The modeled Library Membership Issuance System in UPPAAL consists of several components to mimic the user interactions, authentication, membership purchase, payment process, and bank approval.

Defined global declarations:

1. **Variables**
    - GOLD_COST and PLAT_COST: Represents the cost values for Gold and Platinum membership types, set respectively at 500 and 1500 units.
    - PLAT and GOLD: Booleans indicating whether a Platinum or Gold membership has been purchased. Both are initially set to false.
    - balance: Represents the user's account balance, initially set to 1000 units.
    - cost: Tracks the cost of the membership being purchased, initially set to 0.
2. **Channels**
    - AuthCheck: Initiates user authentication.
    - AuthFail: Signals failed authentication.
    - AuthSuccess: Signals successful authentication.
    - Pay: Initiates payment from the library system.
    - PaymentSucc: Indicates successful payment approval.
    - PaymentFailed: Indicates failed payment approval.

To model the system, we use 3 submodels.

1. Library_Index Model:

- User Interaction: Manages user interactions, membership purchases, and authentication processes.
- Membership Purchase: Allows users to select Gold or Platinum membership types for purchase.
- Membership Details: Allows for printing of membership details after users have signed in.
- Payment Completion: Requires payment completion before exiting the system.

2. Library_Auth Model:

- Authentication Module: Responsible for user authentication for signing into the system.
- Email check: Assumes login is through email, inputs and checks for the same.
- Initiation and Verification: Initiates the authentication process through the AuthCheck channel and communicates authentication outcomes via appropriate channels.

3. Bank_Payment_Gateway Model:

- Bank Payment Simulation: Simulates the bank's payment system.
- Transaction Approval: Receives payment requests via the Pay channel and approves or denies transactions, communicating outcomes through respective channels (PaymentSucc or PaymentFailed).
- Failure check: **Assumes** 2 failure conditions, first of incorrect payment detail input and the second is a comparison of balance with the transaction amount.

Model Flow

Library: The model starts at the initial state ‘Start’ in the Library model. It then moves the ‘SignIn’ state, after which it initiates an AuthCheck in the LibraryAuth system using the ‘Authcheck!’ synchronization channel.

Library_Auth: As the synchronization channel ‘Authcheck!’ is received, the model moves from the ‘LibraryAuth’ initial state to InputEmail state and then the CheckEmail state. There are 2 branches to this state, which signify a success or failed auth.

Library: If the Library_Auth model returns AuthFail, then we again reach the ‘SignIn’ state and have to check the details again, but if it is successful, we move to the ‘Menu’ state. The menu state has 2 branches, ‘ViewMembershipDetails’ and ‘BuyMembershipCard’. If we move to the ‘ViewMembershipDetails’ state, then the next state is ‘PrintDetails’ which models printing out the member details, and then returns back to the ‘Menu’ state. Whereas if we choose the ‘BuyMembershipCard’ state, we first check a guard to ensure that we haven't purchased both the cards already. If this condition is true, we can either move to the ‘BuyPlatinumCard’ state or the ‘BuyGoldCard’ state, signifying buying a platinum or a gold card. These transitions have a guard for checking that the specific card of type has not been purchased already by checking the PLAT and GOLD variables. After these states each state respectively transitions to the PaymentWait[Gold/Plat] state while initializing the Pay channel and also assigning the value of the cost variable to the cost of the card type being purchased.

Bank_Payment_Gateway: the Pay synchronization moves the model from the initial ‘PaymentGateway’ state to the ‘InputDetails’ state and then the ‘CheckDetails’ state which, has 2 branches, where one signifies the payment failed and initiates the ‘PaymentFailed!’ channel and the other signifies a successful detail check and moves to the ‘CheckBalance’ state. The CheckBalance state has 2 branches signifying success and failure as well. The guards respectively compare the cost of the card being purchased and the current balance variable. If balance >= cost, then payment succeeds , initializes the ‘PaymentSucc!’ channel and balance is updated by subtracting the cost

of the card purchased. Whereas if balance < cost, then the payment fails and the channel ‘PaymentFailed!’ is initiated.

Library: If the payment succeeds we move to the respective success states ‘PlatSucc’ and ‘GoldSucc’. And if the payment fails their respective Fail states. All these states lead to the ‘Signout’ state, which then moves the SignIn state and the model can repeat its flow.

Queries

Safety:

1. The first query for safety checks for the absence of deadlocks in all paths always.
2. The second query checks that the value of balance is always non-negative.

Liveness:

1.