Chinook PlayList.

# ERD for UI/UX process

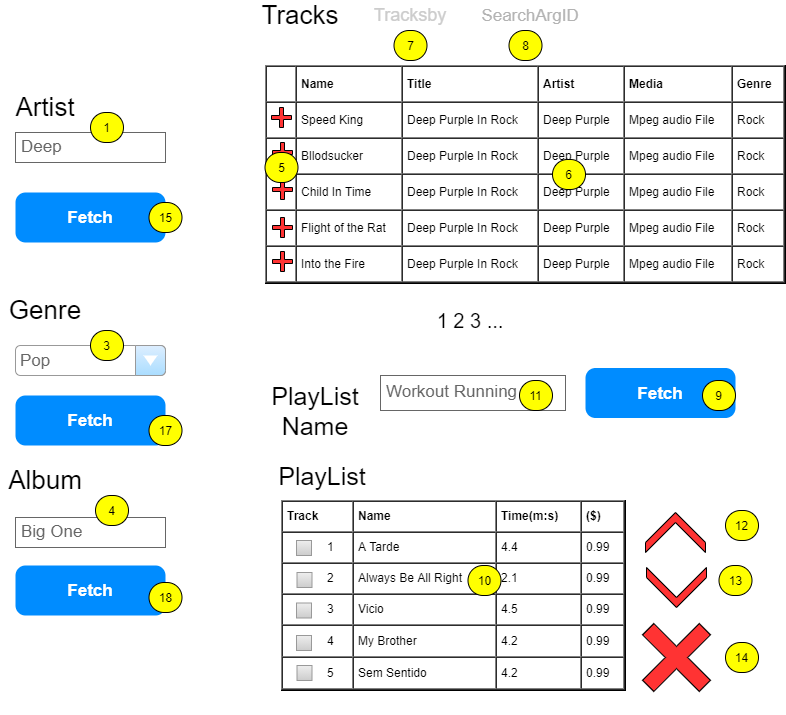


The playlists will contain purchased tracks and non-purchased tracks. Only tracks that have been purchased would play (this is functionality of another subsystem).

The manage playlist OLTP sample will not include generation of Invoices for playlists. This would be another button (Purchase) on the Possible UI/UX Interface.

When creating your prototype, you can expect some changes as you create your system and review it with the client. A prototype is a starting point.

# Possible UX Interface



|  |  |  |
| --- | --- | --- |
| **Controls** | | **Events** |
| GenreDDL (**3**) | | None / wired via ObjectDataSource |
| **BLL Class(es) and Method (s)** | | |
| GenreController | List<SelectionList> List\_GenreNames()  Retrieve a list of genre names and genre id for DDL | |
| **SQL Table(s): (C,R,U,D)** | | **ViewModels** |
| Genres (R) | | SelectionList |

|  |  |  |
| --- | --- | --- |
| **Controls** | | **Events** |
| Fetch (**15**), ArtistArg (1)  Tracks (**6**), TracksBy (**7**), SearchArgID (**8**) | | OnClick / wired via ObjectDataSource   * Move selection value to TracksBy which identifies text picked, (hidden field) * Validate artists partial name is present * ArtistArg id to SearchArgID; (hidden field) * Bind data to Tracks |
| **BLL Class(es) and Method (s)** | | |
| TracksController | List<TrackList> List\_TracksForPlaylistSelection(string tracksby, string arg)  Retrieve a list of tracks for supplied selection type and argument strings. This is a partial string lookup for data.  Return trackid, song, title, artist, genre, composer, timelength, size, unitprice. | |
| **SQL Table(s): (C,R,U,D)** | | **ViewModels** |
| Tracks (R) | | TrackList |

|  |  |  |
| --- | --- | --- |
| **Controls** | | **Events** |
| Fetch (**17**), GenreDLL(3)  Tracks (**6**), TracksBy (**7**), SearchArgID (**8**) | | OnClick / wired via ObjectDataSource   * Move selection value to TracksBy which identifies DDL picked, (hidden field) * DDL id to SearchArgID; (hidden field) * Bind data to Tracks |
| **BLL Class(es) and Method (s)** | | |
| TracksController | List<TrackList> List\_TracksForPlaylistSelection(string tracksby, string arg)  Retrieve a list of tracks for supplied selection type and argument strings. Return trackid, song, title, artist, genre, composer, timelength, size, unitprice. | |
| **SQL Table(s): (C,R,U,D)** | | **ViewModels** |
| Tracks (R) | | TrackList |

|  |  |  |
| --- | --- | --- |
| **Controls** | | **Events** |
| Fetch (**,18**), AlbumArg(4)  Tracks (**6**), TracksBy (**7**), SearchArgID (**8**) | | OnClick / wired via ObjectDataSource   * Move selection value to TracksBy which identifies DDL picked, (hidden field) * Validate an album name is present * AlbumArg id to SearchArgID; (hidden field) * Bind data to Tracks |
| **BLL Class(es) and Method (s)** | | |
| TracksController | List<TrackList> List\_TracksForPlaylistSelection(string tracksby, string arg)  Retrieve a list of tracks for supplied selection type and argument strings. Return trackid, song, title, artist, genre, composer, timelength, size, unitprice. | |
| **SQL Table(s): (C,R,U,D)** | | **ViewModel** |
| Tracks (R) | | TrackList |

|  |  |  |
| --- | --- | --- |
| **Controls** | | **Events** |
| Fetch (**9**), Playlist (**10**), PlayListName (**11**) | | Fetch OnClick   * Validate data present * call BLL * Bind returned data to PlayList. |
| **BLL Class(es) and Method (s)** | | |
| PlaylistTracksController | List< UserPlaylistTrack > List\_TracksForPlaylist(string playlistname, string username)  Retrieve a list of tracks for supplied playlist name and username. Return trackid, title, timelength, unitprice, tracknumber. Display in PlayList. | |
| **SQL Table(s): (C,R,U,D)** | | **ViewModel** |
| PlayList (R), PlaylistTracks(R), Tracks(R) | | UserPlaylistTrack |

|  |  |  |
| --- | --- | --- |
| **Controls** | | **Events** |
| Tracks: Plus Sign (**5**), PlayListName (**11**),  PlayList (10) | | OnCommand   * Collect track information (trackid) and playlist name * Validate presence * send to BLL for adding track to playlist * refresh playlist. |
| **BLL Class(es) and Method (s)** | | |
| PlaylistTracksController | void Add\_TrackToPlaylist(string playlistname, string username, int trackid, string song)  TRX (all errors to be returned)   * create playlist if needed, set tracknumber to 1, * playlist exists, find current highest tracknumber to increment by 1 * **verify desired track not already on list, if so reject** * add new track to playlist tracks | |
| **SQL Table(s): (C,R,U,D)** | | **ViewModel** |
| PlayList(C opt), PlaylistTracks(C) | |  |

|  |  |  |
| --- | --- | --- |
| **Controls** | | **Events** |
| Up/Down (**12,13**), PlayListName (**11**), PlayList (**10**) | | OnCLick   * Collect track id, track number, playlist name * Validate   + only 1 track can move   + data presence * send to BLL for moving track in playlist; * refresh playlist. |
| **BLL Class(es) and Method (s)** | | |
| PlaylistTracksController | void Move\_TrackInPlaylist(string playlistname, string username, int trackid, int tracknumber, string direction)  TRX (all errors to be returned)   * determine if move necessary * if already a top and moving up, reject move * if already a bottom and moving down, reject move * move up, swap with track above by altering tracknumbers * move down, swap with track below by altering tracknumbers. | |
| **SQL Table(s): (C,R,U,D)** | | **ViewModel** |
| Playlist(R), PlaylistTracks(U) | |  |

|  |  |  |
| --- | --- | --- |
| **Controls** | | **Events** |
| Delete (**14**), PlayListName (**11**), PlayList (**10**) | | OnCLick   * Collect track id(s), ~~track number~~ and playlist name * Validate   + at least one track selected   + data presence * send to BLL for removing track from playlist; * refresh playlist. |
| **BLL Class(es) and Method (s)** | | |
| PlaylistTracksController | void Remove\_TrackFromPlaylist(string playlistname, string username~~, int trackid,~~ List<int> trackid)  TRX (all errors to be returned)   * find tracks to keep, * remove tracks to delete, * shift remaining tracknumbers up by renumbering tracks. | |
| **SQL Table(s): (C,R,U,D)** | | **ViewModel** |
| Playlist(R), PlaylistTracks(U/D) | |  |



|  |  |  |
| --- | --- | --- |
| **SelectionList**  Common class used by several entities to supply a set of fields for a dropdownlist | | |
| DisplayField | string |  |
| ValueField | int |  |

|  |  |  |
| --- | --- | --- |
| **UserPlaylistTrack** | | |
| trackid | int | PlaylistTracks |
| title | string | Tracks |
| timelength | int | Tracks |
| unitprice | decimal | Tracks |
| tracknumber | int | PlaylistTracks |

|  |  |  |
| --- | --- | --- |
| **TrackList** | | |
| trackid | Int | Tracks |
| song | String | Tracks |
| title | string | Albums |
| artist | String | Artists |
| Genre | String | Genres |
| Composer | Strig | Tracks |
| timelength | Int | Tracks |
| size | Int | Tracks |
| unitprice | decimal | Tracks |