COMP3011 Technical Report: EventHub API

Module: COMP3011 – Web Services and Web Data
Student: Nathaniel Sebastian (sc232ns)
Date: 5th February 2026
Word Count: ~1,500 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	✅	**39 tests passing**

---

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: 39 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
Provenance	SHA256 hash stored in `ImportRun` 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. Security Implementation Evidence

Security Measure	Implementation	File
Password Hashing	PBKDF2-SHA256	`app/core/auth.py`
JWT Signing	HS256, 30-min expiry	`app/core/auth.py`
RBAC	`is_admin` flag, 403 on `/admin/*`	`app/core/auth.py`, `app/api/admin.py`
Rate Limiting	120/min global, 10/min login	`app/core/rate_limit.py`
Request Tracing	`X-Request-ID` on all responses	`app/core/middleware.py`
Security Headers	nosniff, DENY, no-store/no-cache	`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`

---

6. Analytics Endpoints

Endpoint	Computation	Use Case

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

Evaluation:

- **Strengths:** Trending score balances recency with popularity; seasonality provides actionable insights for event planners.
- **Limitations:** Recommendations are category-based only (no collaborative filtering); trending score is not normalised by event age.
- **Trade-off Rationale:** Simpler algorithms chosen for coursework scope; documented as extensible via Redis caching and ML in production.

---

7. Testing Strategy

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
Middleware	2	Headers, rate limiting
Attendees	4	CRUD, uniqueness
Health	1	Returns metadata
ETag	3	Generation, 304, mismatch
Error Handling	1	Sanitization
Total	**39**	

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

---

8. 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 `updated_at`	Recalculates per request	Simpler; no schema changes needed
JWT without refresh	Refresh tokens	30-min hard limit	Simpler for coursework; users re-login
SQLite/Postgres dual	Postgres-only	Dev overhead	Alembic abstracts dialect; faster local iteration

---

9. Deployment

Platform: Render.com [3]
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 (39 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) [2]
3. Researched ETag/If-None-Match RFC 7232 compliance

Failures Caught:

- Missing `requests` dependency → ModuleNotFoundError
- Placeholder test with `pass` → False coverage
- Deprecated `Query(regex=...)` → Warning

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-notices (Open Government Licence v3.0)

[2] OWASP, "Rate Limiting," https://cheatsheetseries.owasp.org/cheatsheets/Denial_of_Service_Cheat_Sheet.html

[3] Render, "Web Services Documentation," https://render.com/docs/web-services

[4] IETF, "RFC 7232: HTTP/1.1 Conditional Requests," https://tools.ietf.org/html/rfc7232

COMP3011 CW1, University of Leeds