MODULE *juniorv3*

Redeclaration of *specdatamodels* variables

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

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*

VARIABLES
Represents the current month
*month*,
Represents the status of the database. Design requirements require
that all persistant application state be stored here
*database*,
Required by spec
*events*

$vars \triangleq \langle events, month, database \rangle$

Provides all the data models required by the spec
INSTANCE *specdatamodels*

$Now \triangleq Len(events)$

$Months \triangleq 0 \,..\, 10$

Strong Typing

$Month \triangleq Nat$

Database Rows

$UserRow \triangleq [$
*subscribed* : BOOLEAN ,
Forget *cancelled*
*inTrial* : BOOLEAN ,
*trialStartTime* : *Nat*,
*billedForMonth* : *Nat*,
*hasHadTrialOrSubscription* : BOOLEAN ,
*hasCancelled* : BOOLEAN ,

```
    cancelMonth : Nat,
    subscribeMonth : Nat
]

BillQueueItem ≜ [
    user : USERS,
    fee : Fees,
    when : Nat
]

TypeOk ≜
    ∧ EventsOk
    ∧ month ∈ Month
    ∧ database.users ∈ [USERS → UserRow]
    ∧ database.billQueue ∈ Seq(BillQueueItem)
```

*API* endpoints

```
StartSubscription(u) ≜
    Not subscribed
    ∧ ∧ database.users[u].subscribed = FALSE
      ∧ ∨ database.users[u].inTrial  = FALSE
        ∨ database.users[u].trialStartTime = month

    ∧ database' =
      [database EXCEPT
            ![“users”][u].subscribed = TRUE,
            ![“users”][u].hasHadTrialOrSubscription = TRUE,
            ![“users”][u].hasCancelled = FALSE,
            ![“users”][u].inTrial = FALSE,
            ![“users”][u].subscribeMonth = month,
            ![“billQueue”] =
                Append(database.billQueue,
                    [user ↦ u,
                    fee ↦ SubscriptionFee,
                     bill not
                    when ↦ month])
      ]
    Observability required by stub
    ∧ events' = Append(events, [type ↦ “startsubscription”, user ↦ u])
    ∧ UNCHANGED month

CancelSubscription(u) ≜
    LET updatedBillQueue ≜
```

```
        IF ∃ i ∈ 1 .. Len(database.billQueue) :
             ∧ database.billQueue[i].user = u
             ∧ database.billQueue[i].fee = SubscriptionFee
         THEN database.billQueue
         ELSE Append(database.billQueue,
                     [user ↦ u, fee ↦ SubscriptionFee, when ↦ month])
    IN

     Subscribed
    ∧ ∨ database.users[u].subscribed = TRUE
      ∨ ∧ database.users[u].inTrial  = TRUE
        ∧ database.users[u].trialStartTime < month
    ∧ database' =
        [database EXCEPT
            !["users"][u].subscribed = FALSE,
            !["users"][u].inTrial = FALSE,
            !["users"][u].hasCancelled = TRUE,
            !["users"][u].cancelMonth = month,

             Charge cancellation fee
            !["billQueue"] =
                Append(updatedBillQueue,
                    [user ↦ u,
                     fee ↦ CancellationFee,
                     when ↦ month + 1])
        ]

     Observability required by stub
    ∧ events' = Append(events, [type ↦ "cancelsubscription", user ↦ u])
    ∧ UNCHANGED ⟨month⟩

StartTrial(u) ≜
    ∧ database.users[u].inTrial = FALSE
    ∧ database.users[u].subscribed = FALSE
    ∧ database.users[u].hasHadTrialOrSubscription = FALSE
    ∧ database' = [database EXCEPT
                        !["users"][u].inTrial = TRUE,
                        !["users"][u].trialStartTime = month,
                        !["users"][u].hasHadTrialOrSubscription = TRUE
                   ]

     Observability required by stub
    ∧ events' = Append(events, [type ↦ "starttrial", user ↦ u])
    ∧ UNCHANGED ⟨month⟩
```

$CancelTrial(u) \triangleq$

In active trial

$\land\ database.users[u].inTrial = \text{TRUE}$

$\land\ database.users[u].trialStartTime = month$

And not subscribed

$\land\ database.users[u].subscribed = \text{FALSE}$

$\land\ database' = [database \text{ EXCEPT}$

$!["\text{users}"][u].inTrial = \text{FALSE}$

$]$

Observability required by stub

$\land\ events' = Append(events, [type \mapsto "\text{canceltrial}", user \mapsto u])$

$\land\ \text{UNCHANGED } \langle month \rangle$

$WatchVideo(u) \triangleq$

$\land\ \lor\ database.users[u].subscribed = \text{TRUE}$

$\lor\ database.users[u].inTrial = \text{TRUE}$

Remove video access at the end of *cancelled* month

$\lor\ \land\ database.users[u].hasCancelled = \text{TRUE}$

$\land\ database.users[u].cancelMonth = month$

Observability required by stub

$\land\ events' = Append(events, [type \mapsto "\text{watchvideo}", user \mapsto u])$

$\land\ \text{UNCHANGED } \langle month, database \rangle$

Stub method, do not changed

$Bill(u, fee) \triangleq$

$\land\ events' = Append(events, [type \mapsto "\text{bill}",$

$user \mapsto u,$

$fee \mapsto fee])$

$PaymentFailed(u, fee) \triangleq$

$\land\ database' = [database \text{ EXCEPT}$

$![\text{"users"}][u].subscribed = \text{FALSE},$

$![\text{"users"}][u].hasCancelled = \text{FALSE},$

$![\text{"users"}][u].inTrial = \text{FALSE}$

$]$

Observability required by stub

$\land\ events' = Append(events, [type \mapsto "\text{paymentfailed}",$

$user \mapsto u,$

$fee \mapsto fee])$

$\land\ \text{UNCHANGED } \langle month \rangle$

Recurring Operations

```
 This the the state that calls the Payment Failed API
ExistingBillFailed ≜
    ∨ ∃ i ∈ 1 .. Len(events) :
        Only a past bill can fail
        ∧ events[i] ∈ BillEvent
        ∧ PaymentFailed(events[i].user, events[i].fee)


BillSubscribedUsers ≜
    ∧ ∃ u ∈ USERS :
        That is subscribed
        ∧ ∨ database.users[u].subscribed = TRUE
          Subscribed from a trial so bill
          ∨ ∧ database.users[u].inTrial = TRUE
            ∧ database.users[u].trialStartTime < month
        Ensure users are not double billed
        ∧ database.users[u].billedForMonth < month
        ∧ database′ =
            [database EXCEPT
                Add subscription fee
                ![“billQueue”] =
                    Append(database.billQueue,
                           [user ↦ u,
                           fee ↦ SubscriptionFee,
                           when ↦ month]),
                ![“users”][u].billedForMonth = month
            ]
    ∧ UNCHANGED ⟨events, month⟩

ProcessBills ≜
    ∧ Len(database.billQueue) > 0
    ∧ ∃ i ∈ 1 .. Len(database.billQueue) :
        LET bill ≜ database.billQueue[i]IN
        ∧ bill.when = month
        Charge cancellation fees only a month after cancelled
        and still cancelled
        ∧  ∨ bill.fee ≠ CancellationFee
           ∨ ∧ bill.fee = CancellationFee
             ∧ ∨ database.users[bill.user].subscribed = FALSE
               Subscribed too late to cancel cancellation fee
               ∨ database.users[bill.user].subscribeMonth ≥ bill.when

        ∧ Bill(bill.user, bill.fee)
        ∧ database′ =
            [database EXCEPT
                Removes head of queue
```

```
                ![“billQueue”] =
                    SubSeq(database.billQueue, 1, i − 1) ∘
                    SubSeq(database.billQueue, i + 1, Len(database.billQueue))
            ]
    ∧ UNCHANGED ⟨month⟩

Stub method that prevents the month from passing until all operations are complete. Represent worker methods, etc
HandledMonth ≜
    ∧ ¬ENABLED BillSubscribedUsers
    ∧ ¬ENABLED ProcessBills

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 = [
            Users start with everything unset
        users ↦
            [u ∈ USERS ↦
              [
                subscribed ↦ FALSE,
                inTrial ↦ FALSE,
                trialStartTime ↦ 0,
                billedForMonth ↦ 0,
                hasHadTrialOrSubscription ↦ FALSE,
                hasCancelled ↦ FALSE,
                cancelMonth ↦ 0,
                subscribeMonth ↦ 0
              ]
            ],
            Bill queue starts empty
        billQueue ↦ ⟨⟩
      ]

Next ≜
    Required by stub
    ∨ 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

$\lor ExistingBillFailed$

$\lor BillSubscribedUsers$

$\lor ProcessBills$

* Modification History

* Last modified Sun *Jun* 19 17:44:12 *MST* 2022 by *elliotswart*

* Created Sun *Jun* 19 17:26:11 *MST* 2022 by *elliotswart*