UI Revamp Plan & Task List

Objectives

- Establish a theme-first design system so typography, color, and spacing are applied consistently across the app surface.
- Modernise navigation chrome and status bar handling to respond to platform theme and reduce per-screen overrides.
- Replace hardcoded UI copy/data with dynamic sources (excluding the learning mock data repository) to unblock localization and growth features.

Guiding Principles

- Respect the feature-first architecture: presentation layer consumes tokens/components, deeper layers stay unaware of UI concerns.
- Prefer composition over ad-hoc styling—shared widgets should expose configuration via theme or explicit props, not magic strings.
- Keep the `learning_mock_data_repository` untouched; wire dynamic data through domain interfaces so we can swap sources later.

Workstreams & Tasks

1. Theme & Tokens

- ☐ Define typography scale, color roles, and spacing tokens in `vidyaras_app/lib/src/shared/presentation/theme/` and surface them through `ThemeData` extensions.
- ☐ Replace inline text styles with `TextTheme` calls and add semantic getters for headings/body text.
- ☐ Introduce a spacing helper (e.g., `AppSpacing`) consumed by layout widgets to eliminate raw `EdgeInsets` and `SizedBox` numbers.

2. App Shell & Status Bar

- ☐ Centralize `SystemUiOverlayStyle` management (update `vidyaras_app/lib/main.dart` and `app_theme.dart`) to react to light/dark mode and route needs.

- [ ] Build a reusable `AppScaffold` / `AppBar` wrapper that encapsulates status bar color, title/subtitle, and optional actions.
- [ ] Normalize `SafeArea` usage so only the shell controls system insets and child screens rely on padding utilities.

3. Component Library & Layout

- [ ] Refactor button components to accept explicit trailing icon parameters (remove `label.contains('Next')` logic) and align padding/height with the spacing scale.
- [ ] Expand shared header, card, and list item widgets to accept theming tokens instead of raw colors; migrate existing screens to use them.
- [ ] Add utility widgets for section headers, empty states, and error blocks so repetition in Courses, Tests, and Profile can collapse onto shared patterns.

4. Screen Refactors

- [ ] Home: swap inline gradients/colors in `HomeHeader`, category pills, and stats cards for themed variants; ensure scroll behavior plays nicely with the new app bar.
- [ ] Courses: move category/filter definitions to a provider or config model and restyle tab/search areas with the new spacing/typography tokens.
- [ ] Tests: restyle the tab bar/header using shared components, and align the history/available lists with the reusable card layouts.
- [ ] My Courses: apply theme tokens to progress detail views, extract the join-live CTA into a reusable component, and keep data fetches unchanged.
- [ ] Profile & Shared Screens: migrate app bars, snackbars, and share flows to the new component set while keeping referral logic intact.

5. Dynamic Data & Copy

- [ ] Externalize user-facing strings into the localization layer (`flutter_localizations`) and replace hardcoded copy in presentation widgets.
- [ ] Source category and stats metadata from application/domain providers, leaving `learning_mock_data_repository` untouched while real endpoints are wired.
- [ ] Audit share messages, progress labels, and error states to ensure they handle missing/async data gracefully.

Dependencies & Sequencing

- Confirm updated color/typography tokens with design before broad refactors.
- Land theme/token work (Workstreams 1–2) before refactoring individual screens to minimise churn.
- Coordinate with backend/domain owners when promoting dynamic data sources to avoid breaking existing flows.

Validation Checklist

- [ ] `flutter analyze` passes with no new warnings.
- [ ] `flutter test` covers critical widgets/providers affected by the refactor.
- [ ] Visual QA on light/dark mode and small/large devices for each major screen.