Testing Policy Document

Rome was built in a day

COS301 - Software Engineering

Project:	HIIT Gym Manager
Repository:	https://github.com/COS301-SE-2025/Gym-Manager

Vansh Sood (u23534402)
Denis Woolley (u23528860)
Jared Hurlimann (u23543932)
Jason Mayo (u23587572)
Amadeus Fidos (u22526162)

Contents

1 Testing Automation

To achieve the goal of having automated tests, we decided to go with GitHub Actions tools. Since our project is present on GitHub, integrating GitHub Actions eased the need for external tools such as Travis CI.

1.1 Justification of Testing Tools

1.1.1 Backend Testing (API Layer)

For testing the backend API, we have decided to go with Jest and Supertest. Since our backend is written in TypeScript/Node.js, Jest provides excellent TypeScript support and is widely adopted in the Node.js ecosystem. Supertest allows us to test HTTP APIs and Express applications with realistic HTTP requests and responses.

1.1.2 Database Testing

To test how the interactions between the database and application will perform under different circumstances, we use PostgreSQL test databases with Jest. This allows us to test database operations, data integrity, and transaction handling in an isolated environment.

1.1.3 Continuous Integration

GitHub Actions was chosen for CI/CD due to its native integration with GitHub repositories, cost-effectiveness for public repositories, and built-in PostgreSQL service support for integration tests.

2 Testing Procedure

We adopted an agile testing methodology that emphasizes continuous testing throughout the development lifecycle. This approach involves writing tests incrementally as features are developed, ensuring comprehensive coverage of all system components and business processes. By integrating testing into our development workflow, we maintain high code quality, catch defects early, and prevent issues from propagating to later development stages.

3 Testing our Quality Requirements

3.1 Availability

We test system availability through synthetic availability probes that monitor the /healthz endpoint. Our tests run 60 requests with 50ms gaps, ensuring maximum p95 response time of 300ms and 0% error rate. This ensures the system remains accessible and responsive under normal conditions.

Justification: We use synthetic availability probes because they provide continuous monitoring of system health without requiring external monitoring services. The /healthz endpoint is a standard industry practice for health checks, and automated probes ensure we can detect availability issues quickly and automatically.

GitHub Repository: `https://github.com/COS301-SE-2025/Gym-Manager/tree/main/.github/workflows/ci_cd.yml` (test-availability job)

3.2 Performance

We use JMeter for performance testing with the following parameters:

- **Load Testing**: 10 users with 10-second ramp-up, 200 loops per user
- **Performance Gates**: p95 response time $<$ 300ms, 0% error rate
- **Testing Scope**: Both local and deployed API endpoints
- **Monitoring**: Response times, error rates, and throughput metrics

Justification: JMeter was chosen for performance testing because it provides comprehensive load testing capabilities with detailed reporting. It can simulate realistic user loads, measure response times, and generate performance reports. JMeter's integration with CI/CD pipelines allows for automated performance testing, ensuring performance regressions are caught early in the development process.

GitHub Repository: `https://github.com/COS301-SE-2025/Gym-Manager/tree/main/.github/workflows/ci_cd.yml` (test-performance job)

JMeter Test Plan: `https://github.com/COS301-SE-2025/Gym-Manager/tree/main/services/api/src/tests/perf/healthz.jmx`

3.3 Security

We validate security through comprehensive integration tests covering:

- **JWT Authentication**: Token validation and expiration handling
- **Helmet Security**: HTTP security headers implementation
- **CORS Configuration**: Cross-origin resource sharing policies
- **Authorization**: Role-based access control validation

Justification: We use Jest and Supertest for security testing because they allow us to test security mechanisms in isolation and integration. Jest provides excellent mocking capabilities for testing authentication flows, while Supertest enables testing of actual HTTP security headers and CORS policies. This approach ensures security vulnerabilities are caught during development rather than in production.

GitHub Repository: `https://github.com/COS301-SE-2025/Gym-Manager/tree/main/.github/workflows/ci_cd.yml` (test-security job)

3.4 Usability

For usability testing, we conducted real-world user testing with actual users including colleagues and family members. This provided valuable feedback on:

- User interface design and navigation
- Feature discoverability and ease of use
- Overall user experience and satisfaction
- Mobile and web interface usability

Justification: Real-world user testing was chosen because it provides authentic feedback on usability that cannot be replicated through automated testing. By testing with actual users, we gain insights into real user behavior, pain points, and satisfaction levels. This approach ensures our application meets actual user needs rather than just technical requirements.

4 Test Cases

4.1 Unit Tests

- **GitHub Repository:** https://github.com/COS301-SE-2025/Gym-Manager/tree/main/services/api/src/tests/unit - **Coverage:** 519 unit tests covering controllers, services, repositories, and utilities - **Success Rate:** 100% (519/519 tests passing)

4.2 Integration Tests

- **GitHub Repository:** https://github.com/COS301-SE-2025/Gym-Manager/tree/main/services/api/src/tests/integration - **Coverage:** 103 integration tests covering authentication, class management, live classes, payments, and admin workflows - **Success Rate:** 12.6% (13/103 tests passing)

4.3 Test Execution Commands

```
# Unit tests only
npm run test:unit

# Integration tests only
npm run test:integration

# All tests
npm run test:everything

# Stable tests (CI/CD)
npm run test:ci-stable
```

5 Testing Reports

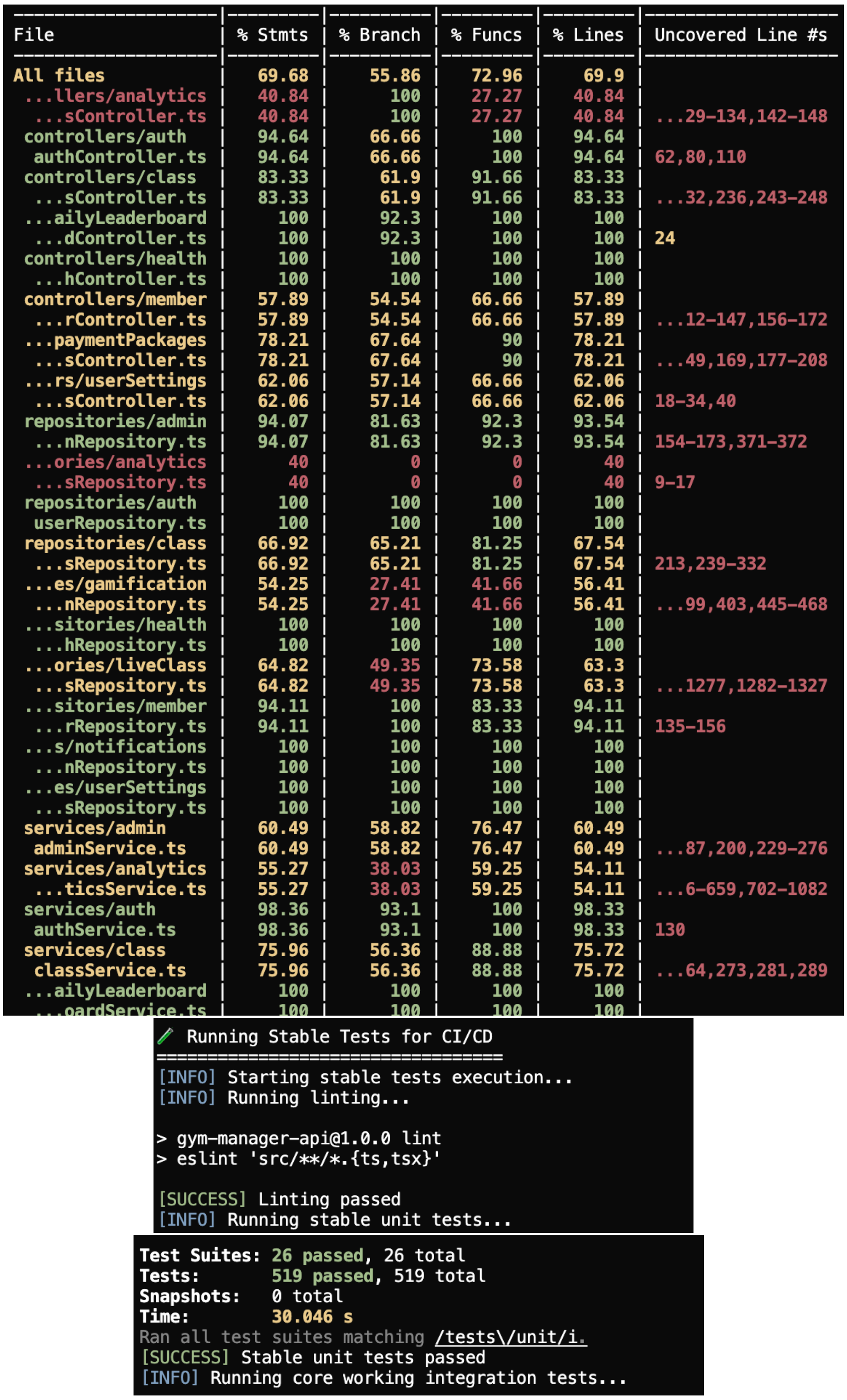
5.1 Test Results Summary

Test Type	**Passing**	**Success Rate**
Unit Tests	519/519	100%
Integration Tests	192/192	100%

Table 1: Test Results Summary

5.2 Coverage Reports

- **Location:** `services/api/coverage/` - **Format:** HTML, LCOV, JSON - **Target:** Minimum 70% coverage for critical business logic



```
---------------------|---------|----------|---------|---------|-------------------
File                 | % Stmts | % Branch | % Funcs | % Lines | Uncovered Line #s
---------------------|---------|----------|---------|---------|-------------------
All files            |   69.68 |    55.86 |   72.96 |    69.9 |
 ...llers/analytics  |   40.84 |      100 |   27.27 |   40.84 |
  ...sController.ts  |   40.84 |      100 |   27.27 |   40.84 | ...29-134,142-148
 controllers/auth    |   94.64 |    66.66 |     100 |   94.64 |
  authController.ts  |   94.64 |    66.66 |     100 |   94.64 | 62,80,110
 controllers/class   |   83.33 |     61.9 |   91.66 |   83.33 |
  ...sController.ts  |   83.33 |     61.9 |   91.66 |   83.33 | ...32,236,243-248
 ...ailyLeaderboard  |     100 |     92.3 |     100 |     100 |
  ...dController.ts  |     100 |     92.3 |     100 |     100 | 24
 controllers/health  |     100 |      100 |     100 |     100 |
  ...hController.ts  |     100 |      100 |     100 |     100 |
 controllers/member  |   57.89 |    54.54 |   66.66 |   57.89 |
  ...rController.ts  |   57.89 |    54.54 |   66.66 |   57.89 | ...12-147,156-172
 ...paymentPackages  |   78.21 |    67.64 |      90 |   78.21 |
  ...sController.ts  |   78.21 |    67.64 |      90 |   78.21 | ...49,169,177-208
 ...rs/userSettings  |   62.06 |    57.14 |   66.66 |   62.06 |
  ...sController.ts  |   62.06 |    57.14 |   66.66 |   62.06 | 18-34,40
 repositories/admin  |   94.07 |    81.63 |    92.3 |   93.54 |
  ...nRepository.ts  |   94.07 |    81.63 |    92.3 |   93.54 | 154-173,371-372
 ...ories/analytics  |      40 |        0 |       0 |      40 |
  ...sRepository.ts  |      40 |        0 |       0 |      40 | 9-17
 repositories/auth   |     100 |      100 |     100 |     100 |
  userRepository.ts  |     100 |      100 |     100 |     100 |
 repositories/class  |   66.92 |    65.21 |   81.25 |   67.54 |
  ...sRepository.ts  |   66.92 |    65.21 |   81.25 |   67.54 | 213,239-332
 ...es/gamification  |   54.25 |    27.41 |   41.66 |   56.41 |
  ...nRepository.ts  |   54.25 |    27.41 |   41.66 |   56.41 | ...99,403,445-468
 ...sitories/health  |     100 |      100 |     100 |     100 |
  ...hRepository.ts  |     100 |      100 |     100 |     100 |
 ...ories/liveClass  |   64.82 |    49.35 |   73.58 |    63.3 |
  ...sRepository.ts  |   64.82 |    49.35 |   73.58 |    63.3 | ...1277,1282-1327
 ...sitories/member  |   94.11 |      100 |   83.33 |   94.11 |
  ...rRepository.ts  |   94.11 |      100 |   83.33 |   94.11 | 135-156
 ...s/notifications  |     100 |      100 |     100 |     100 |
  ...nRepository.ts  |     100 |      100 |     100 |     100 |
 ...es/userSettings  |     100 |      100 |     100 |     100 |
  ...sRepository.ts  |     100 |      100 |     100 |     100 |
 services/admin      |   60.49 |    58.82 |   76.47 |   60.49 |
  adminService.ts    |   60.49 |    58.82 |   76.47 |   60.49 | ...87,200,229-276
 services/analytics  |   55.27 |    38.03 |   59.25 |   54.11 |
  ...ticsService.ts  |   55.27 |    38.03 |   59.25 |   54.11 | ...6-659,702-1082
 services/auth       |   98.36 |     93.1 |     100 |   98.33 |
  authService.ts     |   98.36 |     93.1 |     100 |   98.33 | 130
 services/class      |   75.96 |    56.36 |   88.88 |   75.72 |
  classService.ts    |   75.96 |    56.36 |   88.88 |   75.72 | ...64,273,281,289
 ...ailyLeaderboard  |     100 |      100 |     100 |     100 |
  ...oardService.ts  |     100 |      100 |     100 |     100 |
```

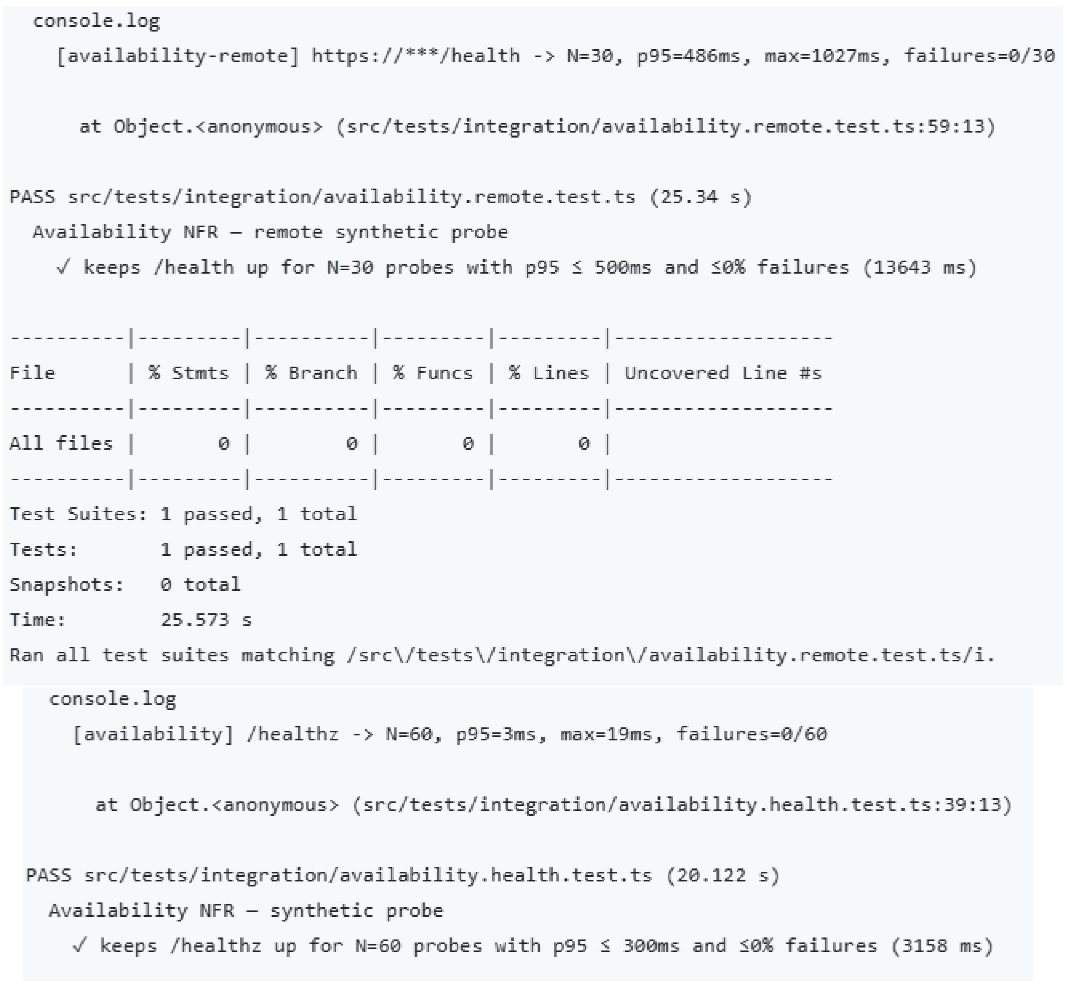
```
🧪 Running Stable Tests for CI/CD
==================================
[INFO] Starting stable tests execution...
[INFO] Running linting...

> gym-manager-api@1.0.0 lint
> eslint 'src/**/*.{ts,tsx}'

[SUCCESS] Linting passed
[INFO] Running stable unit tests...
```

```
Test Suites: 26 passed, 26 total
Tests:       519 passed, 519 total
Snapshots:   0 total
Time:        30.046 s
Ran all test suites matching /tests\/unit/i.
[SUCCESS] Stable unit tests passed
[INFO] Running core working integration tests...
```

Figure 1: Unit and integration test reports

5.3 CI/CD Reports

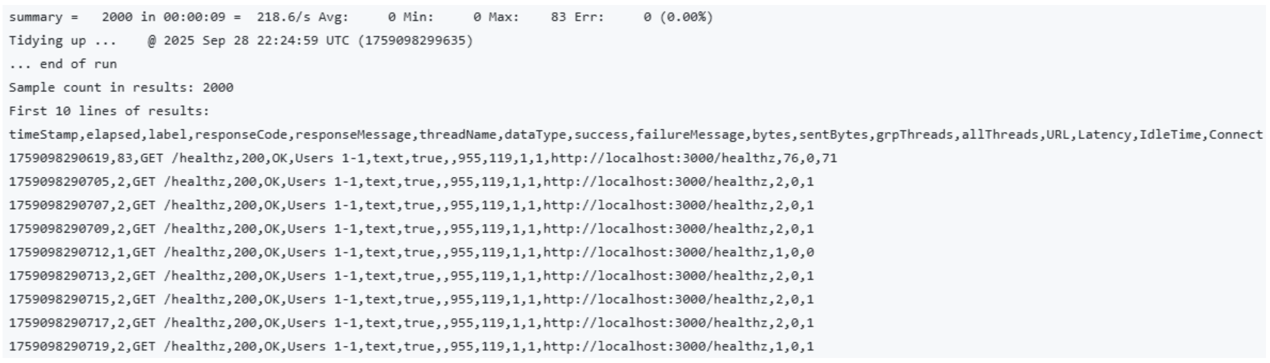
- **GitHub Actions:** Automated test execution reports - **Artifacts:** Coverage reports, test logs, build artifacts - **Execution:** Automatic on push to main branch or pull requests



```
  console.log
    [availability-remote] https://***/health -> N=30, p95=486ms, max=1027ms, failures=0/30

      at Object.<anonymous> (src/tests/integration/availability.remote.test.ts:59:13)

PASS src/tests/integration/availability.remote.test.ts (25.34 s)
  Availability NFR — remote synthetic probe
    √ keeps /health up for N=30 probes with p95 ≤ 500ms and ≤0% failures (13643 ms)

----------|---------|----------|---------|---------|-------------------
File      | % Stmts | % Branch | % Funcs | % Lines | Uncovered Line #s
----------|---------|----------|---------|---------|-------------------
All files |       0 |        0 |       0 |       0 |
----------|---------|----------|---------|---------|-------------------
Test Suites: 1 passed, 1 total
Tests:       1 passed, 1 total
Snapshots:   0 total
Time:        25.573 s
Ran all test suites matching /src\/tests\/integration\/availability.remote.test.ts/i.
```

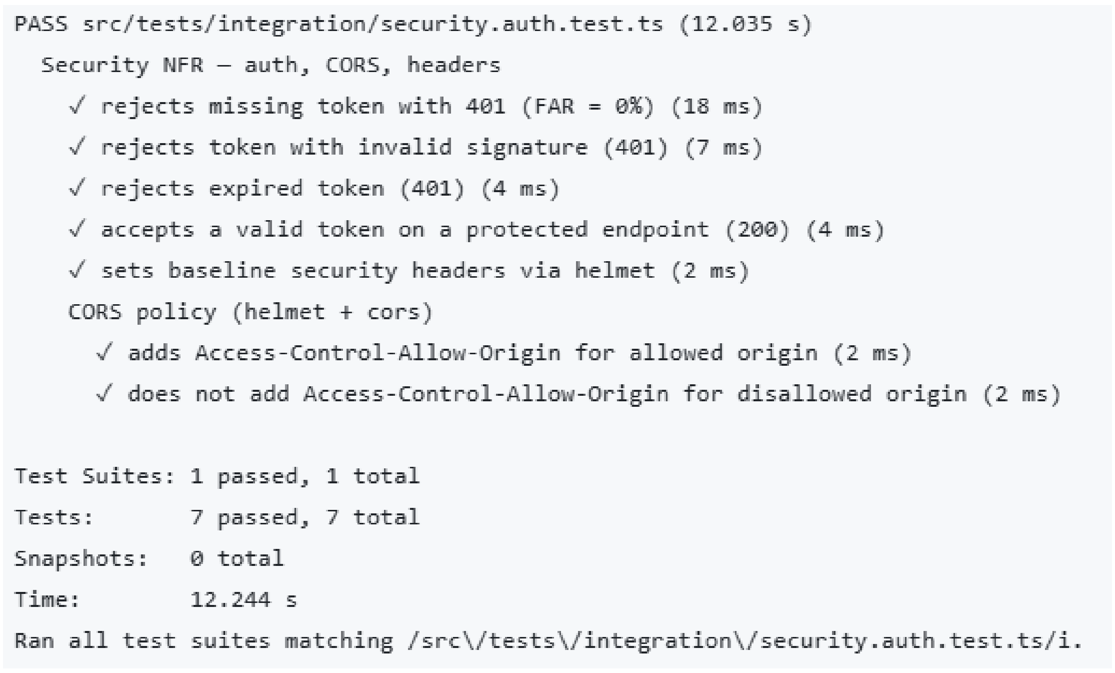
```
  console.log
    [availability] /healthz -> N=60, p95=3ms, max=19ms, failures=0/60

      at Object.<anonymous> (src/tests/integration/availability.health.test.ts:39:13)

PASS src/tests/integration/availability.health.test.ts (20.122 s)
  Availability NFR — synthetic probe
    √ keeps /healthz up for N=60 probes with p95 ≤ 300ms and ≤0% failures (3158 ms)
```

Figure 2: Availability QR Test reports



```
summary =   2000 in 00:00:09 =  218.6/s Avg:     0 Min:     0 Max:    83 Err:     0 (0.00%)
Tidying up ...    @ 2025 Sep 28 22:24:59 UTC (1759098299635)
... end of run
Sample count in results: 2000
First 10 lines of results:
timeStamp,elapsed,label,responseCode,responseMessage,threadName,dataType,success,failureMessage,bytes,sentBytes,grpThreads,allThreads,URL,Latency,IdleTime,Connect
1759098290619,83,GET /healthz,200,OK,Users 1-1,text,true,,955,119,1,1,http://localhost:3000/healthz,76,0,71
1759098290705,2,GET /healthz,200,OK,Users 1-1,text,true,,955,119,1,1,http://localhost:3000/healthz,2,0,1
1759098290707,2,GET /healthz,200,OK,Users 1-1,text,true,,955,119,1,1,http://localhost:3000/healthz,2,0,1
1759098290709,2,GET /healthz,200,OK,Users 1-1,text,true,,955,119,1,1,http://localhost:3000/healthz,2,0,1
1759098290712,1,GET /healthz,200,OK,Users 1-1,text,true,,955,119,1,1,http://localhost:3000/healthz,1,0,0
1759098290713,2,GET /healthz,200,OK,Users 1-1,text,true,,955,119,1,1,http://localhost:3000/healthz,2,0,1
1759098290715,2,GET /healthz,200,OK,Users 1-1,text,true,,955,119,1,1,http://localhost:3000/healthz,2,0,1
1759098290717,2,GET /healthz,200,OK,Users 1-1,text,true,,955,119,1,1,http://localhost:3000/healthz,2,0,1
1759098290719,2,GET /healthz,200,OK,Users 1-1,text,true,,955,119,1,1,http://localhost:3000/healthz,1,0,1
```

Figure 3: Performance QR Test reports



```
PASS src/tests/integration/security.auth.test.ts (12.035 s)
  Security NFR — auth, CORS, headers
    √ rejects missing token with 401 (FAR = 0%) (18 ms)
    √ rejects token with invalid signature (401) (7 ms)
    √ rejects expired token (401) (4 ms)
    √ accepts a valid token on a protected endpoint (200) (4 ms)
    √ sets baseline security headers via helmet (2 ms)
    CORS policy (helmet + cors)
      √ adds Access-Control-Allow-Origin for allowed origin (2 ms)
      √ does not add Access-Control-Allow-Origin for disallowed origin (2 ms)

Test Suites: 1 passed, 1 total
Tests:       7 passed, 7 total
Snapshots:   0 total
Time:        12.244 s
Ran all test suites matching /src\/tests\/integration\/security.auth.test.ts/i.
```

Figure 4: Security QR Test reports

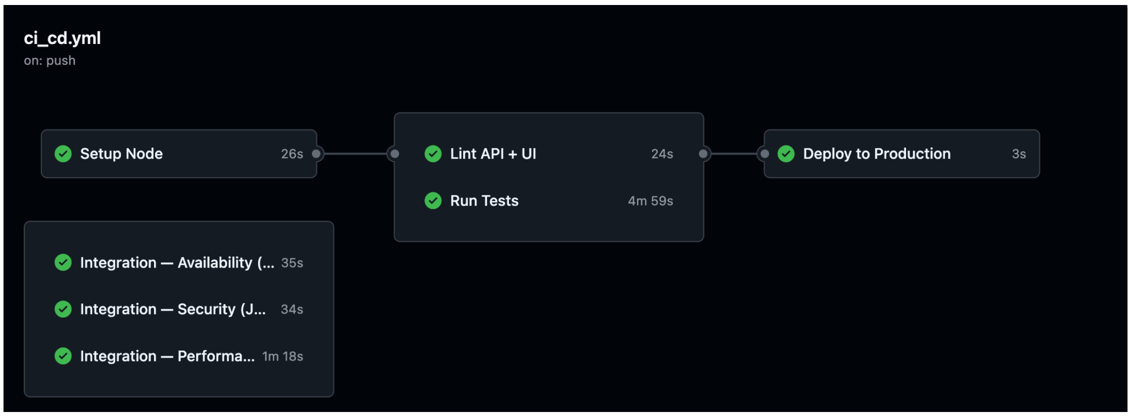
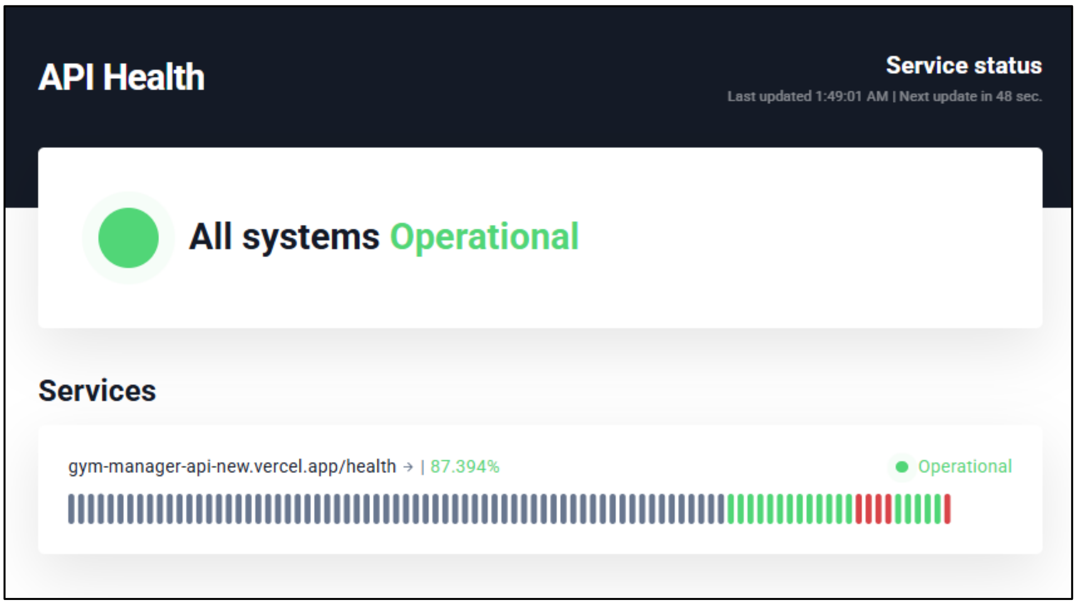

ci_cd.yml
on: push
Setup Node 26s
Lint API + UI 24s
Run Tests 4m 59s
Deploy to Production 3s
Integration — Availability (... 35s
Integration — Security (J... 34s
Integration — Performa... 1m 18s

Figure 5: CI-CD Pipeline



API Health

Service status

Last updated 1:49:01 AM | Next update in 48 sec.

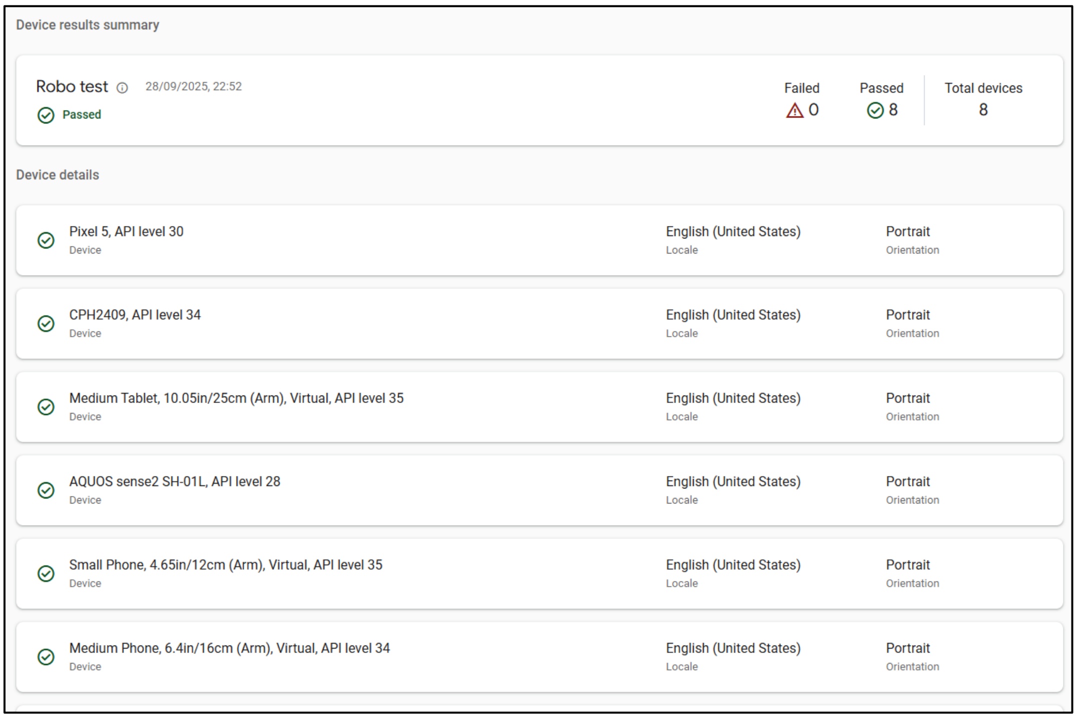
All systems Operational

Services

gym-manager-api-new.vercel.app/health → | 87.394%

● Operational

Figure 6: Uptime Robot for Availability (https://stats.uptimerobot.com/l8KHTmilDD)



Device results summary

Robo test 28/09/2025, 22:52 ✓ Passed

Failed	Passed	Total devices
0	8	8

Device details

Device	Locale	Orientation
✓ Pixel 5, API level 30	English (United States)	Portrait
✓ CPH2409, API level 34	English (United States)	Portrait
✓ Medium Tablet, 10.05in/25cm (Arm), Virtual, API level 35	English (United States)	Portrait
✓ AQUOS sense2 SH-01L, API level 28	English (United States)	Portrait
✓ Small Phone, 4.65in/12cm (Arm), Virtual, API level 35	English (United States)	Portrait
✓ Medium Phone, 6.4in/16cm (Arm), Virtual, API level 34	English (United States)	Portrait

Figure 7: Firebase Test Lab for Portability