructors / Project evaluators
* Team developers (3 students)
* QA / Testers
* Business stakeholders (product owner, supply planner)

### **1.4 Definitions / Acronyms**

* DFS — Demand Forecasting System
* API — Application Programming Interface
* UI — User Interface
* RTM — Requirements Traceability Matrix
* RMSE, MAPE — standard forecast error metrics

## **2. Overall description**

### **2.1 Product perspective**

DFS is a standalone web application (single server or container) composed of:

* Data ingestion & ETL module
* Feature engineering & preprocessing module
* Model training & evaluation module
* Forecast generation & scheduler
* Visualization dashboard (web UI)
* REST API layer
* Admin & security module

### **2.2 Major product functions (high-level)**

* Ingest time series and related features (promotions, prices, holidays).
* Clean and transform data (missing values, outliers).
* Train multiple forecasting models (ARIMA, Exponential Smoothing, Prophet, XGBoost / LightGBM).
* Evaluate and compare models using standard metrics.
* Generate and persist forecasts and prediction intervals.
* Visualize historicals, forecasts, model diagnostics, and scenario simulations.
* Export forecasts (CSV/Excel) and provide REST endpoints for downstream systems.
* Alerting for data anomalies and forecast drift.

### **2.3 User roles and characteristics**

* **Developer / Data Scientist (team):** implements ingestion, models, diagnostics.
* **Business Analyst / Planner:** loads data, views forecasts, runs scenario comparisons, exports CSV.
* **System Administrator (developer role for prototype):** manages local configs and backups.

### **2.4 Operating environment**

**Target development / demo environment (minimum viable):**

* **Developer machine (local laptop)** — macOS (your machine) or any Linux/Windows laptop for development and demo. No server provisioning required.
* **Python 3.9+** (use a virtual environment like venv or pipenv) for all backend and modeling code.
* **SQLite** for metadata and small persistence (file-based, no DB server to manage).
* **Backend:** lightweight Python web framework (Flask or FastAPI) running locally (single-process for prototype).
* **Frontend:** simple UI using React or server-rendered pages (Bootstrap + minimal JS). Works in modern browsers (Chrome/Firefox).
* **Data I/O:** **CSV upload/download only** for the prototype. (No DB/SFTP connectors required.)
* **Model libraries:** scikit-learn, statsmodels, XGBoost/LightGBM only if installation is straightforward. Avoid heavy or hard-to-build packages unless necessary.

### **2.5 Constraints**

* Two-month project scope: prototype-level feature completeness prioritized.
* Use open-source libraries only (no paid model services).
* Dataset sizes expected to be small-to-moderate (suitable for local execution and SQLite). Very large datasets or real-time high-concurrency use are out of scope.
* CSV-only ingestion for prototype — DB connectors and SFTP are optional future work.

## **3. External interface requirements**

### **3.1 User interfaces**

* Responsive, simple web UI with:  
  + CSV upload & preview wizard
  + Dataset explorer (table + time series preview plot)
  + Model training UI (select algorithm & basic hyperparameters)
  + Forecast results page (interactive plot, forecast table, metrics)
  + Simple admin page for model registry and logs (prototype)
* Accessibility: basic keyboard navigation and readable fonts.

### **3.2 Hardware interfaces**

* Standard laptop/desktop; no special hardware.

### **3.3 Software interfaces**

* **Input sources:** CSV upload (primary). DB connectors optional/future.
* **Model libraries:** scikit-learn, statsmodels, (Prophet/XGBoost/LightGBM if feasible).
* **Export formats:** CSV, XLSX.
* **Authentication integrations:** Optional; prototype may use a simple local user system or no auth for demo.

### **3.4 Communications**

* REST API for programmatic forecast retrieval; recommended to run locally during demo. HTTPS recommended for remote hosting (future work).

## **4. System features — Functional Requirements (DF-F-###)**

Each requirement lists priority and acceptance criteria. Tests (TC-IDs) are illustrative.

### **DF-F-001 — Data ingestion: CSV upload**

**Requirement:** The system shall allow users to upload historical sales data as CSV with a required datetime column and a value column.  
 **Priority:** High  
 **Acceptance criteria:** Upload completes for valid CSV; system shows preview (first 100 rows) and detected columns; validation messages for missing required columns. Test: TC-ING-01

### **DF-F-002 — Schema detection & validation**

**Requirement:** Detect date/time column, value column, and optionally category/exogenous columns; validate types and report errors.  
 **Priority:** High  
 **Acceptance criteria:** Incorrect schemas flagged with clear error messages. Test: TC-ING-03

### **DF-F-003 — Data cleaning & missing-value handling**

**Requirement:** Provide configurable strategies for missing values (interpolate, forward-fill, drop) and apply selected strategy.  
 **Priority:** High  
 **Acceptance criteria:** Missing values handled per selection; resulting series has no nulls per chosen strategy. Test: TC-PRE-01

### **DF-F-004 — Outlier detection**

**Requirement:** Detect and flag outliers using configurable methods (IQR or z-score) and allow user to accept/reject automated corrections.  
 **Priority:** Medium  
 **Acceptance criteria:** Outliers marked on a plot; user can accept corrections which update data. Test: TC-PRE-02

### **DF-F-005 — Feature engineering**

**Requirement:** Generate time-based features (day-of-week, month, holiday flags) and allow adding simple exogenous variables via CSV.  
 **Priority:** High  
 **Acceptance criteria:** Feature set visible in dataset schema and used by models when selected. Test: TC-FE-01

### **DF-F-006 — Model selection**

**Requirement:** Allow selection of one or more forecasting algorithms from a provided list and configure basic hyperparameters.  
 **Priority:** High  
 **Acceptance criteria:** Models (e.g., ARIMA/ETS, simple ML regressors) selectable; hyperparameter form applied. Test: TC-MOD-01

### **DF-F-007 — Training & cross-validation**

**Requirement:** Perform time-series cross-validation (rolling/expanding window) and display cross-validated metrics (RMSE, MAPE).  
 **Priority:** High  
 **Acceptance criteria:** CV completes and a table of metrics per fold and aggregated metrics is shown. Test: TC-MOD-02

### **DF-F-008 — Hyperparameter tuning (basic / optional)**

**Requirement:** Support simple automated tuning (grid search) for selected model hyperparameters with a limited number of trials.  
 **Priority:** Medium  
 **Acceptance criteria:** Tuning runs, best hyperparameters shown, and best-model saved (optional). Test: TC-MOD-03

### **DF-F-019 — Forecast generation**

**Requirement:** Generate forecasts for configurable horizons (days/weeks/months) and produce prediction intervals at configurable confidence levels.  
 **Priority:** High  
 **Acceptance criteria:** Forecast plot shows forecast line and shaded prediction interval; table exportable. Test: TC-FC-01

### **DF-F-010 — Model store & versioning (prototype)**

**Requirement:** Persist trained models and metadata locally (SQLite or model files with JSON metadata). Allow selection of saved models for forecasts.  
 **Priority:** Medium  
 **Acceptance criteria:** Saved models listed with metadata (type, training date, dataset id). Selected model used to generate forecast. Test: TC-MOD-04

### **DF-F-011 — Forecast dashboard**

**Requirement:** Present an interactive dashboard with historicals, forecasts, overlays of events (promotions/holidays), and metric summaries.  
 **Priority:** High  
 **Acceptance criteria:** Dashboard displays interactive time series and metrics; user can toggle series. Test: TC-UI-01

### **DF-F-012 — Scenario simulation**

**Requirement:** Allow users to create simple scenario inputs (e.g., change promotion flag) and re-generate forecasts for scenario comparison.  
 **Priority:** Medium  
 **Acceptance criteria:** User creates a scenario; system shows delta vs baseline forecast. Test: TC-UI-02

### **DF-F-013 — Export & integration**

**Requirement:** Allow exporting forecasts as CSV/XLSX and provide a minimal REST API endpoint to fetch forecasts in JSON (optional).  
 **Priority:** High  
 **Acceptance criteria:** Downloads of CSV/XLSX succeed; API returns JSON for local authenticated requests (if implemented). Test: TC-API-01

### **DF-F-014 — Anomaly detection & alerts (basic)**

**Requirement:** Detect significant deviations in incoming data vs expected patterns and create alerts visible in UI. Webhook/email alerts are optional.  
 **Priority:** Medium  
 **Acceptance criteria:** Alert created when deviation threshold exceeded; UI displays alert detail. Test: TC-OPS-01

### **DF-F-015 — Scheduler / Automated runs (simplified)**

**Requirement (revised):** Provide manual "Run now" for retraining/forecast generation and an optional simple scheduler (e.g., apscheduler) for local repeat runs if time permits.  
 **Priority:** Low–Medium  
 **Acceptance criteria:** Manual runs work reliably; optional scheduler executes local jobs. Test: TC-OPS-02

### **DF-F-016 — User management & RBAC (prototype)**

**Requirement:** Include a simple local user account mechanism with light-weight roles (Admin, DataScientist, Analyst). Full enterprise RBAC is future work.  
 **Priority:** Medium  
 **Acceptance criteria:** Role assignment works in prototype; restricted operations blocked for lower roles where implemented. Test: TC-SEC-01

### **DF-F-017 — Audit logs (basic)**

**Requirement:** Maintain a simple audit log of key operations: data import, model training, forecast generation, exports.  
 **Priority:** Medium  
 **Acceptance criteria:** Audit entries created and viewable via UI for recent actions. Test: TC-OPS-03

## **5. Non-functional requirements (DF-NF-###)**

### **DF-NF-001 — Performance (relaxed for laptop prototype)**

**Requirement:** Forecast generation (using a saved model) for a single time series shall complete within **30 seconds** on a typical development laptop for datasets with ≤ 100k points.  
**Priority:** High  
**Acceptance criteria:** Bench test shows median runtime ≤ 30s for sample datasets. Test: TC-PERF-01

### **DF-NF-002 — Scalability (prototype-level)**

**Requirement:** The system shall support **sequential** training of multiple models on the demo machine. Parallel training is optional and not required.  
 **Priority:** Medium  
 **Acceptance criteria:** Multiple model trainings run sequentially without crashing; system remains responsive. Test: TC-PERF-02

### **DF-NF-003 — Availability (prototype)**

**Requirement:** For the prototype/demo, the system shall be available during scheduled demo times on the development machine (no formal SLA).  
 **Priority:** Low  
 **Acceptance criteria:** Developer can demonstrate the system during scheduled demo. Test: TC-OPS-04

### **DF-NF-004 — Maintainability & Extensibility**

**Requirement:** Codebase must be modular and documented; include automated unit tests achieving at least 60% coverage for core modules where feasible.  
 **Priority:** Medium  
 **Acceptance criteria:** Tests present and coverage reports meet threshold for core modules. Test: TC-DEV-01

### **DF-NF-005 — Explainability & Traceability**

**Requirement:** Model training results shall include diagnostic plots and feature-importance (for ML models); each forecast must be traceable to the model and dataset used.  
 **Priority:** High  
 **Acceptance criteria:** Feature importance view available (where applicable); forecast metadata includes model id and dataset id. Test: TC-UX-03

## **6. Security**

### **6.1 Security objectives (prototype)**

1. Protect sensitive data stored locally where feasible.
2. Provide basic accountability: simple logs and traceability of key actions.

### **6.2 Security requirements (prototype)**

* **DF-SR-001 (Encryption in transit):** If the app is hosted remotely, use HTTPS/TLS. For local demos this is optional but recommended. Test: TC-SEC-02.
* **DF-SR-002 (Encryption at rest):** Sensitive config (API keys) should not be hard-coded; use environment variables. Full encryption at rest is future work. Test: TC-SEC-03.
* **DF-SR-003 (RBAC — prototype):** Implement a simple local role system (Admin, DataScientist, Analyst). Full LDAP/OAuth integration is future work. Test: TC-SEC-01.
* **DF-SR-004 (Audit logging):** Log administrative actions and data exports with timestamp and user. Test: TC-OPS-03.
* **DF-SR-005 (Input validation):** Validate and sanitize CSV inputs to avoid code injection or malformed records. Test: TC-SEC-04.
* **DF-SR-006 (Secrets management):** Use environment variables for credentials and avoid hard-coded secrets. Test: TC-DEV-02.

## **7. Quality attributes & Acceptance tests**

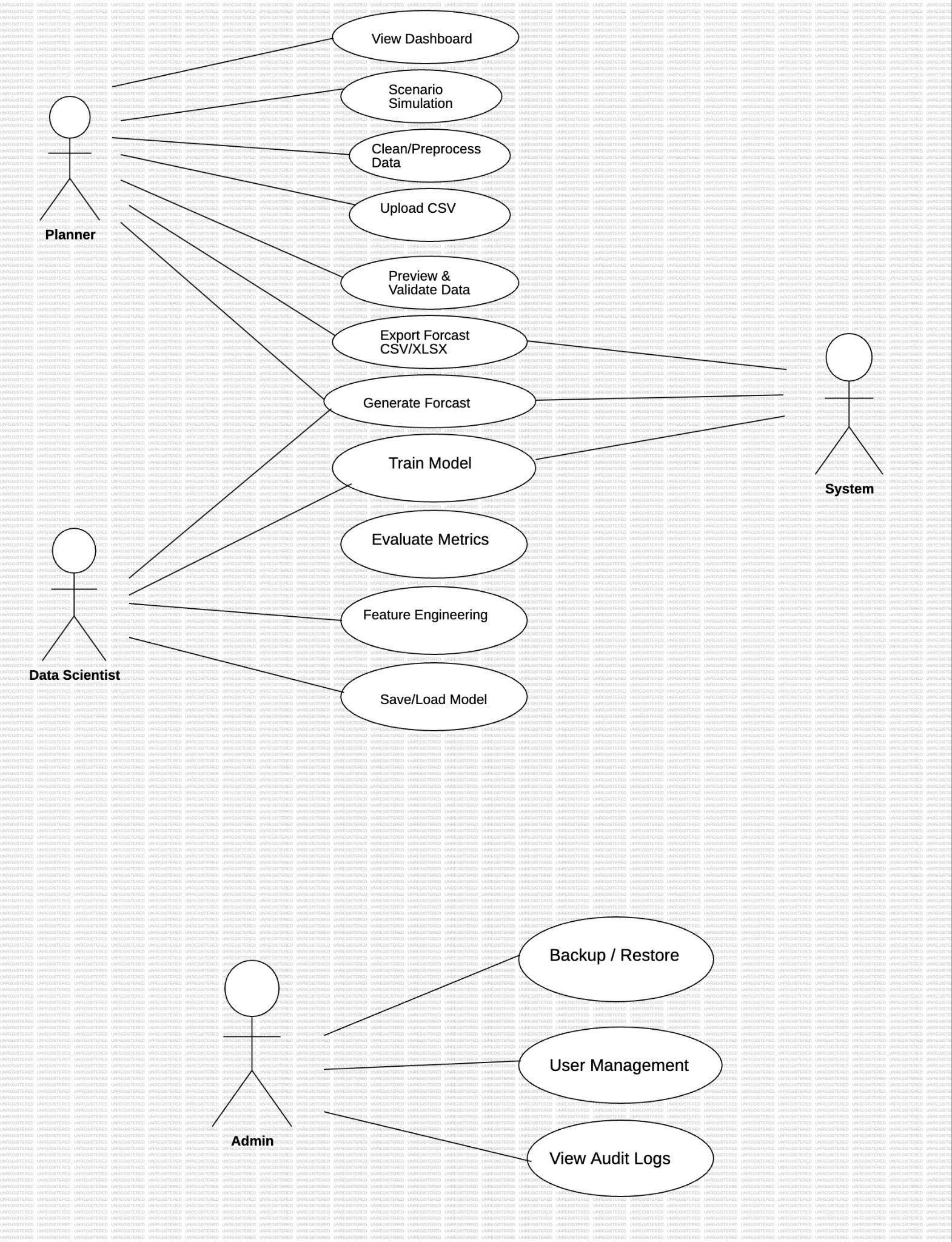
### **7.1 Acceptance criteria (exit conditions)**

* All **High-priority** functional requirements implemented and passing tests.
* Core NFRs for prototype demonstrated (performance on a development laptop, maintainability).
* No critical security vulnerabilities outstanding.
* RTM completed linking high-priority requirements to tests and implementation modules.

### **7.2 Acceptance Test suites**

* Ingestion Tests
* Preprocessing Tests
* Modeling Tests
* Forecast Tests
* Security Tests
* Operational Tests

## **8. System models and diagrams**



# **9)Full Requirements Traceability Matrix**

| **Req ID** | **Short description** | **Module** | **Test case** | **Owner** | **Priority** | **Status** |
| --- | --- | --- | --- | --- | --- | --- |
| DF-F-001 | CSV upload + preview | Ingestion | TC-ING-01 | B | High | N |
| DF-F-002 | DB connector (optional) | Ingestion (future) | TC-ING-02 (opt) | A | Low (Opt) | N |
| DF-F-003 | Schema detection & validation | Ingestion | TC-ING-03 | B | High | N |
| DF-F-004 | Missing-value handling | Preprocessing | TC-PRE-01 | C | High | N |
| DF-F-005 | Outlier detection & correction | Preprocessing | TC-PRE-02 | C | Medium | N |
| DF-F-006 | Feature engineering (time features/exog) | Feature Engine | TC-FE-01 | C | High | N |
| DF-F-007 | Model selection UI | Model Engine / UI | TC-MOD-01 | B/C | High | N |
| DF-F-008 | Time-series CV & metrics | Model Engine | TC-MOD-02 | C | High | N |
| DF-F-009 | Hyperparameter tuning (basic/opt) | Model Engine | TC-MOD-03 | C | Medium | N |
| DF-F-010 | Forecast generation + intervals | Forecast Engine | TC-FC-01 | A/C | High | N |
| DF-F-011 | Model store & versioning (SQLite/files) | Model Registry | TC-MOD-04 | A | Medium | N |
| DF-F-012 | Forecast dashboard | UI | TC-UI-01 | B | High | N |
| DF-F-013 | Scenario simulation | UI / Engine | TC-UI-02 | B/C | Medium | N |
| DF-F-014 | Export CSV/XLSX + API (opt) | API / UI | TC-API-01 | A/B | High | N |
| DF-F-015 | Anomaly detection & alerts (basic) | Ops / Preproc | TC-OPS-01 | C | Medium | N |
| DF-F-016 | Scheduler / Manual Run | Ops | TC-OPS-02 | A | Low-Med | N |
| DF-F-017 | User mgmt & RBAC (prototype) | Auth / UI | TC-SEC-01 | A/B | Medium | N |
| DF-F-018 | Audit logs (basic) | Logging / Ops | TC-OPS-03 | A | Medium | N |
| DF-NF-001 | Performance (<=30s on laptop) | Model Engine | TC-PERF-01 | C/A | High | N |
| DF-NF-002 | Scalability (sequential training) | Model Engine | TC-PERF-02 | A/C | Medium | N |
| DF-NF-003 | Availability (demo) | Deployment / Ops | TC-OPS-04 | A | Low | N |
| DF-NF-004 | Maintainability & tests (60% core) | Repo / Dev | TC-DEV-01 | All | Medium | N |
| DF-NF-005 | Explainability & traceability | UI / Model Engine | TC-UX-03 | C/B | High | N |
| DF-SR-001 | TLS for remote hosting | API / Ops | TC-SEC-02 | A | High (if remote) | N |
| DF-SR-002 | Auth & session management (bcrypt/argon2) | Auth | TC-SEC-01 | A | High | N |
| DF-SR-003 | Lightweight RBAC | Auth / UI | TC-SEC-01 | A/B | High | N |
| DF-SR-004 | CSV sanitization & input validation | Ingestion | TC-SEC-02 | B | High | N |
| DF-SR-005 | Secrets & config via env vars | DevOps | TC-DEV-02 | A | High | N |
| DF-SR-006 | Logging & audit trail | Logging / Ops | TC-OPS-03 | A | Medium | N |
| DF-SR-007 | Backup & restore script | Ops | TC-OPS-04 | A | Medium | N |
| DF-SR-008 | Dependency vulnerability check | Dev | TC-DEV-02 | All | Medium | N |
| DF-SR-009 | Data minimization & retention policy | Documentation | TC-SEC-05 | All | Medium | N |
| DF-SR-010 | Secure defaults (DEBUG off) | Config / Repo | TC-DEV-03 | A | Medium | N |

Notes:

* Owner mapping: **A = Backend/Infra**, **B = Frontend/UX**, **C = Modeling/ML**. Adjust owners to fit your team.
* Status: start with **N**; update to **P** or **A** as you implement/test.
* Test case IDs (TC-\*) correspond to the test hints/acceptance criteria in the SRS. Flesh each TC into step-by-step test scripts in your test plan.