Developer Overview – SSO Wishlist v2

📱 App Overview

A Glide-based web app built to allow Star Stable Online (SSO) players to browse, search, and manage their wishlist and owned lists of Generation 3 horses. This includes robust custom features for filtering, sorting, searching, and viewing the horse collection.

🔗 Current Structure

Tables:

- **horses** – Main data table for all Gen 3 horses.
- **users** – Stores user-specific settings like filter preferences, sorting, and view options.
- **sort_options** – Contains the sort criteria values used in choice components.

Columns in `users`:

Column Name	Type	Purpose
`user_id`	Row ID	Unique identifier for each user.
`user_sort_preference`	Text	Stores the selected sort type.
`search_haystack`	Text	User's search input.
`filter_breeds`	Text	Comma-separated list of selected breeds.
`filter_release_min`	Date	Lower bound of release date filter.
`filter_release_max`	Date	Upper bound of release date filter.
`filter_price_range`	Text	User-selected price range label.
`filter_locations`	Text	Comma-separated list of selected locations.
`filter_magic`	Boolean	Include/exclude magic horses.
`filter_availability`	Text	Comma-separated list of selected availability types.
`view_card_zoom`	Number	Sets card zoom scale.
`view_page_size`	Number	Limits number of horses shown per page.
`filtered_horse_count`	Number	Count of horses currently shown after filters/search.

Columns in `horses`:

Column Name	Type	Description
`id`	Row ID	Unique identifier for each horse.
`breed`	Text	Breed name.
`coat_name`	Text	Coat variation name.
`image_url`	URL	Image link.
`magic`	Boolean	Whether the horse is a magic horse.
`availability`	Text	Options include limited, permanent, rotation, etc.
`price`	Number	Cost in-game.
`release_date`	Date	When the coat was added to the game.
`location`	Text	Where to find the horse in-game.
`price_band`	Template	Logical label bucket based on price (e.g. low, mid, high).
`rel_user_profile`	Relation	Links each horse to current user row via `user_id`.
`user_sort_preference`	Lookup	Pulls value from related user row.
`sort_value`	Lookup	Synonym for `user_sort_preference`.
`sort_target`	If-Then-Else	Outputs correct column value based on `sort_value`.
`sort_order`	Template	Indicates asc/desc manually for now.
`sort_final`	Template	Merges `sort_target` and `sort_order` for clean sorting logic.
`search_match`	If-Then-Else	True if horse name/breed/coat matches `search_haystack`.
`user_sort_preference_value`	Lookup	Supporting column for sync with user selection modals.

🧰Modal System Setup

Three buttons at the top of the Browse screen (`Filter`, `Sort`, `View`) open full-screen **Form pages**:

1. Browse: Filter (Form Page)

Purpose: Allow users to select filtering preferences.

Components: - Choice (multi-select): `filter_breeds`, `filter_locations`, `filter_availability` - Date Pickers: `filter_release_min`, `filter_release_max` - Chips: `filter_price_range`, `filter_magic`

Apply Button: - Action: "Set Column Values" targeting the user profile row

---

2. Browse: Sort (Form Page)

Purpose: Let users select how horses are sorted.

Component: - Choice chips: From `sort_options`, writes to `user_sort_preference`

Apply Button: - Action: Writes value to `user_sort_preference` in `users`

---

3. Browse: View (Form Page)

Purpose: Customize appearance of horse cards.

Components: - Choice: `view_page_size` – how many horses to show at once - Choice: `view_card_zoom` – determines card scaling

---

🔍Custom Search

Search Bar (Text Entry): - Writes to `search_haystack`

Computed Column in `horses`: - If name, breed, or coat includes `search_haystack`, set `search_match = true`

Filter Applied in Collection Component: - Only include rows where `search_match = true`

📊Collection Filtering & Sorting

Filters: (Set in Collection → Options → Filter) - Apply logic based on user-specific columns: - Breed IN `filter_breeds` - Release Date BETWEEN `filter_release_min` AND `filter_release_max` - Price falls within selected `filter_price_range` - Location IN `filter_locations` - Magic matches `filter_magic` - Availability IN `filter_availability`

Sort: - Sort by `sort_final` column - Direction handled by sort options with A→Z or Z→A setting

---

🕐Horse Count (e.g. “30 Horses Listed”)

Approach: - Create a relation from users → horses (filtered) - Use a **rollup column** to count that relation - Bind that value to the label shown above the collection

---

🔗Status Checklist

- [x] `user_sort_preference` logic connected
- [x] `sort_target` logic restructured for IF logic
- [x] Modal views created for Filter, Sort, View
- [x] Apply button correctly updates user-specific columns
- [x] Custom search logic planned
- [ ] Collection filters still need to be connected to filter user preferences
- [ ] Horse count logic needs implementation
- [ ] Page size and zoom need to be applied
- [ ] Final pass of columns/tables for any untracked computed values

---

This document reflects the full backend and UI logic plan, ready for onboarding a new developer or resuming from scratch.

Let me know if you’d like this exported as a PDF or attached to your case study assets.