MODULE *speccannonical*

EXTENDS $Naturals$, $Sequences$, $FiniteSets$

Redeclaration of specdatamodels variables

VARIABLE $events$

CONSTANT $USERS$

CONSTANTS
$SubscriptionFee$,
$CancellationFee$,
$FailedPaymentFee$

Logic to Test Replace stubs below with implementation. Because there is no forward declaration, we invert what we'd ideally like to do, which is to import the requirements into each implementation. Our logic testing relies on determining if a given state is enabled or not.

VARIABLE $database$, $month$
INSTANCE $stubs$

$Spec \triangleq Init \land \Box[Next]_{vars}$

Trace requirements to specification

Not Traceable Functional: 1,2,3,6,7,9,14 *NonFunctional*: 1,2,3

Definitions

```
InTrial(u, end) ≜
    ∃ i ∈ 1 .. end :
        ∧ events[i] ∈ StartTrialEvent    Has started trial
        ∧ events[i].user = u
        6. Start Trial endpoint request
        6.3 If the requesting User has never been Subscribed or In Trial, that User SHALL be In
            Trial
        ∧ ¬∃ j ∈ i .. end :    And not canceled
            ∧ events[j] ∈
                8 Cancel Trial endpoint request
                8.2 [Partial] If the requesting User is In Trial, the User SHALL be Not Subscribed
                CancelTrialEvent ∪
                2. Start Subscription endpoint request
                2.2 If the requesting User is In Trial, the trial SHALL end and the requesting
                    User SHALL be Subscribed
                StartSubscriptionEvent
            ∧ events[j].user = u
    11 [Partial] When a User is In Trial at the end of the month that the trial was started,
        they SHALL be *Subscribed*
    $\land \neg\exists\, j \in i \,..\, end :$
        $\land\ events[j] \in MonthPassEvent$


$UnsubscribedAfterEvent(u,\ i,\ end) \ \triangleq$
  $\exists\, j \in i \,..\, end :$  And not unsubscribed after
    $\land\ events[j] \notin MonthPassEvent$
    $\land\ events[j].user = u$
    Cancel Subscription endpoint request 4.2.1 User SHALL be Not *Subscribed* at the end of
    the
       current month
    $\land\ \lor\ \land\ events[j] \in CancelSubscriptionEvent$
           $\land\ \exists\, k \in j \,..\, end : events[k] \in MonthPassEvent$
      16. User has payment failed
      16.1 mark the User as Not *Subscribed*
      $\lor\ events[j] \in PaymentFailedEvent$


$SubscribedFromStartSubscription(u,\ end) \ \triangleq$
  2.4 If the requesting User is scheduled to be Not *Subscribed* due to cancellation, the requesting
     User SHALL remain Subscribed
  Implemented because a *StartSubscriptionEvent* after Cancel undoes the cancel.
  $\exists\, i \in 1 \,..\, end :$
    $\land\ events[i] \in StartSubscriptionEvent$  Has subscribed
    $\land\ events[i].user = u$
    $\land\ \neg UnsubscribedAfterEvent(u,\ i,\ end)$

$AboutToCancel(u,\ end) \ \triangleq$
  $\exists\, i \in 1 \,..\, end :$
    $\land\ events[i] \in CancelSubscriptionEvent$
    $\land\ \neg\exists\, j \in i \,..\, end :$
        $events[j] \in MonthPassEvent\ \cup$
                    $StartSubscriptionEvent$

$SubscribedFromTrial(u,\ end) \ \triangleq$
  11 [Partial] When a User is In Trial at the end of the month that the trial was started, they
     SHALL be *Subscribed*
  $\exists\, i \in 1 \,..\, end :$
    $\land\ events[i] \in StartTrialEvent$  Has started trial
    $\land\ events[i].user = u$
    $\land\ \neg InTrial(u,\ end)$  Requirement fulfilled through *InTrial*
    $\land\ \neg UnsubscribedAfterEvent(u,\ i,\ end)$
```

Cancel Trial endpoint request 8.2 [Partial] If the requesting User is In Trial, the User SHALL be Not *Subscribed*

$\land \neg \exists\, j \in i \mathrel{..} end :$ And not canceled
$\quad \land\ events[j] \in CancelTrialEvent$
$\quad \land\ events[j].user = u$

$$
\begin{aligned}
&Subscribed(u, end) \triangleq \\
&\quad \lor SubscribedFromStartSubscription(u, end) \\
&\quad \lor SubscribedFromTrial(u, end)
\end{aligned}
$$

Invariants

2 When a request is received by the Start Subscription endpoint

$StartSubscriptionAccessControl \triangleq$
$\quad \forall\, u \in USERS :$
$\qquad \text{LET } authorized \triangleq \neg Subscribed(u, Now) \lor AboutToCancel(u, Now) \text{ IN}$

2.1: If the requesting User is *Subscribed*, the request SHALL return with 409 Conflict

$\lor \land \neg authorized$
$\quad \land \neg \text{ENABLED } StartSubscription(u)$

2.2 [Partial]: If the requesting User is In Trial, the trial SHALL end and the requesting User SHALL be *Subscribed*

2.3: If the requesting User is Not *Subscribed*, the requesting User SHALL be *Subscribed*

$\lor \land authorized$
$\quad \land \text{ENABLED } StartSubscription(u)$

4 When a request is received by the Cancel Subscription endpoint

$CancelSubscriptionAccessControl \triangleq$
$\quad \forall\, u \in USERS :$
$\qquad \text{LET } authorized \triangleq Subscribed(u, Now) \land \neg AboutToCancel(u, Now) \text{ IN}$

4.1 If the requesting User is not *Subscribed*, the request SHALL return with 409 Conflict

$\lor \land \neg authorized$
$\quad \land \neg \text{ENABLED } CancelSubscription(u)$

4.2 [Partial]: If the requesting User is *Subscribed*, the User SHALL ... [Cancellation Requirements]

$\lor \land authorized$
$\quad \land \text{ENABLED } CancelSubscription(u)$

6.3 [Partial] If the requesting User is has never been *Subscribed*, or is In Trial

```
$EligibleForTrial(u) \triangleq$
    $\neg \exists\, i \in 1 \,..\, Len(events) :$
        $\land\ events[i] \in$
            $StartSubscriptionEvent \cup$
            $StartTrialEvent$
        $\land\ events[i].user = u$
```

6 When a request is received by the Start Trial endpoint

```
$StartTrialAccessControl \triangleq$
    $\forall\, u \in USERS :$
```

6.1 If the requesting User is *Subscribed* or In Trial, the request SHALL return with 409 Conflict

6.2 If the requesting User has previously been *Subscribed* or In Trial, the request SHALL return with 409 Conflict

```
    $\lor\ \land \neg EligibleForTrial(u)$
      $\land \neg \text{ENABLED}\ StartTrial(u)$
```

6.3 If the requesting User has never been *Subscribed* or In Trial, that User SHALL be In Trial

```
    $\lor\ \land EligibleForTrial(u)$
      $\land \text{ENABLED}\ StartTrial(u)$
```

8 When a request is received by the Cancel Trial endpoint

```
$CancelTrialAccessControl \triangleq$
    $\forall\, u \in USERS :$
```

8.1 If the requesting User is not In Trial, the request SHALL return with 409 Conflict

```
    $\lor\ \land \neg InTrial(u,\ Now)$
      $\land \neg \text{ENABLED}\ CancelTrial(u)$
```

8.2 [Partial] If the requesting User is In Trial, the User SHALL be Not *Subscribed*

```
    $\lor\ \land InTrial(u,\ Now)$
      $\land \text{ENABLED}\ CancelTrial(u)$
```

10 When a request is received by the Watch Video endpoint

```
$WatchVideoAccessControl \triangleq$
    $\forall\, u \in USERS :$
```

10.1 If the requesting User is not In Trial or *Subscribed*, the request SHALL return with 409 Conflict

```
    $\lor\ \land \neg InTrial(u,\ Now) \land \neg Subscribed(u,\ Now)$
      $\land \neg \text{ENABLED}\ WatchVideo(u)$
```

10.2 If the requesting User is In Trial or *Subscribed*, the system SHALL allow the User to Watch Video

```
    $\lor\ \land InTrial(u,\ Now) \lor Subscribed(u,\ Now)$
      $\land \text{ENABLED}\ WatchVideo(u)$
```

Runs a given operation between: $1-$ first month for the first month, and month $i - month\ i+1$

```
$TrueForEveryUserMonth(op(\_, \_, \_), checkFirstMonth) \triangleq$
    LET $numMonthPass \triangleq Cardinality(\{i \in 1 \,..\, Len(events) : events[i]$
                                                              $\in MonthPassEvent\})$
    IN
     If checking the first month
     $\land \lor \neg checkFirstMonth$
        $\lor \land checkFirstMonth$
         There does not exist
          $\land \neg \exists\, i \in 1 \,..\, Len(events) :$
             a first month
             $\land events[i] \in MonthPassEvent$
             $\land \neg \exists\, j \in 1 \,..\, i : events[j] \in MonthPassEvent$
             Where the $op$ is false for any user
             $\land \exists\, u \in USERS :$
                $\neg op(u, 1, i)$

     There does not exist a pair of consecutive months
     $\land \neg \exists\, i \in 1 \,..\, Len(events) :$
         $\land events[i] \in MonthPassEvent$
         $\land \exists\, j \in i+1 \,..\, Len(events) :$
             $\land events[j] \in MonthPassEvent$
             $\land \neg \exists\, k \in (i+2) \,..\, (j-1) :$
                 $events[k] \in MonthPassEvent$
             where $op$ is not true for all users
             $\land \exists\, u \in USERS :$
                $\neg op(u, i, j)$
```

15 When a User is Billed the system SHALL call the Bill endpoint of the Payment Processor. This requirement is satisfied by how requirements 4.2.2, 12 and 13 are tested. They test that appropriate Bill message was dispatched

12 When a User becomes *Subscribed*

12.1 they shall be Billed the Subscription Fee before the end of the month

```
$SubscribedThisMonth(u, start, end) \triangleq$
     $\land \neg Subscribed(u, start)$
     $\land Subscribed(u, end - 1)$

$UserSubscribedThisMonthBilledSubscriptionFee(u, start, end) \triangleq$
    LET $shouldBill \triangleq SubscribedThisMonth(u, start, end)$IN
     Only applies if subscribed this month
     $\lor \neg shouldBill$
     $\lor \land shouldBill$
        $\land \exists\, i \in start \,..\, end :$
        $\land$ $events[i] \in BillEvent$
        $\land$ $events[i].user = u$
        $\land$ $events[i].fee = SubscriptionFee$

$SubscribedNewUsersBilledSubscriptionFee \triangleq$
    $TrueForEveryUserMonth(UserSubscribedThisMonthBilledSubscriptionFee, \text{TRUE})$
```

13 When a User is *Subscribed* at the start of a month, they shall be Billed the Subscription Fee

```
$SubscribedUserBilledThisMonth(u, start, end) \triangleq$
   LET $subscribed \triangleq Subscribed(u, start)$ IN
    Only applies if subscribed at start of month
    $\lor \neg subscribed$
    $\lor \land subscribed$
       $\land \lor \exists i \in start \,..\, end :$
              $\land events[i] \in BillEvent$
              $\land events[i].user = u$
              $\land events[i].fee = SubscriptionFee$
          If the user failed a payment this is a separate workflow
          $\lor \exists i \in start \,..\, end :$
              $\land events[i] \in PaymentFailedEvent$
              $\land events[i].user = u$

$SubscribedUsersBilledStartOfMonth \triangleq$
    $TrueForEveryUserMonth(SubscribedUserBilledThisMonth, \text{FALSE})$
```

12.2 If the requesting User has Post Due Payments they SHALL be Billed in that amount before the end of the month, and Post Due Payments shall be zeroed

16 When a callback is received to the Payment Failed endpoint for a User, the system SHALL
16.2 set Post Due Payment for the User to:
(failed payment amount) + *CancellationFee*

```
$PotentialStartingEvent(u, event) \triangleq$
    $\land event \in StartSubscriptionEvent \cup$
                $StartTrialEvent$
    $\land event.user = u$

$IsPaymentFailedEvent(u, event) \triangleq$
    $\land event \in PaymentFailedEvent$
    $\land event.user = u$

$UserBilledForFailureBetweenRange(u, start, end, fee) \triangleq$
   $\exists i \in start \,..\, end :$
       $\land events[i] \in BillEvent$
       $\land events[i].user = u$
       $\land events[i].fee = FailedPaymentFee$
```

```
$UserBilledForPostDuePaymentsIfSubscribed(u, start, end) \triangleq$
    LET $starts \triangleq \{i \in 1 \, .. \, start : PotentialStartingEvent(u, events[i])\}$ IN
    LET $paymentFailed \triangleq \{i \in 1 \, .. \, start : IsPaymentFailedEvent(u, events[i])\}$ IN

    $\forall p \in paymentFailed :$

      LET $resubscribedAfterFailedPayment \triangleq$
            $\exists i \in p \, .. \, end :$
                $\land \; i \in starts$
      IN

       $\lor \; \neg resubscribedAfterFailedPayment$
       $\lor \; \land \; resubscribedAfterFailedPayment$
            There doesn't exist a failed payment
          $\land \; \neg \exists i \in p \, .. \, end :$
                That has a subscription directly after it
                $\land \; i \in starts$
                $\land \; \neg \exists j \in p \, .. \, i :$
                    $j \in starts$
                Where the user was not billed for the failed payment
                $\land \; \neg UserBilledForFailureBetweenRange(u, i, end, events[p].fee)$

$SubscribedUsersBilledPostDuePayements \triangleq$
    $TrueForEveryUserMonth(UserBilledForPostDuePaymentsIfSubscribed, \text{TRUE})$
```

4 Cancel Subscription endpoint
4.2.2 if the user is Not *Subscribed* at the end of the current month, they SHALL be Billed a Cancellation Fee

```
$UserCancelledLastMonth(u, start, end) \triangleq$
      $start - 1$ because it doesn't count cancellations that take effect
      at start
      $\land \; Subscribed(u, start - 1)$
      $\land \; \neg Subscribed(u, start)$

$UserCancelledLastMonthBilled(u, start, end) \triangleq$
      Only applies if user cancelled this month
      $\lor \; \neg UserCancelledLastMonth(u, start, end)$
      $\lor \; \land \; UserCancelledLastMonth(u, start, end)$
        $\land \; \lor \; \exists i \in start \, .. \, end :$
                $\land \; events[i] \in BillEvent$
                $\land \; events[i].user = u$
                $\land \; events[i].fee = CancellationFee$
           If the user failed a payment this is a separate workflow
           $\lor \; \exists i \in start \, .. \, end :$
                $\land \; events[i] \in PaymentFailedEvent$
                $\land \; events[i].user = u$
```

$CancelingUsersBilledCancelationFees \triangleq$
$\quad TrueForEveryUserMonth(UserCancelledLastMonthBilled, \text{FALSE})$

State Constraints

$EventLengthLimit \triangleq$
$\quad Len(events) < 10$

$MonthLimit \triangleq$
$\quad \text{LET } monthPassEvents \triangleq SelectSeq(events, \text{LAMBDA } x : x.type = \text{“monthpass”})$
$\quad \text{IN}$
$\quad Len(monthPassEvents) < 5$

$StateLimit \triangleq$
$\quad \land EventLengthLimit$
$\quad \land MonthLimit$

* Modification History
* Last modified Sun *Jun* 19 17:43:11 *MST* 2022 by *elliotswart*
* Created *Thu Jun* 16 19:34:18 *MST* 2022 by *elliotswart*