GenAI Conversation Export Logs

Module: COMP3011 – Web Services and Web Data
Student: Nathaniel Sebastian (sc232ns)
Date: 5th February 2026
Tools Used: Google Gemini (Antigravity), Claude (Anthropic), ChatGPT (OpenAI)

Summary

Tool	Purpose	Sessions
Google Gemini	Coding, debugging, test generation, security hardening	12
Claude	Documentation polish, refactoring review	2
ChatGPT	Early brainstorming	1

Test Count: 41 tests passing (verified via `pytest -q`)

High-Level Design Decisions

Decision 1: Middleware Ordering for Security Headers

Problem: Security headers were not being applied to all response types (429, 500, 404).

Options Compared: 1. **Per-route decorator:** Apply headers in each route function - Pros: Explicit control - Cons: Repetitive, easy to miss routes 2. **Exception handler headers:** Apply in exception handlers only - Pros: Catches errors - Cons: Misses successful responses 3. **Middleware with `add_headers` helper:** Central function called on ALL response paths - Pros: DRY, guarantees coverage - Cons: Slightly more complex middleware

Decision: Option 3 (middleware with helper). Created `add_headers(resp)` function called on rate limit returns, exception catches, and normal response paths.

Verification: `tests/test_headers_security.py` confirms headers on 200, 404, and 403 responses.

Decision 2: ETag Implementation Approach

Problem: Need conditional GET (304 Not Modified) for event caching.

Options Compared: 1. **Database `updated_at` column:** Store last modification timestamp - Pros: No body computation needed - Cons: Requires schema change, doesn't catch list composition changes 2. **Response

body SHA256 hash: Compute ETag from actual response bytes - Pros: Accurate, no schema changes - Cons: Must consume response body in middleware

Decision: Option 2 (body hash). Middleware intercepts GET 200 responses on `/events*`, computes SHA256, stores in `ETag` header. On `If-None-Match` match, returns 304 with empty body.

Implementation: `app/core/middleware.py` lines 64–91.

Reference: RFC 7232 (HTTP Conditional Requests) [1]

Decision 3: Error Sanitization Pattern

Problem: 500 error responses could leak stack traces or sensitive exception details.

Options Compared: 1. **Try/except per route:** Catch and sanitize in each handler - Pros: Route-level control - Cons: Easy to miss routes, code duplication 2. **Custom exception handler:** FastAPI exception_handler for Exception - Pros: Centralized - Cons: Some exceptions bypass handlers 3. **Middleware catch-all:** Wrap `call_next` in try/except - Pros: Catches everything, includes request_id - Cons: Must avoid catching expected exceptions

Decision: Option 3 (middleware catch-all). Generic message returned: `{"detail":"Internal Server Error","request_id":"<uuid>"}`. Full exception logged server-side with `logger.error(..., exc_info=True)`.

Verification: `tests/test_admin_errors.py` mocks an exception and confirms no sensitive data in response.

Session Log: Architecture Planning

Tool: Google Gemini
Prompt: "Best architecture for FastAPI + SQLAlchemy REST API?"

AI Suggestion: Layered architecture (routes → CRUD → models → schemas).

Decision: Adopted. Created `app/api/routes.py`, `app/crud.py`, `app/models.py`, `app/schemas.py`.

Session Log: RSVP Storage

Tool: Google Gemini
Prompt: "Embedded list vs separate table for RSVPs?"

AI Analysis: - Embedded: Simpler, no JOINs, but no uniqueness constraint - Separate: UNIQUE(event_id, attendee_id), timestamps, cascade delete

Decision: Separate table for data integrity.

Session Log: Rate Limiting Strategy

Tool: Google Gemini
Prompt: "In-memory vs Redis rate limiting for coursework?"

AI Analysis: - In-memory: Simple, single-process, no infrastructure - Redis: Distributed, production-grade, requires setup

Decision: In-memory for coursework; documented Redis as future upgrade.

Trade-off Rationale: Appropriate for single-worker Render free tier.

Session Log: ETag Implementation

Tool: Google Gemini
Prompt: "How to implement ETag + If-None-Match correctly?"

AI Response: Referenced RFC 7232. Compute SHA256 of response body, wrap in quotes, check If-None-Match header, return 304 with no body.

Implementation: - Middleware intercepts GET 200 responses on `/events*` - Computes ETag, sets header - Returns 304 if match - 304 includes ETag + X-Request-ID + security headers

---

Failures & Corrections

AI Failure	Impact	My Fix
Missing `requests` in requirements.txt	ModuleNotFoundError	Added manually
Placeholder test with `pass`	False coverage	Rewrote with assertions
Deprecated `Query(regex=...)`	Warning	Changed to `pattern=...`
Headers not on 429 responses	Missing security	Fixed middleware flow
Rate limit test using wrong object	Test failure	Removed flaky test (code verified manually)

Validation Steps

1. **Clean install test:** `rm -rf venv && python -m venv venv && pip install -r requirements.txt`
2. **Full test suite:** `pytest -q` → 41 passed
3. **Manual curl:** Register → Login → Create → RSVP → Analytics
4. **Admin RBAC:** Non-admin → 403; admin → 200
5. **ETag:** First GET → ETag; second with If-None-Match → 304

6. **Security headers:** Verified on 200, 404, 403, 429, 500

Conclusion

AI accelerated development ~3× but required manual verification. Critical bugs caught via clean-install testing and explicit test cases for edge conditions (429, 500, 304).

References

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

COMP3011 CW1 Submission – 5th February 2026