# Missile Model / TEWA Project – Comprehensive Progress Report

Project: Threat Evaluation & Weapon Assignment (TEWA) System  
Owner: Hardy (with ChatGPT support)  
Date: 01 Oct 2025 (IST)  
Technology Stack: Python 3.10, Django 5.1, PostgreSQL, REST API, Angular (Frontend), LuciadRIA (GIS), CSV Import, Unit & Geodesy Helpers, UDP Messaging

## 1. Executive Summary

The TEWA system has been fully scaffolded, backend models defined, and core kinematics/math modules implemented. We have consolidated API routes, standardized geometry/engagement services for UDP messaging, implemented scoring and normalization functions, and established CSV-based track import. Core unit and geodesy helpers were added, and initial seed data fixtures for scenarios, defended assets, tracks, and model parameters were created. Testing coverage for kinematics, scoring, and API endpoints has been implemented. The system is now ready for frontend integration, bulk scoring enhancements, and real-time track ingestion.

## 2. Project Scaffold & Setup

Objective: Initialize project structure for backend and basic API endpoints.

Actions Taken:

* Django project missile\_model created.
* Applications core and tewa created within project structure.
* Virtual environment .venv configured.
* PostgreSQL database wired, migrations applied, and superuser created.
* Basic REST endpoints scaffolded (API root, health check, TEWA API).

Commands Executed:

django-admin startproject missile\_model

python manage.py startapp core

python manage.py startapp tewa

python manage.py migrate

python manage.py createsuperuser

Status: ✅ Completed

## 3. Core Domain Models

Objective: Define primary entities and relationships for TEWA.

Models Implemented:

1. DefendedAsset – Name, lat/lon, radius, TimeStamped, indexes on name.
2. Scenario – Start/end times, notes, TimeStamped for operational runs.
3. Track – Aircraft track info, optional scenario FK, unique (scenario, track\_id), spatial index (lat, lon), TimeStamped.
4. TrackSample – Time-stamped track samples, unique (track, timestamp), TimeStamped.
5. ThreatScore – Computed threat score per (scenario, track, DA) with indices.
6. ModelParams – Weights and scale factors for CPA/TCPA/TDB/TWRP, clamping options, TimeStamped.

Migrations Applied:

python manage.py makemigrations tewa

python manage.py migrate

Notes:

* Track timestamp renamed to created\_at to align with TimeStamped model.
* Unique constraints and indexes added to optimize scenario queries.

Status: ✅ Completed

## 4. Units & Geodesy Baseline

Objective: Implement unit conversions and geodesic helpers for kinematics calculations.

Actions Taken:

* Defined core units (meters, seconds).
* Implemented planar and geodesic conversion functions.
* Added ENU (East-North-Up) transformations for Lat/Lon coordinates.
* Helpers used in kinematics and scoring computations.

Testing:

python manage.py test core.tests -v 2

Status: ✅ Completed

## 5. Seed Data & Fixtures

Objective: Populate initial development dataset with sample scenarios, DAs, tracks, and model parameters.

Actions Taken:

* Fixture tewa/fixtures/tewa\_seed.json created containing:
  + 1 Scenario
  + 2 Defended Assets (DA-Alpha, DA-Bravo)
  + 3 Tracks (T1, T2, T3)
  + TrackSamples for each Track
  + ModelParams
* Management command seed\_demo implemented for repeatable seeding.
* Fixed fixture errors after adding TimeStamped fields (created\_at, updated\_at).

Commands:

python manage.py loaddata tewa\_seed

python manage.py seed\_demo

Status: ✅ Completed

## 6. Kinematics Math Module

Objective: Implement deterministic threat evaluation calculations.

Implemented Models:

1. CPA – Closest Point of Approach
2. TCPA – Time to CPA
3. TDB – Time to DA Boundary
4. TWRP – Time to Weapon Release Point

Notes:

* ENU-based coordinate transformations used.
* Functions return raw and normalized values.
* Edge cases handled: negative TCPA, moving away, stationary targets.

Testing:

python manage.py test core.tests -v 2

Status: ✅ Completed

## 7. Normalization & Scoring

Objective: Combine CPA/TCPA/TDB/TWRP into a single threat score.

Actions Taken:

* Normalized metrics using scales defined in ModelParams.
* Weighted scoring using w\_cpa, w\_tcpa, w\_tdb, w\_twrp.
* Scores clamped to [0,1] if required.
* Functions decoupled using ParamsLike protocol for type safety.

Testing:

* Unit tests for scoring functions passed.

Status: ✅ Completed

## 8. Scenario Engine

Objective: Compute threat scores for all tracks in a scenario against selected DAs at a given timestamp.

Actions Taken:

* compute\_scores\_at\_timestamp() implemented for timestamp-based scoring.
* Supports method="linear" (interpolation) and method="latest".
* Scores persisted to ThreatScore.
* API endpoint /api/tewa/compute\_at implemented and tested.

Testing:

python manage.py test tewa.tests.test\_api\_compute\_at -v 2

Status: ✅ Completed

## 9. CSV Import for Tracks

Objective: Support bulk track and TrackSample ingestion via CSV.

Actions Taken:

* CSV parsing handles: track\_id, lat, lon, alt\_m, speed\_mps, heading\_deg, timestamp.
* Management command import\_tracks implemented.
* Default timestamp logic added to handle missing values.
* Example CSV:

track\_id,lat,lon,alt\_m,speed\_mps,heading\_deg,timestamp

T1,28.6139,77.209,3500,220,200,2025-09-30T06:05:00Z

T2,28.7,77.1,3200,250,170,2025-09-30T06:06:00Z

T3,28.5,77.6,4800,210,315,2025-09-30T06:07:00Z

Command:

python manage.py import\_tracks docs/tracks.csv

Status: ✅ Completed

## 10. TEWA Backend Refactor

Objective: Consolidate routing, geometry services, and UDP message flows.

Actions Taken:

* apps/engagements/api/urls.py merged and deduplicated.
* Function-based endpoints grouped by domain (tracks/, trajectory/, engagements/, ops/).
* Geometry services refactored (apps/engagements/services/geometry.py):
  + Standardized message headers and byte field handling
  + Helpers for \_bytes\_fixed, \_payload, \_maybe\_len
  + Geodesic math utilities cleaned (get\_heading\_from\_position, DistanceBetweenPos)
  + Lifecycle methods consolidated (ERASE, NO-WA, Disengage, Ban-Zone alerts)
* Aliased imports to prevent drift and duplication.
* Preserved legacy UDP broadcast/unicast behaviors.

Verification:

* HTTP smoke tests (health, tracks, feasible allocation, TEWA compute)
* UDP capture tests for correct destinations and port order
* Manual lifecycle validation of ERASE → NO-WA → Disengage

Status: ✅ Completed

## 11. Tests & Verification

Scope:

* Unit tests for kinematics, scoring, and CSV import
* Integration tests for scenario engine and API endpoints
* Manual verification of UDP flows and geometry lifecycle
* All tests passed and type-checking errors resolved

Status: ✅ Completed

## 12. Developer Ergonomics

* Type hints added across modules.
* ParamsLike protocol decouples scoring logic from ORM models.
* Serializer payloads cast to Mapping and nested dicts guarded.
* Pylance warnings resolved for attribute access and argument types.

## 13. Operational Notes

* Run migrations carefully for renamed fields (track.timestamp → track.created\_at).
* Fixtures require created\_at / updated\_at fields for TimeStamped models.
* Environment variables govern database, CORS, and debug settings.

Useful Env Example:

DJANGO\_SECRET\_KEY=change-me

DJANGO\_DEBUG=True

DJANGO\_ALLOWED\_HOSTS=127.0.0.1,localhost

DJANGO\_CSRF\_TRUSTED\_ORIGINS=http://localhost:8000

CORS\_ALLOWED\_ORIGINS=http://localhost:4200

USE\_SQLITE=True

## 14. Next Steps / Pending Tasks

1. TEWA API enhancements (bulk scoring, filtering, thresholds)
2. Angular frontend integration (heatmaps, dashboards)
3. Real-time GPS/IMU track ingestion
4. Extended CSV features (batch import, validation)
5. Optional: JWT auth, pagination, and telemetry hooks
6. Unit tests for edge cases: negative TCPA, stationary targets, already-inside DA

## 15. Quick Reference (API Endpoints)

| Domain | Endpoint |
| --- | --- |
| Health | /api/health/ |
| Tracks | /tracks/list/, /tracks/latest/, /tracks/ingest/ |
| Trajectory | /trajectory/compute/, /trajectory/height-chart/ |
| Engagements | /engagements/summary/, /engagements/osa/proxy/, /engagements/igla/status/, /engagements/tewa/, /engagements/orders/send/, /engagements/orders/ack/, /engagements/allocate/, /engagements/assign-widget/, /engagements/feasible-solution/ |
| TEWA Compute | /api/tewa/score/, /api/tewa/compute\_at/ |

## 16. Project Status Summary

| Task | Status |
| --- | --- |
| Project scaffold & setup | ✅ Completed |
| Core domain models | ✅ Completed |
| Units & geodesy baseline | ✅ Completed |
| Seed data fixtures | ✅ Completed |
| Kinematics math module | ✅ Completed |
| Normalization & scoring | ✅ Completed |
| Scenario engine | ✅ Completed |
| CSV import (tracks) | ✅ Completed |
| TEWA Backend refactor | ✅ Completed |
| Testing & verification | ✅ Completed |

Overall Progress: ~95–100% for backend core functionality.

This document contains everything done so far, including scaffolding, models, fixtures, kinematics, scoring, CSV import, API endpoints, UDP geometry services, refactors, and tests. It’s comprehensive enough for another AI or developer to understand the current state of the project and continue development.