PROJECT ABSTRACT

Movie Watchlist App using React Native

A cross-platform mobile app built with React Native and Expo: browse and search a movie catalogue, keep a watchlist, mark films watched and rate them with stars — offline-first with local storage. Delivered built-to-order with source code, demo build, report, PPT and viva kit.

01. Title

Movie Watchlist App using React Native

02. Introduction / Background

Everyone keeps a mental list of films to watch — and loses it. Notes apps are not built for movies: no posters, no ratings, no distinction between 'want to watch' and 'watched'. Existing streaming apps lock watchlists inside one service, while a student-built app can be service-agnostic and fully offline. React Native lets one JavaScript codebase ship to both Android and iOS, and Expo removes the native-toolchain pain that stops most student mobile projects. This project builds that app: catalogue browsing, search, a persistent watchlist, watched history and star ratings, all working without an internet connection.

03. Problem Statement

Movie recommendations arrive constantly — from friends, social media, trailers — but there is no lightweight place to capture them with their metadata. Streaming-service watchlists are siloed per platform and need subscriptions; generic notes lose posters, ratings and structure. The project addresses this by building a dedicated, offline-first mobile watchlist: save a film in two taps, see what is queued versus watched, and rate films to build a personal taste record — with data persisted on the device via AsyncStorage.

04. Objectives

- Build a React Native (Expo) mobile app with Home, Search and Watchlist tabs.
- Implement a browsable movie catalogue with posters, genres, years and ratings.
- Implement watchlist management: add/remove, to-watch vs watched states, star ratings.
- Persist all user data on-device with AsyncStorage for fully offline use.
- Deliver the complete student kit: source code, Expo demo build instructions, report, PPT and viva Q&A.

05. Proposed Methodology

The app is developed with Expo's managed workflow in React Native. Navigation uses React Navigation (bottom tabs + stack for the detail screen). The catalogue is a bundled JSON dataset of films with metadata; the search screen filters it client-side by title and genre. Watchlist state (film id, watched flag, star rating) is held in a context provider and persisted to AsyncStorage on every change, rehydrated at launch. The detail screen shows synopsis, metadata chips and the interactive star-rating control. The UI is built with StyleSheet-based components following mobile design conventions (tab bar, cards, touch feedback).

06. System Architecture / Working

On launch the app rehydrates the watchlist from AsyncStorage into the global store. The Home tab renders trending and top-rated sections from the bundled catalogue; tapping a poster pushes the detail screen with synopsis and metadata. From the detail screen (or any row card) the user adds the film to the watchlist, marks it watched, or assigns a star rating — each action updates the store and writes through to AsyncStorage immediately. The Watchlist tab splits saved films into 'To watch' and 'Watched' sections with counts. Search filters the catalogue locally as the user types, so every feature works offline.

07. Hardware / Software Requirements

- Node.js 18+, Expo CLI, React Native, React Navigation
- Android Studio emulator or a physical Android device (Expo Go) for testing
- AsyncStorage for on-device persistence; no backend or database server needed
- No special hardware; the demo runs on any Android phone via Expo Go

08. Expected Outcomes

- Working mobile app with three tabs: Home, Search, Watchlist.
- Film detail screens with synopsis, metadata and star ratings.
- Watchlist with to-watch/watched separation, persisted across restarts.
- Fully offline operation; installable demo via Expo build.
- Complete documentation: report, PPT, setup guide and viva Q&A.

09. Applications

- Personal movie tracking for students and film-club members.
- Teaching example of React Native fundamentals: navigation, state, persistence.
- Reference build for other catalogue apps (books, games, restaurants).
- Base for a future version with TMDB API integration and social sharing.

10. Future Scope

- Live catalogue via the TMDB API with real posters and trailers.
- Recommendation engine based on rated films.
- Social features: shared lists and friend activity.
- Reminder notifications for watchlist films leaving streaming services.

11. References

- React Native documentation (reactnative.dev).
- Expo documentation — managed workflow and builds (docs.expo.dev).
- React Navigation documentation — tab and stack navigation.
- AsyncStorage documentation — asynchronous on-device key-value storage.