Post-Class Summary: Open Banking Business Models

Key Frameworks

Business Model Canvas Applied to Open Banking Aggregators

The Business Model Canvas decomposes any venture into nine interlocking blocks. For an open-banking aggregator, four blocks do most of the work — Customer Segments, Key Partners, Revenue Streams, and Channels — while the other five look structurally ordinary. The Customer Segments block identifies a developer or enterprise-fintech buyer who pays for access to someone else's data. The Key Partners block captures licensed banks whose participation is regulatory rather than commercial; those partners rarely want the aggregator to succeed. The Revenue Streams block monetises via per-call metering, tiered subscription, or bundled premium-data overlays. The Channels block is a developer portal with SDKs and documentation, not a retail marketing surface. The canvas makes the aggregator's structural oddity visible: the paying customer and the consenting account-holder are different people, and the supplying bank would often prefer that neither existed.

Platform Economics Applied to Open Banking Aggregators

An open-banking aggregator is a two-sided platform with an unusual feature: the supply side is mandated rather than commercial. Licensed banks participate because regulation requires them to, not because they are paid to. On the other side are developers and downstream fintechs that must be cultivated. Cross-side effects exist — more developer adoption raises the aggregator's leverage in bank partnership discussions, more bank coverage makes the aggregator more useful to developers — but they operate on top of a mandated floor rather than a commercial one. The platform challenge is therefore different from a payments network's: the aggregator's strategic lever is not which side to seed first, but how to convert mandated supply into cultivated demand before competitors standardise the connector layer into a utility.

Unbundling-Rebundling Applied to Open Banking Aggregators

Christensen's disruption framework explains both the entry and the trajectory of an open-banking aggregator. Entry proceeds by unbundling one service the bank bundled with every other: raw data access. Over time, successful aggregators rebundle — they add adjacent products once the consent-and-carriage machinery exists, because the marginal cost of a second product through the same consent primitive is low. The characteristic sequencing for an aggregator is data-adjacent products first (account information, balance snapshots, categorisation), payment-adjacent products next (payment initiation, recurring payments, refunds), and vertical-specific packages last (e-commerce checkout, fraud overlays, identity binding). Each wave reuses the consent primitive and funds the operational capacity the next wave needs; the product ordering is the business model.

Value Chain Deconstruction Applied to Open Banking Aggregators

Evans and Wurster argued that information-rich value chains deconstruct when digital coordination reduces the cost of operating across firm boundaries. The open-banking value chain — connector supply, data normalisation, consent orchestration, product packaging, go-to-market distribution, support and observability — is a textbook case. An aggregator typically rents the regulated-supply link (from licensed banks) and the shared-infrastructure link (SaaS observability) while owning the customer-surface links: normalisation, consent orchestration, packaging, and go-to-market. The inversion — owning the customer surface while renting the regulated supply — is the aggregator's structural answer to mandated-platform dynamics. The margin profile depends on which links are owned; the durability of the moat depends on the engineering depth that the owned links accumulate.

Regulatory Arbitrage Applied to Open Banking Aggregators

Most open-banking aggregators begin life in a regulatory classification that gives them a cost or speed advantage over aggregators who also serve end-users. Operating as a pure-connectivity provider or as a technical-services vendor avoids the full stack of consumer-protection duties a branded, end-user-facing aggregator would bear. The arbitrage is always temporary — regulators eventually widen the perimeter, as they should. The strategic question is whether the aggregator converts its head start into durable capability: engineering investment in connector resilience, data-schema normalisation, and consent-orchestration at scale. When the classification gap closes, the engineering stack built during the window itself becomes the barrier to the next entrant. Arbitrage that is not converted is merely subsidy; arbitrage that is converted becomes a moat.

Company Cases Summary

Company	Value Creation Mechanism	Key Framework	What Makes It Different
Plaid	Wide developer footprint with premium-data overlays stacked on top of raw connectivity	Platform Economics	Toll Booth today with a trajectory toward the Platform quadrant
TrueLayer	Deliberate product-stack arc: data-adjacent, then payments-adjacent, then vertical-specific products	Unbundling-Rebundling	Consent primitive is reused across waves; ordering is the business model
Tink	Owns data normalisation, consent orchestration, product packaging, and go-to-market; rents raw connector supply	Value Chain Deconstruction	Owns the customer surface, rents the regulated supply
Yapily	Narrow, pure-connectivity classification that avoids end-user-facing regulatory perimeter	Regulatory Arbitrage → Engineering Moat	Classification gap financed an engineering stack that outlasts the gap
Salt Edge	Cross-region operator; same connector stack earns different BM in mandated vs contract-access markets	Context Dependency	Template travels; host regime sets the monetisation mix

The Five-Test Framework

Use these five tests to evaluate any open-banking aggregator:

1. **Friction test.** Identify the friction the aggregator claims to remove and verify that the friction actually costs developers money, not only convenience. A friction that is merely cosmetic does not survive a bank shipping its own direct API at portfolio scale.
   Application: Plaid removes the portfolio-level friction of maintaining connectors to many banks; the friction is real because no single bank's direct API eliminates it.

2. **Platform test.** Determine whether the aggregator benefits from cross-side effects that tighten over time: does developer adoption reinforce bank-partnership leverage and vice versa, or does each new bank require a separate enterprise negotiation?
   Application: Plaid's developer footprint creates real cross-side leverage in bank-partnership discussions, but only in markets where direct-API competition is limited.

3. **Rebundling test.** Assess whether the product ordering is deliberate (data first, payments next, verticals last) or opportunistic. Opportunistic ordering signals weak operational capacity ahead of readiness.
   Application: TrueLayer's sequencing illustrates the disciplined arc; each wave reuses the consent primitive and funds the operational overhead the next wave will require.

4. **Infrastructure test.** Ask whether the aggregator adds infrastructure that the mandated-supply side cannot efficiently host (schema normalisation at scale, consent orchestration across banks) or merely a thin wrapper around the bank's own APIs. Addition is durable; thin wrapping is a race to commoditisation.
   Application: Tink adds data-normalisation and consent-orchestration infrastructure that individual banks have no economic incentive to build at cross-bank scale.

5. **Arbitrage test.** Evaluate whether the regulatory classification the aggregator exploited is being converted into an engineering moat — connector resilience, uptime monitoring, consent-flow capacity — or whether the classification is merely closing under the aggregator's feet.
   Application: Yapily illustrates successful conversion; a bank-agnostic pure-connectivity vendor that ran its narrow-classification window without building the engineering stack ends up either acquired or shut down.

Connections to Other Topics

The open-banking aggregator business model connects directly to several other course themes. Neobank business models rely on the same API-platform primitives to build their non-branch distribution, so the unbundling lens here complements the neobank rebundling lens in an adjacent package. Embedded-finance business models flip the question: where an aggregator sells access to bank data wholesale to developer applications, a banking-as-a-service provider sells banking capability wholesale to non-bank brands, and the margin split between brand-owner and rails-owner in embedded finance mirrors the connector-ownership split here analytically. Finally, the regulatory-arbitrage test developed in this package links directly to the RegTech material in the risk and regulation lesson: the compliance and engineering apparatus that converts arbitrage into moat is precisely the category of investment that RegTech vendors are selling, and the aggregators that convert best tend to be those that buy and integrate those vendors earliest.

