Project Name	...
Online team meeting	https://fau.zoom-x.de/j/3913240515?pwd=TWo2NkZmaVZFWDVKckNKVzQyRG4zdz09
Production system (if any)	TBT
Test system (if any)	TBT
GitHub repository	https://github.com/amosproj/amos2025ws01-opensearch-load-tester
GitHub feature board	https://github.com/orgs/amosproj/projects/89/views/2
GitHub imp-squared backlog	https://github.com/orgs/amosproj/projects/93/views/1
Team T-shirt (white)	https://www.shirtinator.de/en/s/nasAq-u7RjO3FQ4BPPUSLw
Team T-shirt (black)	https://www.shirtinator.de/en/s/0VuwTYIwRviJfsV7ERwamA
Additional materials	...
Team maling list	oss-amos-proj1@lists.fau.de

Last Name	First Name	GitHub User Name	Email Address
Lorenz	Alexander	Hydraneut	alexander.lorenz@fau.de
Buchner	Lea	LeaBuchner	lea.buchner@fau.de
Engelhard	Dirk	engelharddirk	dirk.engelhard@fau.de
Strachwitz	Carlo	Carlit0	carlostrachwitz@gmail.com
Belz	Sara	SaraBelz	sara.belz@fau.de
Becker	Eugen	BeEugen	eugen.becker@fau.de
Hofmann	Leo	Leolingio	leohof23@gmail.com
Knecht	Sebastian	SeboKnt	sebastian.knecht@fau.de

		Product Owner					
#	Meeting Day	Review	Planning	Software Developer	Release Manager	Scrum Master	Comment
1	2025-10-15	both	both	Everyone else	N/A	Alexander Lorenz	
2	2025-10-22	Lea Buchner	Dirk Engelhard	Everyone else	N/A	Alexander Lorenz	
3	2025-10-29	Dirk Engelhard	Lea Buchner	Everyone else	Carlo Strachwitz	Alexander Lorenz	
4	2025-11-05	Lea Buchner	Dirk Engelhard	Everyone else	Carlo Strachwitz	Alexander Lorenz	**Build process review**
5	2025-11-12	Dirk Engelhard	Lea Buchner	Everyone else	Leo Hofmann	Alexander Lorenz	
6	2025-11-19	Lea Buchner	Dirk Engelhard	Everyone else	Sara Belz	Alexander Lorenz	
7	2025-11-26	Dirk Engelhard	Lea Buchner	Everyone else	Sebastian Knecht	Alexander Lorenz	Mid-term due
8	2025-12-03	Lea Buchner	Dirk Engelhard	Everyone else	Eugen Becker	Alexander Lorenz	
9	2025-12-10	Dirk Engelhard	Lea Buchner	Everyone else	Eugen Becker	Alexander Lorenz	
10	2025-12-17	Lea Buchner	Dirk Engelhard	Everyone else	Leo Hofmann	Alexander Lorenz	Team Workshop
11	2026-01-07	Dirk Engelhard	Lea Buchner	Everyone else	Sara Belz	Alexander Lorenz	No class but team meeting
12	2026-01-14	Lea Buchner	Dirk Engelhard	Everyone else	Sebastian Knecht	Alexander Lorenz	
13	2026-01-21	Dirk Engelhard	Lea Buchner	Everyone else	Leo Hofmann	Alexander Lorenz	
14	2026-01-28	Lea Buchner	Dirk Engelhard	Everyone else	Sara Belz	Alexander Lorenz	
15	2026-02-04	Dirk Engelhard	Lea Buchner	Everyone else	Sebastian Knecht	Alexander Lorenz	Demo day!+retrospective
Product owners, software developers, and Scurm Master are set and ideally don't change over time; the critical part is the Release Manager role you need to define here							

Goals	achieve the project goal
	get inside into the scrum and agile process
	improve practical development skills
	create something worth using
Meeting norms	everybody should feel safe
	be punctual
	Absence from meetings should be (if possible) declared a day in advance
Working norms	open door policy (everybody should be allowed to do mistakes)
	maintaining good documentation
	Backlog items are assigned by the developers themselve before they work (outside the team meeting)
Coordination norms	every Backlog item has at least 1 deticated person (if work has already started on it)
	Contributers assigne them self to bakclog items
Communication norms	meetings take place in english unless stated otherwise
	stick to the topic of the channel
	We use discord as our primary method of communication
	we check discord at least once a day (and react if needed)
	We use Whatsapp for urgent communication (less than an hour)
Consideration norms	If you see something say something
	We decide with majority vote
	If we think we could help we extend it
Cont. improvement norms	constructive feedback is always welcome
	We jointly review the happiness index
Rewards	celebrate successes
	Appreciation and praise
Sanctions	apoligize sincerly
	and find a funny way to apoligize
Signatures	
Scrum Master	Alexander Lorenz
Product owner	Dirk Engelhard
Product owner	Lea Buchner
Software developer	Leo Hofmann
Software developer	Eugen Becker
Software developer	Sebastian Knecht
Software developer	Sara Belz
Software developer	Carlo Strachwitz

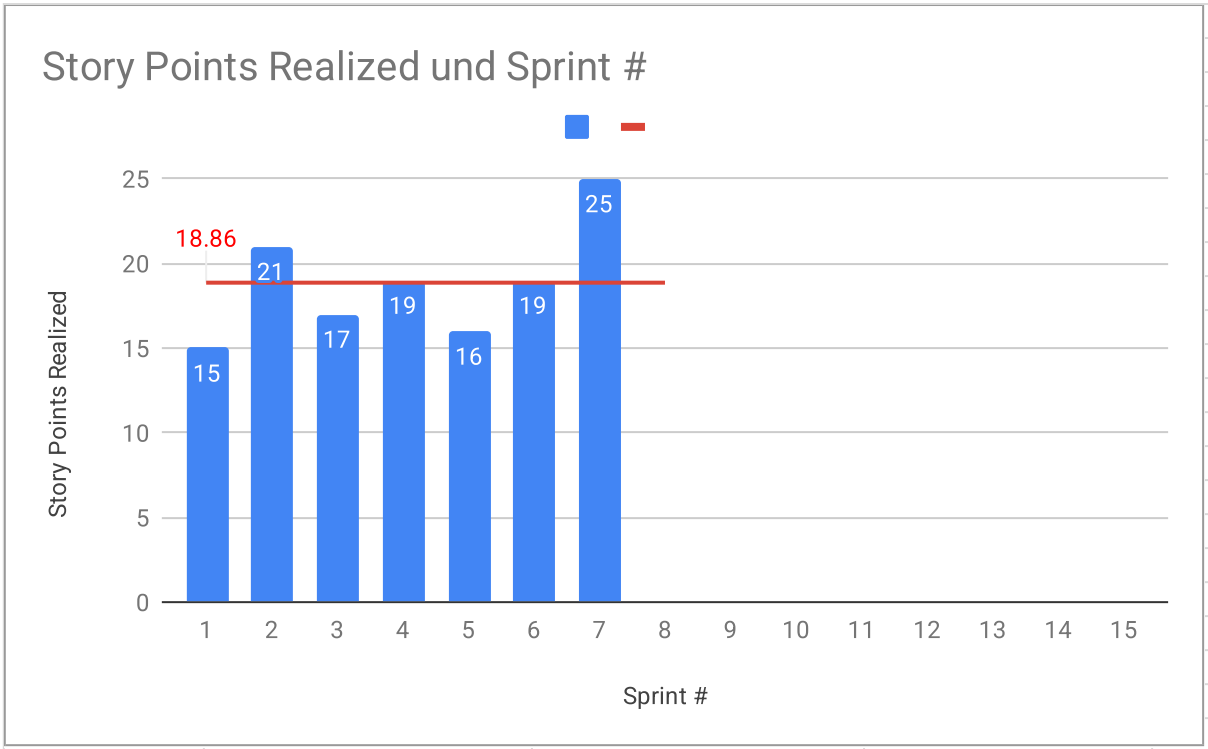
Product Vision	Project Mission
Make OpenSearch deployments predictable, right-sized, and resilient by providing a standard, evidence-based benchmarking and load-testing platform that reveals real-world limits and behaviors.	Build a modular, extensible load-tester that reliably measures OpenSearch performance limits and produces reproducible artifacts for capacity planning and reliability engineering.

Term	Definition
OpenSearch	A search and analytics engine (Elasticsearch fork). In this project, OpenSearch is the target system for load and ingest testing.
Load Generator / load-generator	Component that produces requests/load against OpenSearch to perform performance and stress tests. Located in the load-generator module.
Load Test	A planned execution that runs a certain number of queries, a load pattern, or a duration against OpenSearch to measure behavior, throughput and latency.
Load Runner / LoadRunnerService	Service component that orchestrates query execution for a test (scheduling, threads, rate limiting, metric collection).
Query (Search Query)	A single search or aggregation request, represented as a JSON template under src/main/resources/queries/ (e.g. q1_ano_payroll_range.json). Templates may include parameters/placeholders.
Query Template	JSON template for a query with placeholders to be filled at runtime.
QueryRunRequest	DTO describing a request to run a query in the load generator (which query, repetitions, rate, etc.). Found in controller/QueryRunRequest.java.
Query Execution / QueryExecution	Abstraction for executing a query against a target. Implementations perform the actual call (or a no-op for dry runs).
OpenSearchQueryExecution	Concrete QueryExecution implementation that sends requests to OpenSearch.
NoOpQueryExecution	A test or dry-run implementation of QueryExecution that does not send real requests.
QueryExecutionFactory	Factory producing appropriate QueryExecution instances depending on configuration or target.
QueryRegistry	Registry or catalog of available query templates and metadata (names, paths, parameters).
Index	OpenSearch index: a logical collection of documents defined by mappings.
Document	A single JSON record stored in an index. Test data is created and indexed as documents.
Mapping	Schema-definition for an index (fields and types).
Aggregation	OpenSearch operation to group or summarize data (counts, sums, bucket aggregations). Examples: q5_ano_clients_aggregation.json.
Range Query	A query that filters documents based on numeric or date ranges (e.g., payroll ranges).
Match / Term Query	Full-text or exact-match queries in OpenSearch.
Bulk API / Bulk Indexing	Mechanism for inserting many documents in a single request to increase throughput.
Ingestion Rate	Number of documents indexed per time unit.
Throughput	Number of successfully processed requests per second (often QPS).
QPS (Queries Per Second)	Metric indicating how many queries are executed per second.
Concurrency	Number of parallel requests/threads during a test.
Latency	Time between request and response. Typical metrics: average, p95, p99.
Response Time	General term for latency; often reported with distribution statistics.
Error Rate	Fraction of failed responses (HTTP errors, timeouts) out of total requests.
Metrics Collector / metrics-reporter	Component that gathers runtime metrics (latency, throughput, errors) and exports or stores them. See metrics-reporter module.
MetricsCollectorService	Service in load-generator that aggregates metrics from test runs and forwards them to a reporter or sink.
Test Data	Data that is indexed into OpenSearch for testing, often generated synthetically (see testdata-generator).
testdata-generator	Module responsible for generating and optionally storing test datasets (models: AnoRecord, DuoRecord).

Term	Definition
Ano / AnoRecord	Domain model class in the testdata-generator module. Represents a specific record schema used by some queries (fields defined in the model file).
Duo / DuoRecord	Another domain model in testdata-generator with a different schema; used by queries referencing duo data.
Recordable	Interface/abstraction implemented by AnoRecord and DuoRecord to provide serialization or indexing behavior.
PersistentDataGeneratorService	Service that persists generated test data (e.g., to files or external storage).
DynamicDataGeneratorService	Service that generates test data dynamically at runtime (randomized or parameterized fields).
FileStorageService	Utility/service for reading and writing generated test data to/from files.
OpenSearchDataService	Component that communicates with OpenSearch to index, delete, or query data.
Client (business entity)	Business object representing a customer/client. Queries like q7_duo_client_by_customer_number.json reference client data.
Booking	Business domain entity (e.g., a financial booking or reservation). Appears in queries like q4_duo_booking_by_client_and_state.json.
SLA (Service Level Agreement) - general concept	Target or acceptance values for latency/throughput/error rates used to validate system behavior under load.
Dry-run	Mode where queries are not actually sent to OpenSearch (see NoOpQueryExecution).
Warmup	Pre-measurement phase in load tests to warm caches and stabilize the system before capturing metrics.
Backpressure	System behavior when the target cannot keep up with the requested throughput; relevant to error rates and latency spikes.

Sprint #	Sprint goal
1	None
2	None
3	None
4	Optional
5	Internal component integration of each service
6	First End-To-End test run
7	Add Warm-Up and Ramp-Up Phase
8	Query capabilities
9	Query randomization
10	Distributed Load Execution
11	Query Scenarios
12	Visualization for demo
13	Stabilization, tuning, packaging
14	DEMO-DAY readiness
15	

Sprint #	Story Points Realized	Avg
1	15	18.86
2	21	18.86
3	17	18.86
4	19	18.86
5	16	18.86
6	19	18.86
7	25	18.86
8		18.86
9		18.86
10		18.86
11		18.86
12		18.86
13		18.86
14		18.86
15		18.86
	PLEASE CREATE THE VELOCITY CHART ON A NEW TAB USING THE DATA FROM THIS TAB	

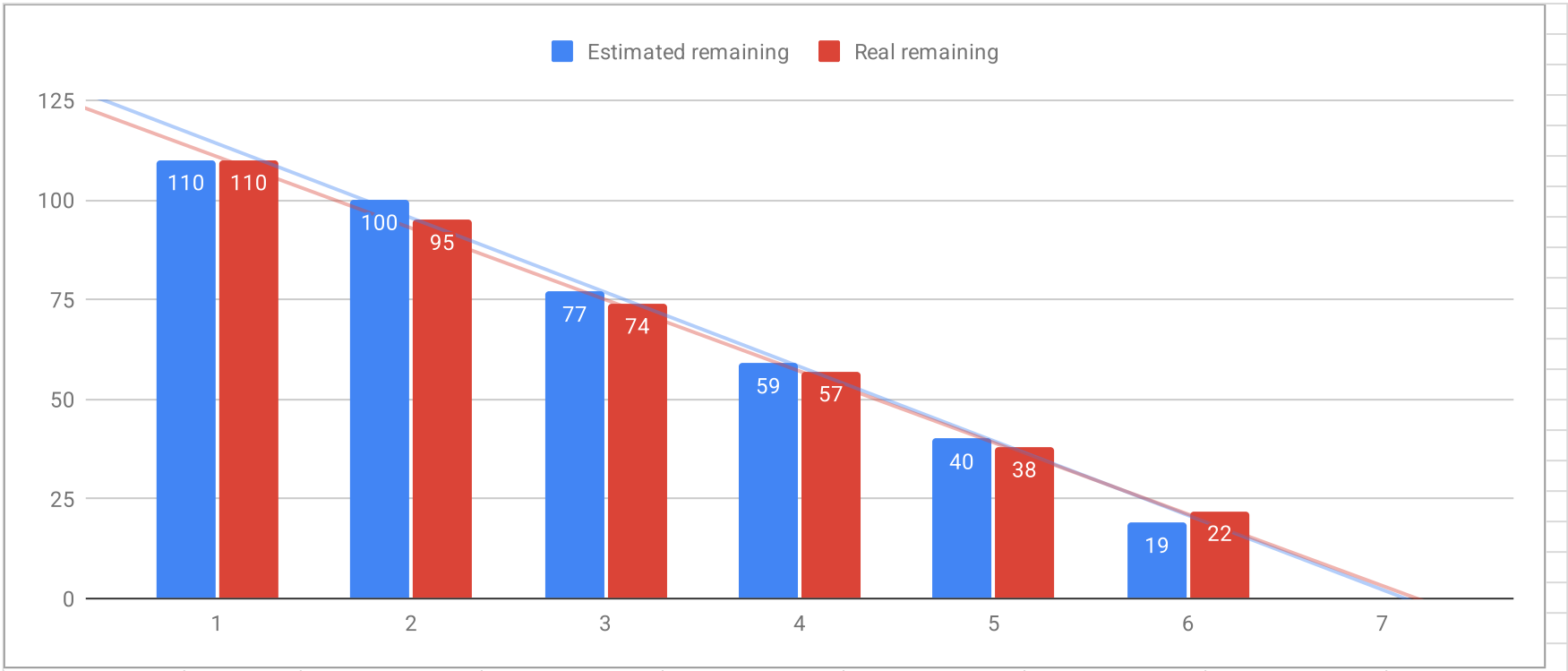


Average Speed:	18.86	Points/Sprint		

Sprint	Goal	Feature Name	Est. size	Est. remaining	Real size	Real remaining
Release						
Total			110	110		
Sprints						
1	**Deliver first increment of initialized software modules and an architecture plan**		10	110	15	110
2	Deliver increment with Build process implementation and first module prototypes		23	100	21	95
3	**Deliver increment with broad prototype implementation coverage**		18	77	17	74
4	Deliver increment with first component integrations implemented		19	59	19	57
5	**Deliver increment with more integration and technical test running capability**		21	40	16	38
6	Deliver Increment with simple full test run routine executable		19	19	19	22
7						
Features						
1	**Deliver first increment of initialized software modules and an architecture plan**					
		Research - OpenSearch	3		3	
		Research - Understand DATEV Data Structure (Industry Partner Input) for Load Tester Query Design	1		1	
		Developer Environment Setup	1		1	
		GitHub Repository Management Decision			2	
		Architecture evaluation/decision	2		5	
		Design Team Shirt	1		1	
		Initialize SBOM				
		Minimal Module Initialization	2		2	
2	**Deliver increment with Build process implementation and first module prototypes**					
		Define Representative Query Scenarios	5		3	
		Prepare Build Process Review	2		2	
		Research - Benchmark Tools for OpenSearch Load Tester	3		3	
		OpenSearchClient prototype	3		3	
		Build Process Decision	2		2	
		Data Generation Prototype	3		3	
		Architecture adaptation	2		2	
		Build Process Implementation	2		2	
		Github Signoff/Co-Author Automation	1		1	
3	**Deliver increment with broad prototype implementation coverage**					
		MetricsCollector Prototype Implementation	5		3	

Sprint	Goal	Feature Name	Est. size	Est. remaining	Real size	Real remaining
		QueryExecution Prototypee Implementation	3		2	
		LoadRunner Component Prototype Implementation	2		2	
		Logging Mechanism Implementation	2		2	
		Data Generation <> OpenSearch Client Integration	3		5	
		Environment Variable Support for Dockerized Services	2		2	
		Automatic Shutdown of Testdata Generator Service	1		1	
4	**Deliver increment with first component integrations implemented**					
		MetricsCollector Prototype Implementation	5		3	
		QueryExecution Prototype Implementation	3		2	
		LoadRunner Component Prototype Implementation	2		2	
		Data Generation with OpenSearch Client Integration	3		5	
		Logging Mechanism Implementation	2		2	
		Implementation of Batch Processing for Data Pre-Loading	3			
		Unified Code Styling & Editor Configurations	1		2	
		Ano/Duo Separation	3		3	
5	**Deliver Increment with internal component integrations broadly implemented**					
		Load Generator Port Setup Dockerized Services	3		2	
		Load Generator Internal Component Integration	5		5	
		ReportController Component Prototype Implementation	5		3	
		Build Process Video	2		2	
		Definition of Done	3		2	
		Implementation of MetricsReporterClient Prototype	3		2	
6	**Deliver Increment with simple full test run routine executable**					
		Implementation of Load Test Scenario	5		5	
		Implementation of Batch Processing for Data Pre-Loading	3		3	
		ReportCreator Component Prototype Implementation	5		8	
		Wiki Creation	3		3	
		CI/CD Pipeline Setup	3			

Mid-Project Burndown Chart



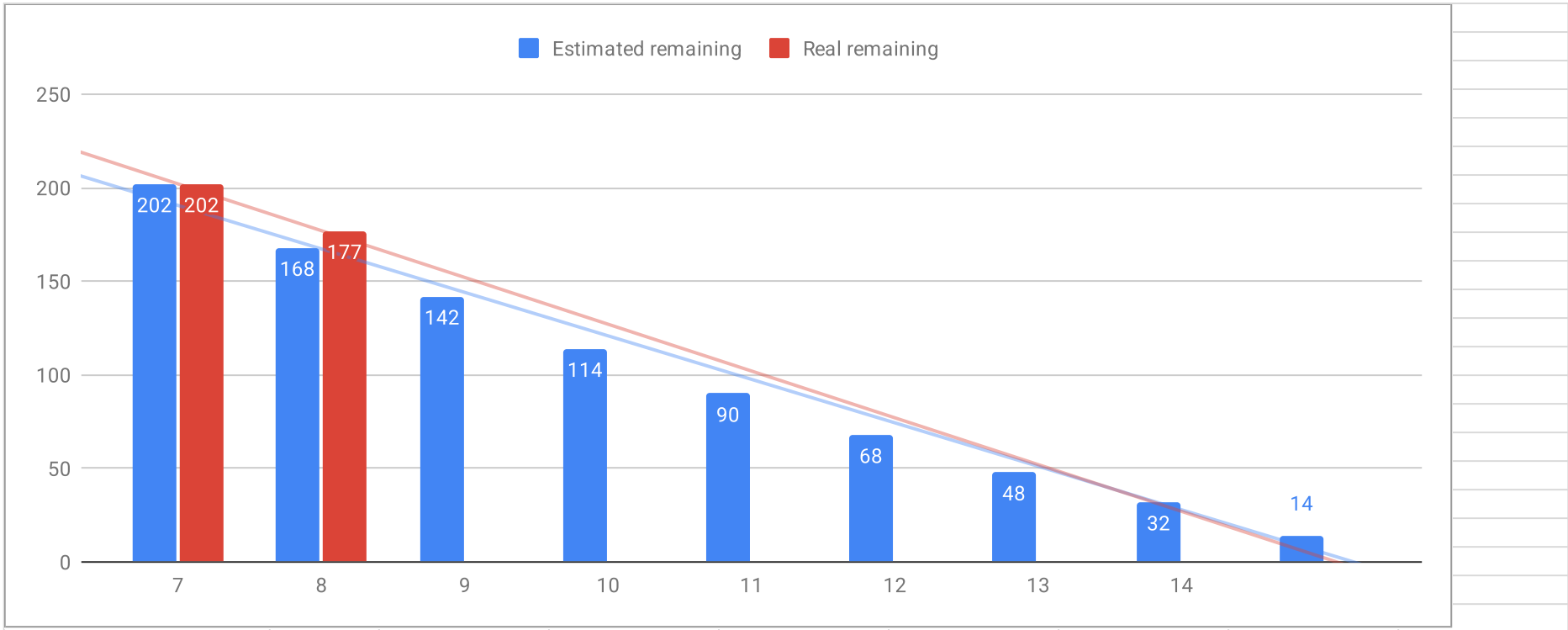
Development Speed:	18.86	Points/Sprint						

Sprint	Goal	Feature Name	Est. size	Est. remaining	Real size	Real remaining
Release						
Total			202	202		
Sprints						
7	**Deliver Increment with full test run executable supporting warm-up and synchronized loa**		34	202	25	202
8	**Deliver Increment with impoved query capabilities**		26	168	0	177
9	**Deliver Increment with broad query randomization**		28	142		
10	**Deliver Increment with Distributed load execution**		24	114		
11	**Deliver Increment with set of query scenarios**		22	90		
12	**Deliver Increment with a data report visualization for demo purposes**		20	68		
13	**Deliver Increment with Stabilization, tuning, packaging**		16	48		
14	**Deliver Increment with finished, DEMO-DAY-Ready product**		18	32		
			14	14		
				114		0
Features						
7	**Deliver Increment with full test run executable supporting warm-up and synchronized load start**					
		Complex Queries	3			
		Big Data Handling in MetricsReporter	5		5	
		Code Refactoring	5		5	
		Shared Library Creation	1		1	
		Warm-Up Phase	3			
		Configurable Queries per Second	3		5	
		Load Generator Request Synchronization	5			
		Optimize Random Generation of Test Data	3		3	
		Configurable Test Duration	3		3	
		Load Generator functionality Definition	3		3	
8	**Deliver Increment with impoved query capabilities**					
		Query Parsing Engine Improvements	4			
		Support for Multi-field Search Queries	3			
		Support for Aggregation Queries	5			
		Query Latency Sampling Enhancements	3			
		Improved Error Handling for Malformed Queries	2			
		Configurable Query Weighting	5			
		Documentation for Advanced Query Usage	2			

Sprint	Goal	Feature Name	Est. size	Est. remaining	Real size	Real remaining
9	**Deliver Increment with broad query randomization**					
		Random Query Template Generator	6			
		Value Pool Expansion (IDs, Categories, Date Ranges)	4			
		Randomized Query Complexity Levels	5			
		Random Payload Generation for POST Queries	4			
		Probabilistic Query Path Selection	3			
		Random Scenario Injection (latency spikes, errors)	4			
10	**Deliver Increment: Distributed load execution**					
		Multi-Node Load Generation Support	6			
		Coordinator Node Architecture	4			
		Load Distribution Algorithm	4			
		Resilient Messaging Between Nodes	4			
		Distributed Metrics Aggregation	4			
11	**Deliver Increment: Set of query scenarios**					
		Define Scenario Structure & DSL	5			
		Pre-built Scenarios (Search, Aggregations, Mixed)	5			
		Time-based Scenario Control	4			
		Scenario Looping & Weighted Groups	4			
		Scenario Summary & Reporting	4			
12	**Deliver Increment: Data report visualization for demo**					
		Basic Web Dashboard	4			
		Real-time Graphs (QPS, Latency, Errors)	5			
		Trend Lines & Historical Data Comparison	3			
		Export Reports (PDF/CSV)	4			
		Dashboard Mobile-Friendly UI	2			
		Demo Visualization Themes	2			
13	**Deliver Increment: Stabilization, tuning, packaging**					
		Load Generator Optimization (GC, Memory)	4			
		Query Execution Tuning	3			
		Packaging into Standalone Binary	3			
		End-to-End Stability Tests	4			
		Documentation Cleanup	2			
14	**Deliver Increment with finished, DEMO-DAY-Ready produc**					
		Final Demo Workflow	3			
		Polished Dashboard & UX	4			
		Scenario Library Complete	3			

Sprint	Goal	Feature Name	Est. size	Est. remaining	Real size	Real remaining
		Full System QA	4			
		Presentation + Demo Script	4			

Final Project Burndown Chart



Avg. development speed (across all sprints):	18.86	Points/Sprint

#	Feature Definition of Done	Sprint Release Definition of Done	Project Release Definition of Done
Overview	A single new feature or user story implemented in Trunk Load (e.g., new query type, new CLI option, new scenario).	The subset of functionality and artifacts that must be finished and deliverable at the end of a sprint (sprint-level deliverable).	The final project release: full handover of the stress-testing toolset, reproducible scenarios, baselines and deliverables.
Scope	Code, tests and docs needed to implement the feature in load-generator, metrics-reporter or testdata-generator.	All features/bugfixes scoped to the sprint board's "Done" column, plus required scenarios and deliverables in Deliverables/sprint-XX/.	All features and fixes required for the final deliverable, automated scenarios (ramp, steady, spike, soak, chaos), reporting assets and reproducibility instructions in Deliverables/.
Acceptance Criteria	- Implements the story's acceptance criteria from the issue. - At least one concrete example scenario demonstrating the feature exists. - No secrets added and no new critical static-analysis findings. - Excludes wide refactors unless explicitly requested. - Code compiles without errors locally and in CI. - Formatting and linting checks pass. - Minimal unit tests cover the relevant logic. - Code is documented where appropriate (Wiki/internal README).	- All sprint issues marked Done have merged PRs. - Each merged item satisfies its Feature or Bugfix DoD. - All completed features are integrated and do not break the full system. - Architectural guidelines are respected. - Full Docker Compose setup starts without errors: ○OpenSearch + Dashboards ○Load Generator ○Metrics Reporter / Metrics Storage (if present) ○Report Controller ○Logging stack (Promtail/Loki) if used○Metrics Reporter / Metrics Storage (if present) - The increment can be demonstrated in the Sprint Review without failures.	- All required scenarios implemented and automated. - Reproducible runs produce archived results (raw + aggregated charts). - Required performance/resilience targets met for mandatory scenarios. - Documentation is complete (test plan, runbook, interpretation guide, reproducibility checklist). - No open critical bugs. - All promised components implemented: ○Fully working Load Generator ○Query Execution + Templates ○Metrics Reporter ○Report Controller (prototype level is OK if IP handles the rest) ○Stable OpenSearch integration ○Data generation for "ano" and "duo" - Final code review approved by the team. - No critical warnings or dead code. - Repository clean with no unmerged branches.
Tests & Validation	- Unit tests: new/changed classes covered (place under module src/test/java). - Integration test: Testcontainers or short docker-compose integration that exercises the feature against OpenSearch. - Smoke run: a short end-to-end scenario (e.g., 30–60s) that demonstrates feature behavior and produces output. - Basic manual testing completed. - Logs checked and no unexpected errors appear. - Code review completed and approved by at least one teammate. - DTOs, controllers, YAML config files updated as needed	(- Run the sprint acceptance scenarios: population → run → collect → validate.) (- Validation scripts assert basic pass/fail (no critical errors, p95/p99 within sprint thresholds).) All commits pass CI: ○Build ○Unit tests ○Static analysis (lint) ○Formatting - No critical errors in logs.	- Full end-to-end validation runs executed (including at least one soak and one chaos test) and validated by scripts. - Regression/reproducibility check: repeat a representative scenario and verify results within accepted variance (e.g., ±10%). - Unit + integration suites green. - Docker Compose environment stable and reproducible. - Entire stack starts from scratch without errors. - Final load tests executed and documented. - Logs structured and free of critical problems. - Final demonstration with DATEV/IP completed. - Test coverage includes: ○Happy-path scenarios ○Error handling (e.g., OS down, timeout, invalid configs, multi-threading) - Minimal stress tests documented.
Documentation	- Usage example added to module README.md and/or wiki.	- Sprint notes updated in the wiki. - Architecture diagrams updated if anything changed. - API documentation updated with new endpoints.	- Deliverables/ contains final reports, scenario definitions, scripts to reproduce, dashboard exports and a README with exact reproduction commands. - Documentation/ contains runbook, interpretation guide, and troubleshooting steps, final architecture, docker setup, API endpoints, execution examples, "How to add new queries", "how to generate and upload data to OpenSearch" as part of wiki. - Final presentation delivered
CI / Automation (when existing)	- PR CI must run build + unit tests + linter. - Integration smoke test recommended in merge pipeline (if it's fast); otherwise on main/merge job.	- Merge/main pipeline runs smoke acceptance scenarios (short) or a job that can be manually triggered to run them. - CI artifacts for the sprint (result JSON/CSV and summary) are archived in pipeline artifacts.	- Release pipeline builds artifacts (JARs, Docker images) and archives results. - Nightly or release-run perf jobs executed and artifacts persisted; CI flags regressions compared to baselines. - Dependency/security scans completed.
Metrics & Thresholds	- If performance-related: attach baseline and verify no significant regression (>10% on critical metric) unless approved. - Smoke validation: error rate < 5% (adjustable per scenario) and p95 reported. PR Checklist - Issue/story linked - Unit tests added + passing - Integration/smoke test added - Documentation updated - Reviewer approved	- Sprint acceptance thresholds defined per scenario (e.g., for that sprint: p95 <= 600 ms, error rate <= 2%). - If thresholds not met, the sprint cannot be marked finished until remediation or explicit acceptance by PO. PR Checklist - All sprint issues linked and merged - Acceptance scenarios included and runnable - Deliverables/sprint-XX populated with results - CI artifacts archived - PO/TA acknowledged	- Project-level mandatory thresholds must be met (define concretely before final validation, e.g.): - Steady-state p95 <= X ms, p99 <= Y ms, error rate <= Z%. - Soak: sustained stability for N minutes without memory growth or node crashes. - Chaos: cluster remains available and recovers within R minutes. - Reproducibility: repeated runs within ±10% of throughput and latency targets (tunable). PR / Release Checklist - All scenarios automated and in repo - Deliverables folder contains final reports and artifacts - Release build artifacts created (JARs, Docker images) - Baselines and comparison scripts in place - Security/dependency checks done - PO/TA and technical reviewers sign off

#	Feature Definition of Done	Sprint Release Definition of Done	Project Release Definition of Done
Sign-off	At least one technical reviewer; feature owner or PO must approve if feature affects user behavior or public interfaces.	Product Owner acceptance of sprint deliverables and at least one technical reviewer sign-off.	Formal sign-off by Product Owner / Teaching Assistant and at least two technical reviewers; release tagged and artifacts archived.

Type	Link / reference
Wiki	https://github.com/amosproj/amos2025ws01-opensearch-load-tester/wiki

#	Context	Name	Version	License	Comment
1	environment	Eclipse Temurin OpenJDK	25 (LTS)	GPL-2.0 with Classpath Exception	Java runtime used for building and running the application
2	environment	Docker	28.5.1	Apache-2.0	Container environment for running OpenSearch and Load Tester services
3	tool	Apache Maven	3.9.11	Apache-2.0	Build automation and dependency management tool
4	library	spring-boot-starter	3.5.7	Apache-2.0	Entry point for Spring Boot dependency management
5	library	spring-boot	3.5.7	Apache-2.0	Core framework providing auto-configuration and runtime support
6	library	spring-context	6.2.12	Apache-2.0	Dependency injection and application context management
7	library	spring-aop	6.2.12	Apache-2.0	Aspect-oriented programming support
8	library	spring-beans	6.2.12	Apache-2.0	Bean creation and configuration framework
9	library	spring-expression	6.2.12	Apache-2.0	Spring Expression Language (SpEL) processing
10	library	spring-boot-autoconfigure	3.5.7	Apache-2.0	Automatically configures components based on classpath
11	library	spring-boot-starter-logging	3.5.7	Apache-2.0	Default logging configuration using Logback
12	library	logback-classic	1.5.20	EPL-1.0	Logging backend implementing SLF4J API
13	library	logback-core	1.5.20	EPL-1.0	Core utilities for Logback logging
14	library	log4j-to-slf4j	2.24.3	Apache-2.0	Redirects Log4j 2 logs to SLF4J
15	library	jul-to-slf4j	2.0.17	MIT	Bridges java.util.logging to SLF4J
16	library	jakarta.annotation-api	2.1.1	EPL-2.0	Jakarta annotations used by Spring components
17	library	spring-core	6.2.12	Apache-2.0	Core utilities and classloading framework of Spring
18	library	spring-jcl	6.2.12	Apache-2.0	Logging abstraction used internally by Spring
19	library	snakeyaml	2.4	Apache-2.0	YAML parser for configuration files
20	library	micrometer-core	1.15.5	Apache-2.0	Core metrics collection API
21	library	micrometer-commons	1.15.5	Apache-2.0	Shared utilities for metrics instrumentation
22	library	micrometer-observation	1.15.5	Apache-2.0	Captures timing and observation data for metrics
23	library	HdrHistogram	2.2.2	CC0-1.0	High-resolution latency measurement library
24	library	LatencyUtils	2.0.3	CC0-1.0	Helper utilities for latency tracking
25	library	micrometer-registry-prometheus	1.15.5	Apache-2.0	Exposes Micrometer metrics to Prometheus
26	library	prometheus-metrics-core	1.3.10	Apache-2.0	Core Prometheus metrics collection
27	library	prometheus-metrics-model	1.3.10	Apache-2.0	Data model for Prometheus metrics
28	library	prometheus-metrics-config	1.3.10	Apache-2.0	Configuration utilities for Prometheus metrics
29	library	prometheus-metrics-tracer-common	1.3.10	Apache-2.0	Common tracing utilities for Prometheus exporters
30	library	prometheus-metrics-exposition-formats	1.3.10	Apache-2.0	Serialization formats for Prometheus metrics
31	library	prometheus-metrics-exposition-textformats	1.3.10	Apache-2.0	Text exposition format for Prometheus

#	Context	Name	Version	License	Comment
32	library	lombok	1.18.34	MIT	Generates boilerplate code (getters/setters, builders, etc.)
33	library	opensearch-rest-high-level-client	2.16.0	Apache-2.0	REST client for interacting with OpenSearch clusters
34	library	opensearch	2.16.0	Apache-2.0	Core OpenSearch library
35	library	opensearch-common	2.16.0	Apache-2.0	Common interfaces and utilities for OpenSearch modules
36	library	opensearch-core	2.16.0	Apache-2.0	Core APIs for OpenSearch operations
37	library	opensearch-compress	2.16.0	Apache-2.0	Compression utilities used internally by OpenSearch
38	library	zstd-jni	1.5.5-5	BSD-2-Clause	Java bindings for Zstandard compression
39	library	opensearch-secure-sm	2.16.0	Apache-2.0	Security manager integration for OpenSearch
40	library	opensearch-x-content	2.16.0	Apache-2.0	Serialization/deserialization utilities in OpenSearch
41	library	jackson-dataformat-smile	2.19.2	Apache-2.0	Jackson module for Smile binary JSON
42	library	jackson-dataformat-cbor	2.19.2	Apache-2.0	Jackson module for CBOR binary JSON
43	library	opensearch-geo	2.16.0	Apache-2.0	Geospatial data and query support for OpenSearch
44	library	opensearch-telemetry	2.16.0	Apache-2.0	Telemetry and monitoring features for OpenSearch
45	library	lucene-core	9.11.1	Apache-2.0	Core search and indexing engine powering OpenSearch
46	library	lucene-analysis-common	9.11.1	Apache-2.0	Text analysis and tokenization utilities
47	library	lucene-backward-codecs	9.11.1	Apache-2.0	Backward-compatible codecs for old Lucene versions
48	library	lucene-grouping	9.11.1	Apache-2.0	Implements result grouping and field collapsing
49	library	lucene-highlighter	9.11.1	Apache-2.0	Highlighting support for search result snippets
50	library	lucene-join	9.11.1	Apache-2.0	Supports join queries between documents
51	library	lucene-memory	9.11.1	Apache-2.0	In-memory index structures for fast search
52	library	lucene-misc	9.11.1	Apache-2.0	Miscellaneous utilities and experimental features
53	library	lucene-queries	9.11.1	Apache-2.0	Advanced query utilities and filters
54	library	lucene-queryparser	9.11.1	Apache-2.0	Parses textual queries into Lucene queries
55	library	lucene-sandbox	9.11.1	Apache-2.0	Experimental Lucene modules
56	library	lucene-spatial-extras	9.11.1	Apache-2.0	Spatial/geographic query utilities
57	library	lucene-spatial3d	9.11.1	Apache-2.0	3D geospatial shapes and indexing
58	library	lucene-suggest	9.11.1	Apache-2.0	Suggestion/autocomplete engine
59	library	opensearch-cli	2.16.0	Apache-2.0	Command-line tools for OpenSearch management
60	library	jopt-simple	5.0.4	MIT	Command-line option parser
61	library	joda-time	2.12.7	Apache-2.0	Date and time handling library
62	library	t-digest	3.2	Apache-2.0	Statistical distribution estimation for percentiles
63	library	log4j-api	2.24.3	Apache-2.0	Core Log4j 2 API

#	Context	Name	Version	License	Comment
64	library	log4j-jul	2.24.3	Apache-2.0	Bridge for Java Util Logging to Log4j
65	library	jna	5.13.0	LGPL-2.1-or-later OR Apache-2.0	Java Native Access for system-level operations
66	library	jzlib	1.1.3	BSD-4-Clause	Java implementation of zlib compression
67	library	reactor-core	3.7.12	Apache-2.0	Reactive programming support (Project Reactor)
68	library	reactive-streams	1.0.4	MIT-0	Minimal reactive streams specification
69	library	protobuf-java	3.22.3	BSD-3-Clause	Google Protocol Buffers serialization
70	library	opensearch-rest-client	2.16.0	Apache-2.0	Low-level REST client for OpenSearch
71	library	httpclient	4.5.14	Apache-2.0	Apache HTTP client for network communication
72	library	httpcore	4.4.16	Apache-2.0	Core transport components for Apache HTTP client
73	library	httpasyncclient	4.1.5	Apache-2.0	Asynchronous HTTP client for non-blocking I/O
74	library	httpcore-nio	4.4.16	Apache-2.0	NIO transport layer for asynchronous HTTP
75	library	commons-codec	1.18.0	Apache-2.0	Utility library for encoding/decoding formats
76	library	commons-logging	1.2	Apache-2.0	Lightweight logging abstraction
77	library	mapper-extras-client	2.16.0	Apache-2.0	Additional mapping features for OpenSearch
78	library	parent-join-client	2.16.0	Apache-2.0	Supports parent-child relationships in OpenSearch
79	library	aggs-matrix-stats-client	2.16.0	Apache-2.0	Advanced aggregation statistics module
80	library	rank-eval-client	2.16.0	Apache-2.0	Ranking evaluation tools for OpenSearch
81	library	lang-mustache-client	2.16.0	Apache-2.0	Template rendering support (Mustache)
82	library	compiler	0.9.14	Apache-2.0	Annotation processing and compilation utilities
83	library	jackson-dataformat-yaml	2.19.2	Apache-2.0	YAML serialization and deserialization
84	library	jackson-databind	2.19.2	Apache-2.0	Object–JSON mapping framework
85	library	jackson-annotations	2.19.2	Apache-2.0	Annotations for Jackson serialization/deserialization
86	library	jackson-core	2.19.2	Apache-2.0	Core streaming JSON processor
87	library	slf4j-api	2.0.17	MIT	Unified logging abstraction layer

Last Name	First Name	Value					
Lorenz	**Alexander**	5		**5.00**	**OK**		
Buchner	**Lea**	5					
Engelhard	**Dirk**						
Strachwitz	**Carlo**						
Belz	**Sara**			0	No size		
Becker	**Eugen**			1	Trivial size		
Hofmann	**Leo**			2	Small size		
Knecht	**Sebastian**			3	Medium size		
				5	Large size		
				8	Very large size		
				13	Too large (size)		
How to play planning poker							
1. Everyone type their number into their value field, don't hit return yet							
2. Someone, perhaps a product owner, count down 3.. 2.. 1..							
3. Then, everyone hit return to submit their value							