PROCESS LOG - EventFinderJava Android Application
==

This document logs important issues fixed during development, including:
a) Prompt (Clarity) - What the user requested
b) Issues (Specificity) - The specific problem identified
c) Fix (Correctness) - How the issue was resolved
d) Explanation (Depth) - Why the fix works

==
ISSUE #1: Unliked Events Not Immediately Removed from SearchFragment UI
==
a) Prompt (Clarity):
User reported: "WHEN EVER i TRY TO UNLIKE A FAV EVENT FROM SEARCH RES PAGE WHICH HAS NO SEARCH IT OR IF THAT EVENT IS NOT PART OF SEARCH RES THEN IT SHOULD BE REMOVED IMMEDIATELY FROM THE SEARCH RES PAGE IF IT IS PART OF SEARCH RES THEN IT SHOULD STAY"

b) Issues (Specificity):
- When an event was unliked on the EventDetails page, it was not immediately removed from the likedEvents list in SearchFragment
- The UI did not update to reflect the unliked state
- Events that were not part of searchResults should be removed immediately when unliked
- Events that were part of searchResults should remain visible but update their favorite status

c) Fix (Correctness):
- Modified updateEventFavoriteState() method to correctly remove events from likedEvents when unliked
- Updated addToFavorites() and removeFromFavorites() to call updateEventFavoriteState() after successful backend operations
- Added logic to check if an unliked event is part of searchResults before deciding whether to remove it from display
- Enhanced checkAndAddNewFavorites() to fetch all backend favorite IDs and remove events from likedEvents that are no longer in the backend

d) Explanation (Depth):
The fix ensures synchronization between the backend favorite state and the local likedEvents list. When an event is unliked, the method checks if it exists in searchResults. If it does, the event remains visible but its favorite status is updated. If it doesn't exist in searchResults, it's immediately removed from the display. This maintains data consistency and provides immediate visual feedback to the user.

==
ISSUE #2: Keyword Not Persisting When Navigating Back from EventDetails

==
a) Prompt (Clarity):
  User requested: "now without effecting current behaviour when I navigate back from event details page to search page, the keyword should stay popped up"

b) Issues (Specificity):
  - When navigating back from EventDetails page to SearchFragment, the previously entered keyword was lost
  - The editKeyword field was empty, requiring users to re-enter their search term

c) Fix (Correctness):
  - Added a fragment-level variable 'keyword' to store the current search keyword
  - Modified onCreateView() to restore the keyword text in editKeyword if it was previously entered
  - Added logic to trigger fetchKeywordSuggestions() for the restored keyword

d) Explanation (Depth):
  By storing the keyword in a fragment-level variable, it persists across fragment lifecycle events. When the fragment is recreated (onCreateView), the stored keyword is restored to the editKeyword field, maintaining user context and improving UX by not requiring re-entry of search terms.

==
ISSUE #3: Keyboard Not Staying Popped Up on SearchFragment
==
a) Prompt (Clarity):
  User reported: "the keyboard is not staying popped up" and "when i come from home page to search page even if there is fav event or not the keyboard should pop up"

b) Issues (Specificity):
  - Keyboard was hidden when navigating back to SearchFragment from EventDetails
  - Keyboard was not shown when navigating from HomeFragment to SearchFragment
  - Keyboard was being hidden when favorites were updated or displayed

c) Fix (Correctness):
  - Modified onResume() to always attempt to show the keyboard when navigating to the search page
  - Updated showNoEventsState() to no longer hide the keyboard unconditionally
  - Modified updateFavoritesInSearchResults() to ensure keyboard stays visible after updating favorites
  - Added logic to request focus and show keyboard using InputMethodManager with proper timing delays

d) Explanation (Depth):

The keyboard visibility is controlled by InputMethodManager, which requires the view to have focus and a valid window token. By requesting focus on editKeyword and using postDelayed() to ensure the view is fully laid out, we can reliably show the keyboard. The fix ensures the keyboard appears whenever the user navigates to the search page, improving usability.

==
ISSUE #4: Keyboard and Keyword Suggestions Popping Up After Search
==
a) Prompt (Clarity):
User reported that after performing a search and results are displayed, the keyboard and keyword suggestion dropdown should be hidden.

b) Issues (Specificity):
- After search results were displayed, the keyboard remained visible
- Keyword suggestion dropdown remained visible after search completion
- This cluttered the UI and made it difficult to view search results

c) Fix (Correctness):
- Introduced an overloaded showResultsState(boolean hideKeyboardAfterSearch) method
- Modified parseSearchResults() to call showResultsState(true) to explicitly hide keyboard and dismiss dropdown after search
- The default showResultsState() now calls showResultsState(false) to keep keyboard visible for other scenarios

d) Explanation (Depth):
By adding a boolean parameter to control keyboard visibility, we can differentiate between showing results after a search (where keyboard should be hidden) and showing results after other operations (where keyboard should remain visible). This provides fine-grained control over UI state based on user actions.

==
ISSUE #5: Keyboard and Keyword Suggestions Popping Up on Category Tab Click
==
a) Prompt (Clarity):
User requested: "when I click on category again the keyboard and keyword suggest are getting popped make sure its not happening for category"

b) Issues (Specificity):
- When clicking on a category tab (even if it's the same category), the keyboard and keyword suggestion dropdown appeared
- This was disruptive to the user experience when browsing category-filtered results

c) Fix (Correctness):

   - Modified setupCategoryTabs() to explicitly dismiss the keyword dropdown when a category tab is selected
   - Added code to hide the keyboard and clear focus from editKeyword field when category is selected
   - Introduced showResultsStateWithoutKeyboard() method to update UI without affecting keyboard visibility
   - Added isCategorySwitchInProgress flag to prevent keyboard from showing during category switches

d) Explanation (Depth):
   Category tab selection is a navigation action, not a text input action. By explicitly dismissing the dropdown and hiding the keyboard when a category is selected, we prevent the keyboard from interfering with result viewing. The flag ensures that focus changes during category switches don't trigger keyboard display.

==
ISSUE #6: Image and Date/Time Missing for Favorite Events
==
a) Prompt (Clarity):
   User reported that favorite events on HomeFragment were missing images and date/time information after navigation.

b) Issues (Specificity):
   - Initial backend response for favorites sometimes lacked imageUrl, dateRaw, or timeRaw
   - FavoriteEvent objects were created with incomplete data
   - UI displayed favorite events without images or date/time information

c) Fix (Correctness):
   - Modified loadFavorites() to check if imageUrl, dateRaw, or timeRaw are empty for a FavoriteEvent
   - Enhanced fetchEventDetailsForFavorite() to fetch both image URL and date/time from /api/eventdetails endpoint
   - Updated FavoriteEvent objects in the favorites list directly when missing data is fetched
   - Called adapter.notifyItemChanged() on the UI thread to refresh specific items

d) Explanation (Depth):
   The backend favorites API may return incomplete data for performance reasons. By checking for missing fields and fetching them from the event details API, we ensure all favorite events display complete information. Updating the object directly in the list and notifying the adapter for that specific item provides efficient UI updates without full list refreshes.

==
ISSUE #7: Date Format Incorrect (Dot Instead of Comma)

==
a) Prompt (Clarity):
User requested: "after year in date there should be comma not dot"

b) Issues (Specificity):
- Date and time were displayed with a bullet point separator (•) instead of a comma
- Format was "MMM d, yyyy • h:mm AM/PM" instead of "MMM d, yyyy, h:mm AM/PM"

c) Fix (Correctness):
- Modified SearchResultsAdapter.java to change the separator between date and time from bullet point to comma and space
- Updated the date/time formatting logic to use ", " instead of " • "

d) Explanation (Depth):
The date format should match standard conventions. Using a comma and space between date and time provides better readability and follows common date formatting standards. This is a simple string formatting change that improves the visual presentation of event information.

==
ISSUE #8: Unknown and Miscellaneous Genres Not Displayed
==
a) Prompt (Clarity):
User reported: "why is the genre and date not getting displayed for search result" and "the unknown and misc genre also should be displayed"

b) Issues (Specificity):
- Category (genre) label was not displayed for some events
- "Unknown" and "Miscellaneous" genres were being filtered out and not displayed
- The code had implicit filtering that hid certain segment values

c) Fix (Correctness):
- Removed holder.textCategory.setVisibility(View.GONE) that was hiding the category
- Modified SearchResultsAdapter to always display the segment value if it exists
- Removed any filtering logic that excluded "Unknown" or "Miscellaneous" segments
- Ensured textCategory.setVisibility(View.VISIBLE) is called when segment is not null or empty

d) Explanation (Depth):
All event categories should be displayed to provide complete information to users. By removing the visibility hiding and ensuring all segment values (including "Unknown" and "Miscellaneous") are displayed, users can see the full categorization of events. This improves transparency and helps users understand event classifications.

==
ISSUE #9: Search Result Cards Background Color
==
a) Prompt (Clarity):
  User requested: "the cards also should have same grey backgrond which we used for event tab"

b) Issues (Specificity):
  - Search result cards had white background (app:cardBackgroundColor="@color/white")
  - Event details tabs used grey background (@color/eventfinder_gray_card)
  - Inconsistent visual styling between search results and event details

c) Fix (Correctness):
  - Changed cardBackgroundColor in search_result_item.xml from @color/white to @color/eventfinder_gray_card

d) Explanation (Depth):
  Consistent visual design improves user experience and makes the app feel more cohesive. By using the same grey background color for search result cards as used in event details tabs, we create visual consistency across the application. This is a simple styling change that enhances the overall design language.

==
ISSUE #10: Sports, Film, and Miscellaneous Category Events Not Displayed
==
a) Prompt (Clarity):
  User reported: "the sports, film and misc category events are not getting displayed on my app. I checked with my website version we have those events"

b) Issues (Specificity):
  - Events with "Sports", "Film", and "Miscellaneous" categories were not appearing when filtering by these categories
  - The code was filtering events by comparing event.segment with selectedCategory
  - "Miscellaneous" and "Film" are genres, not segments, in the Ticketmaster API
  - The Event class only stored segment information, not genre information
  - Filtering logic only checked segment field, not genre field

c) Fix (Correctness):
  - Added 'genre' field to Event class with a new constructor that accepts both segment and genre
  - Modified parseSearchResults() to extract genre from API response classifications array
  - Updated all Event constructor calls throughout SearchFragment to include genre parameter

- Modified filterResultsByCategory() to check genre field for "Miscellaneous" and "Film" categories
- Updated filtering logic in parseSearchResults() to use genre for genre-based categories
- Updated fetchEventImageForFavorite() to parse and store genre from event details API response

d) Explanation (Depth):

The Ticketmaster API categorizes events using both segments (Music, Sports, Arts & Theatre) and genres (Miscellaneous, Film, etc.). "Miscellaneous" and "Film" are genres that can appear within different segments. The original code only parsed and stored segment information, so when filtering by "Miscellaneous" or "Film", it couldn't find matches because it was only checking the segment field. By adding genre parsing and updating the filtering logic to check the appropriate field based on the category type, we can now correctly display all category-filtered events. This fix ensures that events are properly categorized and filtered regardless of whether they're classified by segment or genre in the API.

==
ISSUE #11: checkAndAddNewFavorites Not Removing Unliked Events from Backend
==

a) Prompt (Clarity):

User requirement: The checkAndAddNewFavorites() method should not only add new favorites but also remove events that are no longer favorited in the backend.

b) Issues (Specificity):

- checkAndAddNewFavorites() only added new favorites from the backend
- Events that were unliked on other devices or through other means remained in likedEvents
- No synchronization to remove events that were deleted from backend favorites
- Local likedEvents list could become out of sync with backend state

c) Fix (Correctness):

- Modified checkAndAddNewFavorites() to fetch all backend favorite IDs first
- Added logic to collect backend favorite IDs into a Set
- Implemented removal of events from likedEvents that are not in the backend favorite IDs set
- Added logging to track removed and added favorites
- Updated display results after removing unliked events

d) Explanation (Depth):

Two-way synchronization is essential for maintaining consistency between local state and backend. By fetching all backend favorite IDs and comparing them with local likedEvents, we can identify events that were removed from favorites elsewhere. Removing these events from likedEvents ensures the UI accurately reflects the current backend state. This is particularly important for multi-device scenarios or when favorites are managed through different interfaces.

==
==========
ISSUE #12: Favorites Not Sorted by Addition Time (Newest First)
==
==========
a) Prompt (Clarity):
User requirement: "The events on the Favorites page are sorted in the order they are added to the favorites list" (newest first).

b) Issues (Specificity):
- Favorites were displayed in the order received from the backend API
- No sorting logic was applied to show newest favorites first
- Users couldn't easily see their most recently added favorites

c) Fix (Correctness):
- Added Collections.sort() in loadFavorites() to sort favorites by timestampAdded
- Implemented Comparator that parses timestamps and compares them
- Used reverse order (time2.compareTo(time1)) to show newest first
- Ensured sorting happens after all favorites are loaded from backend

d) Explanation (Depth):
Chronological sorting by addition time provides better user experience by showing the most recently favorited events first. This allows users to quickly access events they just added. The reverse chronological order (newest first) is intuitive and matches common UI patterns. By sorting after loading all favorites, we ensure accurate ordering regardless of the backend response order.

==
==========
ISSUE #13: Time Elapsed Not Displayed for Favorite Events
==
==========
a) Prompt (Clarity):
User requirement: "Each of the items in the list includes an event catalog image, event name, date/time, and time elapsed since it was added to favorites"

b) Issues (Specificity):
- Time elapsed since favorite was added was not displayed
- FavoritesAdapter did not format or display timestampAdded information
- No visual indication of when an event was favorited

c) Fix (Correctness):
- Added formatTimeElapsed() method in FavoritesAdapter to calculate time difference
- Implemented parsing for both ISO 8601 format and epoch milliseconds timestamps
- Added logic to display time elapsed in human-readable format (seconds, minutes, hours, days ago)
- Set textTimeElapsed TextView with formatted time elapsed string
- Handled edge cases for timestamp parsing errors

d) Explanation (Depth):
   Displaying time elapsed provides context about when favorites were added, helping users understand their favoriting history. The formatTimeElapsed() method calculates the difference between current time and the timestamp when the event was added, then formats it in a human-readable way. Supporting multiple timestamp formats (ISO 8601 and epoch) ensures compatibility with different backend response formats.

==
ISSUE #14: Date Not Displayed Below Event Name in Favorites
==
a) Prompt (Clarity):
   User requested: "the date should also get displayed below the heading for each fav" and "display the date too by using the reference from the picture"

b) Issues (Specificity):
   - Date was not displayed below the event name heading in favorite items
   - FavoritesAdapter did not set textDate visibility or content
   - Layout had textEventDate TextView but it was not being populated

c) Fix (Correctness):
   - Modified FavoritesAdapter.onBindViewHolder() to set textDate text and visibility
   - Added logic to display event.date if available, otherwise hide the TextView
   - Ensured date is formatted as "MMM d, yyyy, h:mm AM/PM" in HomeFragment before setting
   - Added logging to track when date is set or missing

d) Explanation (Depth):
   Displaying the date below the event name provides essential event information at a glance. By checking if event.date is available and setting visibility accordingly, we handle cases where date information might be missing. The date is formatted consistently with the search results format, maintaining visual consistency across the app.

==
ISSUE #15: Arrow Icon Not Matching Design (Head-Only Chevron)
==
a) Prompt (Clarity):
   User requested: "the time ellapse and arrow should be on same row if possible try replacing arrow with the arrow that has only head like in second picture"

b) Issues (Specificity):
   - Arrow icon used was a full arrow (ic_arrow_back rotated) instead of a head-only chevron
   - Icon did not match the design specification shown in reference images
   - Visual appearance was inconsistent with expected UI design

c) Fix (Correctness):
- Created new drawable resource ic_chevron_right.xml with head-only chevron design
- Used vector path to draw a right-pointing chevron (head only, no stem)
- Updated favorite_item.xml to use ic_chevron_right instead of rotated ic_arrow_back
- Positioned chevron icon on the same row as event name and time elapsed

d) Explanation (Depth):
Using a head-only chevron icon provides a cleaner, more modern appearance that matches common navigation patterns. The chevron clearly indicates navigation without visual clutter. Creating a custom vector drawable ensures the icon matches the exact design specification and can be easily customized (color, size) through XML attributes.

==
ISSUE #16: Favorites Layout - Time Elapsed and Arrow Not on Same Row
==
a) Prompt (Clarity):
User requested: "the time ellapse and arrow should be on same row if possible"

b) Issues (Specificity):
- Time elapsed and navigation arrow were not positioned on the same horizontal row as the event name
- Layout structure did not match the design specification
- Event name, time elapsed, and arrow were in separate rows or misaligned

c) Fix (Correctness):
- Restructured favorite_item.xml layout to use a horizontal LinearLayout
- Placed textEventTitle, textTimeElapsed, and chevron icon in the same horizontal row
- Used layout_weight for textEventTitle to allow proper spacing
- Positioned textEventDate on a new line below the name/time/arrow row
- Adjusted margins and gravity to ensure proper alignment

d) Explanation (Depth):
Placing the event name, time elapsed, and navigation arrow on the same row maximizes space efficiency and creates a cleaner visual hierarchy. The event name takes up available space (layout_weight=1), while time elapsed and arrow are positioned at the end. The date is placed below on a separate line to maintain readability while keeping the header row compact.

==
ISSUE #17: No Favorites Message Not Displayed
==
a) Prompt (Clarity):

User requirement: "If there are no favorite events, 'No favorites' should be displayed at the center of the screen"

b) Issues (Specificity):
- Empty state was not shown when favorites list was empty
- No visual feedback when user had no favorited events
- Empty state container existed in layout but was not being managed

c) Fix (Correctness):
- Implemented updateEmptyState() method in HomeFragment to show/hide empty state
- Added logic to set emptyStateContainer visibility based on favorites list size
- Called updateEmptyState() after loading favorites and on error
- Ensured empty state is displayed when favorites.isEmpty() is true

d) Explanation (Depth):
Providing clear feedback when lists are empty improves user experience by confirming the app state. The empty state message ("No favorites") informs users that the list is intentionally empty rather than indicating an error. By checking the favorites list size and updating visibility accordingly, we ensure the empty state is shown at the right times.

==
ISSUE #18: Lambda Expression Final Variable Error
==
a) Prompt (Clarity):
Compiler error: "local variables referenced from a lambda expression must be final or effectively final" at line 357 in HomeFragment.java

b) Issues (Specificity):
- Variable 'position' was being modified within a loop and then used in a lambda expression
- Java requires variables used in lambdas to be final or effectively final
- Code attempted to use a non-final variable in runOnUiThread lambda

c) Fix (Correctness):
- Created final copies of 'position' and 'event.id' before using them in lambda
- Declared 'final int finalPosition = position' and 'final String eventId = event.id'
- Used finalPosition and eventId inside the lambda expression instead of the original variables
- Added validation to check if finalPosition is still valid before notifying adapter

d) Explanation (Depth):
Java's lambda expressions can only access final or effectively final variables to prevent concurrency issues and ensure predictable behavior. By creating final copies of the variables before the lambda, we satisfy this requirement while maintaining the intended functionality. The validation check ensures the position is still valid when the lambda executes, preventing IndexOutOfBoundsException.

==

ISSUE #19: Adapter Notifications Not on UI Thread

==

a) Prompt (Clarity):
  Issue: Adapter notifications were being called from background threads, causing potential UI update errors.

b) Issues (Specificity):
  - fetchEventDetailsForFavorite() runs on a background thread (Volley response handler)
  - adapter.notifyItemChanged() was called directly from the background thread
  - This could cause IllegalStateException or UI update failures
  - Android requires all UI updates to be on the main/UI thread

c) Fix (Correctness):
  - Wrapped adapter notification calls in getActivity().runOnUiThread()
  - Ensured all adapter.notifyItemChanged() and notifyDataSetChanged() calls are on UI thread
  - Added fallback to notifyDataSetChanged() if position validation fails
  - Maintained proper thread safety for all UI updates

d) Explanation (Depth):
  Android's UI framework is single-threaded and requires all UI operations to occur on the main thread. Volley response handlers run on background threads, so any adapter notifications must be posted to the UI thread. Using runOnUiThread() ensures thread safety and prevents crashes. The fallback to notifyDataSetChanged() provides a safe alternative if the specific item position is no longer valid.

==

ISSUE #20: Date Format for Favorites Not Matching Specification

==

a) Prompt (Clarity):
  User requirement: Date should be displayed as "MMM d, yyyy, h:mm AM/PM" (e.g., "Aug 8, 2026, 5:30 PM")

b) Issues (Specificity):
  - Date format in favorites did not match the specified format
  - Date and time were not combined with comma separator
  - Format was inconsistent with search results date display

c) Fix (Correctness):
  - Modified formatDate() in HomeFragment to return "MMM d, yyyy" format
  - Modified formatTime() to return "h:mm AM/PM" format
  - Combined date and time with ", " separator in fetchEventDetailsForFavorite()
  - Updated FavoriteEvent.date field with the combined formatted string

   - Ensured format matches exactly: "MMM d, yyyy, h:mm AM/PM"

d) Explanation (Depth):
   Consistent date formatting across the app improves readability and user experience. The specified format "MMM d, yyyy, h:mm AM/PM" provides clear, readable date and time information. By formatting in HomeFragment and storing the combined string in FavoriteEvent.date, we ensure the format is applied consistently. The comma separator between date and time matches standard date formatting conventions.

==
ISSUE #21: Image Loading for Favorites Not Working Properly
==
a) Prompt (Clarity):
   Issue: Favorite event images were not loading or displaying correctly in the favorites list.

b) Issues (Specificity):
   - Image URLs from backend favorites API were sometimes empty
   - Images were not being fetched from event details API when missing
   - Image loading logic in FavoritesAdapter was not handling empty URLs
   - No fallback mechanism for missing images

c) Fix (Correctness):
   - Modified loadFavorites() to check if imageUrl is empty and fetch from event details if needed
   - Enhanced fetchEventDetailsForFavorite() to fetch and update imageUrl
   - Updated FavoritesAdapter to check for empty imageUrl before attempting to load
   - Added imageRequestQueue cancellation to prevent memory leaks
   - Implemented proper image loading with ImageRequest in FavoritesAdapter

d) Explanation (Depth):
   Images are essential for visual identification of events. When the backend favorites API doesn't include image URLs (for performance reasons), we need to fetch them from the event details API. By checking for empty URLs and fetching them asynchronously, we ensure all favorites display images. Proper request cancellation prevents memory leaks and ensures only the latest image request is processed for each view.

==
ISSUE #22: Event Details Not Fetched for Missing Date/Time in Favorites
==
a) Prompt (Clarity):
   Issue: Favorite events were missing date and time information when the backend favorites API didn't include it.

b) Issues (Specificity):

- Backend favorites API sometimes returned empty dateRaw or timeRaw
- FavoriteEvent objects were created with empty date/time fields
- No mechanism to fetch missing date/time from event details API
- Users saw favorites without date/time information

c) Fix (Correctness):
- Modified loadFavorites() to check if dateRaw or timeRaw are empty
- Added needDate parameter to fetchEventDetailsForFavorite() to indicate if date/time is needed
- Enhanced fetchEventDetailsForFavorite() to parse dates object from event details response
- Added logic to format and combine date and time when fetched from event details
- Updated FavoriteEvent.date field directly in the favorites list

d) Explanation (Depth):
Complete event information improves user experience. When the backend favorites API omits date/time (for performance), we fetch it from the event details API. The needDate parameter allows us to conditionally fetch date/time only when needed, optimizing API calls. By updating the FavoriteEvent object directly in the list and notifying the adapter, we ensure the UI updates efficiently without full list refreshes.

==
ISSUE #23: Position Validation Missing in Adapter Notifications
==
a) Prompt (Clarity):
Issue: Adapter notifications could fail if the item position changed between fetching data and updating UI.

b) Issues (Specificity):
- Position calculated when fetching event details might be invalid when updating UI
- No validation to check if position is still within list bounds
- No check to verify the item at position still matches the expected event ID
- Could cause IndexOutOfBoundsException or updating wrong item

c) Fix (Correctness):
- Added validation to check if finalPosition < favorites.size() before notifying
- Added check to verify favorites.get(finalPosition).id.equals(eventId) before notifying
- Implemented fallback to notifyDataSetChanged() if position validation fails
- Added logging to track when position validation fails and fallback is used

d) Explanation (Depth):
List positions can change between async operations (e.g., user adds/removes favorites while fetching details). Validating the position ensures we're updating the correct item and prevents crashes. The ID check ensures the item at that position is still the one we fetched details for. The fallback to notifyDataSetChanged() provides a safe alternative that updates all items, ensuring correctness even if positions shifted.

==
ISSUE #24: Empty State Not Updated on Error
==
a) Prompt (Clarity):
   Issue: Empty state was not shown when favorites failed to load due to network errors.

b) Issues (Specificity):
   - updateEmptyState() was not called in error handlers
   - Users saw loading state or stale data when API calls failed
   - No feedback when favorites couldn't be loaded

c) Fix (Correctness):
   - Added updateEmptyState() call in Response.ErrorListener for loadFavorites()
   - Cleared favorites list on error before calling updateEmptyState()
   - Ensured empty state is shown when favorites list is empty due to errors
   - Added proper error logging before updating empty state

d) Explanation (Depth):
   Error handling should provide clear feedback to users. When favorites fail to load, showing the empty state (or an appropriate error message) is better than showing stale data or a perpetual loading state. Clearing the list ensures the empty state is correctly displayed. This provides a consistent user experience even when network issues occur.

==
ISSUE #25: Timestamp Parsing for Time Elapsed Not Handling All Formats
==
a) Prompt (Clarity):
   Issue: Time elapsed calculation failed for some timestamp formats returned by the backend.

b) Issues (Specificity):
   - formatTimeElapsed() only handled specific timestamp formats
   - Backend might return timestamps in different formats (ISO 8601, epoch milliseconds, etc.)
   - Parsing errors caused time elapsed to display as empty string
   - No fallback for unrecognized timestamp formats

c) Fix (Correctness):
   - Enhanced formatTimeElapsed() to detect timestamp format (ISO 8601 vs epoch)
   - Added parsing for ISO 8601 format with timezone handling
   - Added parsing for epoch milliseconds format
   - Implemented try-catch to handle parsing errors gracefully
   - Returned empty string on parsing errors instead of crashing

d) Explanation (Depth):
Backend APIs may return timestamps in various formats depending on configuration or version. Supporting multiple formats ensures the time elapsed feature works regardless of the timestamp format. Graceful error handling prevents crashes and provides a fallback (empty string) when timestamps can't be parsed. This makes the feature robust and user-friendly.

==
END OF PROCESS LOG
==