Technical Report

COMP3011 Technical Report: EventHub API

Module: COMP3011 – Web Services and Web Data
Student: Nathaniel Sebastian (sc232ns)
Date: 4th February 2026
GitHub: github.com/NathS04/comp3011-cw1-api
Live API: comp3011-cw1-api.onrender.com

---

1. Introduction

EventHub is a RESTful API for event registration management, designed for university societies and community organisations. It provides CRUD operations for events, attendees, and RSVPs, plus analytics endpoints for insights.

Key Features: - JWT authentication for secure write operations - Data provenance tracking for external dataset imports - Analytics: seasonality, trending, and personalised recommendations - 25 automated tests with full coverage

---

2. Technology Stack

Component	Choice	Rationale
Framework	FastAPI	Async support, automatic OpenAPI docs, Pydantic integration
Database	SQLite/PostgreSQL	Zero-config dev, production-ready via `DATABASE_URL`
ORM	SQLAlchemy 2.x	Industry standard, relationship support, migrations
Migrations	Alembic	Version-controlled schema changes

Component	Choice	Rationale
Auth	JWT (python-jose)	Stateless, fits REST principles
Testing	pytest	Isolated in-memory DB with StaticPool

---

3. Architecture

The system uses a layered architecture:

```
HTTP Client → FastAPI app → Routes → CRUD → Models → Database
```

Design Principle: Routes are thin handlers; business logic lives in `crud.py` for testability.

---

4. Data Model

Core Tables: users, events, attendees, rsvps

Provenance Tables (Novel): - `data_sources`: Tracks external data origins (name, URL, license) - `import_runs`: Logs each import execution (rows, errors, status)

Key Constraints: - Unique emails for users and attendees - Unique RSVP per (event, attendee) pair - Events link to `data_sources` via `source_id`

---

5. Key Design Decisions

RSVP Storage

Choice: Separate `rsvps` table vs. embedded list.
Trade-off: Adds JOINs but enables timestamps, independent queries, and database-level uniqueness.

JWT vs Sessions

Choice: Stateless JWT tokens.
Trade-off: Simpler (no session store) but tokens can't be revoked until expiry.

SQLite Compatibility

Choice: SQLite for dev, PostgreSQL for prod.
Solution: Alembic's `batch_alter_table` for SQLite foreign key constraints.

---

6. Novel Features

Data Import Pipeline

The `scripts/import_dataset.py` script: 1. Registers data source in `data_sources` 2. Creates `ImportRun` record with "running" status 3. Imports rows idempotently using `source_record_id` 4. Logs errors and final counts

Analytics Endpoints

Endpoint	Purpose
`/analytics/events/seasonality`	Monthly event aggregation
`/analytics/events/trending`	RSVP-based trending score
`/events/recommendations`	Personalised suggestions

Trending Formula: `score = (recent_rsvps × 1.5) + (total_rsvps × 0.5)`

---

7. Testing

- **25 tests** covering auth, events, attendees, RSVPs, analytics
- In-memory SQLite with `StaticPool` for isolation
- Both success paths and error cases tested

```
$ pytest -q
25 passed, 2 warnings in 0.72s
```

---

8. Deployment

Platform: Render.com with GitHub auto-deploy

Environment Variables: - `DATABASE_URL` – PostgreSQL connection - `SECRET_KEY` – JWT signing key - `ENVIRONMENT` – production

Verification: `/docs` accessible, `/health` returns `{"ok": true}`

---

9. GenAI Declaration

Tools Used

- **Google Gemini (Antigravity):** Primary – architecture, coding, debugging
- **ChatGPT (GPT-4):** Secondary – feedback interpretation

Usage Approach

1. Described requirements, AI suggested layered structure
2. Generated initial models, schemas, CRUD functions
3. Debugged tracebacks collaboratively
4. Explored alternatives (JWT vs sessions, embedded vs relational RSVPs)

Critical Evaluation

- Always ran generated code before committing
- Caught security issues (hardcoded secrets) in review
- Fixed AI-introduced bugs (circular imports, bcrypt compatibility)

Full logs: See `docs/appendix_genai_logs.md`

---

10. Limitations & Future Work

Current: - Single-tenant (no role-based access) - No email verification - 30-minute token expiry without refresh

Future: - Role-based permissions - Capacity enforcement - Email notifications - Rate limiting

---

Word Count: ~800 words (excluding code/tables)

Report for COMP3011 CW1, University of Leeds