Initial Component Diagram:

[Browser / Client]

[Express App Router Layer]

[Producer Controller / Routes]
- GET /producer (dashboard)
- POST /producer/add-rotation
- POST /producer/remove-rotation
- POST /producer/add-booking
- POST /producer/remove-booking
- GET /producer/producer-data

[Models / Mongoose Schemas]
- Producer (main profile: shows, rotation, playlists, bookings, ROS)
- Track (title, artist, mm:ss, bpm)
- Show (title, slots, genre)
- Genre (format/style)

[MongoDB Database]

The browser sends HTTP requests to the Express router responsible for /producer.

The producer controller loads or creates a Producer profile using the session ID and then retrieves relational data using Mongoose's .populate().

Routes render EJS templates or return JSON for AJAX interactions (rotation/bookings).

Mongoose models store all application data in MongoDB, including tracks, shows, rotation items, playlists, and bookings.

Static CSS and JavaScript are served from public/ and referenced by the EJS dashboard.

Diagram created: *2025-11-20*

Initial Sequence Diagram:

Client -> Express Router: GET /producer
Express Router -> getProducerProfile: load or create profile by sessionId
getProducerProfile -> MongoDB: populate shows, rotation, playlists, bookings
MongoDB --> getProducerProfile: producer document
Router -> EJS View: render producer.ejs with {producer, stats, shows, genres}
EJS View -> Client: HTML + CSS + front-end JS

Client -> Express Router: POST /producer/add-rotation (JSON payload)
Router -> Body Parser: parse JSON
Router -> getProducerProfile: attach producer to req
Router -> Track Model: create + save Track document
Track Model -> MongoDB: insert track
Router -> Producer Model: addRotationItem() + save()
Producer Model -> MongoDB: update document
MongoDB -> Router: confirmation
Router -> Client: JSON

The producer dashboard loads by retrieving the producer's session-based profile, populating related models (genre, shows, rotation tracks, playlists).

Rendering uses EJS server-side templates.

When adding a rotation item, the front-end JavaScript submits a JSON payload; the server creates a new Track, appends a rotation entry to the producer profile, saves, and returns updated stats and rotation in JSON.

Diagram created: ***2025-11-25***

Updated Component Diagram

[Browser / Client]
  - Producer Dashboard (EJS)
  - Front-End JS (fetch for rotation, playlists, booking

[Express App]
  - Middleware: sessions, static files, JSON parser
  - Router: /producer

[Producer Route Handlers]
  - getProducerProfile (session-based)
  - Rotation CRUD
  - Booking CRUD
  - Playlist display
  - Run-of-show features

[Models / DB Layer]
  - Producer (with stats virtual)
  - Track (auto-created from rotation form)
  - Show (populated schedule)
  - Genre

[MongoDB]

Diagram created: ***2025-11-28***

Updated Sequence Diagram:
Client -> Producer JS: submit #track-form
Producer JS -> Router: POST /producer/add-rotation {title, artist, mmss, bin, notes, explicit}
Router -> getProducerProfile: load profile
getProducerProfile -> MongoDB: findOne / create profile
MongoDB --> Router: producer document
Router -> Track Model: save new Track
MongoDB --> Router: track saved
Router -> Producer Model: addRotationItem() + save()
Router -> Producer Model: virtual stats recalculated

Producer Model -> MongoDB: update rotation array
Router --> Client: JSON {rotation, stats}
Client -> Producer JS: update UI tables + update stat cards
UI --> User: refreshed rotation + new totals

Diagram created: ***2025-11-28***