GenAI Conversation Export Logs

Module: COMP3011 – Web Services and Web Data
Student: Nathaniel Sebastian (sc232ns)
Tools Used: Google Gemini (Antigravity), ChatGPT (GPT-4)

Session 1: Architecture Planning

Date: 3 February 2026
Tool: Google Gemini

Prompt:

> I need to build a REST API for event management as coursework. What's the best architecture approach for FastAPI + SQLAlchemy?

AI Response (Summary):

Suggested a layered architecture: - Routes layer for HTTP handling - CRUD layer for business logic - Models for database entities - Schemas for Pydantic validation

Recommended separating concerns so routes are thin and logic is testable in CRUD.

My Decision:

Adopted this approach. Created `app/api/routes.py`, `app/crud.py`, `app/models.py`, `app/schemas.py`.

Session 2: RSVP Data Model Design

Date: 3 February 2026
Tool: Google Gemini

Prompt:

> Should I store RSVPs as a list embedded in the Event model, or as a separate table?

AI Response (Summary):

Explained trade-offs: - Embedded: Simpler queries, no JOINs - Separate table: Enables uniqueness constraints, timestamps, independent queries

Recommended separate table for relational integrity.

My Decision:

Chose separate `rsvps` table with UNIQUE(event_id, attendee_id) constraint. This enabled proper cascade deletes and RSVP history tracking.

Session 3: Authentication Choice

Date: 3 February 2026
Tool: Google Gemini

Prompt:

> JWT vs server-side sessions for a REST API authentication?

AI Response (Summary):

- JWT: Stateless, simpler for pure APIs, no session storage needed
- Sessions: Enable immediate revocation, require Redis/DB storage

For coursework scope, JWT simplicity outweighs revocation limitations.

My Decision:

Implemented JWT with 30-minute expiry. Documented revocation as a known limitation.

Session 4: SQLite Migration Issue

Date: 4 February 2026
Tool: Google Gemini

Prompt:

> Alembic migration fails with “Cannot add a NOT NULL column with default value NULL” on SQLite

AI Response:

SQLite doesn't support ALTER TABLE for certain operations. Use Alembic's `batch_alter_table` context manager which recreates the table.

Code Fix:

```
with op.batch_alter_table('events') as batch_op:
    batch_op.add_column(sa.Column('source_id', sa.Integer(),
        nullable=True))
```

Outcome:

Migration succeeded. Applied this pattern to all subsequent migrations.

Session 5: Test Fixture Design

Date: 4 February 2026
Tool: Google Gemini

Prompt:

> Tests are interfering with each other. How do I isolate database state?

AI Response:

Use in-memory SQLite with `StaticPool` and create/drop tables per test function:

```
@pytest.fixture(scope="function")
def db(engine):
    Base.metadata.create_all(bind=engine)
    yield session
    Base.metadata.drop_all(bind=engine)
```

My Enhancement:

Added transaction rollback for faster cleanup. All 25 tests now run in <1 second.

Session 6: Analytics Implementation

Date: 4 February 2026
Tool: Google Gemini

Prompt:

> How should I implement a trending score for events?

AI Response:

Suggested formula weighting recent activity higher:

```
score = (recent_rsvps × 1.5) + (total_rsvps × 0.5)
```

Explained that 1.5× multiplier rewards events gaining momentum.

My Implementation:

Adopted the formula. Added `window_days` parameter to control "recent" definition.

Session 7: Timezone Bug Fix

Date: 4 February 2026
Tool: Google Gemini

Prompt:

> Error in analytics.py: NameError: name 'timezone' is not defined

AI Response:

Missing import. Add `timezone` to the datetime import:

```
from datetime import datetime, timedelta, timezone
```

Outcome:

Fixed. Tests continue to pass.

Session 8: PDF Generation

Date: 4 February 2026
Tool: Google Gemini

Problem:

Coursework requires PDF documentation, but pandoc/weasyprint have missing dependencies.

Solution Found:

Use Puppeteer (Node.js) to convert HTML to PDF:

```
const page = await browser.newPage();
await page.goto(htmlPath);
await page.pdf({path: outputPath, format: 'A4'});
```

Outcome:

Generated `docs/API_DOCUMENTATION.pdf` (542KB) and `TECHNICAL_REPORT.pdf` (310KB).

Session 9: Harsh Reality Check (Code Quality Upgrade)

Date: 5 February 2026
Tool: Gemini 3 Pro

Problem:

Project was "working" but had "silly" errors that would kill marks (missing `requests` in requirements.txt, unfinished test ending in `pass`).

AI Action:

- Identified that `scripts/import_dataset.py` imported `requests` but it wasn't in `requirements.txt`.
- Found a test `test_recommendations_personalized` that had placeholder logic.
- Found duplicate field definitions in `models.py`.

Correction (Human Led):

- Instructed AI to *not* just "fix it" but to first **verify the failure** (Reproduction).
- Forced AI to implement a *logic-based* test for recommendations (create history -> verify output) rather than a mock.
- Manually verified the final `pytest` run (31 passing) on a clean environment.

Summary and Critical Reflection

What AI Did Well

- **Boilerplate:** Rapidly scaffolded FastAPI CRUD routes and Pydantic schemas.
- **Testing:** Generated comprehensive test cases for standard HTTP flows (404s, 422s).
- **Explanation:** Clearly explained trade-offs between Embedded vs Relational data for RSVPs.

Where AI Failed (and I Fixed It)

1. **Dependency Management:** AI wrote code using `requests` but failed to add it to the package manifest. I manually caught this during a clean install check.
2. **Incomplete Logic:** AI generated a test skeleton (`test_recommendations_personalized`) ending in `pass`, effectively lying about test coverage. I forced a rewrite with real assertions.
3. **Circular Imports:** AI created models that imported schemas that imported models. I refactored to use string forward references to break the cycle.
4. **Deprecated API Usage:** AI suggested `Query(regex=...)` which is deprecated in FastAPI. I updated it to `Query(pattern=...)` to silence warnings.

Conclusion

AI is a powerful accelerator for standard patterns but a "sloppy" engineer for integration and lifecycle management. It requires a human supervisor to enforce rigour, dependency correctness, and actual logical validation.

Exported for COMP3011 CW1 submission