PHP Databases & API Systems - **TECHNICAL DOCUMENT**

Tool Used: phpMyAdmin with MySQL (via MAMP local environment)
Total Tables: 29
Core Entities: user, album, artist, track, playlist, subscription, badge, image, comment, product, etc.

Key Relationship Highlights:

- album.artistID → artist.artistID
- track.albumID → album.albumID
- playlisttrack.playlistID → playlist.playlistID
- comment.trackID → track.trackID
- user.imageID → image.imageID

Join Tables:

- playlisttrack handles the many-to-many relationship between playlists and tracks.
- albumbadge links users to specific albums for badge collection.
- producttype connects product-specific types like CDs, vinyls, and wearables.

Field Data Types:

- INT: For IDs (auto-incremented)
- VARCHAR: Names, emails, titles
- TEXT: Descriptions/comments
- DATETIME: Timestamps (e.g. album release, user creation)
- BOOLEAN: Admin privileges, active states

Constraints:

- Primary Keys are set on all ID fields.
- Foreign Keys enforce data consistency and prevent orphaned records.

2. Virtual Server Setup (Local Server Setup)

Environment: MAMP (Mac) / XAMPP (Windows)
Purpose: To simulate a real live web server with Apache + MySQL for local development.

Steps:

1. **Installed MAMP** to get Apache and MySQL services locally.
2. **Moved the `Php-Databases` folder** to the MAMP `htdocs` directory.

3. Ran MySQL through MAMP and accessed phpMyAdmin via localhost:8888/phpmyadmin.
4. **Created `mytunesdb` database**, and imported the provided `.sql` file to generate all tables.
5. Accessed the app via browser using localhost:8888/Php-Databases/index.php.

Outcome: A fully functional environment to run and test the application without an internet connection.

3. Techniques Used to Build the Dynamic Web Application

Languages Used:

- **PHP**: Handles backend logic and dynamic rendering of content (e.g. login logic, form submissions, conditionals).
- **MySQL**: Manages all data in a relational format (albums, tracks, users, comments, etc.).
- **HTML/CSS**: UI structure with styling using a Neumorphism + Metro-inspired design.
- **JavaScript (minimal)**: Used for confirmation prompts (e.g. delete confirmation).
- **Spotify API (via PHP cURL)**: Pulls real artist metadata such as images, genres, and followers.

Examples:

- Dynamic artist search using `%LIKE%` in SQL.
- Tracklist on album page rendered using a `while` loop from a query.
- Admin-only panel visibility controlled by checking `$_SESSION['userID']` and `admin` flag.

4. Techniques Used to Manipulate Data Through the Web Application

INSERT:

- Registration form: creates new users with prepared statements.
- Upload form: adds uploaded images to the image table, linked to users.

UPDATE:

- Profile picture change updates the `imageID` of a user.
- Subscription changes update the `subscriptionID` of a user.
- Admins can toggle or remove data using update/delete commands.

DELETE:

- Admin panel allows deleting any user.
- Old profile images are removed using PHP's `unlink()` function if replaced.

SELECT:

- Homepage loads all recent tracks, albums, and artists via JOIN queries.
- Search queries filter track.title or album.title using LIKE.

Test Case Description	**Steps to Execute**	**Expected Result**	**Actual Result**	**Pass/Fail**
Test user login with email address	Open login page, Enter a valid email address and a correct password, click "Login"	User is directed to the homepage if the password and email address match	User is directed to the homepage if the password and email address match	pass
Test user login with username	Open login page, Enter a valid email username and a correct password, click "Login"	User is directed to the homepage if the password and username match	User is directed to the homepage if the password and username match	pass
User Registration	Open login page, click create account and create a new user with an email address and username	User has a new account created	User has a new account created	pass
Test music search function	Enter search term. Press the search icon.	Relevant search results are shown	Relevant search results are shown	pass
Test logout functionality	Click profile icon2. Click "Logout"	User is logged out and returned to login screen	User is logged out and returned to login screen	pass
Test profile picture upload	Go to profile, Upload a valid picture, Press upload.	Profile picture updates correctly	Profile picture updates correctly	pass

Test admin account deletion	User logs in as admin, clicks on admin panel and deletes any user	User gets deleted	User gets deleted	pass
Subscribe to premium plan	User Logs in, visits subscriptions and clicks purchase	Payment screen loads and plan updated after confirmation	Payment screen loads and plan updated after confirmation	pass
Cancel subscription	Go to subscriptions and change back to a free plan.	Plan is reset to to free	Plan is reset to to free	pass
Album page with image	Visit album page that has an image	Album image appears	Loaded via LEFT JOIN entityimages	pass
View badges	Visit profile page	Badge earned from pressing album is listed	Matches DB values from `albumbadge`	pass