MODULE *spec*

EXTENDS *Naturals*, *Sequences*, *FiniteSets*

Redeclaration of *specdatamodels* variables

VARIABLE *events*

Represents every potential user in the system
CONSTANT *USERS*

Constants that should be set to single model values, to allow comparisons
Only equality comparisons will be made.
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 import the requirements into each implementation. Our logic testing relies on determining if a given state is enabled or not

VARIABLE *database*, *month*
INSTANCE *juniorv*1

$Next \triangleq$
Required by stub
$\lor$ *MonthPasses*
State modified below
$\lor \exists\, u \in USERS :$
$\lor$ *StartSubscription*(u)
$\lor$ *CancelSubscription*(u)
$\lor$ *StartTrial*(u)
$\lor$ *CancelTrial*(u)
$\lor$ *WatchVideo*(u)
Add more user based states
Payment failing behavior is part of spec not implementation
$\lor$ *ExistingBillFailed*
$\lor$ *BillSubscribedUsers*
$\lor$ *ProcessBills*

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

Trace requirements to specification

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

Defintions

$InTrial(u,\ end) \triangleq$

$\exists\, i \in 1\,..\,end :$

$\land\ events[i] \in StartTrialEvent$ Has started trial

$\land\ events[i].user = u$

6. Start Trial endpoint request
6.3 If the requesting User is has never been *Subscribed*, or In Trial, that User SHALL be In Trial

$\land \neg\exists\, j \in i\,..\,end :$ And not canceled

$\land\ events[j] \in$

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

$CancelTrialEvent \cup$

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*

$StartSubscriptionEvent$

$\land\ events[j].user = u$

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 undos 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*

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

$Subscribed(u,\ end)\ \triangleq$
$\lor\ SubscribedFromStartSubscription(u,\ end)$
$\lor\ SubscribedFromTrial(u,\ end)$

Invariants

2 When a request is received by the Start Subscription endpoint

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

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

$\lor\ \land\ \neg authorized$
$\land\ \neg$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*
$\land$ ENABLED *StartSubscription*(u)

4 When a request is received by the Cancel Subscription endpoint

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

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

$\lor \land \neg$*authorized*
$\land \neg$ENABLED *CancelSubscription*(u)

4.2 [Partial]: If the requesting User is *Subscribed*, the User SHALL ... sub requirements

$\lor \land$ *authorized*
$\land$ ENABLED *CancelSubscription*(u)

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

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

6 When a request is received by the Start Trial endpoint

StartTrialAccessControl $\triangleq$
$\forall\, u \in \mathit{USERS}$:

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

$\lor \land \neg \mathit{EligibleForTrial}(u)$
$\land \neg$ENABLED *StartTrial*(u)

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

$\lor \land$ *EligibleForTrial*(u)
$\land$ ENABLED *StartTrial*(u)

8 When a request is received by the Cancel Trial endpoint

CancelTrialAccessControl $\triangleq$
$\forall\, u \in \mathit{USERS}$:

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

$\lor \land \neg \mathit{InTrial}(u,\ \mathit{Now})$
$\land \neg$ENABLED *CancelTrial*(u)

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

$\lor \land InTrial(u,\ Now)$
$\quad \land$ 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

$\lor \land \neg InTrial(u,\ Now) \land \neg Subscribed(u,\ Now)$
$\quad \land \neg$ENABLED $WatchVideo(u)$

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

$\lor \land InTrial(u,\ Now) \lor Subscribed(u,\ Now)$
$\quad \land$ 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
$\land\ \lor \neg checkFirstMonth$
$\quad \lor \land checkFirstMonth$
There does not exist
$\quad\quad \land \neg\exists\, i \in 1\ ..\ Len(events) :$
a first month
$\quad\quad\quad \land events[i] \in MonthPassEvent$
$\quad\quad\quad \land \neg\exists\, j \in 1\ ..\ i : events[j] \in MonthPassEvent$
Where the *op* is false for any user
$\quad\quad\quad \land \exists\, u \in USERS :$
$\quad\quad\quad\quad \neg op(u,\ 1,\ i)$

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

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

$$
\begin{array}{l}
SubscribedThisMonth(u,\ start,\ end) \triangleq \\
\quad \land \neg Subscribed(u,\ start) \\
\quad \land Subscribed(u,\ end - 1)
\end{array}
$$

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

$$
\begin{array}{l}
SubscribedNewUsersBilledSubscriptionFee \triangleq \\
\quad TrueForEveryUserMonth(UserSubscribedThisMonthBilledSubscriptionFee,\ \text{TRUE})
\end{array}
$$

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

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

$$
\begin{array}{l}
SubscribedUsersBilledStartOfMonth \triangleq \\
\quad TrueForEveryUserMonth(SubscribedUserBilledThisMonth,\ \text{FALSE})
\end{array}
$$

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*

$$
\begin{array}{l}
PotentialStartingEvent(u,\ event) \triangleq \\
\quad \land\ event \in StartSubscriptionEvent \cup \\
\qquad\qquad StartTrialEvent \\
\quad \land\ event.user = u
\end{array}
$$

$$
\begin{array}{l}
IsPaymentFailedEvent(u,\ event) \triangleq \\
\quad \land\ event \in PaymentFailedEvent \\
\quad \land\ event.user = u
\end{array}
$$

$$
\begin{array}{l}
UserBilledForFailureBetweenRange(u,\ start,\ end,\ fee) \triangleq \\
\quad \exists\, i \in start\ ..\ end : \\
\qquad \land\ events[i] \in BillEvent \\
\qquad \land\ events[i].user = u \\
\qquad \land\ events[i].fee = FailedPaymentFee
\end{array}
$$

$$
\begin{array}{l}
UserBilledForPostDuePaymentsIfSubscribed(u,\ start,\ end) \triangleq \\
\quad \text{LET } starts \triangleq \{i \in 1\ ..\ start : PotentialStartingEvent(u,\ events[i])\} \text{IN} \\
\quad \text{LET } paymentFailed \triangleq \{i \in 1\ ..\ start : IsPaymentFailedEvent(u,\ events[i])\} \text{IN} \\
\\
\quad \forall\, p \in paymentFailed : \\
\\
\qquad \text{LET } resubscribedAfterFailedPayment \triangleq \\
\qquad\quad \exists\, i \in p\ ..\ end : \\
\qquad\qquad \land\ i \in starts \\
\qquad \text{IN} \\
\\
\qquad \lor \neg resubscribedAfterFailedPayment \\
\qquad \lor\ \land\ resubscribedAfterFailedPayment \\
\qquad\quad \text{There doesn't exist a failed payment} \\
\qquad\quad \land \neg \exists\, i \in p\ ..\ end : \\
\qquad\qquad\quad \text{That has a subscription directly after it} \\
\qquad\qquad\quad \land\ i \in starts \\
\qquad\qquad\quad \land \neg \exists\, j \in p\ ..\ i : \\
\qquad\qquad\qquad\quad j \in starts \\
\qquad\qquad\quad \text{Where the user was not filled for the failed payment} \\
\qquad\qquad\quad \land \neg UserBilledForFailureBetweenRange(u,\ i,\ end,\ events[p].fee)
\end{array}
$$

$$
\begin{array}{l}
SubscribedUsersBilledPostDuePayements \triangleq \\
\quad TrueForEveryUserMonth(UserBilledForPostDuePaymentsIfSubscribed,\ \text{TRUE})
\end{array}
$$

4 Cancel Subscription endpoint
4.2.2 if the user 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

$\lor \exists\, i \in start\ ..\ end :$

$\land\ events[i] \in PaymentFailedEvent$

$\land\ events[i].user = u$

$CancelingUsersBilledCancelationFees \triangleq$

$TrueForEveryUserMonth(UserCancelledLastMonthBilled,\ \text{FALSE})$

State Constraints

$EventLengthLimit \triangleq$

$Len(events) < 10$

$MonthLimit \triangleq$

LET $monthPassEvents \triangleq SelectSeq(events,\ \text{LAMBDA}\ x : x.type = \text{“monthpass”})$

IN

$Len(monthPassEvents) < 5$

$StateLimit \triangleq$

$\land\ EventLengthLimit$

$\land\ MonthLimit$

* Modification History

* Last modified Sun *Jun* 19 19:37:09 *MST* 2022 by *elliotswart*

* Created *Thu Jun* 16 19:34:18 *MST* 2022 by *elliotswart*