1. Design (30 points)

Initial Component Diagram (10/10 points)

I created a complete component diagram before writing any code, showing how the browser, Express router, controllers, EJS views, and MongoDB all interact in an MVC-style structure.
I also included a short explanation and documented the date the diagram was produced.

Initial Sequence Diagram (10/10 points)

Before building the application, I made a sequence diagram that illustrated the module flow—for example, how a producer loads their dashboard and how a rotation item is added.
I explained the flow in clear language and timestamped the diagram.

Updated Diagrams (10/10 points)

After finishing development, I updated both diagrams so they matched the final delivered system.
The updates included the addition of the stats virtual, the corrected route paths, and AJAX-based UI refreshing.

2. EJS Implementation (50 points)

EJS Syntax (20 points)

Throughout the project, I demonstrated consistent and correct use of EJS syntax.
This includes loops, conditions, partials, and dynamic attributes.
For example, rotation items and bookings are rendered using EJS forEach, and producer statistics are displayed using <%= %> interpolation.

Data Passing (10 points)

I successfully passed data from the server into the EJS templates, including:

- producer profile information
- dynamic stats
- shows, genres, rotation items, and playlists

All values rendered on the page come directly from the server and MongoDB.

EJS Views (10 points)

I created fully dynamic EJS views for the radio station application and used layout partials (header and footer) to keep the UI consistent.
Pages are clean, organized, and use the shared components correctly.

Directory Structure (10 points)

My project uses a proper folder structure:
/routes for route files,
/models for database schemas,
/views for EJS templates, and
/public for static assets (CSS and JS).
This follows standard Express best practices.

3. Functionality (20 points)

All core features work as intended:

- The Producer dashboard loads correctly.
- Buttons and forms perform their functions:
    - rotation adding/removing
    - guest/segment booking
    - playlist creation
- Dynamic stats update correctly after CRUD operations.
- Navigation links lead to the correct pages.

Note: I initially struggled with routing especially the /producer/add-rotation path and getting rotation items to save and display correctly.

4. Database Implementation (30 points)

Database Setup (10 points)

I successfully set up MongoDB and connected it to the Node.js application using Mongoose. The connection configuration loads when the server starts, and the database works consistently across sessions.

Data Model Schema Design (10 points)

I designed a set of schemas that accurately model the application's needs:

- Producer (main profile)
- Track
- Show

- Genre

These schemas support rotation items, playlists, run-of-show, booking segments, and more.

Frontend and Backend Data Sync (10 points)

I migrated data into MongoDB so that nearly everything shown in the UI comes from the database.

5. Session Management and Continuity (20 points)

Data Persistence and Restoration (10 points)

Producer data is tied to the session ID, which means a user can reload the page and still see all of their current rotation items, playlists, bookings, and run-of-show information.
This demonstrates persistent session-based state (except for in-progress form input, which resets normally).

Example scenario:
If a producer adds several rotation tracks and refreshes the page, the added rotation items remain because they are saved in MongoDB and linked to the user's session.

Logout and Session Clearing (10 points)

The logout logic successfully clears the session while keeping the user’s data intact in the database.
When logging back in (or starting a new session), the system loads the stored producer profile correctly.