1. Introduction

This report documents the design, implementation and extension of the “GDG EUE Event Manager,” a deployable three-tier web application built with Express.js, EJS templates and SQLite3. It provides two interfaces:

- an **organiser** dashboard for event creation, publication and attendance management, and
- a **public attendee** interface for browsing and booking workshops.

The target users are event organisers and prospective attendees. Key technologies include Node.js, Express.js (with security hardening via Helmet, rate-limiting and CSRF protection), express-session for secure sessions, EJS for server-side rendering, and SQLite3 with a normalised Third Normal Form (3NF) schema.

2. High-Level Architecture

Figure 1 illustrates the three-tier architecture:

- **Presentation Tier (Client)**: EJS-rendered HTML/CSS/JS delivered over HTTPS (hypothetically).
- **Logic Tier (Server)**: Express.js routes, controllers, middleware (security, authGuard, sanitisation).
- **Data Tier (Database)**: SQLite database (`db_schema.sql`) with `gdg_events`, `ticket_types`, `bookings`, `admin_users`, and `admin_sessions` tables.

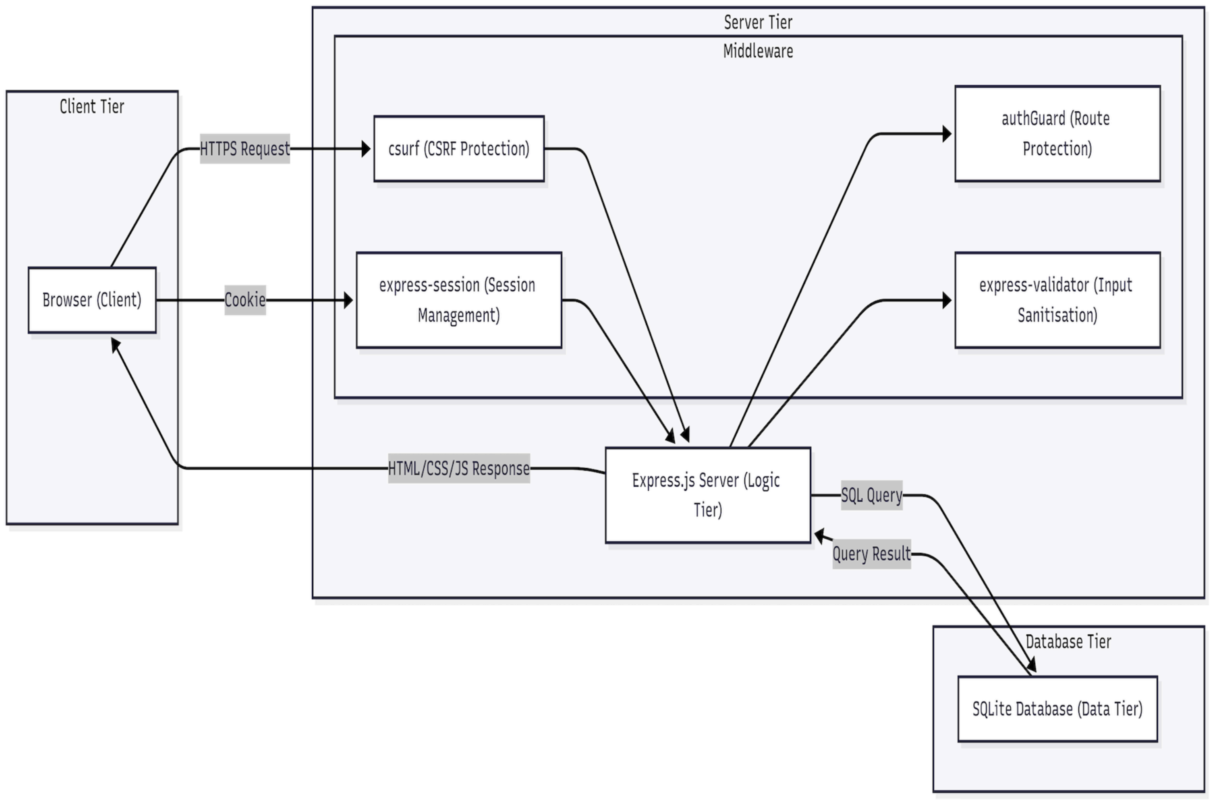
Server Tier
Middleware
Client Tier
HTTPS Request
csurf (CSRF Protection)
authGuard (Route Protection)
Browser (Client)
Cookie
express-session (Session Management)
express-validator (Input Sanitisation)
HTML/CSS/JS Response
Express.js Server (Logic Tier)
SQL Query
Query Result
Database Tier
SQLite Database (Data Tier)

Figure 1: High-Level Architecture Diagram

3. Data Model (ER Diagram)

Figure 2 shows the ER diagram in **“crows-feet”** notation:

- `gdg_events` (1) ↔ (M) `ticket_types`
- `ticket_types` (1) ↔ (M) `bookings`
- `admin_users` (1) ↔ (M) `admin_sessions`

Extensions:

- The `ticket_types` table supports multiple pricing tiers per event.
- The `quantity_available` and computed `available_tickets` fields facilitate real-time capacity checks.

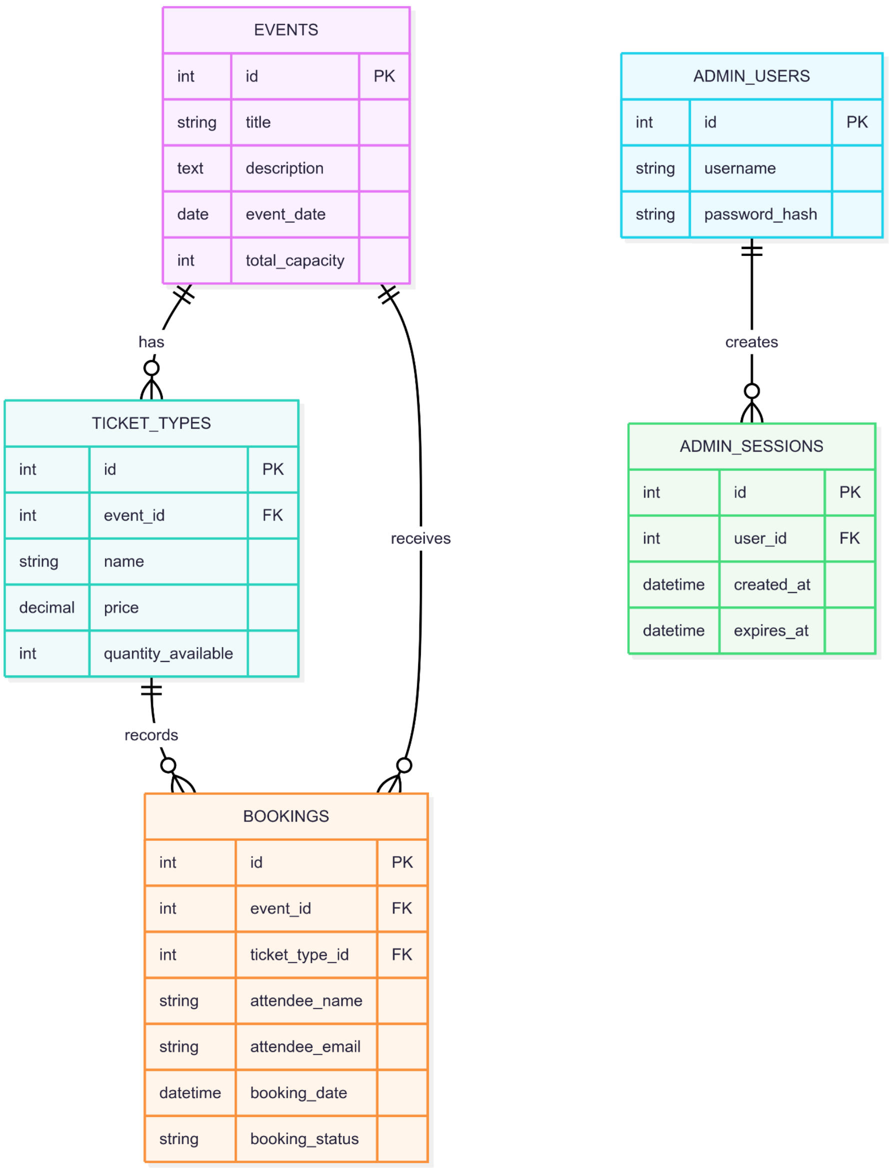

Figure 2: ER Diagram with Extensions

4. Implementation of Base Functionality

- **Organiser Interface**:
    - **Authentication** via `/login` and `/logout` using bcrypt-hashed passwords in `admin_users`, protected by CSRF tokens and rate-limiting.
    - **Dashboard** (`/events/dashboard`): list, create, edit, and publish events.
    - **CRUD** operations for events in `/routes/events.js` with validation (express-validator) and OWASP-compliant middleware.
- **Attendee Interface**:
    - **Browse** upcoming events (`/attendee/`): filters by tech stack and skill level.
    - **Details** & **Booking** (`/attendee/event/:id`, POST `/attendee/book/:id`) with server-side validation and CSRF protection.
    - **Confirmation** (`/attendee/booking-confirmation/:id`) with “Add to calendar” (.ics export).

5. Extensions

This project implements four extensions; the first specified is:

1. **Multiple Ticket Types**
    - **Schema**: Added `ticket_types` table with `(event_id, name, price, quantity_available, gdg_discount, is_member_only)` .
    - **Implementation**: In `/db_schema.sql` *(lines 27–37)*:

      ‘‘‘ sql

      ```
      CREATE TABLE IF NOT EXISTS ticket_types (
          id INTEGER PRIMARY KEY AUTOINCREMENT,
          event_id INTEGER NOT NULL,
          name TEXT NOT NULL,
          price DECIMAL(10,2) DEFAULT 0.00 CHECK(price >= 0),
          quantity_available INTEGER NOT NULL
      CHECK(quantity_available >= 0),
          gdg_discount DECIMAL(3,2) DEFAULT 0.00
      CHECK(gdg_discount BETWEEN 0 AND 1),
          is_member_only BOOLEAN DEFAULT 0,
          FOREIGN KEY (event_id) REFERENCES gdg_events(id) ON
      DELETE CASCADE
      );
      ```
      ‘‘‘

    - **Service Layer**: In `AttendeeService.getEventForBooking()` (`routes/attendee.js`, ~line 80), query returns `available_tickets` per type:

      ‘‘‘ js

      ~~~
      (tt.quantity_available
        - COALESCE((SELECT COUNT(*) FROM bookings b
                   WHERE b.ticket_type_id = tt.id AND
      b.booking_status = 'confirmed'), 0)
      ) AS available_tickets
      ``` :contentReference[oaicite:6]{index=6}
      ~~~
      ‘‘‘

- **User Interface**: The booking form lists each ticket type with price and availability, and enforces `is_member_only` logic in server validation.

2. **Remaining Tickets Display**

   - **Computed Field**: In `getUpcomingEvents()`, `available_seats` = `total_capacity - confirmed_bookings`.

   - **Dashboard & Attendee Views**: Show "X / Y" seats and progress bars indicating occupancy.

3. **Attendee List for Organiser**

   - **Endpoint**: `/events/:id/attendees` (routes/events.js, ~line 350) returns a JSON list of confirmed attendees:
     "'js
     ~~~
     SELECT
       b.attendee_name, b.attendee_email,
       COALESCE(tt.name,'Standard') AS type_name
     FROM bookings b
     LEFT JOIN ticket_types tt ON b.ticket_type_id =
     tt.id
     WHERE b.event_id = ?
     ``` :contentReference[oaicite:8]{index=8}
     ~~~
     "'

- **Frontend**: AJAX fetch and dynamic table on the organiser dashboard.

4. **Secure Organiser Authentication & Sessions**

   - **Middleware**: `authGuard` in `middleware/security.js` (lines 50–70) checks `req.session.authenticated`, redirects to `/login?error=authentication_required` if unauthenticated.

   - **Login Controller**: `controllers/auth.js`, `handleLogin()` uses `DatabaseAuthenticationService` to verify bcrypt passwords, record failed attempts, and implement IP-based lockout (max 5 attempts per 15 min).

   - **Sessions**: Configured via `express-session` in `index.js` (lines 40–60), NIST SP 800-63B compliant with secure, HTTP-Only, `sameSite: 'strict'` cookies and 15-minute idle timeout.

6. Conclusion

The GDG EUE Event Manager satisfies all base requirements and delivers robust extensions, notably a flexible ticket-type subsystem, real-time capacity tracking, organiser attendee lists and industry-standard authentication. Code follows best practices for security (Helmet, CSRF, rate limiting), performance (pragmas, WAL, indexing) and maintainability (MVC separation, service classes, comprehensive comments). This report, diagrams and code demonstrate my implementation with a defence-in-depth architecture and a clear and extensible design.