Adobe GenStudio for Performance Marketing Engineering Test

Integer to Roman Numeral Converter

This project was built as part of Adobe's take-home engineering task. It demonstrates best practices in full-stack development, observability, containerization, and automated testing. The goal is to convert an integer between 1 and 3999 into its Roman numeral representation through a frontend UI and backend API.

Approached the Problem

- I started by carefully reviewing the requirements and figuring out the three main parts of the task: how it works (converting input to output), how it can be monitored (logs, metrics, traces), and how it's ready to be deployed in production (tests, Docker).
- I took a backend-first approach to make sure the core API and monitoring infrastructure were solid before adding the frontend.
- The core logic for converting integers to Roman numerals is implemented using a rule-based approach. The API Input validation ensures users provide a valid integer between 1 and 3999, with descriptive error responses for invalid inputs.
- Roman numeral rules were implemented from scratch based on the Wikipedia specification, following rule-based logic with subtractive notation (e.g., IV for 4, CM for 900).
- The frontend was built using React and Adobe Spectrum to keep the visual design and accessibility consistent in both light and dark modes.
- All input is validated to accept only integers within the valid range. Invalid input results in descriptive errors.
- Each service was put into containers and managed using Docker Compose, including Jaeger for local trace viewing, which makes it easy to develop and test locally.
- The README and automated test coverage ensure the project is easy to run, verify, and extend.

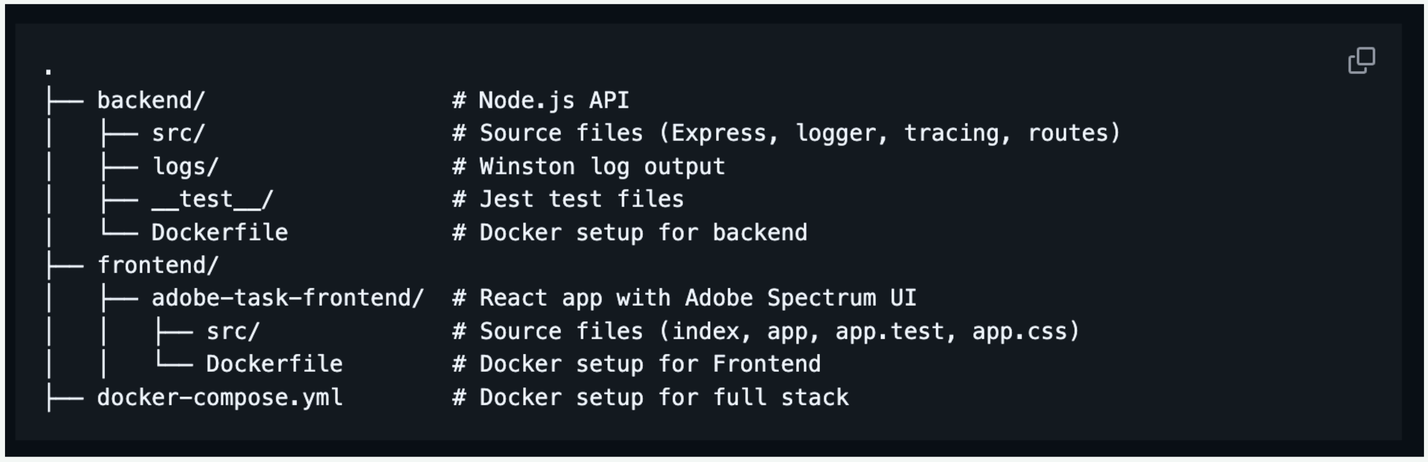
Technologies Used

- **Frontend**: React, TypeScript, Adobe React Spectrum
- **Backend**: Node.js, Express, TypeScript
- **Logging**: Winston
- **Metrics**: Prometheus (via prom-client)
- **Tracing**: OpenTelemetry + Jaeger
- **Testing**: Jest, Supertest
- **Containerization**: Docker, Docker Compose

Source Code

https://github.com/AkshayPatel8140/integer-to-roman-converter

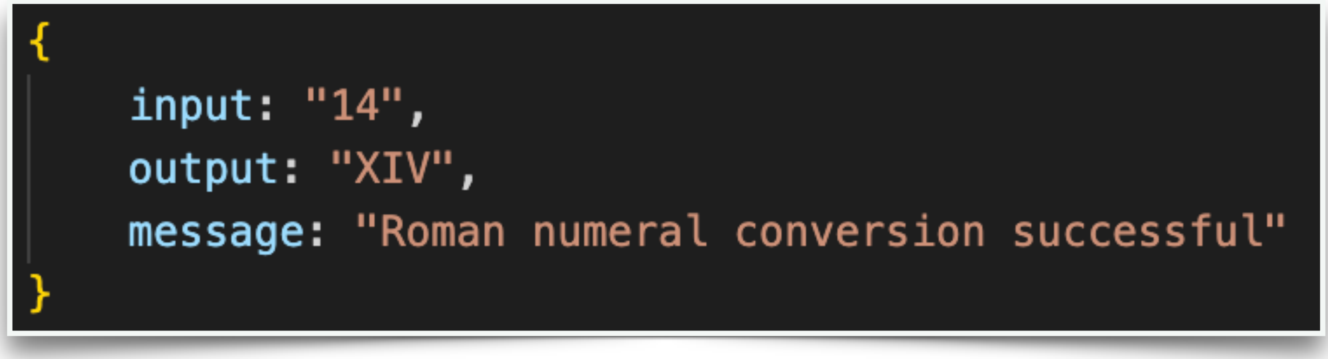
Project Structure



```
.
├── backend/                # Node.js API
│   ├── src/                # Source files (Express, logger, tracing, routes)
│   ├── logs/               # Winston log output
│   ├── __test__/           # Jest test files
│   └── Dockerfile          # Docker setup for backend
├── frontend/
│   ├── adobe-task-frontend/  # React app with Adobe Spectrum UI
│   │   ├── src/            # Source files (index, app, app.test, app.css)
│   │   └── Dockerfile      # Docker setup for Frontend
├── docker-compose.yml      # Docker setup for full stack
```

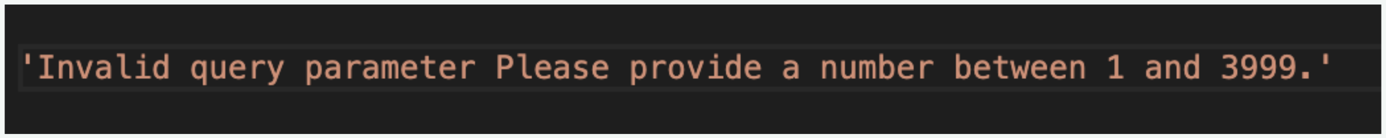
Backend API Details

- **Endpoint**: GET {URL}/romannumeral?query=1234
- **Valid Range**: From 1 to 3999
- **Success Response**:



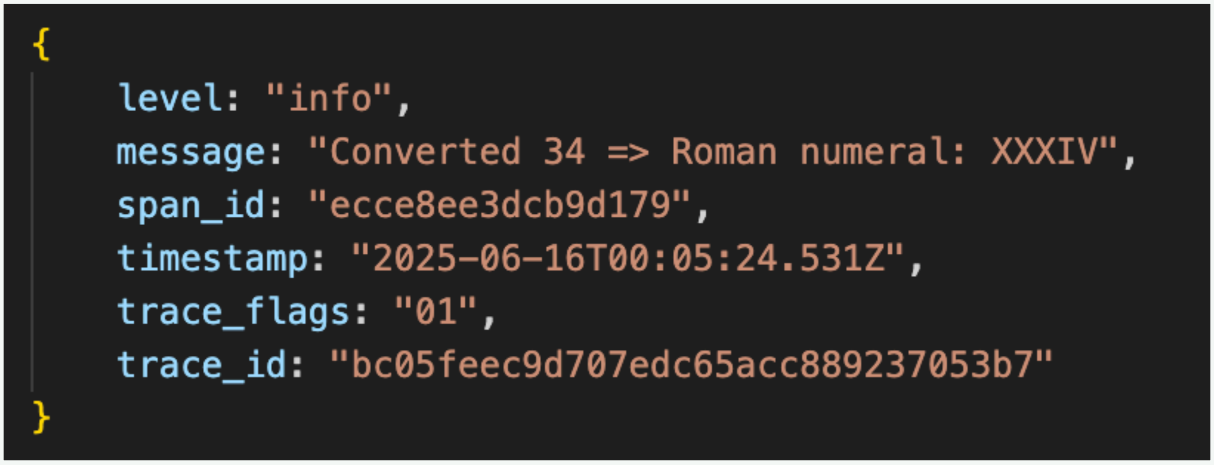
```
{
    input: "14",
    output: "XIV",
    message: "Roman numeral conversion successful"
}
```

- **Error Response**:



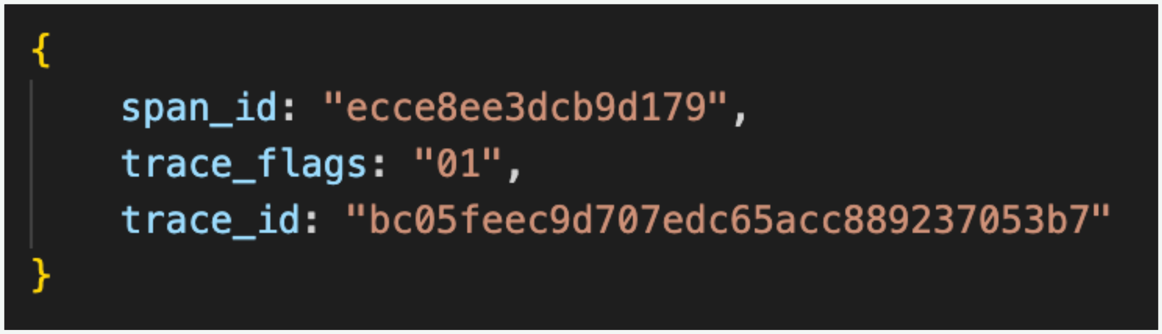
```
'Invalid query parameter Please provide a number between 1 and 3999.'
```

- **Log output**:

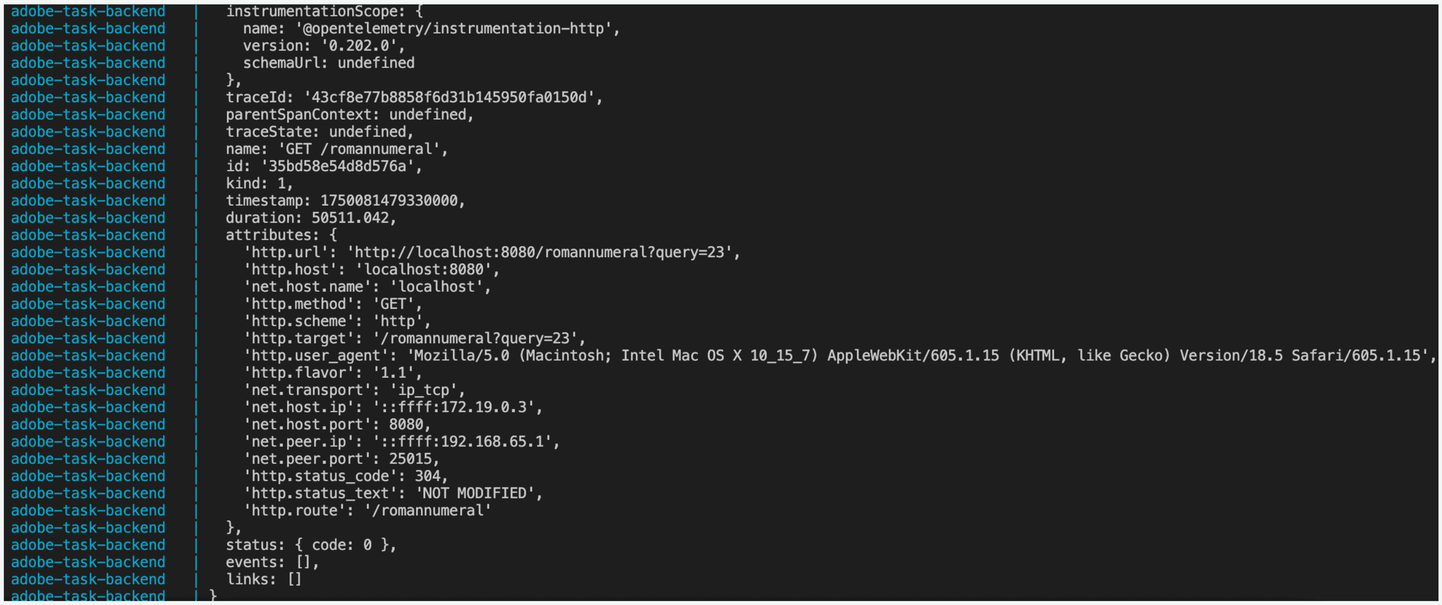


```
{
    level: "info",
    message: "Converted 34 => Roman numeral: XXXIV",
    span_id: "ecce8ee3dcb9d179",
    timestamp: "2025-06-16T00:05:24.531Z",
    trace_flags: "01",
    trace_id: "bc05feec9d707edc65acc889237053b7"
}
```

- **Tracing Output**:



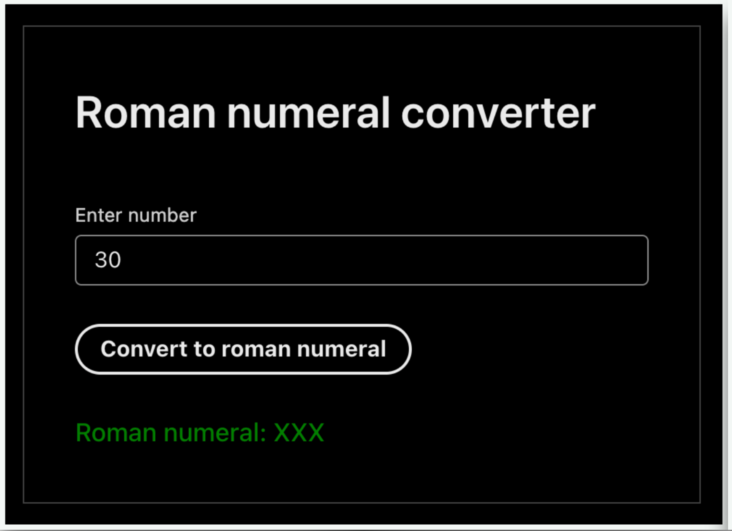
```
{
    span_id: "ecce8ee3dcb9d179",
    trace_flags: "01",
    trace_id: "bc05feec9d707edc65acc889237053b7"
}
```



```
adobe-task-backend    |   instrumentationScope: {
adobe-task-backend    |     name: '@opentelemetry/instrumentation-http',
adobe-task-backend    |     version: '0.202.0',
adobe-task-backend    |     schemaUrl: undefined
adobe-task-backend    |   },
adobe-task-backend    |   traceId: '43cf8e77b8858f6d31b145950fa0150d',
adobe-task-backend    |   parentSpanContext: undefined,
adobe-task-backend    |   traceState: undefined,
adobe-task-backend    |   name: 'GET /romannumeral',
adobe-task-backend    |   id: '35bd58e54d8d576a',
adobe-task-backend    |   kind: 1,
adobe-task-backend    |   timestamp: 1750081479330000,
adobe-task-backend    |   duration: 50511.042,
adobe-task-backend    |   attributes: {
adobe-task-backend    |     'http.url': 'http://localhost:8080/romannumeral?query=23',
adobe-task-backend    |     'http.host': 'localhost:8080',
adobe-task-backend    |     'net.host.name': 'localhost',
adobe-task-backend    |     'http.method': 'GET',
adobe-task-backend    |     'http.scheme': 'http',
adobe-task-backend    |     'http.target': '/romannumeral?query=23',
adobe-task-backend    |     'http.user_agent': 'Mozilla/5.0 (Macintosh; Intel Mac OS X 10_15_7) AppleWebKit/605.1.15 (KHTML, like Gecko) Version/18.5 Safari/605.1.15',
adobe-task-backend    |     'http.flavor': '1.1',
adobe-task-backend    |     'net.transport': 'ip_tcp',
adobe-task-backend    |     'net.host.ip': '::ffff:172.19.0.3',
adobe-task-backend    |     'net.host.port': 8080,
adobe-task-backend    |     'net.peer.ip': '::ffff:192.168.65.1',
adobe-task-backend    |     'net.peer.port': 25015,
adobe-task-backend    |     'http.status_code': 304,
adobe-task-backend    |     'http.status_text': 'NOT MODIFIED',
adobe-task-backend    |     'http.route': '/romannumeral'
adobe-task-backend    |   },
adobe-task-backend    |   status: { code: 0 },
adobe-task-backend    |   events: [],
adobe-task-backend    |   links: []
adobe-task-backend    | }
```

Frontend

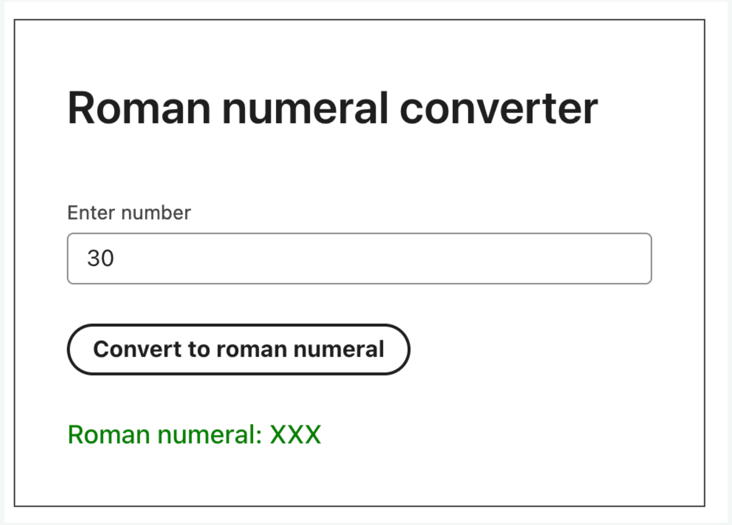
- Takes user input (number from 1–3999)
- Using the Button to trigger conversion, sends it to the backend
- Display area for result or error message
- Supports system light/dark mode using Spectrum theme
- **Adobe React Spectrum** was chosen to ensure accessible, **theme-consistent UI** components that support both **light** and **dark modes**.


Roman numeral converter

Enter number: 30

Roman numeral: XXX

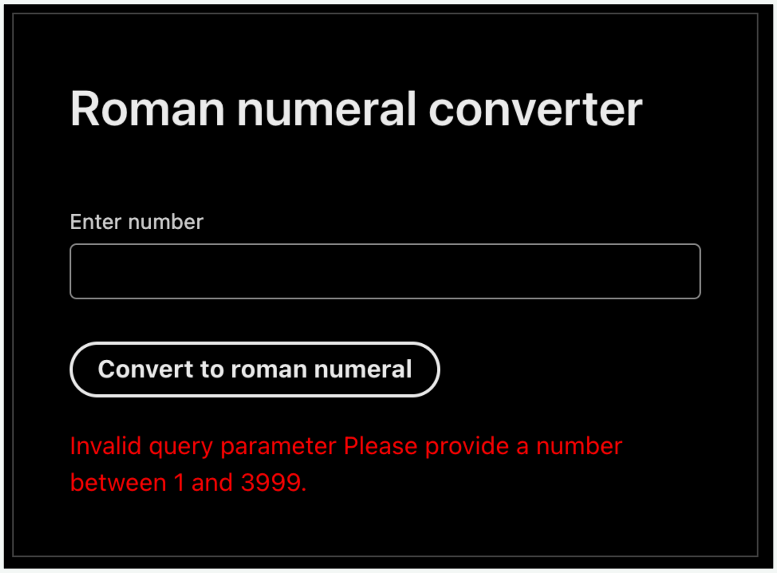
Dark Mode


Roman numeral converter

Enter number: 30

Roman numeral: XXX

Light Mode


Roman numeral converter

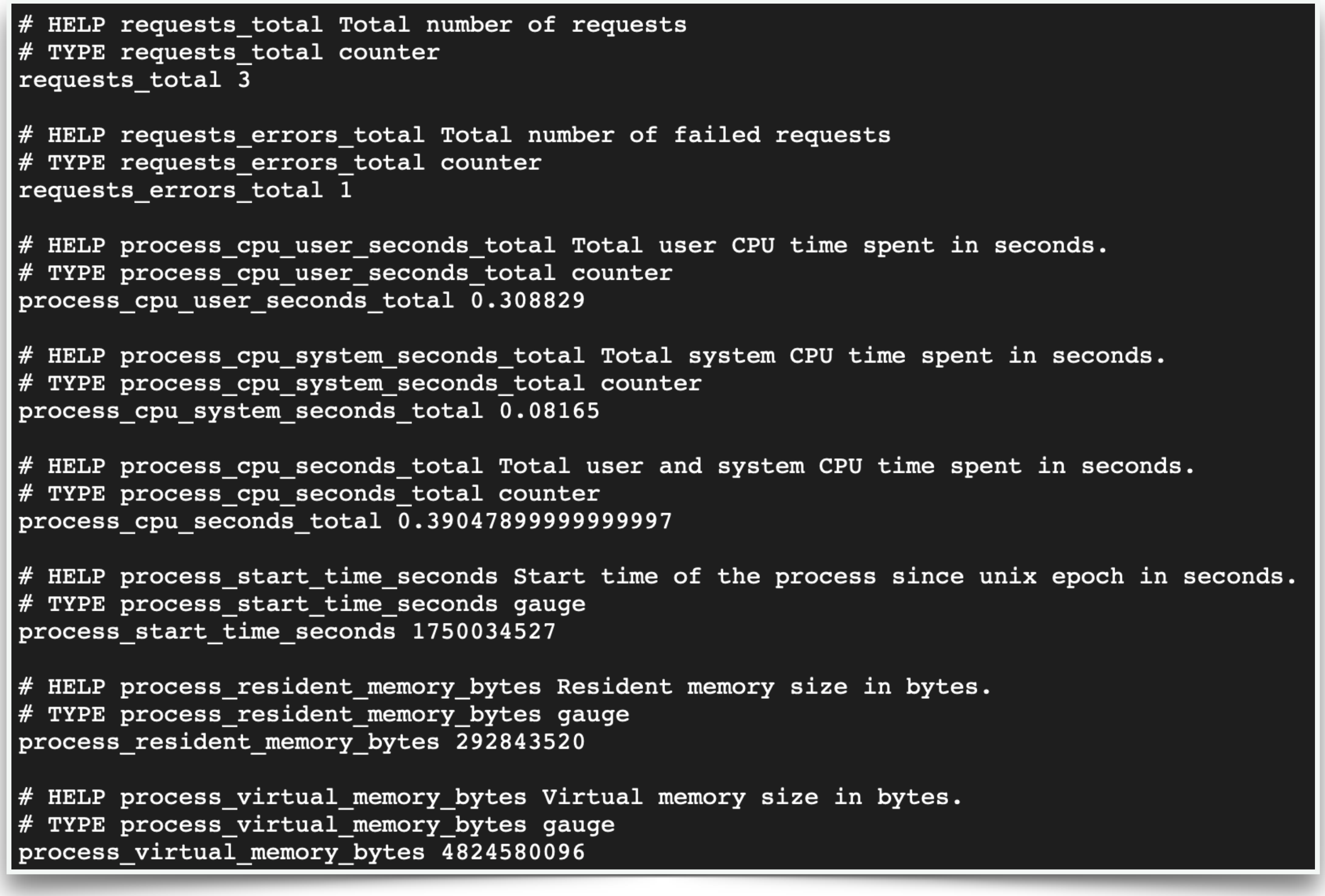
Enter number

Invalid query parameter Please provide a number between 1 and 3999.

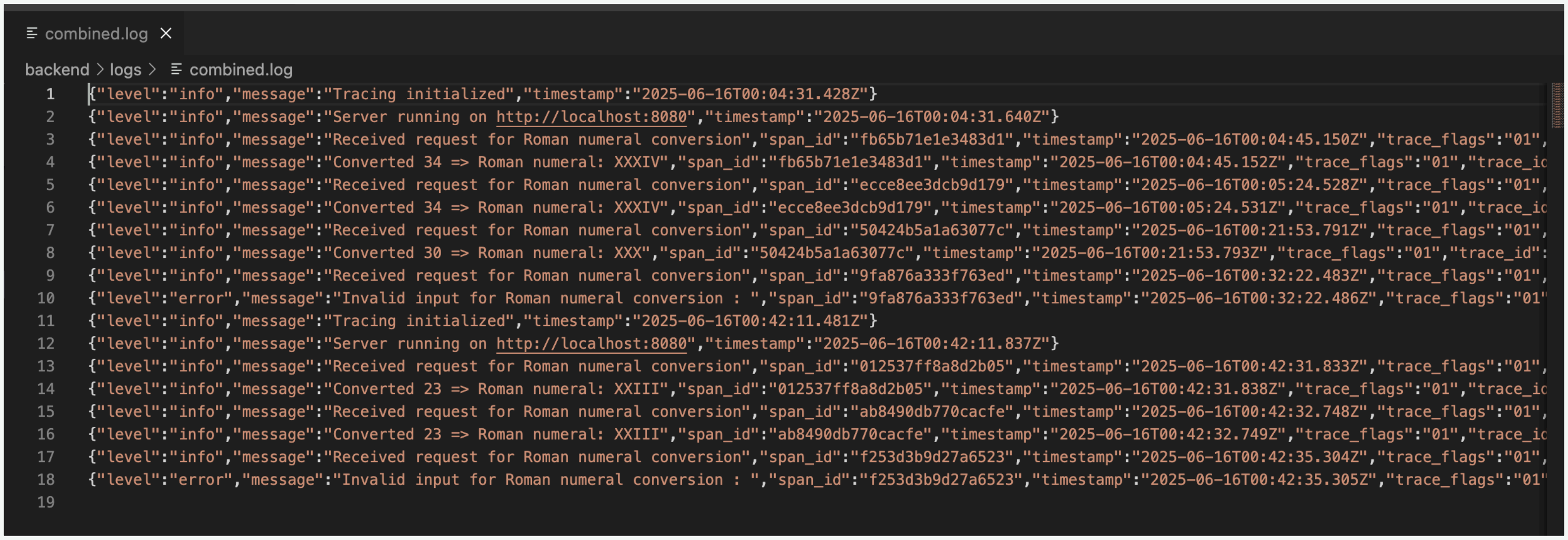
Display error

Observability

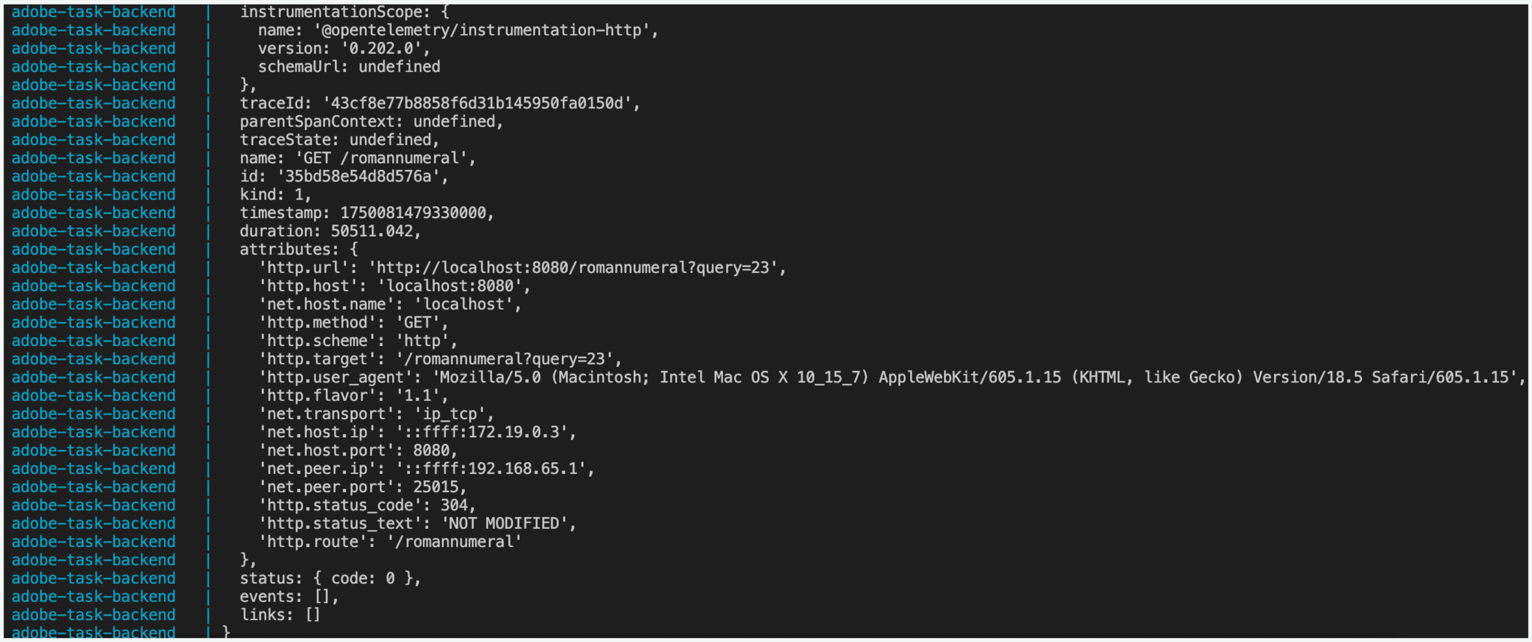
- Prometheus exposes HTTP request count, latency, and error rate via ‘/metrics’.
- OpenTelemetry with Jaeger captures spans for each HTTP request lifecycle, including internal function calls.
- Winston logs structured messages with timestamp, level, request input, response output, and error stacks.
- **Winston** was chosen for its support of structured **JSON logging** and flexible transport options, making it ideal for production-grade logging.
- **Prometheus** (**prom-client**) was selected for its native integration with Node.js and ability to expose **detailed application metrics** like **counters** and **histograms**.
- **OpenTelemetry with Jaeger** was used for distributed tracing due to its **open standard**, **extensibility**, and **ease of local visualization**.



```
# HELP requests_total Total number of requests
# TYPE requests_total counter
requests_total 3

# HELP requests_errors_total Total number of failed requests
# TYPE requests_errors_total counter
requests_errors_total 1

# HELP process_cpu_user_seconds_total Total user CPU time spent in seconds.
# TYPE process_cpu_user_seconds_total counter
process_cpu_user_seconds_total 0.308829

# HELP process_cpu_system_seconds_total Total system CPU time spent in seconds.
# TYPE process_cpu_system_seconds_total counter
process_cpu_system_seconds_total 0.08165

# HELP process_cpu_seconds_total Total user and system CPU time spent in seconds.
# TYPE process_cpu_seconds_total counter
process_cpu_seconds_total 0.39047899999999997

# HELP process_start_time_seconds Start time of the process since unix epoch in seconds.
# TYPE process_start_time_seconds gauge
process_start_time_seconds 1750034527

# HELP process_resident_memory_bytes Resident memory size in bytes.
# TYPE process_resident_memory_bytes gauge
process_resident_memory_bytes 292843520

# HELP process_virtual_memory_bytes Virtual memory size in bytes.
# TYPE process_virtual_memory_bytes gauge
process_virtual_memory_bytes 4824580096
```



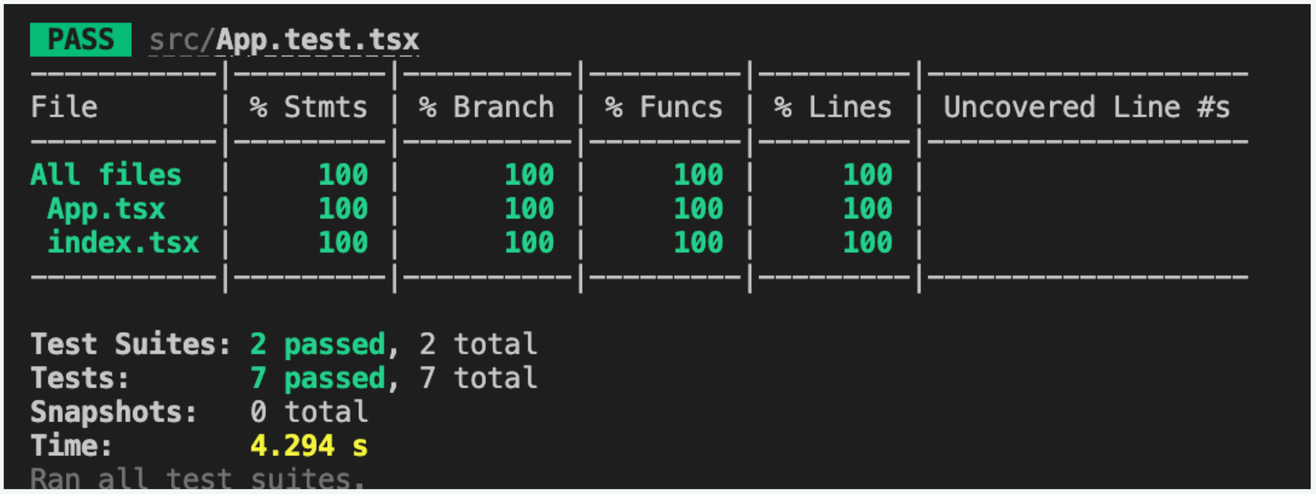
```
{"level":"info","message":"Tracing initialized","timestamp":"2025-06-16T00:04:31.428Z"}
{"level":"info","message":"Server running on http://localhost:8080","timestamp":"2025-06-16T00:04:31.640Z"}
{"level":"info","message":"Received request for Roman numeral conversion","span_id":"fb65b71e1e3483d1","timestamp":"2025-06-16T00:04:45.150Z","trace_flags":"01",
{"level":"info","message":"Converted 34 => Roman numeral: XXXIV","span_id":"fb65b71e1e3483d1","timestamp":"2025-06-16T00:04:45.152Z","trace_flags":"01","trace_id
{"level":"info","message":"Received request for Roman numeral conversion","span_id":"ecce8ee3dcb9d179","timestamp":"2025-06-16T00:05:24.528Z","trace_flags":"01",
{"level":"info","message":"Converted 34 => Roman numeral: XXXIV","span_id":"ecce8ee3dcb9d179","timestamp":"2025-06-16T00:05:24.531Z","trace_flags":"01","trace_id
{"level":"info","message":"Received request for Roman numeral conversion","span_id":"50424b5a1a63077c","timestamp":"2025-06-16T00:21:53.791Z","trace_flags":"01",
{"level":"info","message":"Converted 30 => Roman numeral: XXX","span_id":"50424b5a1a63077c","timestamp":"2025-06-16T00:21:53.793Z","trace_flags":"01","trace_id":
{"level":"info","message":"Received request for Roman numeral conversion","span_id":"9fa876a333f763ed","timestamp":"2025-06-16T00:32:22.483Z","trace_flags":"01",
{"level":"error","message":"Invalid input for Roman numeral conversion : ","span_id":"9fa876a333f763ed","timestamp":"2025-06-16T00:32:22.486Z","trace_flags":"01"
{"level":"info","message":"Tracing initialized","timestamp":"2025-06-16T00:42:11.481Z"}
{"level":"info","message":"Server running on http://localhost:8080","timestamp":"2025-06-16T00:42:11.837Z"}
{"level":"info","message":"Received request for Roman numeral conversion","span_id":"012537ff8a8d2b05","timestamp":"2025-06-16T00:42:31.833Z","trace_flags":"01",
{"level":"info","message":"Converted 23 => Roman numeral: XXIII","span_id":"012537ff8a8d2b05","timestamp":"2025-06-16T00:42:31.838Z","trace_flags":"01","trace_id
{"level":"info","message":"Received request for Roman numeral conversion","span_id":"ab8490db770cacfe","timestamp":"2025-06-16T00:42:32.748Z","trace_flags":"01",
{"level":"info","message":"Converted 23 => Roman numeral: XXIII","span_id":"ab8490db770cacfe","timestamp":"2025-06-16T00:42:32.749Z","trace_flags":"01","trace_id
{"level":"info","message":"Received request for Roman numeral conversion","span_id":"f253d3b9d27a6523","timestamp":"2025-06-16T00:42:35.304Z","trace_flags":"01",
{"level":"error","message":"Invalid input for Roman numeral conversion : ","span_id":"f253d3b9d27a6523","timestamp":"2025-06-16T00:42:35.305Z","trace_flags":"01"
```



```
adobe-task-backend  |   instrumentationScope: {
adobe-task-backend  |     name: '@opentelemetry/instrumentation-http',
adobe-task-backend  |     version: '0.202.0',
adobe-task-backend  |     schemaUrl: undefined
adobe-task-backend  |   },
adobe-task-backend  |   traceId: '43cf8e77b8858f6d31b145950fa0150d',
adobe-task-backend  |   parentSpanContext: undefined,
adobe-task-backend  |   traceState: undefined,
adobe-task-backend  |   name: 'GET /romannumeral',
adobe-task-backend  |   id: '35bd58e54d8d576a',
adobe-task-backend  |   kind: 1,
adobe-task-backend  |   timestamp: 1750081479330000,
adobe-task-backend  |   duration: 50511.042,
adobe-task-backend  |   attributes: {
adobe-task-backend  |     'http.url': 'http://localhost:8080/romannumeral?query=23',
adobe-task-backend  |     'http.host': 'localhost:8080',
adobe-task-backend  |     'net.host.name': 'localhost',
adobe-task-backend  |     'http.method': 'GET',
adobe-task-backend  |     'http.scheme': 'http',
adobe-task-backend  |     'http.target': '/romannumeral?query=23',
adobe-task-backend  |     'http.user_agent': 'Mozilla/5.0 (Macintosh; Intel Mac OS X 10_15_7) AppleWebKit/605.1.15 (KHTML, like Gecko) Version/18.5 Safari/605.1.15',
adobe-task-backend  |     'http.flavor': '1.1',
adobe-task-backend  |     'net.transport': 'ip_tcp',
adobe-task-backend  |     'net.host.ip': '::ffff:172.19.0.3',
adobe-task-backend  |     'net.host.port': 8080,
adobe-task-backend  |     'net.peer.ip': '::ffff:192.168.65.1',
adobe-task-backend  |     'net.peer.port': 25015,
adobe-task-backend  |     'http.status_code': 304,
adobe-task-backend  |     'http.status_text': 'NOT MODIFIED',
adobe-task-backend  |     'http.route': '/romannumeral'
adobe-task-backend  |   },
adobe-task-backend  |   status: { code: 0 },
adobe-task-backend  |   events: [],
adobe-task-backend  |   links: []
adobe-task-backend  | }
```

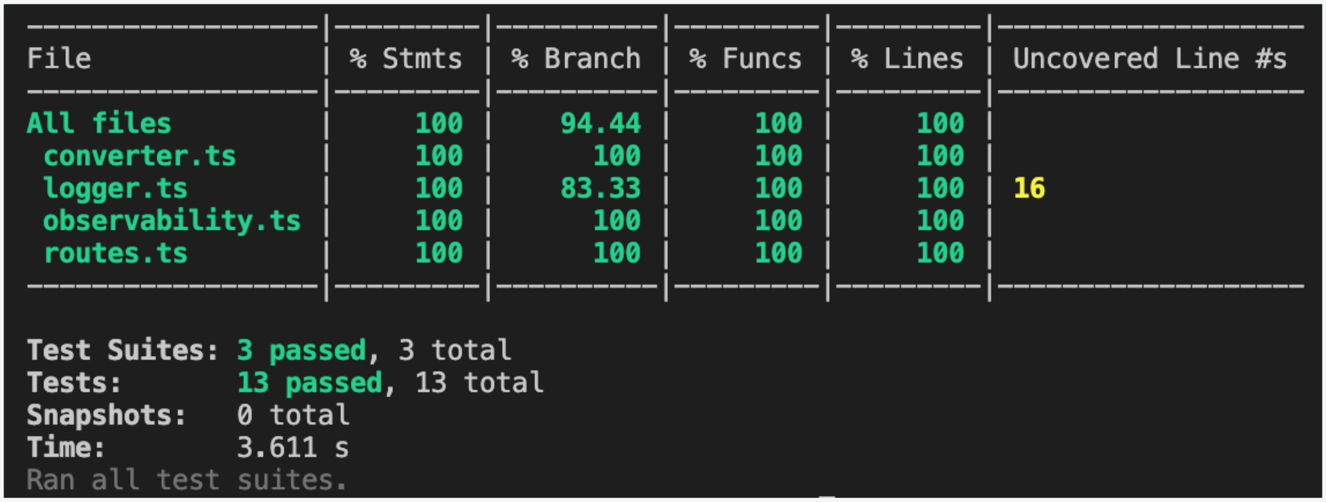
Testing & Coverage

- Frontend Coverage Summary



```
 PASS  src/App.test.tsx
-----------|---------|----------|---------|---------|-------------------
File       | % Stmts | % Branch | % Funcs | % Lines | Uncovered Line #s
-----------|---------|----------|---------|---------|-------------------
All files  |     100 |      100 |     100 |     100 |
 App.tsx   |     100 |      100 |     100 |     100 |
 index.tsx |     100 |      100 |     100 |     100 |
-----------|---------|----------|---------|---------|-------------------

Test Suites: 2 passed, 2 total
Tests:       7 passed, 7 total
Snapshots:   0 total
Time:        4.294 s
Ran all test suites.
```

- Backend Coverage Summary



```
------------------|---------|----------|---------|---------|-------------------
File              | % Stmts | % Branch | % Funcs | % Lines | Uncovered Line #s
------------------|---------|----------|---------|---------|-------------------
All files         |     100 |    94.44 |     100 |     100 |
 converter.ts     |     100 |      100 |     100 |     100 |
 logger.ts        |     100 |    83.33 |     100 |     100 | 16
 observability.ts |     100 |      100 |     100 |     100 |
 routes.ts        |     100 |      100 |     100 |     100 |
------------------|---------|----------|---------|---------|-------------------

Test Suites: 3 passed, 3 total
Tests:       13 passed, 13 total
Snapshots:   0 total
Time:        3.611 s
Ran all test suites.
```

Running the Project

- With Docker (Recommended):

  Refer to download Docker: “https://www.docker.com/"

  docker-compose up --build

- Manually:

  cd backend && npm install && npm start

  cd frontend/adobe-task-frontend && npm install && npm start

Advanced Observability Ideas

- Utilizing the **Correlation ID Middleware**, we can inject a **unique identifier** for each request to facilitate the tracing of logs across services.
- The **Error Rate Dashboard** enables the integration of Prometheus counters with **Grafana panels** for the monitoring of 4xx and 5xx error rates.
- **Custom trace events** allow us to enclose trace spans around **specific code blocks**, such as database lookups or conversion logic.
- **Log rotation** can be implemented using **Winston's daily log file rotation** feature via the \`winston-daily-rotate-file\` plugin.
- **CPU and memory metrics** will be monitored using \`process.memoryUsage()\` or Node.js Performance Hooks and exposed via custom Prometheus gauges.

Alternative Approaches to Observability

- **Logging**: While **Winston** provides flexible and structured logging, **Pino** could be used as a faster alternative for high-throughput applications requiring minimal logging overhead.
- **Metrics**: Prometheus via **prom-client** offers granular control, but **StatsD** with **Datadog** Agent may simplify setup and enhance visibility when using the Datadog ecosystem.
- **Tracing**: **OpenTelemetry + Jaeger** is vendor-neutral and extensible, though teams using AWS or commercial tools might find **X-Ray**, **Zipkin**, or **New Relic APM** easier to integrate and maintain.
- **Visualization & Alerting**: While **Jaeger** provides trace inspection, pairing **Prometheus** with **Grafana** could enhance observability by enabling dashboards for CPU, **memory**, **latency**, and **error rate** trends.

Conclusion

This project demonstrates a complete full-stack solution with best practices for observability, testing, and deployment. It reflects industry-ready design and maintainability.