```
──────────────── MODULE principal ────────────────
Redeclaration of specdatamodels variables
EXTENDS Sequences, Naturals, FiniteSets

CONSTANT USERS

CONSTANTS
    SubscriptionFee,
    CancellationFee,
    FailedPaymentFee

VARIABLES
    month,
    database,
    events

vars ≜ ⟨events, month, database⟩

INSTANCE specdatamodels

Now ≜ Len(events)

Months ≜ 0 . . 10

Strong Typing

Month ≜ Nat

Database Rows
VideoAccessRow ≜
    [type : {"enabled", "disabled"}] ∪
    [
        type : {"disabledasoftimestamp"},
        disabledtime : Month
    ]

TrialStatusRow ≜
    [type : {"eligible", "ineligible", "cancelled"}] ∪
    [
        type : {"started"},
        endtime : Month
    ]

SubscriptionStatusRow ≜
    [type : {"notsubscribed", "subscribed", "tocancel"}] ∪
    [
        type : {"tocancel"},
        canceltime : Month
    ]

$UpcomingChargeItem \triangleq [fee : Fees,\ event : Nat]$

$TypeOk \triangleq$
    $\land\ EventsOk$
    $\land\ month \in Month$
    Represents a singleton record type tracking the billing month
    $\land\ database.billingMonth \in Month$
    Table with VideoAccessRow definition
    $\land\ database.videoAccess \in [USERS \rightarrow VideoAccessRow]$
    Table with TrialStatusRow definition
    $\land\ database.trial \in [USERS \rightarrow TrialStatusRow]$
    Table with SubscriptionStatusRow definition
    $\land\ database.subscription \in [USERS \rightarrow SubscriptionStatusRow]$
    A row index, by user and month, that holds UpcomingCharges
    $\land\ database.upcomingCharges \in [USERS \rightarrow [Months \rightarrow \text{SUBSET}\ UpcomingChargeItem]]$
    Table with PastDueStatus row definition
    $\land\ database.inGoodStanding \in [USERS \rightarrow \text{BOOLEAN}\ ]$

API endpoints

 Database query:
$IsSubscribed(u) \triangleq$
    $\lor\ database.subscription[u].type = \text{“subscribed”}$
    A converted trial that hasn't been processed in database
    $\lor\ \land\ database.trial[u].type = \text{“started”}$
      $\land\ database.trial[u].endtime \leq month$

$StartSubscription(u) \triangleq$
    LET $paymentFailedCharges \triangleq$
          IF $database.inGoodStanding[u] = \text{TRUE}$
           THEN $\{\}$
           ELSE $\{[event \mapsto Now,\ fee \mapsto FailedPaymentFee]\}$
    IN


    LET $thisMonthsCharges \triangleq$
          $database.upcomingCharges[u][month] \cup$
           Add payment failed fee if applicable
          $paymentFailedCharges$
    IN

     Remove cancellation fee for next month
    LET $nextMonthsCharges \triangleq \{$
              $c \in database.upcomingCharges[u][month + 1]$ :
                $c.fee \neq CancellationFee$
    }
    IN
     Transaction precondition
     $\land \neg IsSubscribed(u)$
     Transaction
     $\land database' =$
        $[database$ EXCEPT
            $![$“inGoodStanding”$][u] =$ TRUE,
            $![$“subscription”$][u] = [type \mapsto$ “subscribed”$]$,
            $![$“trial”$][u] = [type \mapsto$ “ineligible”$]$,
            $![$“videoAccess”$][u] = [type \mapsto$ “enabled”$]$,
             Add charges
            $![$“upcomingCharges”$][u][month] = thisMonthsCharges$,
            $![$“upcomingCharges”$][u][month + 1] = nextMonthsCharges$
        $]$
     Observability required by stub
     $\land events' = Append(events, [type \mapsto$ “startsubscription”, $user \mapsto u])$
     $\land$ UNCHANGED $month$

$CancelSubscription(u) \triangleq$
    LET $cancelTime \triangleq month + 1$IN
    LET $cancellationFee \triangleq [$
                                $event \mapsto Now$,
                                $fee \mapsto CancellationFee]$
    IN
    LET $subscriptionFee \triangleq$
          $[event \mapsto Now$,
           $fee \mapsto SubscriptionFee]$
    IN
     Adds subscription fee for this month if not already added
    LET $thisMonthsCharges \triangleq$ IF $\neg\exists\, charge \in database.upcomingCharges[u][month]$ :
                                        $charge.fee = SubscriptionFee$
                                    THEN $database.upcomingCharges[u][month] \cup$
                                         $\{subscriptionFee\}$
                                    ELSE $database.upcomingCharges[u][month]$
    IN
     Transaction precondition
     $\land IsSubscribed(u)$
     Transaction
     $\land database' =$
        $[database$ EXCEPT
            $![$“trial”$][u] = [type \mapsto$ “ineligible”$]$,
            $![$“subscription”$][u] = [type \mapsto$ “tocancel”, $canceltime \mapsto cancelTime]$,
            $![\text{“videoAccess”}][u] = [\mathit{type} \mapsto \text{“disabledasoftimestamp”},$
                                          $\mathit{disabledtime} \mapsto \mathit{cancelTime}],$
            $![\text{“upcomingCharges”}][u][\mathit{month}] = \mathit{thisMonthsCharges},$
            Queue up cancellation fee for next month
            $![\text{“upcomingCharges”}][u][\mathit{month} + 1] =$
                $\mathit{database.upcomingCharges}[u][\mathit{month}] \cup$
                $\{\mathit{cancellationFee}\}$
        $]$
    Observability required by stub
    $\land\ \mathit{events}' = \mathit{Append}(\mathit{events}, [\mathit{type} \mapsto \text{“cancelsubscription”}, \mathit{user} \mapsto u])$
    $\land$ UNCHANGED $\langle \mathit{month} \rangle$

$\mathit{StartTrial}(u) \triangleq$
    LET $\mathit{endTime} \triangleq \mathit{month} + 1$IN

    Transaction precondition
    $\land\ \mathit{database.trial}[u].\mathit{type} = \text{“eligible”}$
    Transaction
    $\land\ \mathit{database}' =$
        $[\mathit{database}$ EXCEPT
            $![\text{“trial”}][u] = [\mathit{type} \mapsto \text{“started”},$
                                $\mathit{endtime} \mapsto \mathit{endTime}],$
            $![\text{“videoAccess”}][u] =$
                                $[\mathit{type} \mapsto \text{“enabled”}]$
        $]$
    Observability required by stub
    $\land\ \mathit{events}' = \mathit{Append}(\mathit{events}, [\mathit{type} \mapsto \text{“starttrial”}, \mathit{user} \mapsto u])$
    $\land$ UNCHANGED $\langle \mathit{month} \rangle$

$\mathit{CancelTrial}(u) \triangleq$
    Transaction precondition
    $\land\ \mathit{database.trial}[u].\mathit{type} = \text{“started”}$
    The trial has not already ended
    $\land\ \mathit{database.trial}[u].\mathit{endtime} > \mathit{month}$
    Transaction
    $\land\ \mathit{database}' =$
        $[\mathit{database}$ EXCEPT
            $![\text{“trial”}][u] = [\mathit{type} \mapsto \text{“cancelled”}],$
            $![\text{“videoAccess”}][u] =$
                                $[\mathit{type} \mapsto \text{“disabled”}]$
        $]$
    Observability required by stub
    $\land\ \mathit{events}' = \mathit{Append}(\mathit{events}, [\mathit{type} \mapsto \text{“canceltrial”}, \mathit{user} \mapsto u])$
    $\land$ UNCHANGED $\langle \mathit{month} \rangle$
```

```
Database query
$WatchVideoAuthorized(u) \triangleq$
    $\lor\ database.videoAccess[u].type = \text{“}\textsf{enabled}\text{”}$
    $\lor\ \land\ database.videoAccess[u].type = \text{“}\textsf{disabledasoftimestamp}\text{”}$
      $\land\ database.videoAccess[u].disabledtime > month$

$WatchVideo(u) \triangleq$
    $\land\ WatchVideoAuthorized(u)$
    Observability required by stub
    $\land\ events' = Append(events, [type \mapsto \text{“}\textsf{watchvideo}\text{”}, user \mapsto u])$
    $\land\ \text{UNCHANGED}\ \langle month, database \rangle$


$Bill(u, fee) \triangleq$
    $\land\ events' = Append(events, [type \mapsto \text{“}\textsf{bill}\text{”},$
                                      $user \mapsto u,$
                                      $fee \mapsto fee])$

$PaymentFailed(u, fee) \triangleq$
    $\land\ database' =$
        $[database\ \text{EXCEPT}$
            $![\text{“}\textsf{inGoodStanding}\text{”}][u] = \text{FALSE},$
            $![\text{“}\textsf{subscription}\text{”}][u] = [type \mapsto \text{“}\textsf{notsubscribed}\text{”}],$
            $![\text{“}\textsf{trial}\text{”}][u] = [type \mapsto \text{“}\textsf{ineligible}\text{”}],$
            $![\text{“}\textsf{videoAccess}\text{”}][u] = [type \mapsto \text{“}\textsf{disabled}\text{”}],$
             Remove all upcoming changes
            $![\text{“}\textsf{upcomingCharges}\text{”}][u][month] = \{\},$
            $![\text{“}\textsf{upcomingCharges}\text{”}][u][month + 1] = \{\}$
        $]$
    $\land\ events' = Append(events, [type \mapsto \text{“}\textsf{paymentfailed}\text{”},$
                                      $user \mapsto u,$
                                      $fee \mapsto fee])$
    $\land\ \text{UNCHANGED}\ \langle month \rangle$

Recurring Operations

This the the state that calls the Payment Failed $API$
$ExistingBillFailed \triangleq$
    $\lor\ \exists\, i \in 1 \,..\, Len(events) :$
        Only a past bill can fail
        $\land\ events[i] \in BillEvent$
        $\land\ PaymentFailed(events[i].user, events[i].fee)$

Trial users that have passed trial period are subscribed
$ConvertTrialUser(u) \triangleq$
    $\land\ database.trial[u].type = \text{“}\textsf{started}\text{”}$
    $\land\ database.trial[u].endtime \leq month$  The trial has ended
    Transaction
    ∧ database′ =
       [database EXCEPT
           ![“subscription”][u] =  [type ↦ “subscribed”],
           ![“trial”][u] = [type ↦ “ineligible”],
           ![“videoAccess”][u] = [type ↦ “enabled”]
       ]
    ∧ UNCHANGED ⟨month, events⟩

 Cancelled users that have passed cancellation period are unsubscribed
ProcessCancelledUser(u) ≜
    ∧ database.subscription[u].type = “tocancel”
    ∧ database.subscription[u].canceltime ≤ month
    ∧ database′ = [database EXCEPT
             unsubscribe
           ![“subscription”][u] =  [type ↦ “notsubscribed”]
       ]
    ∧ UNCHANGED ⟨month, events⟩

 Any subscribed user is billed this month
BillUserForSubscription(u) ≜
   LET subscriptionFee ≜
         [event ↦ Now,
          fee ↦ SubscriptionFee]
   IN
    ∧ database.subscription[u].type = “subscribed”
     Add subscription fee for this month
    ∧ database′ =
       [database EXCEPT
              ![“upcomingCharges”][u][month] =
                  database.upcomingCharges[u][month] ∪
                  {subscriptionFee}
       ]
    ∧ UNCHANGED ⟨month, events⟩

 Bill users for their current month charges
ProcessCharges ≜
    ∧ ∃ u ∈ USERS :
       LET monthlyCharges ≜ database.upcomingCharges[u][month]IN
        If there are upcoming charges for this month
        ∧ Cardinality(monthlyCharges) > 0
        Dequeue a bill
        ∧ ∃ charge ∈ monthlyCharges :
            Submit to payment processor
            ∧ Bill(u, charge.fee)
            Delete from queue
            ∧ database′ =
                [database EXCEPT
                    ![“upcomingCharges”][u][month] = monthlyCharges \ {charge}
                ]
    ∧ UNCHANGED month

Stub method that prevents the month from passing until all operations are complete. Represent
worker methods, etc.
HandledMonth ≜
    ∧ ¬ENABLED ProcessCharges
    ∧ ∀ u ∈ USERS :
        ∧ ¬ENABLED ConvertTrialUser(u)
        ∧ ¬ENABLED ProcessCancelledUser(u)
        ∧ ¬ENABLED BillUserForSubscription(u)


 DO NOT MODIFY
MonthPasses ≜
    ∧ HandledMonth
    ∧ month′ = month + 1
    ∧ events′ = Append(events, [type ↦ “monthpass”])
    ∧ UNCHANGED ⟨database⟩

Specification
Init ≜
    ∧ events = ⟨⟩ Events must be intialized empty, per stub
    ∧ month = 0
    ∧ database = [
                No months have been billed for yet
            billingMonth ↦ 0,
                No user starts with access
            videoAccess ↦ [u ∈ USERS ↦ [type ↦ “disabled”]],
                Every user starts eligible for trial
            trial ↦ [u ∈ USERS ↦ [type ↦ “eligible”]],
                Every user starts not subscribed
            subscription ↦ [u ∈ USERS ↦ [type ↦ “notsubscribed”]],
                All users start in good standing
            inGoodStanding ↦ [u ∈ USERS ↦ TRUE],
                No bills to submit
            upcomingCharges ↦ [u ∈ USERS ↦ [x ∈ Months ↦ {}]]
        ]
Next ≜
   \* Required by stub
   ∨ MonthPasses
   \* State modified below
   ∨ ∃ u ∈ USERS:
```

$\lor \mathit{StartSubscription}(u)$

$\lor \mathit{CancelSubscription}(u)$

$\lor \mathit{StartTrial}(u)$

$\lor \mathit{CancelTrial}(u)$

$\lor \mathit{WatchVideo}(u)$

$\lor \mathit{ConvertTrialUser}(u)$

$\lor \mathit{ProcessCancelledUser}(u)$

$\lor \mathit{ConvertTrialUser}(u)$

$\lor \mathit{BillUserForSubscription}(u)$

* Payment failing behavior is part of spec, not implementation

$\lor \mathit{ExistingBillFailed}$

$\lor \mathit{ProcessCharges}$

* Modification History

* Last modified Sun *Jun* 19 17:45:52 *MST* 2022 by *elliotswart*

* Created *Fri Jun* 17 00:28:26 *MST* 2022 by *elliotswart*