Pinball App Architecture Blueprint

1. System Overview

What the app is
Pinball App is a dual-client mobile application (Android + iOS) for Lansing Pinball League users. It combines league analytics, a machine library, and a personal practice/workflow tracker.

Core purpose
Provide one place to:
- View league data (`Stats`, `Standings`, `Targets`).
- Browse machine references (playfields, rulesheets, videos, game notes).
- Track personal practice activity and compare performance over time.

Main features
- League dashboard with card previews and drill-down screens.
- Multi-filter league analytics.
- Offline-first content/cache system with starter-pack fallback.
- Library search/sort/filter, detail pages, embedded videos, markdown rulesheets.
- Practice subsystem with:
  - Quick entry logging.
  - Per-game workspace (summary/input/log/resources).
  - Group dashboard and group editor.
  - Journal timeline merging practice + library activity.
  - Insights (score stats + head-to-head comparison).
  - Mechanics tracking and trend history.
  - Settings, league CSV import, reset workflow.

Target users
- Lansing Pinball League players.
- Competitive and casual players who want to study machines and log improvement.

---

2. Technology Stack

Languages, frameworks, libraries
- Android:
  - Kotlin
  - Jetpack Compose (Material3)
  - Coil (image loading)
  - CommonMark + compose-richtext (markdown rendering)
  - AndroidX lifecycle/activity/splashscreen
- iOS:
  - Swift
  - SwiftUI (+ Combine)
  - WebKit (embedded YouTube/rulesheet HTML rendering)
  - Codable + UserDefaults persistence

Storage systems
- Android:
  - SharedPreferences (practice state, UI prefs, library activity).
  - File-based cache in app files directory (`pinball-data-cache`).
- iOS:
  - UserDefaults (Codable blobs + `@AppStorage` keys).
  - File-based cache in caches directory (`pinball-data-cache`).

Networking / API layers
- No writable backend API in app code.
- HTTP fetches from static content host: `https://pillyliu.com/pinball/...`
- Remote datasets: CSV, JSON, Markdown.
- Cache metadata:
  - `/pinball/cache-manifest.json`
  - `/pinball/cache-update-log.json`
- External link integrations:
  - YouTube video playback/thumbnail URLs.
  - External source links (rulesheet/playfield URLs).

---

3. C4 Architecture Diagrams

3.1 System Context (C1)

```mermaid
flowchart LR
    U["League Player"] --> A["Pinball App (Android)"]
    U --> I["Pinball App (iOS)"]

    A --> P["pillyliu.com static data host"]
    I --> P

    P --> D1["League CSV data"]
    P --> D2["Library JSON + markdown + images"]
    P --> D3["Cache manifest + update log"]

    A --> Y["YouTube"]
    I --> Y

    A --> SA["Android local storage (SharedPreferences + file cache)"]
    I --> SI["iOS local storage (UserDefaults + file cache)"]
```

3.2 Container Diagram (C2)

```mermaid
flowchart TB
    subgraph Android["Android App Container"]
        AUI["Compose UI Layer (Tabs + Screens)"]
        ADOM["Feature/ViewModel-like domain logic"]
        ACACHE["PinballDataCache (file cache + manifest sync)"]
        APREF["SharedPreferences persistence"]
    end

    subgraph iOS["iOS App Container"]
        IUI["SwiftUI UI Layer (TabView + NavigationStack)"]
        IDOM["ObservableObject stores/view models"]
        ICACHE["PinballDataCache actor (file cache + manifest sync)"]
        IDEF["UserDefaults + AppStorage persistence"]
    end

    DATA["pillyliu.com static resources"]
    YT["YouTube + web sources"]

    AUI --> ADOM --> ACACHE --> DATA
    ADOM --> APREF

    IUI --> IDOM --> ICACHE --> DATA
    IDOM --> IDEF

    AUI --> YT
    IUI --> YT
```

3.3 Component Diagrams (C3)

League module

```mermaid
flowchart LR
    LH["League Hub"] --> ST["Stats Screen"]
    LH --> SD["Standings Screen"]
    LH --> TG["Targets Screen"]

    ST --> SC["Stats parser + filters + table + machine stats"]
    SD --> SDC["Standings parser + season selector + ranking table"]
    TG --> TGC["Targets mapper + sort/filter + benchmark table"]

    SC --> CACHE["PinballDataCache"]
    SDC --> CACHE
    TGC --> CACHE
```

Library module

```mermaid
flowchart LR
    LL["Library List"] --> LD["Library Detail"]
    LD --> RS["Rulesheet Screen"]
    LD --> PF["Playfield Screen"]

    LL --> LF["Search + sort + bank filter + grouping"]
    LD --> LI["Game info markdown + media + sources"]
    LD --> LLOG["LibraryActivityLog"]

    RS --> RDATA["Rulesheet markdown loader + renderer"]
    PF --> PVIEW["Fullscreen zoom/pan image viewer"]

    LF --> CACHE["PinballDataCache"]
    LI --> CACHE
    RDATA --> CACHE
```

Practice module

```mermaid
flowchart TB
```

```
    PH["Practice Home"] --> PQ["Quick Entry"]
    PH --> PG["Game Workspace"]
    PH --> PGD["Group Dashboard"]
    PH --> PJ["Journal Timeline"]
    PH --> PI["Insights"]
    PH --> PM["Mechanics"]
    PH --> PS["Settings"]

    PGD --> PGE["Group Editor"]
    PGE --> PGS["Game Selection"]

    PG --> PGSUM["Summary"]
    PG --> PGIN["Input"]
    PG --> PGLOG["Log"]
    PG --> PGRES["Resources"]

    subgraph Store["PracticeStore"]
        PSTATE["In-memory practice state"]
        PIO["Load/save JSON state"]
        POPS["Mutations + analytics + league helpers"]
    end

    PQ --> Store
    PG --> Store
    PGD --> Store
    PGE --> Store
    PJ --> Store
    PI --> Store
    PM --> Store
    PS --> Store

    Store --> CACHE["PinballDataCache"]
    Store --> PREFS["SharedPreferences/UserDefaults"]
    PJ --> LLOG["LibraryActivityLog"]
```

3.4 Code-Level Diagram (C4, feasible core)

```mermaid
classDiagram
    class PinballDataCache {
      +loadText(path/url)
      +forceRefreshText(path/url)
      +hasRemoteUpdate(path/url)
      +cachedUpdatedAt(path/url)
    }

    class PracticeStore {
      +loadIfNeeded()
      +importLeagueScoresFromCsv()
      +comparePlayers()
      +addScore/addNote/addJournal()
      +resetPracticeState()
    }

    class PinballGame {
      +slug/id
      +name
      +group/pos/bank
      +videos
      +playfield candidates
    }

    class PracticePersistedState {
      +groups
      +scores
      +notes
      +journal
      +settings
      +rulesheet progress
      +game summary notes
    }

    class LibraryActivityLog {
      +log()
      +events()
      +clear()
    }

    class StatsScreen
    class StandingsScreen
    class TargetsScreen
    class LibraryScreen
    class PracticeScreen

    StatsScreen --> PinballDataCache
    StandingsScreen --> PinballDataCache
    TargetsScreen --> PinballDataCache
    LibraryScreen --> PinballDataCache
    PracticeScreen --> PracticeStore
    PracticeStore --> PinballDataCache
    PracticeStore --> PracticePersistedState
    PracticeStore --> PinballGame
    LibraryScreen --> LibraryActivityLog
    PracticeScreen --> LibraryActivityLog
```

---

4. Screen and Feature Inventory

Root navigation (both platforms)
- Tabs:
  - `League`
  - `Library`
  - `Practice`
  - `About`
- Navigation style:
  - Android: tab state + internal route enums.
  - iOS: `TabView` + `NavigationStack` per tab.

League screens

Screen	Purpose	Buttons/Controls/Filters	Navigation Targets	Data Reads	Data Writes
League Hub	Entry dashboard for league data	3 destination cards: `Stats`, `Standings`, `Targets`	To corresponding league screens	`LPL_Stats.csv`, `LPL_Standings.csv`, `LPL_Targets.csv`, `pinball_library.json` (preview)	None
Stats	Score analytics + machine stats	Filters: season, bank, player, machine. Refresh control. iOS nav menu: `Clear all filters` + per-filter menus.	Back to League Hub	`LPL_Stats.csv`, redacted players CSV	In-memory filter
Standings	Season rankings and bank totals	Season selector (`Season N` menu), refresh control	Back to League Hub	`LPL_Standings.csv`, redacted players CSV	In-memory season selection
Targets	Per-game target benchmarks	Sort (`Location/Bank/A-Z`), bank filter (`All banks`/`Bank N`)	Back to League Hub	`LPL_Targets.csv` + `pinball_library.json` (mapping/sort metadata)	In-memory sort/filter state

Library screens

Screen	Purpose	Buttons/Controls/Filters	Navigation Targets	Data Reads	Data Writes
Library List	Search and browse games	Search text box; sort menu (`Location`, `Bank`, `A-Z`); bank filter (`All banks`, `Bank N`)	Open `Library Detail`	`pinball_library.json`	Saves last viewed library game (`library-last-vie
Library Detail	Show game media + references	`Rulesheet`, `Playfield`; video tile buttons; source links (`Rulesheet (source)`, `Playfield (source)`)	To `Rulesheet`, `Playfield`	game info markdown (`/pinball/gameinfo/{slug}.md`
Rulesheet Viewer	Read rulesheet markdown/web	Back, resume/continue controls (platform-specific), confirm dialogs in some flows	Back to detail/practice	`/pinball/rulesheets/{slug}.md`	Rulesheet progress ratio stored in pract
Playfield Viewer	Fullscreen zoom/pan image viewer	Back; gesture zoom/pan (platform-specific)	Back to detail/practice	Playfield URLs (derived/local/remote candidates)	None

Practice screens and dialogs

Screen	Purpose	Buttons/Controls/Filters	Navigation Targets	Data Reads	Data Writes
Practice Home	Main launchpad	`Resume` chip; `Game List`; quick entry buttons (`Score`, `Study`, `Practice`, `Mechanics`); hub cards (`Group Dashboard`, `Journal Timeline`, `Insights`, `Mechanics`); settings icon	To all practice
Welcome Name Prompt	First-time name collection	`Save`, `Not now`, player name text field	Dismiss to home	Current player name state	Saves player profile name, prompt flags
Quick Entry (sheet/dialog)	Fast multi-mode logging	Mode/game pickers; mode-specific fields; `Save`, `Cancel`	Return to current/game route	Games list, current quick-entry defaults	Appends score/note/journal/study/mechanics e
Group Dashboard	Group status and game recommendations	Create (`+`), edit (`pencil`), group select, priority toggle, start/end date buttons, per-game open, delete game from group (context menu)	To Group Editor or Game Workspace
Group Editor	Create/update groups	`Cancel`, `Delete` (edit), `Create/Save`, template selectors (`None/Bank/Duplicate`), apply template buttons, title selector, reorder, active/priority toggles, type segmented control, position up
Group Game Selection	Pick titles for group	Search field, selectable game list/cards, `Done`	Back to group editor	Games list	Mutates selected game IDs/slugs for group draft
Journal Timeline	Unified activity history	Filter segmented control (`All`, `Study`, `Practice`, `Scores`, `Notes`, `League`)	Tap row opens Game Workspace	Practice journal + library activity log	Stores selected journal filte
Insights	Performance analytics + head-to-head	Game dropdown; opponent dropdown (`Select player`); refresh comparison button	None (within practice stack)	Practice scores; imported league data/player options	Saves comparison
Mechanics	Skill logging and trend review	Skill picker; competency slider; mechanics note field; `Log Mechanics Session`; `Dead Flip Tutorials` link	External tutorial site	Mechanics-tagged notes/history	Adds note/journal ent
Practice Settings	Profile/import/reset settings	Player name field + `Save Profile`; league player menu + `Import LPL CSV`; cloud sync toggle; `Reset Practice Log`	Reset confirm dialog	Available league players	Updates profil
Reset Confirm	Guard destructive reset	Text field (`Type reset`), confirm/cancel buttons	Back to settings	Current practice state	Clears practice state + clears library activity log
Group Date Picker	Edit start/end dates	Date picker + `Save`/`Clear`/`Cancel`	Back to dashboard/editor	Existing group dates	Updates group start/end date
Game Workspace	Per-game detailed workflow	Top game picker menu; subview segmented control (`Summary`, `Input`, `Log`); `Save Note`; `Rulesheet`; `Playfield`; video tile selection	To rulesheet/playfield screens from resources
Game Workspace Input subview	Task-specific logging shortcuts	Buttons for `Rulesheet`, `Tutorial`, `Gameplay`, `Playfield`, `Practice`, `Mechanics`, `Log Score` (Android) or task sheets (iOS)	Opens entry sheets/dialogs	Current
Game Task Entry / Score Entry sheets	Structured per-task data input	Task forms, `Save`, `Cancel`	Back to Game Workspace	Current game + enum defaults	Writes study/score/note/journal data

About screen

Screen	Purpose	Buttons/Controls/Filters	Navigation Targets	Data Reads	Data Writes
About	League intro/info	External links: `lansingpinleague.com`, `Facebook Group`	Browser	Static in-app copy/assets	None

---

5. Screen Interaction Diagrams

Stats screen state + interactions

```mermaid
stateDiagram-v2
    [*] --> Loading
    Loading --> Ready: CSV loaded
    Loading --> Error: fetch/parsing failed
    Ready --> Empty: filters produce 0 rows
    Ready --> Filtered: any filter changed
    Filtered --> Ready: clear/reset filters
    Ready --> Refreshing: refresh tapped
    Refreshing --> Ready: success
    Refreshing --> Error: failure
```

```mermaid
flowchart LR
    U["User selects filters"] --> F["Apply season/player/bank/machine predicates"]
    F --> T["Recompute filtered table"]
    T --> S["Recompute machine stats card"]
    U2["User taps refresh"] --> R["Force refresh via cache layer"]
    R --> T
```

Library list/detail flow

```mermaid
flowchart TD
    L["Library List"] -->|Tap game card| D["Library Detail"]
    D -->|Rulesheet button| R["Rulesheet Viewer"]
    D -->|Playfield button| P["Playfield Viewer"]
    D -->|Tap video tile| V["Embedded video player updates"]
    L -->|Search/Sort/Bank filter| L
    D -->|Back| L
    R -->|Back| D
    P -->|Back| D
```

Practice home + quick entry flow

```mermaid
flowchart TD
    H["Practice Home"] --> Q["Quick Entry Sheet/Dialog"]
    Q -->|Save valid| M["PracticeStore mutation"]
    M --> J["Journal updated"]
    M --> G["Optional navigate/open game workspace"]
    Q -->|Validation error| Q
    Q -->|Cancel| H
```

Practice game workspace state

```mermaid
stateDiagram-v2
    [*] --> GameSelected
    GameSelected --> SummaryTab
    GameSelected --> InputTab
    GameSelected --> LogTab
    SummaryTab --> SavingNote: Save Note
    SavingNote --> SummaryTab
    InputTab --> EntryDialog: task/score action
    EntryDialog --> InputTab: save/cancel
    LogTab --> GameSelected: switch tab
    GameSelected --> Rulesheet: open resource
    GameSelected --> Playfield: open resource
    Rulesheet --> GameSelected
    Playfield --> GameSelected
```

---

6. Sequence Diagrams (Behavior)

App launch

```mermaid
sequenceDiagram
    participant User
    participant UI as App UI
    participant Cache as PinballDataCache
    participant Storage as Local Storage
    participant Remote as pillyliu.com

    User->>UI: Launch app
    UI->>Cache: initialize/load
    Cache->>Storage: read cache index + starter seeded files
    Cache->>Remote: refresh manifest/update-log (best effort)
    UI->>Storage: load persisted practice/library prefs
    UI->>Cache: request initial datasets per active tab
    Cache->>Remote: fetch missing/stale files
```

~~~
    Cache-->>UI: cached or fresh content
```

### Opening a game (Library -> Detail)

```mermaid
sequenceDiagram
    participant User
    participant List as LibraryList
    participant Detail as LibraryDetail
    participant Cache as PinballDataCache
    participant Log as LibraryActivityLog

    User->>List: Tap game card
    List->>Log: log(browseGame)
    List->>Detail: navigate(game)
    Detail->>Cache: load gameinfo markdown
    Detail-->>User: render image/meta/videos/game info
```

### Applying a filter (Stats)

```mermaid
sequenceDiagram
    participant User
    participant StatsUI as StatsScreen/ViewModel
    participant Data as Loaded rows

    User->>StatsUI: Change season/player/bank/machine
    StatsUI->>Data: apply predicates
    StatsUI->>StatsUI: recompute derived stats + table widths
    StatsUI-->>User: updated table + stats panels
```

### Saving user data (Quick Entry / Game note)

```mermaid
sequenceDiagram
    participant User
    participant Screen as Practice UI
    participant Store as PracticeStore
    participant Persist as SharedPreferences/UserDefaults

    User->>Screen: Submit entry
    Screen->>Store: validate and mutate state
    Store->>Store: append score/note/journal/group changes
    Store->>Persist: serialize state JSON
    Persist-->>Store: success/failure
    Store-->>Screen: banner/error state
```

### Syncing/updating remote data

```mermaid
sequenceDiagram
    participant UI as Screen/ViewModel
    participant Cache as PinballDataCache
    participant Remote as manifest+files endpoint
    participant FS as local cache files

    UI->>Cache: loadText(path)
    Cache->>FS: check cached file
    alt Cached exists
        Cache-->>UI: return cached immediately
        Cache->>Remote: async revalidate
        Remote-->>Cache: new/unchanged file
        Cache->>FS: overwrite if changed
    else Not cached
        Cache->>Remote: download file
        Remote-->>Cache: file response
        Cache->>FS: write file + index
        Cache-->>UI: return data
    end
```

---

## 7. Data Model and Storage

### Core entities (domain-level)

- `PinballGame`
  - `slug/id`, `name`, `group`, `pos`, `bank`, `manufacturer`, `year`, media/rulesheet fields, `videos[]`.
~~~

- `Video`
  - `label`, `url` (and optional `kind` in iOS model).
- Practice entities
  - Groups: `PracticeGroup` / `CustomGameGroup`.
  - Scores: `ScoreEntry` / `ScoreLogEntry`.
  - Notes: `NoteEntry` / `PracticeNoteEntry`.
  - Journal: `JournalEntry`.
  - Derived analytics: `ScoreSummary`, `MechanicsSkillSummary`, `HeadToHeadComparison`, etc.
- Activity log
  - `LibraryActivityEvent` with kind (`browse/openRulesheet/openPlayfield/tapVideo`).

Remote source datasets
- `/pinball/data/pinball_library.json`
- `/pinball/data/LPL_Stats.csv`
- `/pinball/data/LPL_Standings.csv`
- `/pinball/data/LPL_Targets.csv`
- `/pinball/data/redacted_players.csv`
- `/pinball/gameinfo/{slug}.md`
- `/pinball/rulesheets/{slug}.md`
- `/pinball/cache-manifest.json`
- `/pinball/cache-update-log.json`

Local storage locations
- Android
  - `SharedPreferences`: `practice-upgrade-state-v2` and related keys.
  - Cache filesystem: `pinball-data-cache` + `cache-index.json`.
- iOS
  - `UserDefaults`: `practice-state-json` (+ legacy key) and app-storage keys.
  - Cache filesystem: `Caches/pinball-data-cache` + `cache-index.json`.

Data loading, caching, update behavior
- Offline-first:
  - Prefer local cache if available.
  - Async/background revalidation to keep UI responsive.
- Starter pack seeding:
  - Assets/bundle preloaded for priority files and baseline data.
- Metadata-driven invalidation:
  - Manifest hashes + update-log removal events reconcile local cache.
- Graceful fallback:
  - If network fails and stale cache exists, stale content is served.
  - Missing-allowed paths are represented as missing entries.

ER/Data model diagram

```mermaid
erDiagram
    PINBALL_GAME ||--o{ VIDEO : has
    PINBALL_GAME ||--o{ SCORE_ENTRY : has
    PINBALL_GAME ||--o{ NOTE_ENTRY : has
    PINBALL_GAME ||--o{ JOURNAL_ENTRY : has
    PINBALL_GAME ||--o{ LIBRARY_ACTIVITY_EVENT : has

    GROUP ||--o{ GROUP_GAME_LINK : contains
    PINBALL_GAME ||--o{ GROUP_GAME_LINK : included_in

    PRACTICE_STATE ||--o{ GROUP : stores
    PRACTICE_STATE ||--o{ SCORE_ENTRY : stores
    PRACTICE_STATE ||--o{ NOTE_ENTRY : stores
    PRACTICE_STATE ||--o{ JOURNAL_ENTRY : stores
    PRACTICE_STATE ||--o{ SETTINGS : stores

    PINBALL_GAME {
      string id_or_slug
      string name
      int group
      int pos
      int bank
      string manufacturer
      int year
    }
    SCORE_ENTRY {
      string id
      string game_id
      float score
      string context
      datetime_or_ms timestamp
      bool league_imported
    }
    NOTE_ENTRY {
      string id
      string game_id
      string category
      string detail
```

~~~
      string note
      datetime_or_ms timestamp
    }
    JOURNAL_ENTRY {
      string id
      string game_id
      string action
      string summary_or_payload
      datetime_or_ms timestamp
    }
    GROUP {
      string id
      string name
      string type
      bool is_active
      bool is_priority
      date start_date
      date end_date
    }
```

---

## 8. Data Flow Diagrams

```mermaid
flowchart LR
    REM["Remote static files (CSV/JSON/MD)"] --> CACHE["PinballDataCache"]
    START["Starter pack assets/bundle"] --> CACHE
    CACHE --> PARSE["Parsers / ViewModels / Store loaders"]
    PARSE --> UI["League/Library/Practice UI"]
    UI --> STORE["PracticeStore mutations"]
    STORE --> PERSIST["SharedPreferences/UserDefaults JSON"]
    UI --> ACT["LibraryActivityLog"]
    ACT --> PERSIST
    PERSIST --> UI
```

```mermaid
flowchart TD
    A["User action (filter/log/edit)"] --> B["UI event handler"]
    B --> C["State mutation (in-memory)"]
    C --> D["Derived analytics recompute"]
    C --> E["Persist to local storage"]
    D --> F["Re-render screen"]
    E --> G["Available after relaunch/offline"]
```

---

## 9. Navigation Map

```mermaid
flowchart TD
    ROOT["Root Tabs"] --> LEAGUE["League"]
    ROOT --> LIB["Library"]
    ROOT --> PRAC["Practice"]
    ROOT --> ABOUT["About"]

    LEAGUE --> L_STATS["Stats"]
    LEAGUE --> L_STAND["Standings"]
    LEAGUE --> L_TARG["Targets"]

    LIB --> LIB_LIST["Library List"]
    LIB_LIST --> LIB_DETAIL["Library Detail"]
    LIB_DETAIL --> LIB_RULE["Rulesheet Viewer"]
    LIB_DETAIL --> LIB_PLAY["Playfield Viewer"]

    PRAC --> P_HOME["Practice Home"]
    P_HOME --> P_QUICK["Quick Entry"]
    P_HOME --> P_GROUP_D["Group Dashboard"]
    P_HOME --> P_JOURNAL["Journal Timeline"]
    P_HOME --> P_INSIGHTS["Insights"]
    P_HOME --> P_MECH["Mechanics"]
    P_HOME --> P_SETTINGS["Practice Settings"]
    P_HOME --> P_GAME["Game Workspace"]

    P_GROUP_D --> P_GROUP_E["Group Editor"]
    P_GROUP_E --> P_GROUP_SEL["Group Game Selection"]

    P_GAME --> LIB_RULE
    P_GAME --> LIB_PLAY
```
~~~

Deep links
- No explicit deep-link URL handler implementation found in app code.
- Internal cross-tab navigation exists (iOS `AppNavigationModel.openLibraryGame`), but not OS-level URL deep links.
- Assumption: deep links are currently not exposed publicly.

---

10. Error, Offline, and Edge Cases

Data load failures
- League and library screens render error/empty messages when dataset fetch/parsing fails.
- Practice load/save failures set error strings and fallback to empty/default state (especially on decode failure).

Offline behavior
- Cache-first strategy serves local/starter content while offline.
- If a file was never cached and no starter fallback exists, screen can show empty/error state.
- Revalidation failures keep stale cached data.

Sync/update conflicts
- No multi-device conflict resolution yet.
- “Cloud sync” is explicitly optional placeholder/phase label; state remains device-local.
- Assumption: last local write wins within current device session.

Empty states
- Common explicit empty states:
  - No rows for selected filters.
  - No games/groups selected.
  - No videos listed.
  - No journal events.
  - No head-to-head overlap for selected players.

Input validation and guardrails
- Quick entry validates required fields by mode (score values, tournament name, etc.).
- Reset requires explicit `"reset"` confirmation text.
- Group editor validates naming/order and supports delete confirmations.

---

11. Final Architecture Summary

Pinball App is a two-client, offline-first mobile architecture with shared product behavior across Android and iOS. Both apps consume static league/library datasets from `pillyliu.com`, cache them aggressively with manifest-based fresh

Data flow is straightforward: remote static content enters through `PinballDataCache`, gets parsed into UI/store state, and user-generated practice data is persisted locally (SharedPreferences/UserDefaults JSON). The `Practice` module

Key architectural decisions are:
- Offline-first cache with starter-pack bootstrap and async revalidation.
- Static-content backend (read-only app perspective).
- Strong modular separation by feature domain.
- Local-first persistence for user practice workflows.
- Incremental, composable UI navigation per tab with nested feature routes.