Rust + Solana Advanced Developer Kit (PDK)

MODULE 1: Advanced Rust Mastery

Lifetimes (Deep Dive)

- Function-level lifetimes
- Struct lifetimes
- Lifetime elision rules
- 'static vs non-static lifetimes
- Common compiler errors with lifetimes & how to fix

Unsafe Rust & Memory Layout

- When to use unsafe
- Dereferencing raw pointers
- Memory layout of Rust structs (especially with repr(C))
- MaybeUninit and manual memory management

Smart Pointers & Interior Mutability

- Box, Rc, Arc, RefCell, Cell, Mutex
- When to use interior mutability in Solana programs
- Borrow / BorrowMut trait behavior

Traits & Generics

- Generic traits + trait bounds (where, impl)
- Associated types
- dyn Trait vs generic Trait bounds
- Blanket implementations

Macros

- Declarative macros (macro_rules!)
- Procedural macros (build your own derive)
- Using cargo-expand to debug macros
- Anchor macros dissection

Rust + Solana Advanced Developer Kit (PDK)

MODULE 2: Anchor & Solana Internals

Anchor Macro Internals

- What #[derive(Accounts)] actually generates

- How #[account(mut)] works

- How Anchor resolves PDAs under the hood

- Anchor error handling (#[error_code], require!, custom errors)

PDA & Seeds (Mastery)

- Best practices for seeds

- Manual PDA verification vs Anchor auto-verification

- Deterministic seeds for upgrade-safe logic

Account Serialization

- Borsh serialization (default for Anchor)

- Manual serialization with borsh::BorshSerialize

- Zero-copy deserialization (#[account(zero_copy)])

- Fixed vs dynamic account sizing

Cross-Program Invocations (CPI)

- CPI via CpiContext

- CPI to non-Anchor programs

- Handling signer authorities

- Transferring SOL, SPL tokens via CPI

System Programs & CPI Utilities

- Using system_program::transfer

- Creating accounts via CPI

- Using Token Program via CPI

- Close accounts securely

Rust + Solana Advanced Developer Kit (PDK)

MODULE 3: Real-World Smart Contract Design

Upgradeable Program Patterns

- Anchor upgrade flow
- Program ownership / buffer management
- Secure upgrade pattern (e.g. with multisig guard)

Advanced Access Control

- Multi-signer authority
- Role-based access (admin/user structure)
- Timelocks or cliff periods
- Off-chain signature verification (secp256k1_program)

Real Project Architectures

- Vault pattern (NFT/token locking)
- Escrow w/ dispute resolution
- Vesting contracts (linear, cliff-based)
- DAO execution contract (proposal + execution queue)

Testing & Debugging

- anchor test + local test validator
- Writing integration tests with multiple accounts
- Using solana logs + anchor test --skip-local-validator
- Using cargo-expand and RUST_LOG for deep debugging

BONUS: Dev Tools Checklist

Tool	Purpose
cargo-expand	View macro-expanded code (great for Anchor macros)
cargo-audit	Check for vulnerable crates
solana-test-validator	Local cluster for full program testing
solana logs	Debug transactions live

Rust + Solana Advanced Developer Kit (PDK)

anchor idl fetch - Fetch deployed program IDL

secp256k1_program - Off-chain signature verification in on-chain logic