■ AI Prompt — Implement **OIDC Implicit** using existing V3 patterns (code-aware)

Goal: Implement (or finish hardening) the **OIDC Implicit** flow so it is visually and behaviorally identical to our V3 flows—**without duplicating logic**. Reuse existing components, hooks, and utils from this repo (see file paths below). Keep Implicit behind a feature flag and ship with full logging, validation, and tests.

> Note: We know Implicit is legacy vs. Auth Code + PKCE. We still need it for parity/testing. Enforce strict `state`/`nonce`, hash-only parsing, and JWKS verification via `jose`.

---

0) Guardrails & Parity

- **Visual/UX parity** with V3 flows: - Reuse `styled-components` theme + shared components (stepper, buttons, toasts, status bar). - Match copy tone, tooltips, and step order. - **Reuse > duplicate**: - Prefer extracting shared bits into `src/utils/*` over cloning. - **Unified logging**: - Use `src/utils/logger.ts` with module tags + emojis. - **Config resolution order**: - `.env` → `settings.json` → `localStorage` via `src/services/config` + `src/utils/credentialManager`. - **Hardening**: strict validation, graceful errors, idempotent callback.

---

1) Files to (Re)use & Where to Plug In

- **Context / Session** - `src/contexts/NewAuthContext.tsx` (token presence, session, helpers) - **Flow Pages & Callback** - Start/Flow page: `src/pages/flows/ImplicitFlowOIDC.tsx` (already scaffolded) - Callback page: `src/components/callbacks/ImplicitCallback.tsx` - **UI Kit** - `src/components/StepByStepFlow.tsx` (stepper) - `src/components/TokenDisplay.tsx`, `src/components/ColorCodedURL.tsx`, `src/components/ConfigurationButton.tsx`, `src/components/PageTitle.tsx` - **Config & Discovery** - `src/services/config` (central config) - `src/services/discoveryService.ts` (OIDC metadata, jwks_uri) - `src/config/pingone.ts` (PingOne env helpers) - **Flow Utilities (reusable)** - `src/utils/oauth.ts` (`jose` helpers, randoms, PKCE, etc.) - `src/utils/callbackUrls.ts` (`getCallbackUrlForFlow('implicit')`) - `src/utils/tokenStorage.ts` + `src/utils/storage.ts` (consistent token storage) - `src/utils/tokenHistory.ts`, `src/utils/tokenLifecycle.ts` (status/expiry) - `src/utils/flowConfiguration.ts`, `src/utils/flowConfigDefaults.ts` (step metadata) - `src/utils/secureJson.ts`, `src/utils/urlValidation.ts` - `src/utils/logger.ts` (■ required) - **Types** - `src/types/*` (oauth/auth/storage/errors)

---

2) Routes & Navigation

- Ensure routes exist and are registered: - **Start**: `/flows/implicit` → `ImplicitFlowOIDC.tsx` - **Callback**: `/callbacks/implicit` → `ImplicitCallback.tsx` - Compute redirect via: - `getCallbackUrlForFlow('implicit')` from `src/utils/callbackUrls.ts`

---

3) Functional Spec

3.1 Start / Build Authorize Request (`ImplicitFlowOIDC.tsx`) - Required inputs (validate like V3): - `authorization_endpoint`, `client_id`, `redirect_uri`, `scope` - Defaults: - `response_type`: `"id_token token"` (toggle to `"id_token"` supported) - Scopes: from `flowConfigDefaults` or `config` (e.g., `openid profile email`) - Generate & persist **`state`** and **`nonce`**: - Use secure random from `src/utils/oauth.ts` (e.g., `generateRandomString`) - Persist alongside timestamp; one-time use - Construct authorize URL (encode & log): - Params: `client_id`, `redirect_uri`, `response_type`, `scope`, `state`, `nonce`, optional `prompt=login` - Use `StepByStepFlow` with identical CTA/buttons/spinner behavior as V3 pages. - **Log examples** (use logger): - `[■ OIDC-IMPLICIT] Building authorize URL…` - `[■ OIDC-IMPLICIT] state/nonce ready len={n}`

3.2 Redirect to OP - Disable controls + show spinner during navigation (same as V3).

3.3 Callback Parsing & Validation (`ImplicitCallback.tsx`) - Parse **`window.location.hash`** only: - Read `id_token`, `access_token?`, `token_type`, `expires_in`, `state`, `scope` - Validate: - **state**: must match pending; then invalidate (one-time) - **id_token**: required - **nonce**: decode JWT and compare `nonce` claim - **aud/iss/exp/iat/azp**: verify against discovered metadata and config - Verify signature with `jose`: - `createRemoteJWKSet(jwks_uri)` + `jwtVerify` (reuse in `src/utils/oauth.ts` or add a `verifyIdToken` helper there) - Store tokens via `src/utils/tokenStorage.ts`: - Keep absolute expiry; **no refresh_token** by design - Clear hash + `history.replaceState` after success - Surface result with same success/error cards as other callbacks.

3.4 Post-Auth UX - Token status panel (same as V3): - Use `TokenDisplay` + decode modal, expiry countdown - If `access_token` is present, allow sample API call using existing API client patterns - Status bar: show env ID, region, version (keep parity with V3).

---

4) Security & Hardening

- **Hash-only** token transport; never accept query params. - Exact `redirect_uri` match; enforce HTTPS in prod. - **state/nonce**: cryptographically strong, one-time, timestamped; purge on use/timeout. - Clock skew tolerance consistent with V3 (2–5 min). - Handle OP key rotation (unknown `kid` → refresh JWKS and retry). - CSP: avoid `unsafe-inline`; restrict origins. - Feature flag: `config.oidc.implicit.enabled` (default **off** in prod).

---

5) Code Reuse (concrete refactors)

Create or extend small, shared helpers (in `src/utils/`):

- `buildAuthorizeUrl(base: BaseAuthz, opts: { responseType: string; scope: string[]; state: string; nonce?: string })` - `stateNonce` service: `create()`, `verifyAndConsume()` (backed by session/local storage) - `verifyIdToken(idToken: string, expectations)` wrapping `jose` + discovery - `useCallbackProcessor(strategy: 'hash'|'query')` (if not present, keep simple util for hash) - Reuse `tokenStorage.put/get/clear(flowKey='implicit')` - Reuse `getCallbackUrlForFlow('implicit')` for redirect wiring

> Rule of thumb: If any new code would be ≥70% identical to an existing V3 utility, **extract** and inject differences via params.

---

6) Telemetry & Logging

All major stages emit logs via `logger`:

- Build URL: `[■ OIDC-IMPLICIT] authorize URL ready scopes=${scopes}` - Callback seen: `[■ CALLBACK] hash keys=${keys}` - Verify pass: `[■ VERIFY] id_token valid exp=${expIso} kid=${kid}` - Store: `[■ TOKEN] stored flow=implicit hasAccess=${!!access_token}` - Fail: `[■ VERIFY-FAIL] reason=${code} msg=${err.message}`

Keep entries emoji'd, timestamped, module-tagged, non-blocking.

---

7) Config Additions

Add a feature-flagged block (respect `.env` → `settings.json` → `localStorage`) via `services/config`:

```json { "oidc": { "implicit": { "enabled": true, "responseType": "id_token token", "scopes": ["openid", "profile", "email"], "nonceLength": 32, "stateLength": 32 } } } ```

---

8) Test Plan

- **Unit** - `buildAuthorizeUrl` produces correct query for both `"id_token"` and `"id_token token"`. - `stateNonce` one-time semantics (reuse across refresh). - `verifyIdToken` handles unknown `kid` → JWKS refresh. - **Integration** - Simulate OP redirect with `#` fragment; ensure UI shows decoded claims, expiry, and copy-buttons. - Negatives: wrong state, wrong nonce, expired token, bad issuer. - **E2E** - `/flows/implicit → OP → /callbacks/implicit → dashboard` happy path. - Refresh/idempotency on callback. - **Accessibility** - Focus order, aria labels, keyboard nav consistent with V3.

---

9) Acceptance Criteria

- [ ] **Exact** styling/copy parity with V3 step pages/components. - [ ] No duplicated business logic; shared utils added to `src/utils/*`. - [ ] Strict `state`/`nonce`; JWKS verification via `jose`. - [ ] Tokens stored via `tokenStorage` with visible Status Bar + decode modal. - [ ] Hash cleared post-process; back/refresh safe. - [ ] Feature flag `oidc.implicit.enabled` gates UI/route. - [ ] Unit + integration + E2E green; coverage ≥ V3 baseline.

Do all of the above using only the existing patterns in this repo—`NewAuthContext`, `StepByStepFlow`, `tokenStorage`, `discoveryService`, `callbackUrls`, and `logger`—so the Implicit flow “feels” like Authz v3 and stays maintainable.
```