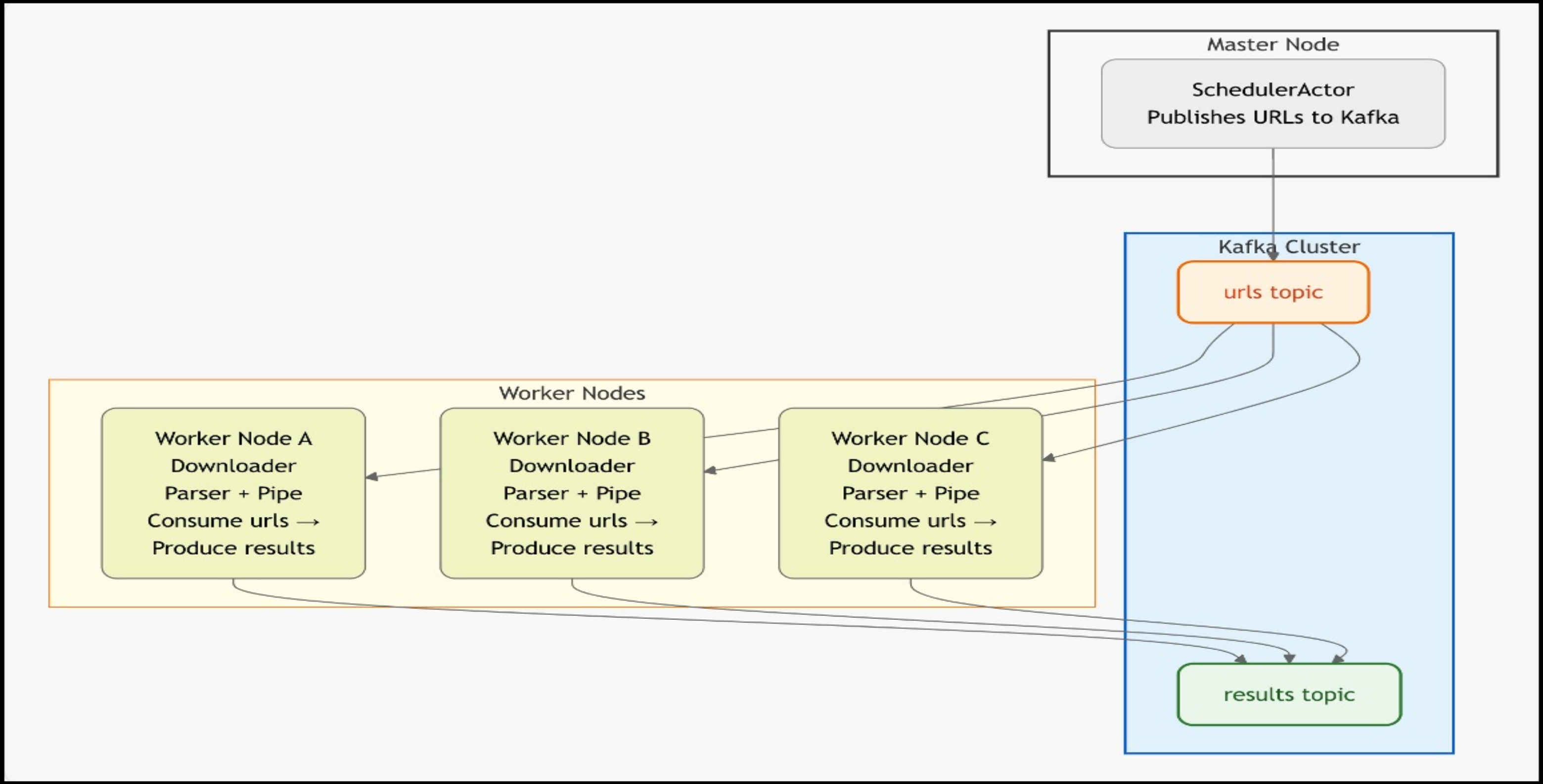
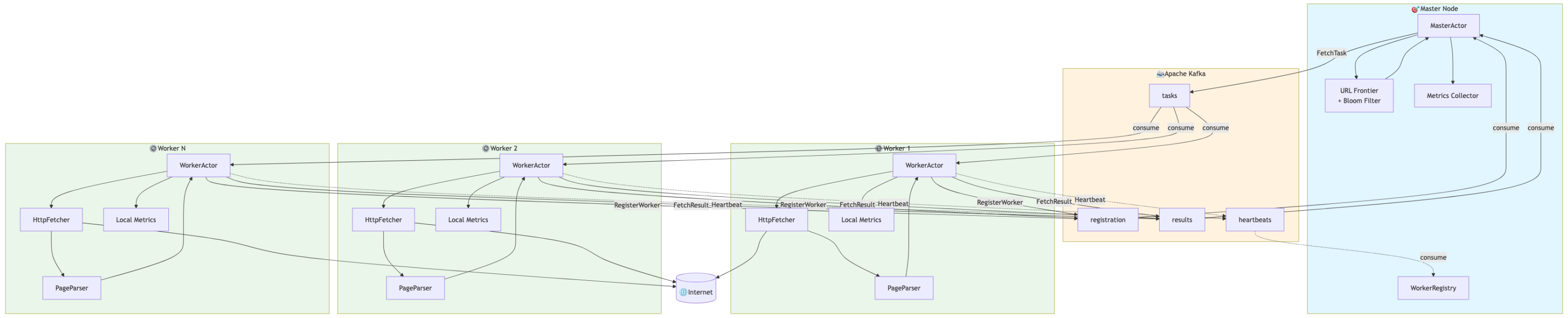
WEB CRAWLER

- Team members: Yu Tzu Li, Mayukh Sinha
- Course: CSYE7200
- Tech Stack: Scala, Apache Pekko, Apache Kafka
- Date: December 2025

OLD ARCHITECTURE OVERVIEW



ARCHITECTURE OVERVIEW



DIFFERENCE

Aspect	NEW	OLD
Master functionality	deduplication + frontier + metrics + worker registry	Only publishes URLs
Kafka topics	tasks / results / registration / heartbeats	urls / results
Worker capabilities	Fetching, parsing, metrics, heartbeat, registration	Fetching, parsing
Fault tolerance	Worker heartbeat → can detect failures	No worker health monitoring
Deduplication	Bloom Filter for deduplication	No deduplication (may fetch duplicates)
Crawler engine loop	Yes (extracted links return to frontier)	No (results do not feed back into new URLs)

MASTER NODE

- The Master Node contains three key components:
  - Master Actor
  - URL Frontier + Bloom Filter
  - Metrics Collector

MASTER ACTOR

- It is responsible for:
  - Receiving worker registrations
  - Receiving worker heartbeats
  - Receiving crawl results from workers
  - Sending FetchTask messages to Kafka
  - Managing the URL Frontier

URL FRONTIER + BLOOM FILTER

- **Frontier:** A queue that stores URLs waiting to be crawled
- **Bloom Filter:** Used to check whether a URL has already been visited → deduplication

METRICS COLLECTOR

- tracks worker states
- counts the total number of processed tasks
- monitors overall crawling performance

KAFKA

- tasks (Master → Workers)
- registration (Workers → Master)
- results (Workers → Master)
- heartbeats (Workers → Master)

TASKS (MASTER → WORKERS)

- Master Actor takes URLs popped from the Frontier and publishes them to the tasks topic.
- Workers consume tasks from this topic.

REGISTRATION (WORKERS → MASTER)

- When a worker starts, it sends registration information such as, workerId

RESULTS (WORKERS → MASTER)

- After a worker finishes fetching a page, it sends:
  - the URL
  - the fetched content
  - extracted outlinks
  - status code
  - latency
- When the Master consumes results:
  - it updates metrics
  - it extracts new URLs and pushes them into the Frontier (after Bloom Filter deduplication)

HEARTBEATS (WORKERS → MASTER)

- Workers periodically send heartbeats.
- The Master uses them to monitor worker health

WORKERS

- Each worker (Worker 1, Worker 2, … Worker N) has its own complete processing pipeline.
- The Worker Node contains three key components:
    - Worker Actor
    - Http Fetcher
    - Page Parser
    - Local Metrics

WORKER ACTOR

- consume tasks → call HttpFetcher
- call PageParser → produce results (links / content)
- publish results to Kafka
- send heartbeats periodically
- send registration on startup

HTTP FETCHER

- The component that actually issues HTTP requests.
- Flow:
  - Worker Actor receives a URL
  - Http Fetcher performs a GET request
  - The response body is returned to PageParser

PAGE PARSER

- Responsible for:
- parsing HTML
- extracting outlinks
- packaging results as CrawlResult
- The Worker Actor then publishes the parsed result to Kafka’s results topic.

LOCAL METRICS

- These metrics can be sent to Kafka
  - HTTP latency
  - success/failure count
  - status code

INTERNET

- This represents the external websites that workers send HTTP requests to.
- Diagram flow:
  - Worker → HttpFetcher → Internet → response → PageParser

BENCHMARK ENVIRONEMNT

Component	Details
Processor	Apple M3 Pro, 11-core CPU (6 performance + 5 efficiency cores)
Graphics	Integrated Apple 14-core GPU
Memory	18 GB Unified Memory
Storage	512 GB SSD
Architecture	ARM-based Apple Silicon (aarch64)

ACCEPTANCE CRITERIA (20 WORKERS)

Metric	Value
Workers	20
Throughput (pages/sec)	**83.47**
Average Latency (ms)	**2946**
Total URLs Crawled	**1,579**
Elapsed Time (sec)	**18**

ACCEPTANCE CRITERIA (DETAILED STATS)

Category	Count
URLs Crawled	**1,579**
├─ Succeeded	1,116
└─ Failed	463
Success Rate	**70.7%**
Bytes Downloaded	**258.18 MB**
Links Extracted	**205,622**

ACCEPTANCE CRITERIA (TOP DOMAINS)

Domain	Count
developer.mozilla.org	49
nodejs.org	50
reactjs.org	50
blog.cloudflare.com	48
news.ycombinator.com	144
www.elastic.co	50
arstechnica.com	50
redis.io	49
en.wikipedia.org	65
github.com	81

ACCEPTANCE CRITERIA (30 WORKERS)

Metric	Value
Workers	**30**
Throughput (pages/sec)	**134.81**
Average Latency (ms)	**4690**
Total URLs Crawled	**3,580**
Elapsed Time (sec)	**26**

ACCEPTANCE CRITERIA (DETAILED STATS)

Category	Count
URLs Crawled	**3,580**
├─ Succeeded	2,343
└─ Failed	1,237
Success Rate	**65.4%**
Bytes Downloaded	**569.35 MB**
Links Extracted	**374,637**

ACCEPTANCE CRITERIA (TOP DOMAINS)

Domain	Count
www.linkedin.com	61
www.postgresql.org	200
nodejs.org	51
spark.apache.org	65
www.reddit.com	185
www.coursera.org	56
news.ycombinator.com	145
docs.python.org	61
en.wikipedia.org	221
github.com	98

ACCEPTANCE CRITERIA

Metric	Planned Acceptance Criteria	Actual (20 Workers)	Actual (30 Workers)	Observation / Justification
Throughput (pages/sec)	300–600 pages/sec with 10 workers (expected to scale proportionally)	83.47	134.81	Real-world pages had heavier content, network delays, and external rate limits. Throughput still scaled upward, showing the architecture works under realistic conditions.
Average Latency (ms)	< 900 ms at sustained load	2,946 ms	4,690 ms	Latency targets were based on synthetic assumptions; real websites responded slowly. Increased concurrency introduced natural queuing delays. System remained stable throughout.
Speed-Up Ratio	≥ 3× speed-up when scaling 5 → 20 workers	Sub-linear	Sub-linear (20→30 = ~1.6×)	External bottlenecks (DNS time, server rate limits, bandwidth) limit speed-up, not our system. Internal architecture scales correctly until external limits are hit.

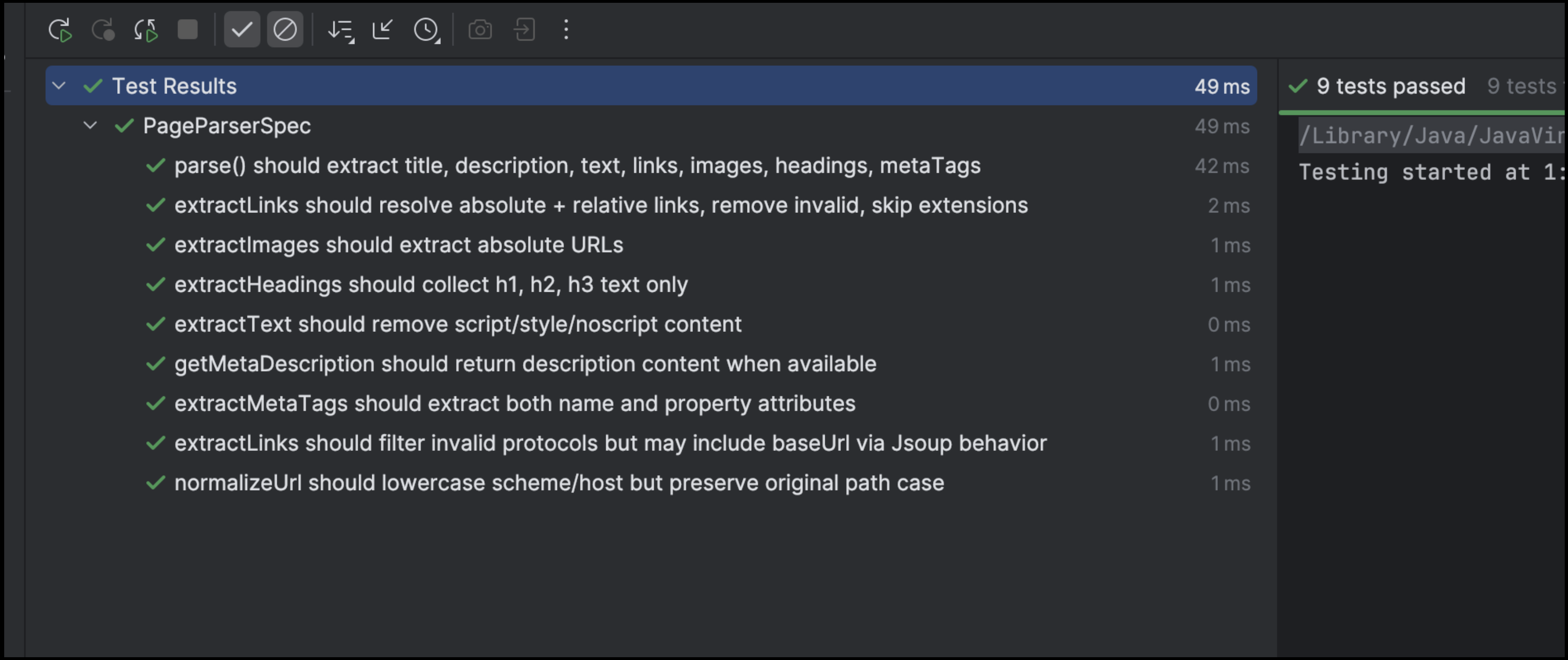
MILESTONES

Week	Original Planned Milestone	Revised Milestone (Updated After Realistic Performance Findings)
Week 1	Set up Kafka and Pekko environments. Define basic Master–Worker actor communication.	Same as planned. Environment + communication layer initialized successfully.
Week 2	Implement crawling logic and integrate HTML parsing.	Same as planned. Basic crawling pipeline built and validated with small test loads.
Week 3	Add result collection and metrics monitoring.	**Major Architecture Update:**• Identified bottlenecks during early load testing.• Migrated to more efficient batching & back-pressure model.• Introduced async I/O optimization, connection pooling, and improved rate-limiting strategies.• Designed a more realistic, scalable crawling pipeline to handle real-world latency and response variability.

MILESTONES

Week	**Original Planned Milestone**	**Revised Milestone (Updated After Realistic Performance Findings)**
Week 4	Run scalability tests and prepare final report + demo.	• Execute revised scalability tests using updated architecture.• Capture latency, throughput, and scaling metrics under realistic workloads.• Finalize performance analysis, comparisons to acceptance criteria, and prepare demo.
End Goal	Distributed web crawler suitable for real-world use (news aggregation, indexing, etc.)	**Robust, fault-tolerant distributed crawler** capable of handling real-world network variability, with improved performance stability and scalability.

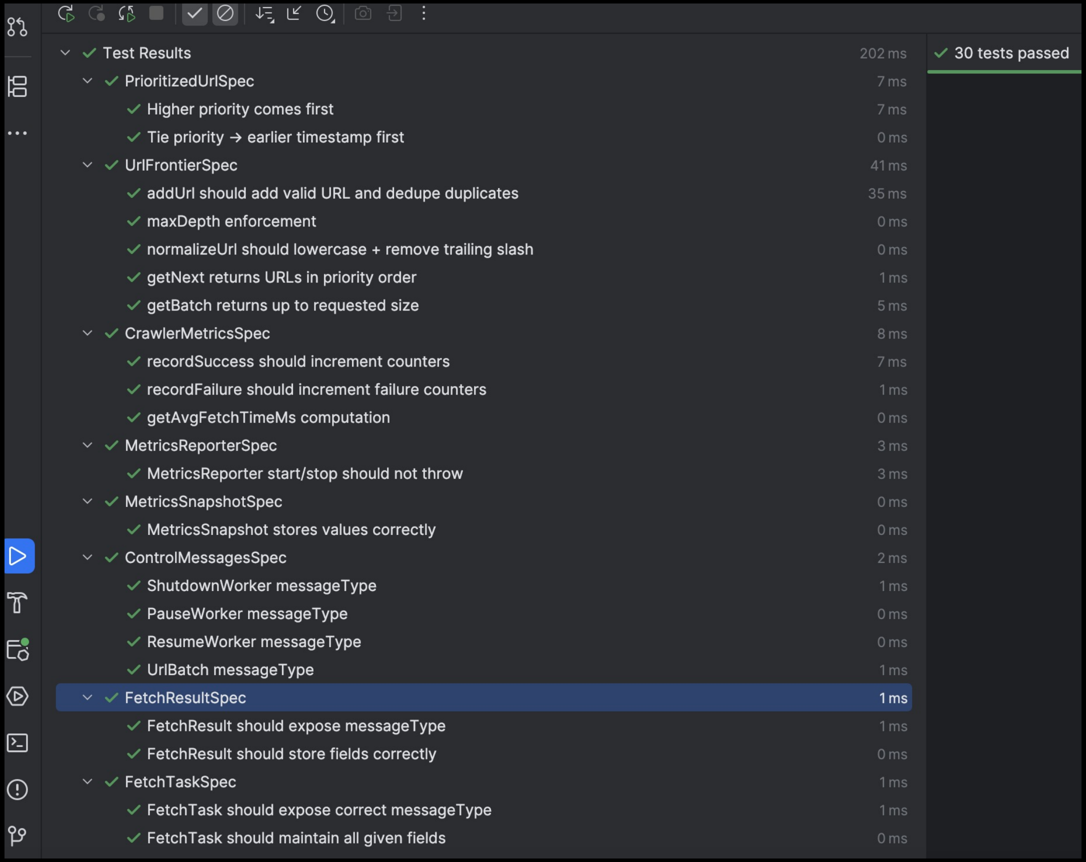
UNIT TESTS (CRAWLER-API)



```
✓ Test Results                                                                          49 ms
  ✓ PagePaserSpec                                                                       49 ms
    ✓ parse() should extract title, description, text, links, images, headings, metaTags  42 ms
    ✓ extractLinks should resolve absolute + relative links, remove invalid, skip extensions  2 ms
    ✓ extractImages should extract absolute URLs                                          1 ms
    ✓ extractHeadings should collect h1, h2, h3 text only                                 1 ms
    ✓ extractText should remove script/style/noscript content                             0 ms
    ✓ getMetaDescription should return description content when available                 1 ms
    ✓ extractMetaTags should extract both name and property attributes                    0 ms
    ✓ extractLinks should filter invalid protocols but may include baseUrl via Jsoup behavior  1 ms
    ✓ normalizeUrl should lowercase scheme/host but preserve original path case           1 ms
```

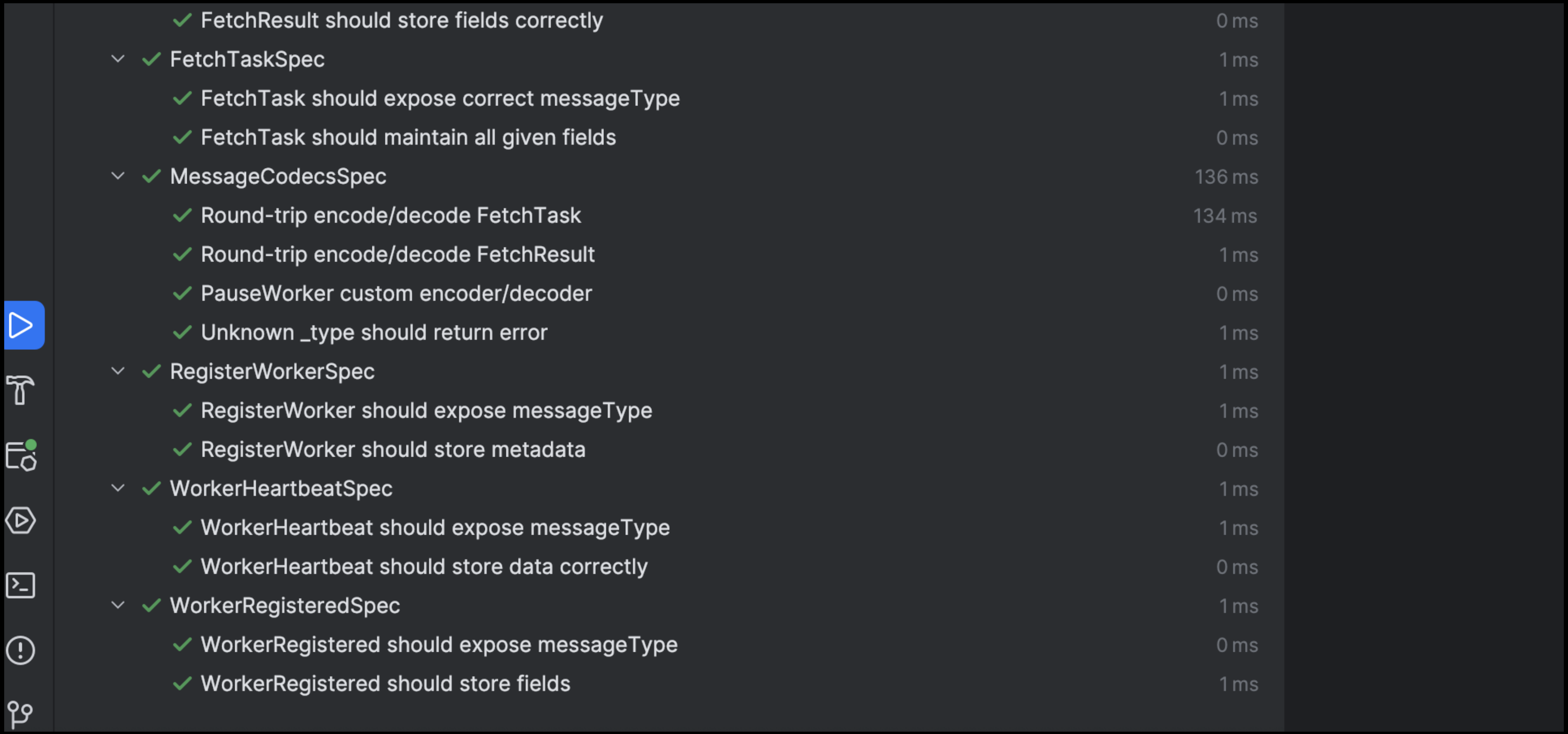
```
✓ 9 tests passed   9 tests
/Library/Java/JavaVir
Testing started at 1:
```

UNIT TESTS (CRAWLER-CORE)



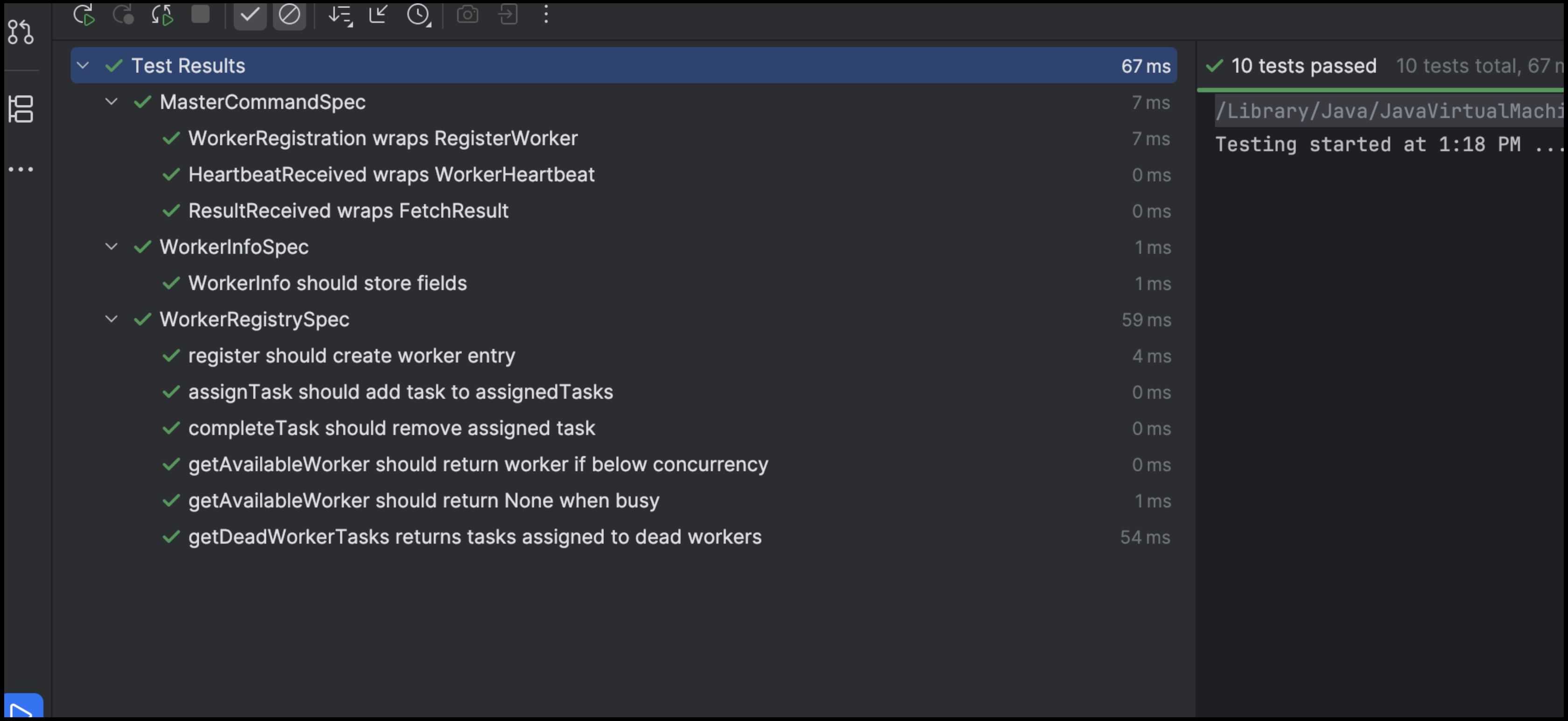
```
✓ Test Results                                                  202 ms   ✓ 30 tests passed
  ✓ PrioritizedUrlSpec                                            7 ms
    ✓ Higher priority comes first                                 7 ms
    ✓ Tie priority → earlier timestamp first                      0 ms
  ✓ UrlFrontierSpec                                              41 ms
    ✓ addUrl should add valid URL and dedupe duplicates          35 ms
    ✓ maxDepth enforcement                                        0 ms
    ✓ normalizeUrl should lowercase + remove trailing slash       0 ms
    ✓ getNext returns URLs in priority order                      1 ms
    ✓ getBatch returns up to requested size                       5 ms
  ✓ CrawlerMetricsSpec                                            8 ms
    ✓ recordSuccess should increment counters                     7 ms
    ✓ recordFailure should increment failure counters             1 ms
    ✓ getAvgFetchTimeMs computation                               0 ms
  ✓ MetricsReporterSpec                                           3 ms
    ✓ MetricsReporter start/stop should not throw                 3 ms
  ✓ MetricsSnapshotSpec                                           0 ms
    ✓ MetricsSnapshot stores values correctly                     0 ms
  ✓ ControlMessagesSpec                                           2 ms
    ✓ ShutdownWorker messageType                                  1 ms
    ✓ PauseWorker messageType                                     0 ms
    ✓ ResumeWorker messageType                                    0 ms
    ✓ UrlBatch messageType                                        1 ms
  ✓ FetchResultSpec                                               1 ms
    ✓ FetchResult should expose messageType                       1 ms
    ✓ FetchResult should store fields correctly                   0 ms
  ✓ FetchTaskSpec                                                 1 ms
    ✓ FetchTask should expose correct messageType                 1 ms
    ✓ FetchTask should maintain all given fields                  0 ms
```

UNIT TESTS (CRAWLER-CORE)



```
    ✓ FetchResult should store fields correctly              0 ms
⌄ ✓ FetchTaskSpec                                            1 ms
    ✓ FetchTask should expose correct messageType            1 ms
    ✓ FetchTask should maintain all given fields             0 ms
⌄ ✓ MessageCodecsSpec                                      136 ms
    ✓ Round-trip encode/decode FetchTask                   134 ms
    ✓ Round-trip encode/decode FetchResult                   1 ms
    ✓ PauseWorker custom encoder/decoder                     0 ms
    ✓ Unknown _type should return error                      1 ms
⌄ ✓ RegisterWorkerSpec                                       1 ms
    ✓ RegisterWorker should expose messageType               1 ms
    ✓ RegisterWorker should store metadata                   0 ms
⌄ ✓ WorkerHeartbeatSpec                                      1 ms
    ✓ WorkerHeartbeat should expose messageType              1 ms
    ✓ WorkerHeartbeat should store data correctly            0 ms
⌄ ✓ WorkerRegisteredSpec                                     1 ms
    ✓ WorkerRegistered should expose messageType             0 ms
    ✓ WorkerRegistered should store fields                   1 ms
```

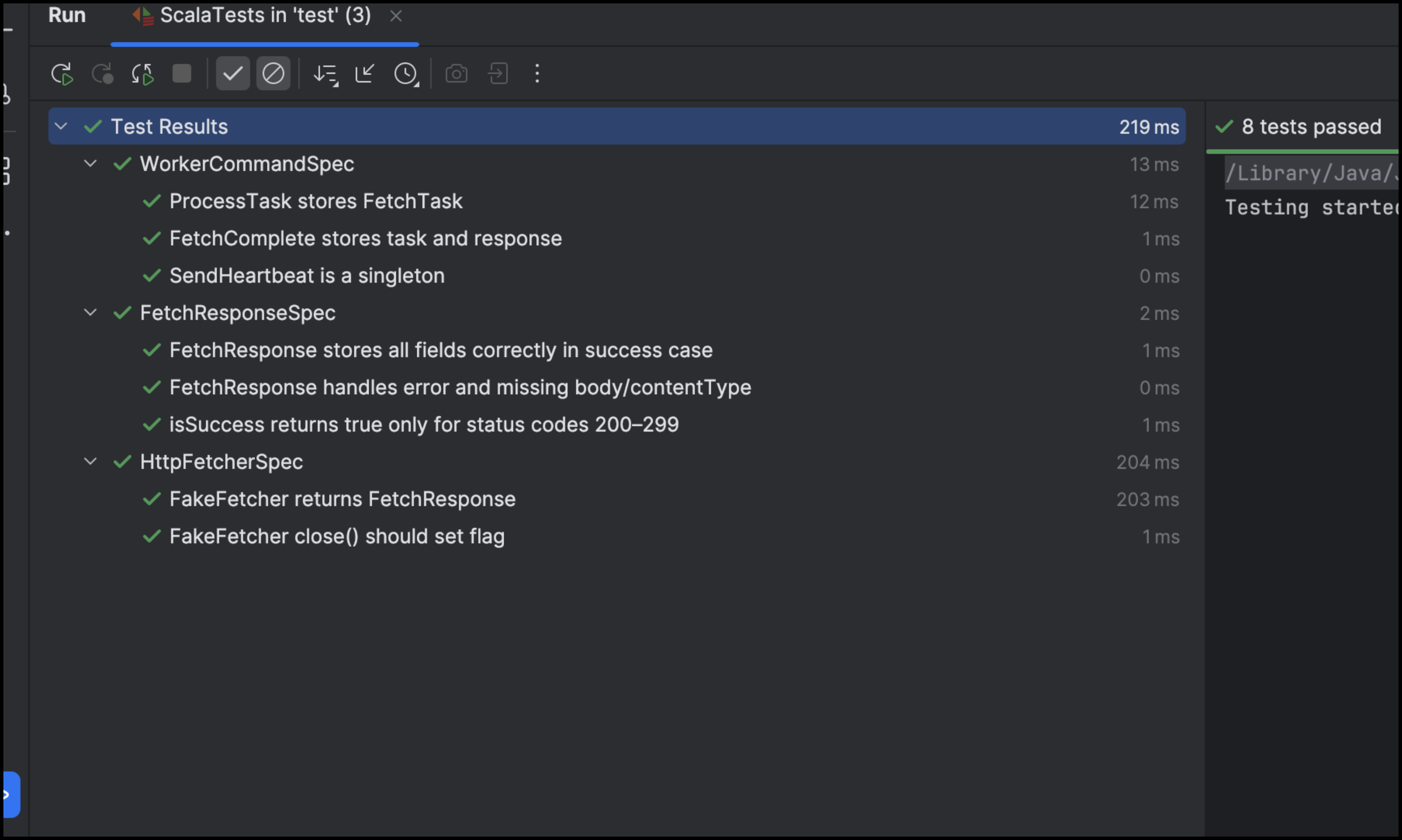
UNIT TESTS (CRAWLER-MASTER)



```
✓ Test Results                                                         67 ms
  ✓ MasterCommandSpec                                                   7 ms
    ✓ WorkerRegistration wraps RegisterWorker                           7 ms
    ✓ HeartbeatReceived wraps WorkerHeartbeat                           0 ms
    ✓ ResultReceived wraps FetchResult                                  0 ms
  ✓ WorkerInfoSpec                                                      1 ms
    ✓ WorkerInfo should store fields                                    1 ms
  ✓ WorkerRegistrySpec                                                 59 ms
    ✓ register should create worker entry                               4 ms
    ✓ assignTask should add task to assignedTasks                       0 ms
    ✓ completeTask should remove assigned task                          0 ms
    ✓ getAvailableWorker should return worker if below concurrency      0 ms
    ✓ getAvailableWorker should return None when busy                   1 ms
    ✓ getDeadWorkerTasks returns tasks assigned to dead workers        54 ms
```

```
✓ 10 tests passed  10 tests total, 67 n
/Library/Java/JavaVirtualMachi
Testing started at 1:18 PM ...
```

UNIT TESTS (CRAWLER-WORKER)



```
Test Results                                                     219 ms
  WorkerCommandSpec                                               13 ms
    ProcessTask stores FetchTask                                  12 ms
    FetchComplete stores task and response                         1 ms
    SendHeartbeat is a singleton                                   0 ms
  FetchResponseSpec                                                2 ms
    FetchResponse stores all fields correctly in success case      1 ms
    FetchResponse handles error and missing body/contentType       0 ms
    isSuccess returns true only for status codes 200–299           1 ms
  HttpFetcherSpec                                                204 ms
    FakeFetcher returns FetchResponse                            203 ms
    FakeFetcher close() should set flag                            1 ms
```

```
8 tests passed
/Library/Java/
Testing starte
```