Decent Monetization

Pillar	Kaizen
Priority	High
Status	Canceled
Source	CEO
Created by	Nicolaus Sherrill
Linear Project	https://linear.app/decent-labs/project/decent-app-monetization-6cdc24995c68/overview
PM	Ⓒ Che Bajandas
Size	Large

Overview

This project will focus on monetization of the **existing Decent product**, which at the time of writing this is essentially everything aside from the Token Sales feature set.

Monetization will happen through a freemium model, with features assigned to various subscription tiers, the lowest being free. At a high level, users need to be able to subscribe to paid tiers through the app settings which in turn unlock features that have been gated behind a paywall.

> 💡 NOTE: The scope of this feature has been adjusted to take a phased approach. Keep a careful eye out for features that are anticipated to be reserved for v2.

Reference

-

Objectives

- Categorize existing features of the Decent app using tiers, ranging from free to enterprise
- Enable a subscription system that:
  - Allows users to subscribe to a paid tier of their choice for a designated time period
  - Allows users to submit payment according to the chosen tier and based on Decent's payment requirements
  - Allows Decent to collect subscription fees
  - Unlocks features in the app for paid users according to the subscription tier
- Update the app to allow for features to be disabled unless the appropriate subscription tier is confirmed as paid for
- Provide a clear UX indicating when a feature is disabled because of a paywall that nudges users towards a subscription

Terms

- **Subscription**: What a DAO signs up for / pays for. Indicates a paid for period of time where services are provided, in this case, the availability of premium app features.

- **Tier**: The choice of subscription level. Defines which premium features are unlocked. A tier includes all features up to that tier, including all features unlocked by lower tiers.
- **Feature**: A capability of the app which may be a portion of the UX or an entire page. This is being used in this context to divide the app experience into subscription tiers. EX: “roles” might be a feature, but more granularly, payments on a role might be a feature. This is more about how we want to bundle app capabilities together than a technical reference.
- **Payment**: In this context, covering the cost of a subscription. Essentially, how customers pay Decent for subscriptions / premium features. Alternative: “Checkout”
- **Paywall**: The in app experience of feature gating, or the locking of features, depending on subscription status.

User Stories

> 💡 The following is scoped to V2 of this project; some aspects will not be included in the initial launch or will be maintained differently than described here.

New User

As a new user, I begin using the Decent app to set up a DAO for my project using the default free options that allow me to do so. Thanks to the basic setup and use of a DAO for simple governance, I am able to get a feel for how the app works, see the potential of it, and decide if I’d like to get more out of it.

When I visit certain pages in the app, I am presented with a page that has a clear indication that the contents of the page are only available to users on a paid plan. The design of the walled off page gives a clear picture of what the feature is that I am not currently able to access and offers a CTA that directs me to the relevant place in the Settings to sign up for a subscription unlocking the content.

From the Settings for my new DAO, I see a tab that is clearly indicated as for setting up a subscription. On the subscription management tab of the Settings, I see the subscription tiers and that I currently am “signed up” for the “free”/”basic” tier. Each tier in the list gives a brief description of the features made available by selecting that subscription tier as well as the price. There’s also a concise, but complete list of each feature that is unlocked by subscribing to that tier.

After selecting a tier and submitting, I am presented with payment options that allow me to set up a one time or recurring payment to Decent that will keep the selected tier active, with the relevant features unlocked, for my DAO until the paid for period elapses, the end of a payment stream, or at the end of the current paid period when I cancel my recurring payments.

When I view the DAO, and when my members and fellow governors view the DAO, the paid features that I unlocked with the subscription are visible. In fact, the paid features are visible to anyone who visits the DAO pages, though naturally use of the features is gated as normal behind whether or not the viewer has proposal permission on the DAO. If/when I stop paying for the subscription tier I signed up for, I am reverted automatically to the free/basic tier and access to the paid features is gated once again. At any time I can upgrade to a higher tier or sign up for a tier I stopped paying for again to regain access to those features.

Existing User

As an existing user, I am used to having access to all of the features available on Decent. I may not use all of them, but I’ve never seen any sort of pay wall before, nor mention of subscriptions.

If I *have* a relationship with the Decent team as an existing paying customer or otherwise with direct access to the development team, I am informed about upcoming changes that will lock some features behind a pay wall and subscription plan. I am given ample time to prepare for such a change and may work closely with the sales team at this stage if I am not seeing a subscription tier that matches my needs in terms of feature offerings and/or price.

If I *don’t have* a relationship with the Decent team and I have been using the Decent app entirely in a self service way, then I will learn about upcoming changes through social media or support channels and through the new paywall UX.

Decent Administrator

As someone working at Decent, I have the option to enable or disable a subscription for a client, effectively bypassing need for the client to go through the subscription activation process in app. This helps in special cases where a different deal is reached with a client than what is offered through the app interface, such as a lower price point for a subscription tier or a demo / promotional period.

▼ [V2] Payment Structure

How we expect to charge our customers and receive payment.

- Payment frequency
  - Monthly, recurring payment
  - Yearly, one-time payment, at discounted rate
- Payment source
  - Individual wallet (EOA) → immediate transaction
  - Project treasury → goes to proposal
- Payment method
  - Crypto only for MVP
  - USDC only to start (”contact sales” option as fallback)

▼ [V2] Payment Management Breakdown

We’ll need to be clear on the expected behavior in a variety of potential scenarios / use cases with regard to managing the current subscription tier and payment method.

> 💡 The following assumes that when a new subscription starts, as specified by the conditions of the subscription update, any previously scheduled recurring withdrawals are cancelled and superseded either by new recurring withdrawals or a lump sum yearly payment.

Subscription Upgrades, Downgrades, and Extensions

> 💡 Subscriptions always end at *the end of the day* of the last day, or more precisely, 12AM the following day.

> 💡 Where required, the time zone for determining start and end times/dates should be: **TBD**

> 💡 Cancellation is achieved by downgrading to the free tier and follows the same logic as other downgrade options, but without an associated payment.

Scenario	Start	Duration	End	Payment Type	Cost	Cost Formula
New monthly subscription (no currently active subscription)	Day of first payment	User specified months	User specified months from start	Recurring	Exact rate from pricing table per month	$monthlyRate$
New yearly subscription (no currently active subscription)	Day of payment	User specified years	User specified years from start	One time	Exact rate from pricing table per year	$yearlyRate * numberOf$

Scenario	Start	Duration	End	Payment Type	Cost	Cost Formula
Update from paid monthly to new payment or approval amount - same subscription tier	Day of change	User specified months	User specified months from change date	Recurring	Exact rate from pricing table per month	$monthlyRate$
Upgrade from paid monthly to higher subscription tier, monthly	Day of upgrade	User specified months	User specified months from upgrade date	Recurring	Prorated credit for unused time on old plan, then new rate	Credit: $((oldMonthlyRate/days$ $daysRemainingInCycle)$ Then charge: $newMonthlyRate$
Downgrade from paid monthly to lower subscription tier, monthly	End of current billing cycle	User specified months	User specified months from downgrade date	Recurring	New rate starts at next billing cycle	$monthlyRate$
Upgrade from paid yearly to higher subscription tier, yearly	Day of upgrade	User specified years	User specified years from upgrade date	One time	Prorated credit for unused time, charge for new plan	$((newYearlyRate * num$ $((oldYearlyRate *$ $numberOfYears)/daysI$ $daysRemainingInTotali$
Downgrade from paid yearly to lower subscription tier, yearly	End of current paid period	User specified years	User specified years from downgrade date	One time	New rate at end of current period	$yearlyRate * numberOf$
Update from paid monthly to yearly payment - same or higher tier	Day of change	User specified years	User specified years from change date	One time	Prorated credit for current month, charge for yearly	$((yearlyRate * numberO$ $((currentMonthlyRate/($ $daysRemainingInCycle)$
Update from paid monthly to yearly payment - lower tier	End of current billing cycle	User specified years	User specified years from end of cycle	One time	New yearly rate at end of monthly cycle	$yearlyRate * numberOf$
Update from paid yearly to monthly - higher tier	Day of change	User specified months	User specified months from change date	Recurring	Prorated credit for remaining yearly period, then monthly charges	Credit: $((yearlyRate *$ $numberOfYears)/daysI$ $daysRemainingInTotali$ Then charge: $monthlyRate$
Update from paid yearly to monthly - lower tier	End of current paid period	User specified months	User specified months from end of period	Recurring	New monthly rate at end of yearly period	$monthlyRate$

▼ Variable Definitions

Subscription Rate Variables

- `monthlyRate`: Monthly subscription rate from pricing table
- `yearlyRate`: Yearly subscription rate from pricing table
- `numberOfYears`: Number of years being purchased upfront

- `newMonthlyRate` / `oldMonthlyRate`: Rates for tier changes

Time Period Variables

- `daysInTotalPaidPeriod`: Total days in the paid yearly/multi-year period
- `daysRemainingInTotalPaidPeriod`: Days remaining until end of paid period
- `daysInBillingCycle`: Days between monthly payment dates
- `daysRemainingInBillingCycle`: Days until next monthly billing date
- `daysInStartMonth`: Days in calendar month when subscription starts
- `daysRemainingInStartMonth`: Days remaining in calendar month from start
- `daysInMonth`: Days in current calendar month
- `daysRemainingInMonth`: Days remaining in current calendar month

[V2] Payment Source Changes

Users may elect to change from paying using the project treasury to an EOA or from an EOA to the project treasury separately from upgrading/downgrading.

This only really applies to recurring payments and there shouldn't be a need to change anything other than the payment wallet address and possibly approval amount.

- For project treasury → EOA: the user will will need to approve whatever remaining amount is left in the current subscription. Optionally, they should be allowed to approve above that amount and extend the subscription.
- For EOA → project treasury: this will need to lead to a new proposal where the project approves the remainder of the subscription burden. Same as above, this can also allow for extending the subscription.

Payment Tiers

TBD: Good place to fill in the working titles of the tiers and start considering how we'd describe them in short form copy / market them.

- 🥉 Free
- 🥈 Tier 1 Subscription - "Pro"
- 🥇 Tier 2 Subscription - "Ultimate"
- 💜 Enterprise

Tier Offering Breakdown

Good place to fill in more user friendly/marketable terms and summarize the selling points of the features we're offering.

> 💡 Each tier essentially unlocks everything listed under that tier and all features from the tiers below it, i.e., Tier 1 includes all Free features and Tier 2 includes all Tier 1 and Free features.

Outline of existing app features and how they're expected to breakdown according to tiers. This can be a place to explore what we consider to be the "selling points" for each feature to better emphasize the draw for each in our in app tier breakdown.

Subscription Tier Feature Breakdown

Aa Feature	Tier	Pillar	Selling Points
Governance Deployment	Free	Deployment	

Feature	Tier	Pillar	Selling Points
Governance Token Deployment	Free	Deployment	
Governance Token Import	Free	Deployment	
Global Permissions	Free	Governance	
Treasury Visualization	Free	Governance	
Proposal Markdown Editor	Free	Governance	
Transaction Builder	Free	Governance	
Batched Transactions	Free	Governance	
Custom Templates	Free	Governance	
Mobile Voting	Free	Governance	
Snapshot Integration	Free	Governance	
Transfers	Free	Payments	
Sub DAOs	Tier 1	Governance	Modular governance: veto, freeze, clawback.
Roles - Permissions	Tier 1	Governance	More granular control of proposal permission
Roles - Elections	Tier 1	Governance	Set role term limits
Roles - Streams	Tier 1	Payments	Pay team members or vendors
Roles - Claiming	Tier 1	Payments	Payment configuration and payee claiming all in app
Streaming	Tier 1	Payments	Stream funds from the DAO through a proposal, all in app
Payroll	Tier 1	Payments	
DeFi Integrations (dApp Explorer)	Tier 1	Integrations	
Gasless Voting	Tier 2	Governance	Reduce friction for voters; increase community participation
Additional Token Deployments	Tier 2	Tokenization	
Airdrops	Tier 2	Tokenization	
Revenue Share	Tier 2	Tokenization	
Staking	Tier 2	Tokenization	
Sponsored Transactions	Tier 2	Tokenization	
DAO Design	Enterprise	Governance	
White Label Usage	Enterprise		
Custom Features	Enterprise		

Nodes (AKA SubDAOs)

The behavior of nodes in the subscription model is something that will require some special consideration. Take careful note of the following:

- From the parent side, the paywall will prevent the creation of new nodes or management of existing nodes

- From the node side, the paywall will be represented by putting those nodes into a “read-only” state with a warning banner type UX
  - As mentioned below with Roles, we should avoid locking down the payment ‘claim’ CTA as part of this
- Nodes themselves will not have the option to set up a subscription
- The features availability of a node is dependent on its parent: a node inherits the subscription tier of the parent project

Roles

Similar to nodes, there’s a good argument to make that the Roles page shouldn’t be locked down entirely when a subscription lapses. In particular, if there are payments on a role, we don’t want to punish the recipient and block them from claiming funds.

We’ll treat the Roles page in this way:

- If no roles exist, show the normal full page paywall treatment
- If roles exist, have a (mostly) read only displayed with warning banner
  - The ‘claim’ action for payments should remain active in this state
  - All other actions will be either hidden or inactive

▼ [V2] Negative Cases

User subscribes with EOA while there is a pending proposal to subscribe with project treasury

Until such a time as a project is registered as having an active subscription, there will continue to be the option to set up payment for a subscription in the settings panel. This means that users have the option of either:

1. Setting up an EOA payment for a subscription while there’s a pending proposal to set up a subscription using the project treasury
2. Putting up a new proposal to subscribe using the project treasury when there’s another pending proposal of the same type

Either case may result in an executable proposal which will fail because there is already an active subscription. This is expected and shouldn’t come as a shock to the user(s) involved.

A minimal way that we can make this more transparent to users would be to ensure that there’s a clear error toast message when a subscription proposal fails to execute for this particular reason.

Recurring payment fails unexpectedly

In such a case where the transaction to withdraw funds from the designated source fails unexpectedly, we may want to retry automatically a certain number of times before considering it fully failed.

At such a point as we’ve determined that a payment is not going to succeed and it is not for a standard reason, we’ll want to flash a banner warning that payment is going to lapse. In all likelihood, the resolution for the user at this point is to reconfigure their subscription/payment.

Rather than immediately ending a subscription that has had a payment failure, we should offer a 7 day grace period to allow the project to sort out the issue without losing their paid features right away.

Recurring payment is expected to fail

On the other hand, a payment may fail for a predictable reason: either the payment source wallet is low on funds or the approval amount is about to be reached.

For cases where **the source wallet is low on funds**, we should treat this similar to the unknown payment failure and give the user a grace period to add funds to the wallet before ending the subscription outright. If at all possible, we should preempt this and warn users the month before an anticipated payment failure.

For cases where **the approval limit is about to be reached**, this is effectively the end of the subscription and should be treated as such. This shouldn't be a special case so much as it is what is projected to be the end of the subscription. The user should be warned and prompted to renew/extend their subscription starting on the last pre-approved month.

Features

> 💡 The following currently assumes that a subscription applies to the DAO.
> Whatever tier is paid for unlocks those features for anyone viewing the DAO, not on a per user basis, but on a top level DAO basis.

Subscription Status Backend

The backend will need to keep track of subscriptions. This will likely involve a table organized by DAO address. The backend will also need to provide an endpoint that transmits the subscription status for a particular DAO address.

We can anticipate that the backend subscriptions table and subscription "status" response will need to include at least all of the following:

- Which subscription tier is active
- How much time remains in the subscription (except for the "free" tier)
- [V2] Whether or not the payment source wallet can cover the cost of the next withdrawal

This status endpoint will be used by the app frontend to determine:

- Whether or not to display a paywall on a page by page basis
- Which subscription tier to display as active in the settings

▼ [V2] Subscription Payments Backend

Backend support for subscription payments and subscription status tracking.

> 💡 Recurring payments may come from either the DAO Treasury or an individual user's EOA. As such, the following needs to consider both possibilities: upgrades or downgrades should apply immediately for EOA payments, but need to go through a proposal for DAO Treasury payments.

The backend will need to support the following:

- Payments to Decent following proposed subscription structure
  - See
  - There will be a monthly recurring and yearly one time payment option
  - Monthly subscriptions will use a pre-approval amount to allow for Decent to withdraw from the source wallet each month, which indicates a set duration even for a recurring payment plan
  - Yearly subscriptions will be paid for upfront
- Should support cancelation of a recurring payment through the app interface
  - This may be represented in app as downgrading to the free tier
- Should support changing subscription tier, either upgrading or downgrading
  - Ideally, an upgrade would allow for getting access to the new tier immediately, meaning that there would be the option to pay the difference between the current tier price and the new tier price for the current month
  - Downgrades would happen at the month mark for recurring payments
  - One time payments/yearly subscriptions do not require a downgrade option

- Tracking of current payment status, indicating current subscription tier
- Toggle of feature availability based on current payment status

We also need to consider what happens when a recurring payment fails. If payment is not received at the designated monthly withdrawal time and the failure reason is unexpected, then there should be a a 7 day grace period before the subscription needs to be o treated as cancelled (downgraded to “free”).

This may mean that **we should be indexing the reason why a subscription changed**, anticipating that it could either be the happy path where the paid for period has elapsed, or the sad path where the source wallet ran out of funds or there was some unexpected error with the transfer transaction. Approval can also be revoked intentionally which is a forceful way of cancelling a recurring subscription payment.

Admin Backend

In addition to providing the necessary endpoints the Decent app front end will leverage, the backend will need to enable admin level interaction. Authenticated calls from outside of the app should be possible, bypassing the app interface to make updates to a subscription.

For MVP, this can be fairly sparse, since this gets complicated fast. Admins will need to input the following to remotely manage a subscription:

- A target DAO address
- Subscription tier to enable
- Start and end timestamps (subscription length)
    - Editing the end timestamp should allow for either ending or extending a subscription by pushing the timestamp into the past or farther into the future

For initial launch, this is the entire way that subscriptions will be managed. Customers will interface directly with sales, negotiate a price, and then once a payment stream or one time transfer has been set up/received, the team can input the relevant parameters into using the admin backend to enable the subscription.

This backend service that will allow for writing to the subscriptions table will be leveraged by the frontend in an internal dashboard.

Admin Page

We will need an admin page for internal use. For v1 of the project, this will be the only way that subscriptions are activated. For v2, this will be a place for the team to manage subscriptions for users in special cases.

This will not be in the Decent app and is only be intended to be used by Decent team members. The page will need to be gated behind some level of internal Decent account validation.

See the and sections above for details on what capabilities this will tie into. At a high level, here’s what would be useful to have on the page:

- A list of subscriptions with basic details displayed for each
    - DAO address
    - DAO name
    - Subscription status
    - Subscription tier
    - Subscription start/end time
- Option to edit an existing subscription
    - Set subscription tier
    - Set start and end timestamps

- Option to set up a new subscription
  - Set target DAO address
  - Set subscription tier
  - Set start and end timestamps

[V1] Subscription Page

A page in the app where users can view the subscription tier options and can be directed to contact sales to negotiate pricing and set up a subscription.

The frontend will need to support UX for the following:

- Subscription tab in the Settings modal
- Subscription tab contents
  - List of subscription tiers
  - Feature sets available per tier
  - Indication of which tier is currently active
  - How much time is remaining for the current subscription
- “Contact Sales” CTA
  - Each subscription tier column, except free, will have a CTA to contact sales
  - This should direct to Reclaim
    - If there’s a way to insert a preformatted message, that might be ideal, to make it super easy for the user to submit and including which tier they are interested in

▼ [V2] Subscription Page

A page in the app where users can view the subscription tier options and select a tier to upgrade to.

> 💡 Pricing may not be straightforward when it comes to upgrading, downgrading, or extending mid billing cycle. This may be a concern when building the UX for this section. See for more details on the possible scenarios.

The frontend will need to support UX for the following:

- Subscription tab in the Settings modal
- Subscription tab contents
  - List of subscription tiers
  - Feature sets available per tier
  - Indication of which tier is currently active
    - There will always be an active tier, since the basic one is free
    - Should show some indication of “time remaining”
      - For yearly, this can key off of the end of the paid for time period
      - For monthly, this can key off of the remainder of the approved payment amount
  - “Upgrade” and “Downgrade” CTAs for inactive tiers
- “Upgrade” CTA
  - Active when connected to any wallet
  - Leads to payment popup / checkout flow

- “Downgrade” CTA
  - Active if there’s a recurring payment set up
    - If the DAO is paying → Active when connected with wallet with proposal power
    - Otherwise → Active when connected wallet is paying
  - If the DAO is paying → Leads into proposal creation
  - If connected wallet is paying → Kicks off “downgrade” transaction
  - Emphasizes that features will be available until the end of the pay period
- “Contact Sales” CTA
  - We may want to provide the option for organizations to reach out to discuss alternative payment options or something else along those lines
  - This should direct to Reclaim

▼ [V2] Payment Page (”Checkout”)

An interstitial page/modal where users are able to fill in payment information in order to sign up for a selected subscription tier.

The frontend will need to support UX for the following:

- Interstitial that can be inserted between selecting a subscription tier and submitting a proposal or kicking off an EOA transaction
- Information display for the currently selected subscription tier
  - Tier name
  - Tier description
  - Tier cost
  - Payment token
    - This will be USDC to start
    - While the user can’t select a token, we should show which token the payment will use
- Options for the user to input:
  - Payment source
    - From connected wallet (EOA)
    - From DAO treasury
  - Payment frequency
    - One-time, yearly
    - Recurring, monthly
  - Pre-approval amount (monthly, recurring)
    - Configurable, up to a year
    - Based on subscription tier monthly cost
    - Adjustable by monthly cost increments
- Submit CTA
  - If the payment source is the DAO treasury → Leads into proposal creation
  - If the payment source is the user wallet/EOA → Kicks off transaction

Feature Paywalls

Features throughout the app will need to be gated behind paywalls which prevent their use and give clear direction on how to unlock them.

The frontend will need to:

- Check the DAO's subscription status when a paid feature is loaded
- Determine whether or not a feature is locked depending on the subscription status
- Display the appropriate UX depending on whether or not a feature is locked

The frontend will need to support UX for:

- A locked version of each feature/page that can be gated behind a paywall, including:
  - Some visual indication of what the feature looks like when not locked, such as a blurring or fading effect
  - A brief description explaining why the feature is locked and promoting its merits
  - A CTA that directs to the subscription page

The Roles and Organization paywalls will be more nuanced than the others listed:

- The Roles page will only be fully blocked when no roles exist
- When roles exist, the page will instead revert to a "read-only" mode when there's no active subscription, but with the 'claim' action for payments left active
- For the organization page and nodes, the paywall will block the creation of new nodes and the management of existing ones
- For nodes themselves, when there's not an active subscription that enables their use, they will be switched to a "read-only" mode where no actions can be used, except the roles 'claim' action

Tier 1 - "Pro"

- Organization page
  - See above note
- All node (subDAO) pages
- Roles page
  - See above note
- Proposal Templates → Stream
- Create Proposal → Use dApps → Explore dApps

Tier 2 - "Ultimate"

- Proposal Templates → Airdrop
- Settings → Sponsored Voting (Gasless Voting)
- Settings → Revenue Share
- Settings → Staking
- Staking page