Technical Specification
Movie Browser iOS Application

Overview

Create a fully functional iOS application using **Swift** and **SwiftUI** that allows users to browse, search, and manage movies utilizing the [The Movie Database (TMDb) API](https://developer.themoviedb.org/docs/getting-started). The app will consist of a **Tab Bar** interface with five main sections:

1. **Movies** — Browse movies (endpoint /discover/movie) with search, sorting, genre filtering, and favorite management.
2. **Series** — Browse TV series (endpoint /discover/tv).
3. **Search** — Search movies or series (endpoint /search) with a segmented control (Movies / Series) and debounce.
4. **Saved** — Display user-saved movies and series with search and swipe actions (Share / Delete).
5. **Settings** — Manage profile photo and name, switch language, delete account/data, and view the About screen (TMDb attribution).

The goal is to implement a clean, performant, and user-friendly app architecture with comprehensive unit testing and optional support for accessibility and screenshot testing.

Requirements

1. General

- **Platform:** iOS 17
- **Languages & Frameworks:** Swift + SwiftUI
- **Architecture:** Open choice (MVVM, VIPER, Redux, etc.) — justify your selection.
- **Testing:**
    - **Mandatory:** Unit tests for core business logic and view models.
    - **Optional:** Screenshot/UI tests for key screens.
- **Accessibility:** Optional but recommended to implement dynamic type, VoiceOver labels, and other accessibility features.

2. API Integration

- Use the official **TMDb API**: https://developer.themoviedb.org/docs/getting-started
- Register for a free API key to access movie data.
- Handle network errors gracefully.
- Cache results locally for better performance and offline support (optional).

3. User Interface

Tab Bar Controller with 5 tabs:

Tab 1: Movies

- Display a list of movies fetched from TMDb (popular or latest).
- Each movie cell must show:
    - Poster image
    - Title
    - Release date
    - Genre(s)
    - Average rating
- Provide a **search bar** to filter movies by title.
- Include **sorting options** by:
    - Release date (ascending/descending)
    - Rating (ascending/descending)
    - Alphabetical order
- Provide **filtering by genre(s)** with multi-select capability.
- Allow users to **add/remove movies from favorites** directly from the list.
- Tapping a movie navigates to the **Movie Details** screen.

Tab 2: Series
Browse TV series fetched from TMDb
Data source: /discover/tv
The UI and features mirror those of the Movies tab.

Tab 3: Search
Search movies or series using TMDb /search endpoint.
Include a segmented control (Movies / Series) and apply a ~300 ms debounce before each request.

Tab 4: Saved

- Display all user-saved movies or series by the user.
- Allow removal of movies from favorites.
- Same layout as the movies list but only with favorites.
- Support search and sorting.

Tab 5: Settings

- Upload or change profile photo.
- Edit profile name (e.g., via an alert dialog).
- Switch the app language within the interface.
- Delete account and clear all local data.
- About screen with TMDb attribution.

Movie Details Screen

- Show comprehensive details about the selected movie:
    - Full poster and backdrop images
    - Title and original title
    - Release date
    - Genres
    - Overview/description
    - User rating
    - Popularity
    - Runtime
    - Language
    - Production companies (if available)
- Provide a button to add/remove the movie to/from favorites.

4. Technical Details

- **Networking:** Use URLSession.
- **Image Loading:** Efficient asynchronous loading and caching of movie posters and backdrops.
- **Persistence:** UserDefaults plus SwiftData (iOS 17) or Core Data (developer choice; justify in README).
- **Pagination**: implement infinite scrolling or “Load more” where applicable.
- **Loading States**: show activity indicators or skeleton placeholders during network operations.
- **Code Style**: project must include SwiftLint (default ruleset).
- **Concurrency:** Use Swift concurrency (async/await) or Combine for asynchronous operations.
- **Error Handling:** Provide user-friendly error messages for network or data failures.

5. Additional Features (Optional)

- **Offline support:** Cache movie lists and details to enable offline browsing.
- **Video trailers:** Integrate trailer playback on movie detail screen (using YouTube or TMDb video endpoints).
- Fancy animations are welcome.

6. Testing

- Write **unit tests** covering:
    - Network layer (mocking responses)
    - ViewModels or Presenters logic
    - Persistence layer
- Optionally, write **screenshot/UI tests** for key user flows (movie list, details, favorites).

- Ensure tests are runnable and included in the project.
- If SwiftData or Core Data is used, include CRUD tests.

7. Accessibility (Optional)

- Support Dynamic Type (font size adjustment).
- Use proper accessibility labels and hints for UI elements.
- Ensure the app works well with VoiceOver.

8. Deliverables

- Complete Xcode project with clean, readable, and well-documented code.
- README file with:
    - Project overview
    - Architecture explanation
    - How to run and test the app
    - Any known issues or limitations
    - If needed, briefly note any extra features you added and why