LICS(CS F214) Assignment:

Scholarship Application

Group 3:

Prithvish Chandrayan 2023A7PS0014P

Rishab Chetal 2023A7PS0015P

Varun Vijay 2023A7PS0016P

Romit Jain 2023A7PS0021P

Anish Sai Nimmagadda 2023A7PS0027P

Assumptions

1. The given model uses two students. This is to ensure ease in using the model.
2. Initially, any student is eligible to apply for a scholarship irrespective of their CGPA or their family's yearly income. This is done to ensure that everyone gets a chance to apply irrespective of their conditions.
3. For a student to be eligible for the Merit Cum Need scholarship it has been assumed that their yearly income would be less than 5,00,000 rupees and the student's CGPA is greater than or equal to 7.0.
4. For a student to be eligible for the Merit scholarship, it has been assumed that the CGPA has to be greater than or equal to 9.0 and that their family income should be greater than or equal to 5 LPA.
5. We have prioritised the checking of Merit Cum Need Scholarship over the Merit Scholarship. A student is first checked for Merit Cum Need, and then in the case that the given student doesn't qualify for Merit Cum Need Scholarship, the said student is checked for the Merit Scholarship. This is done because there is a lower CG cutoff for MCN as compared to MS.
6. Once a student applies for a scholarship, they are not allowed to apply for a scholarship again in that session. This was given in the problem statement.
7. It has been assumed that every student will enter their respective email ID and enter it correctly. This has been done to avoid confusion if he types the email ID of an existing user in the server). It has also been assumed that once a student types the correct email ID, eventually he will type the right password to gain access to the portal.
8. Only one student can access the session at a time. This was given in the problem statement.

9. Student CGPA Validity: The CGPA of students is within the range [0, 10], as enforced by the global declaration int[0,10] CGPA[2]. A student’s CGPA varies from zero to ten.
10. Salary Limits: Salaries are specified in multiples of 10,000 rupees, and the values 48 and 120 correspond to 4,80,000 and 12,00,000 rupees, respectively.
11. No Deadlock Assumption: The system is expected to operate without deadlocks under normal conditions. This is a basic necessity of a model.

Our Model

Our model comprises four systems: Portal, Student, SAC, and Server.

The Portal Template serves as a centralized interface for students to interact with the scholarship application system. It facilitates key operations such as user authentication, scholarship application, transcript retrieval, and viewing of user information.

The Student Template is used for performing all the actions which are relayed to other templates. This template also accepts the user ID as a parameter.

The function of the SAC Template is to grant or deny the scholarship of a student who has applied for it, based on their eligibility which is dependent upon the criteria we have set within the template. Additionally, if a student’s CGPA is less than 7.0, then the template automatically rejects both MCN and the Merit-Based Scholarship.

The Server Template returns the transcript to the portal.

In the model, we have created the scenario for two students by instantiating the Student in the System Declarations as ‘student1’ and ‘student2’. The Portal is named ‘portal’, the SAC is called ‘committee’, and the Server is called ‘server’. We have now declared the CGPA and salaries of the 2 students as arrays of size 2. We have a global variable ‘user_id’ whose value is 1 or 2 referring to the respective students. There is a global array of size 2 called ‘can_apply’ which checks if a particular student is eligible for scholarship by checking the value of the corresponding position in the array. If the value is 1, the student can apply for the scholarship. If 0, the student cannot apply for the scholarship.

Workflow of Model

The Model begins at the sign-in state and the user is asked to enter the username. The model checks whether the username already exists or not. If the username does not exist then the user is directed back to the enter username step. Otherwise, the user is made to enter the password. If the password is correct then the user is sent to the homepage; if not, the user enters the password again and again until the correct password is entered. Once the user reaches the homepage, they are given four options: apply scholarship, fetch transcript, view info, and logout. When the user applies for scholarship, the SAC template processes this query and approves or rejects the scholarship. First, the CGPA is checked. If the CGPA is less than 7.0 then the scholarship request is automatically rejected for both MCN and Merit scholarship. If the CGPA is greater than 7.0 then eligibility for MCN scholarship is checked first. If the annual income is less than 5,00,000 then the MCN scholarship is approved otherwise eligibility for Merit scholarships is checked. If the CGPA is greater than 9 then the merit scholarship is approved, otherwise, it is rejected. Once the SAC committee gives a decision on the scholarship, the student is taken back to the homepage. When the user opts to fetch their transcript, the fetch request is sent to the server template which responds by relaying the transcript. Once this process is completed the user is directed back to the homepage. Upon selecting view info, the user is shown their details. The user is taken back to the homepage by selecting the go-back option. Selecting logout terminates the user’s session and redirects them to the sign-in page.

How do we handle the synchronization?

While we were working on the project, we came across some errors when certain channels were not broadcasted in declarations. Upon researching it occurred to us that to achieve one-to-many synchronization, we had to declare those specific channels as broadcast type. This enabled a single event to trigger multiple actions across different templates simultaneously. For example in the Student template when we trigger *app_scholarship!* we want it to react in both the Portal template and the SAC template simultaneously. By declaring the *app_scholarship as a broadcast chan*, we can achieve this goal. This example exemplifies the functionality of the *broadcast* channel, and how it played an integral role for us to successfully handle the synchronization. Initially, we used the broadcast channel only to rectify the problem depicted above. However, upon implementing the broadcast channel, we realised that this also solved another problem for us, that is, ensuring that only one student could access the portal at a time, as opposed to letting multiple students access the portal at the same time. The broadcast channel helped us knock out two birds with one stone, solving two essential problems that had been prevalent in our model, all at once.

Safety and Liveness Properties

Safety

1. A[] (!deadlock)
    - This property ensures that the model will never go into a deadlock state.
2. A[] (can_apply[user_id-1] == 0 imply !(portal.mcn_scholarship))
    - This represents a guard condition that checks whether the student (with index user_id -1) is allowed to apply for the scholarship. Specifically, this condition means that the can_apply flag (for the student) is set to 0, indicating that the student cannot apply for the scholarship.
    - If can_apply[user_id] == 1, it indicates that the student is not eligible or has already applied for the scholarship.

Liveness

1. E<> portal.password imply portal.homepage
    - This property means that there exists a path where if the system reaches the password state, the user will eventually type in the correct password and it will eventually reach the homepage state. In essence, this ensures that once a user enters the correct password, they are guaranteed access to the homepage.
2. E<> portal.homepage imply portal.sign_in
    - This property is used to verify that after reaching the homepage state, the system will eventually return to the sign_in state, which happens when the user logs out. This ensures that the system correctly transitions back to the sign-in screen after a logout, confirming that the logout operation is functional and the user is required to sign in again to access the system.