Pre-Class Discovery Handout: Composability Business Models

Activity 1: Business Model Canvas Detective for a Composable Primitive

Scenario: Pick ONE composable on-chain primitive — a lending protocol such as Aave, an automated-market-maker such as Curve, a yield aggregator such as Yearn, a yield-stripping protocol such as Pendle, or a restaking primitive such as EigenLayer. Fill in the Business Model Canvas below by investigating how the protocol actually creates and captures value. Look beyond marketing copy; concentrate on the mechanics of liquidity attraction and integration.

Canvas Element	Your Analysis
Value Proposition *What composability friction does this primitive remove?*	
Customer Segments *Depositors? Borrowers? Other protocols above?*	
Channels *How does the primitive reach users it never meets directly?*	
Revenue Streams *Type of fee accrual (not amounts)?*	
Key Resources *What does the protocol need that a fork cannot copy?*	

Q1: What is the single most important composability friction that this primitive removes for protocols sitting above it?

Q2: Which other protocols depend on this primitive — and which protocols does it depend on?

Q3: If a perfect technical fork shipped tomorrow, what specifically would the original keep that the fork would lack?

Activity 2: Cross-Protocol Stack Map

Scenario: A composable stack works because lower primitives feed higher ones. Match each protocol to the layer it occupies, then trace which lower layer it consumes. After completing the table, sketch a simple arrow diagram showing dependency direction.

Protocol	Layer	What lower layer does it consume?
Aave	_____	_____
Curve	_____	_____
Yearn	_____	_____
Pendle	_____	_____
EigenLayer	_____	_____

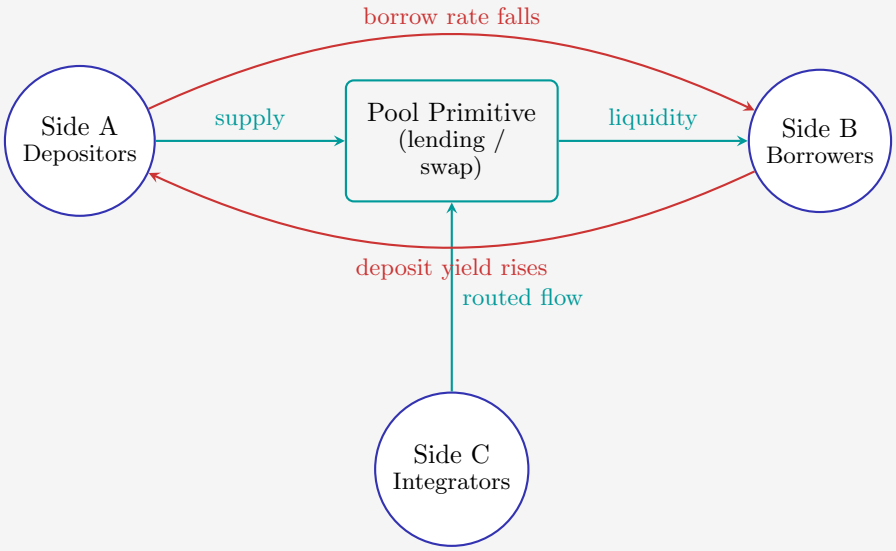
Q1: For each pair, describe in one sentence what the higher layer would lose if its lower layer were removed.

Q2: Which of these protocols ships products on top of its original wedge — and which adjacent products has it added?

Q3: Why might a protocol that begins as a single primitive eventually want to layer adjacent products of its own on top?

Activity 3: The Three-Sided Platform Puzzle

Scenario: A composable lending pool connects three sides of a market: depositors who supply capital, borrowers who consume it, and integrators (aggregators, wallets, structured-product protocols) who route flow into the pool from above. None of the three sides finds the protocol useful unless the other two are present.



Q1: Why does a protocol with deeper deposits attract more integrators (and vice versa)?

Q2: The cold-start problem in three-sided composability: which side should the protocol attract first, and why?

Q3: Once the protocol reaches critical mass, why is it hard for a perfect technical fork to peel any of the three sides away?

Solutions

Activity 1: Business Model Canvas Detective

A1: **Model answer for Aave:** The most important friction Aave removes for upstream protocols is the cost of building a permissioned lending engine from scratch and bootstrapping its own liquidity. Aave gives any aggregator, treasury bot or wallet a single address that already holds depositors and borrowers, exposes a stable interest-rate function, and has a years-long audit history.

A2: Aave is consumed by yield aggregators, structured-product protocols, treasury management bots, and front-end wallets. It depends on settlement on a base-layer chain (and on price oracles for non-stablecoin collateral). The primitive does not market to end users directly; the integrators above it do.

A3: A perfect code fork would lack the deposit base, the integration relationships hard-coded in higher protocols, the accumulated audit trail, the oracle adapters that have been tuned over years, and the governance-token holder community whose attention it depends on. The fork would be a clean contract with empty pools and no consumers.

Canvas elements (Aave):

- **Value Proposition:** A default, audited lending pool with deep liquidity that any upstream protocol can integrate against.
- **Customer Segments:** Primary — depositors and borrowers; secondary — integrating protocols (aggregators, wallets, structured products).
- **Channels:** Other protocols (the integrator layer), wallet front-ends, governance forums.
- **Revenue Streams:** Interest-rate spread between supply and borrow, reserve-factor accrual to the protocol treasury, governance-token value flow tied to protocol-owned activity.
- **Key Resources:** Deposited liquidity, integration relationships, oracle adapters, audit history, the governance-token distribution.

Activity 2: Cross-Protocol Stack Map

A1: Aave → Manufacturing primitive (sits on the settlement layer; consumes oracle feeds). Curve → Manufacturing primitive (sits on the settlement layer; consumes oracle feeds). Yearn → Aggregator (consumes Aave, Curve and other lending or AMM primitives). Pendle → Aggregator (consumes Curve for principal-token pricing and yield-bearing primitives such as Aave or EigenLayer for the underlying yield). EigenLayer → Restaking primitive (consumes the base-layer stake of the host chain).

A2: Yearn ships curated, synthetic, levered and partner vaults on top of its original yield-routing wedge. Pendle has added trading rails and partner integrations beyond the original split-yield trade. Aave has rebundled into multiple chains, an isolated-pool architecture, and a stablecoin issuance product. Each protocol illustrates rebundling on top of its own primitive once liquidity is captured.

A3: A single-product primitive earns a thin slice of fees and is exposed to fee-switch politics from below and fork pressure from above. Layering adjacent products raises lifetime fees per depositor and creates non-portable switching costs that compound the original liquidity moat. Rebundling converts a thin-fee primitive into a multi-product storefront.

Activity 3: The Three-Sided Platform Puzzle

A1: This is a **three-sided cross-network effect**: more deposits make borrowing cheaper, which attracts more borrowers; more borrowers raise the deposit yield, which attracts more depositors;

deeper liquidity makes the pool a more reliable integration target, which attracts more aggregators; more aggregators route still more deposits and borrows back into the pool. Each side's growth reinforces the other two.

A2: Most successful composable primitives attract the **depositor side** first, often by directing governance-token incentives at early supply. The logic: borrowers and integrators will follow the deepest pool, but neither will join an empty one. Attracting the integrator side first is harder because integrators need confidence that the liquidity will still be there in twelve months.

A3: A perfect technical fork enjoys none of the three sides' presence. A new entrant would need to simultaneously attract depositors (who see no yield in an empty pool), borrowers (who see no liquidity in an empty pool) and integrators (who refuse to wire to a primitive that may not exist next year). The incumbent's three-sided utility grows with every additional participant; the fork has to start the cold-start sequence from zero on all three sides at once.

