

COMP3011 Technical Report: EventHub API

Module: COMP3011 – Web Services and Web Data

Student: Nathaniel Sebastian (sc232ns)

Date: 5th February 2026

Word Count: ~1,800 words

Submission Links

Resource	Link
GitHub Repository	github.com/NathS04/comp3011-cw1-api
Live API	comp3011-cw1-api.onrender.com

1. Compliance Checklist

Requirement	Status	Location
GitHub repository	✓	github.com/NathS04/comp3011-cw1-api
API documentation PDF	✓	docs/API_DOCUMENTATION.pdf
Technical report PDF	✓	TECHNICAL_REPORT.pdf
Presentation slides	✓	docs/PRESENTATION_SLIDES.pptx
GenAI logs	✓	docs/GENAI_EXPORT_LOGS.pdf
README.md	✓	Root directory
Deployed API	✓	comp3011-cw1-api.onrender.com
Novel data integration	✓	Leeds TEN XML with SHA256 provenance
Authentication	✓	JWT + PBKDF2
Test suite	✓	41 tests passing

Marker Evidence Cross-Reference

Requirement	Evidence	File Path
Tests pass (41)	pytest -q output	VERIFICATION_INSTRUCTIONS.md

Requirement	Evidence	File Path
Live deployment	<code>/health → environment: prod</code>	comp3011-cw1-api.onrender.com/health
API documentation	Full endpoint reference	<code>docs/API_DOCUMENTATION.pdf</code>
Security headers	7 headers on every response	<code>app/core/middleware.py</code>
Rate limiting	120/min global, 10/min login	<code>app/core/rate_limit.py</code>
Novel dataset	Leeds TEN XML, SHA256 hashed	<code>scripts/import_dataset.py</code> , <code>docs/DATASET_SOURCE.md</code>
GenAI logs	Tools, failures, dates	<code>docs/GENAI_EXPORT_LOGS.pdf</code>
RBAC	Admin-only <code>/admin/*</code> with 403	<code>app/api/admin.py</code> , <code>app/core/auth.py</code>
ETag/304	RFC 7232 compliance	<code>app/core/middleware.py</code>
Presentation	Slide outline + viva prep	<code>docs/PRESENTATION_OUTLINE.md</code>

2. Reproducibility

```
git clone https://github.com/NathS04/comp3011-cw1-api.git && cd comp3011-cw1-api
python3 -m venv venv && source venv/bin/activate
pip install -r requirements.txt
export DATABASE_URL="sqlite:///./app.db" SECRET_KEY="dev-secret"
alembic upgrade head
pytest -q # Expected: 41 passed
uvicorn app.main:app --reload
```

3. Dataset Provenance

Attribute	Value
Source	Leeds Temporary Event Notices [1]
Provider	Leeds City Council (Data Mill North)
Licence	Open Government Licence v3.0
Format	XML
Fields Mapped	Reference_Number → ID, Premises_Name → Title

Attribute	Value
Provenance	SHA256 hash stored in <code>ImportRun</code> table

4. Architecture

Client → Middleware (Rate Limit, Headers, ETag) → FastAPI → Auth → Routes → CRUD → SQLAlchemy → DB

Design Principles: - Layered architecture: thin routes, fat CRUD - Middleware handles cross-cutting concerns (rate limiting, security headers, ETag) - Dual database support via Alembic (SQLite dev, PostgreSQL prod)

5. Outstanding Evidence

5.1 Test Coverage

Category	Count	Coverage
Auth	6	Register, login, validation
Events CRUD	5	Create, read, update, delete, pagination
RSVPs	4	Create, duplicate rejection
Analytics	4	Seasonality, trending, recommendations
Admin/Import	3	Idempotency, provenance
RBAC	2	403 for non-admin
Attendees	4	CRUD, uniqueness
Health	1	Returns metadata
ETag	3	Generation, 304, mismatch
Security Headers	3	On 200, 404, 403
Error Handling	2	Sanitization, request_id
Total	41	

Test isolation: In-memory SQLite with `StaticPool` ; tables created/dropped per test. Runtime: <1.5s.

5.2 Security Header Policy

All responses include:

Header	Value	OWASP Reference
X-Request-ID	UUID v4	Tracing/debugging
X-Content-Type-Options	nosniff	MIME sniffing prevention [2]
X-Frame-Options	DENY	Clickjacking protection [2]
Referrer-Policy	no-referrer	Privacy [2]
Permissions-Policy	geolocation=(), microphone=(), camera=()	Feature restriction [2]
Cross-Origin-Resource-Policy	same-site	Side-channel protection [2]
Cache-Control	no-store (default) / no-cache (ETag)	Sensitive data protection

5.3 Rate Limiting

Scope	Limit	Rationale
Global	120 req/min	Prevents abuse without impacting normal use
/auth/login	10 req/min	Credential stuffing mitigation [3]

429 response includes request_id for correlation: {"detail": "Too Many Requests", "request_id": "<uuid>"}

5.4 ETag/304 Standard Compliance

Implementation follows RFC 7232 [4]:

1. GET /events and /events/{id} compute SHA256 of response body
2. Response includes ETag: "<hash>"
3. Client sends If-None-Match: "<hash>"
4. Server returns 304 Not Modified with empty body if match
5. 304 still includes ETag, X-Request-ID, and security headers

5.5 Provenance Hashing

Each import run stores: - SHA256 hash of source file - Timestamp, duration, row counts - Enables idempotency (duplicate hash = skip)

6. Design Trade-offs

Decision	Alternative	Trade-off	Justification
In-memory rate limiting	Redis	Single-process only	Acceptable for free-tier Render; documented as limitation
ETag via body hash	DB <code>updated_at</code> column	Recalculates per request	Simpler implementation; no schema changes
JWT without refresh	Refresh token flow	30-min hard limit	Simpler for coursework; users re-login
SQLite/PostgreSQL dual	PostgreSQL-only	Dev overhead	Alembic abstracts dialect; faster local iteration
Sanitized 500 errors	Detailed error messages	Less debugging info	Security first; <code>request_id</code> enables log correlation

7. Analytics Endpoints

Endpoint	Computation	Use Case
<code>/analytics/events/seasonality</code>	<code>COUNT(*) GROUP BY month</code>	Identify peak event periods
<code>/analytics/events/trending</code>	<code>(recent_rsvps × 1.5) + (total_rsvps × 0.5)</code>	Surface popular events
<code>/events/recommendations</code>	Filter by user's past RSVP categories	Personalised discovery

Trade-off Rationale: Simpler algorithms chosen for coursework scope; documented as extensible via Redis caching and ML in production.

8. Security Implementation

Security Measure	Implementation	File
Password Hashing	PBKDF2-SHA256	<code>app/core/auth.py</code>
JWT Signing	HS256, 30-min expiry	<code>app/core/auth.py</code>
RBAC	<code>is_admin</code> flag, 403 on <code>/admin/*</code>	<code>app/core/auth.py</code> , <code>app/api/admin.py</code>
Rate Limiting	120/min global, 10/min login	<code>app/core/rate_limit.py</code>

Security Measure	Implementation	File
Request Tracing	X-Request-ID on all responses	app/core/middleware.py
Security Headers	6 headers on all responses	app/core/middleware.py
ETag Caching	SHA256 body hash, 304 support	app/core/middleware.py
Error Sanitization	Generic 500 message, no stack trace	app/core/middleware.py

9. Deployment

Platform: Render.com [5]

Database: Managed PostgreSQL

Config: render.yaml

Variables: DATABASE_URL , SECRET_KEY , ENVIRONMENT=prod , ALLOWED_ORIGINS ,
RATE_LIMIT_ENABLED

Cold Start Note: Free tier spins down after 15 min; first request may take 30–60s.

10. Version Control

Commit history available at: <https://github.com/NathS04/comp3011-cw1-api/commits/main>

Key commits demonstrate incremental development: - Feature additions (analytics, RBAC, ETag) - Bug fixes (middleware headers, rate limit format) - Test additions (41 total)

11. GenAI Usage

Tools: Google Gemini (Antigravity), Claude, ChatGPT

Creative Applications: 1. Compared RSVP storage approaches (embedded vs relational) 2. Evaluated rate limiting options (in-memory vs Redis) [3] 3. Researched ETag/If-None-Match RFC 7232 compliance [4] 4. Compared middleware ordering alternatives for header injection

Failures Caught: - Missing requests dependency → ModuleNotFoundError - Placeholder test with pass → False coverage - Deprecated Query(regex=...) → Warning - Headers not applied to 429 responses → Fixed middleware flow

Full logs: [docs/GENAI_EXPORT_LOGS.pdf](#)

12. Limitations & Future Work

Limitation	Improvement
No token refresh	Implement refresh tokens
In-memory rate limiting	Redis-backed for horizontal scaling
No CSP header	Add Content-Security-Policy
Basic recommendations	Collaborative filtering with user embeddings

References

-
- [1] Leeds City Council, "Temporary Event Notices," Data Mill North, <https://datamillnorth.org/dataset/temporary-event-notice> (Open Government Licence v3.0)
- [2] OWASP, "HTTP Security Response Headers," <https://cheatsheetseries.owasp.org/cheatsheets/HTTP-Headers-Cheat-Sheet.html>
- [3] OWASP, "Rate Limiting," https://cheatsheetseries.owasp.org/cheatsheets/Denial_of_Service-Cheat-Sheet.html
- [4] IETF, "RFC 7232: HTTP/1.1 Conditional Requests," <https://tools.ietf.org/html/rfc7232>
- [5] Render, "Web Services Documentation," <https://render.com/docs/web-services>
-