Editorial for LICS Assignment:

This document is meant to specify the assumptions and give a detailed account of the procedure adopted for completing the task.

We wish to model a library membership issuance system in UPPAAL. The model represents a system where users can sign in, view membership details and history, purchase gold or platinum cards, and initiate transactions, with the bank component validating and processing these transactions, implemented strictly as per the problem requirements.

We model the system using three templates: Lib (for library), User, and Bank.

- The ‘Lib’ template models a library membership issuance system, allowing users to sign in, view their membership details, and choose between buying a Gold or Platinum membership card.

- The ‘User’ template manages user-specific actions, such as viewing membership details, purchasing cards, etc. It also maintains user-specific data, including whether the user is a gold or platinum member and their balance.

- The ‘Bank’ template verifies transaction validity based on the user's balance and responds with success or failure accordingly.

NOTE: The variable names are self-explanatory, but utmost care has been taken to justify their roles here.

Model Flow:

- We have the ‘start’ or the ‘signed_out’ state from where the user moves to the ‘authenticating’ state if the user prompts to sign in.
- Each user has a boolean variable ‘email_registered’ defined representing whether the user is registered with the library or not. The authenticated variable is set equal to this variable.

- If the authenticated variable is false, an ‘authentication failed’ message is sent to the user and the system moves back to the ‘signed_out’ state. Else, if true, the user moves to the ‘home’ screen, where he has multiple options.

- They can view their membership history, or if already a cardholder, they can view membership details. At each point, they also have the option to go back to the home page and sign out. The user can also choose to buy a new membership, having the option of a gold or platinum card.

- After confirming the purchase, the transaction is initiated and leads to the state ‘bank_confirmation’ where it awaits confirmation from the bank.

- The bank checks for the balance in the user's account and accordingly executes the transaction after the ‘Checking_Req_Validity’ state.

- If the balance is more than the required cost of membership, the transaction succeeds and the “Tran_success” state is reached.

- The bank balance of the user is updated after deducting the membership cost and the “Thank_Dialog” state is achieved.

- In the “Thanks_dialog” state a thanking dialog is shown for the successful transaction and the bank goes back to the ”bank_start” state.

- If the balance is less than the cost of the membership then the transaction fails and the bank goes back to the “bank_start” state.

- If the transaction fails, we move to the ‘transaction_failed_dialog’ state and then back to the ‘buy_card_options’ state.

- On the other hand, if the ‘transaction_successful?’ sync triggers, the transaction_in_progress flag is set to false and either ‘gold_member’ or ‘platinum_member’ is set to true. The user is eventually signed out.

- From the ‘view_details’ state, the user is allowed to scan through his details. The user may now move back to the ‘home’ state or choose to end his membership (triggering ‘end_membership!’) which would then result in a transition back to ‘signed_out’.

ASSUMPTIONS:

→Based on the problem statement, here are some reasonable assumptions made by us for modeling the system:

(a)User Authentication:

Assumption: The Library has the emails of valid users, i.e. students and staffs, if a user tries to login with a registered email then authentication is successful, else user needs to recheck email id and try again. Thus, the sign-up option is not provided. In this system by default both the users have their emails registered.
Justification:The college library already has emails of all students and staff thus only these emails are allowed to sign-in.

(b)Membership Purchase:

Assumption: Users can only purchase one type of membership (Gold or Platinum) in a single transaction. Each user is initialized with a balance of 700 so that they can buy only a gold card and not a platinum card.
Justification: Simplifies the model and aligns with the common practice of allowing users to make one purchase at a time.

(c)View Membership History:

Assumption: Users can see the membership history directly from the home screen even if the user doesn't have any membership.
Justification: Membership history doesn't depend on present membership.

(d)Logout After Transaction:

Assumption: User is logged out of the system after every transaction.
Justification: Security and privacy practice to ensure that users' information is protected.

(e)Bank Transaction Approval:

Assumption: The bank approves the transaction only if there is a sufficient balance in the user's account.
Justification: Realistic assumption, as banks typically require sufficient funds for transaction approval.

(f)Membership Card Types:

Assumption: Two types of membership cards are available: Gold and Platinum, each with different rates, the gold card costs 600 and 800 units.
Justification: Aligns with the problem statement, providing users with choices for membership types.

(g)Multiple Users:

Assumption: The system can support multiple users but can only allow one user to be signed-in at a time.
Justification: The system is designed to serve one user only at a time, once the user signs out, other users can come to use it again.

SAFETY PROPERTIES VERIFIED

- A[] !(user1.gold_member && user1.platinum_member)
  - Users can't have both gold and platinum membership together
- A[] !(lib.processing_transaction && !authenticated)
  - Transactions should only be initiated when the user is authenticated.
- A[] not deadlock
  - no deadlock
- A[] !(purchasing_gold && purchasing_platinum)
  - A user cannot purchase both gold and platinum cards in the same transaction.
- A[] !(user1.email_registered) imply !(user1.buy_card_options)
  - If a user's email is not registered then no membership can't be bought
- A[] !(user1.view_membership && !user1.gold_member && !user1.platinum_member)
  - Membership details can't be viewed unless you have a membership card
- A[] !(user1.membership_ended && (user1.gold_member|| user1.platinum_member))
  - If the membership is ended then the user doesn't have any membership card
- A[] !(user1.await_authentication && user2.await_authentication)
  - No more than one user can be signed in at a time
- A[] !(user1.buy_card_options && (user1.gold_member||user1.platinum_member))
  - If a user has a membership they can't buy another membership

LIVENESS PROPERTIES VERIFIED

- A<> (user1.signed_out)

    - Initiating the end of membership should eventually lead to membership termination or cancellation.
- A<> (bank.Tran_Success imply user1.signed_out)
    - The user signs out after every successful transaction
- A<> user1.purchase_failed imply (!user1.gold_member && !user1.platinum_member)
    - If the purchase failed then the user doesn't have any membership

REACHABILITY PROPERTIES VERIFIED

- E<> (!user1.gold_member && !user1.platinum_member) imply user1.view_history
    - Users can view membership history even without any membership card
- E<> (bank.Tran_Success imply user1.view_membership)
    - If a transaction is successful then a path exits where you can eventually view-membership-details
- E<> (user1.membership_ended imply user1.buy_card_options)
    - Users can buy membership again after the membership ended
- E<> (user1.buy_card_options imply (user1.signed_out && (!user1.gold_member && !user1.platinum_member)))
    - Users can sign out even without buying any membership card