ASSIGNMENT SUBMISSION - SCHOLARSHIP APPLICATION PORTAL
COURSE - CS F214 LOGIC IN COMPUTER SCIENCE
INSTRUCTORS- DR. RAJYA LAKSHMI L., DR. JAGAT SESH CHALLA
GROUP NUMBER- 21

The **Scholarship Application Portal** is designed to model the interaction between a user and the portal, simulating scholarship applications, fetching transcripts, and accessing personal information. This system ensures:

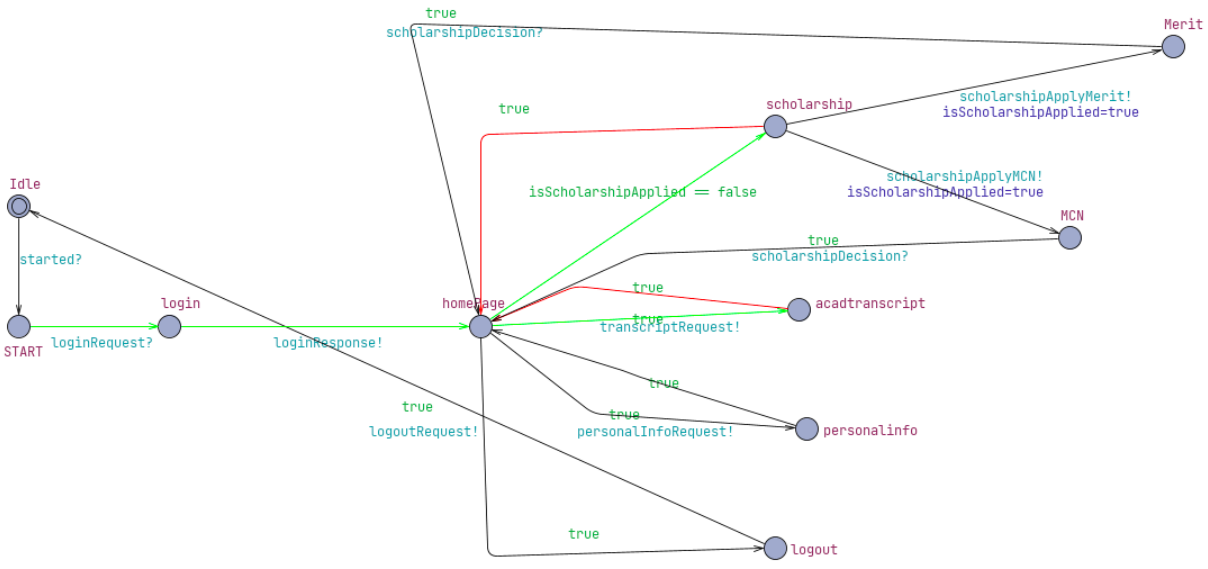
1. Secure authentication for unique users.
2. Prevents multiple scholarship applications in a session.
3. Handles interactions with external components like the ScholarshipComm, CredentialChecker and the Transciptinfo.

The model comprises of the following components listed below in detail with the abstract functionality as a small glimpse of the assignment submission.

KEY COMPONENTS:

1. WEBSITE MODEL:

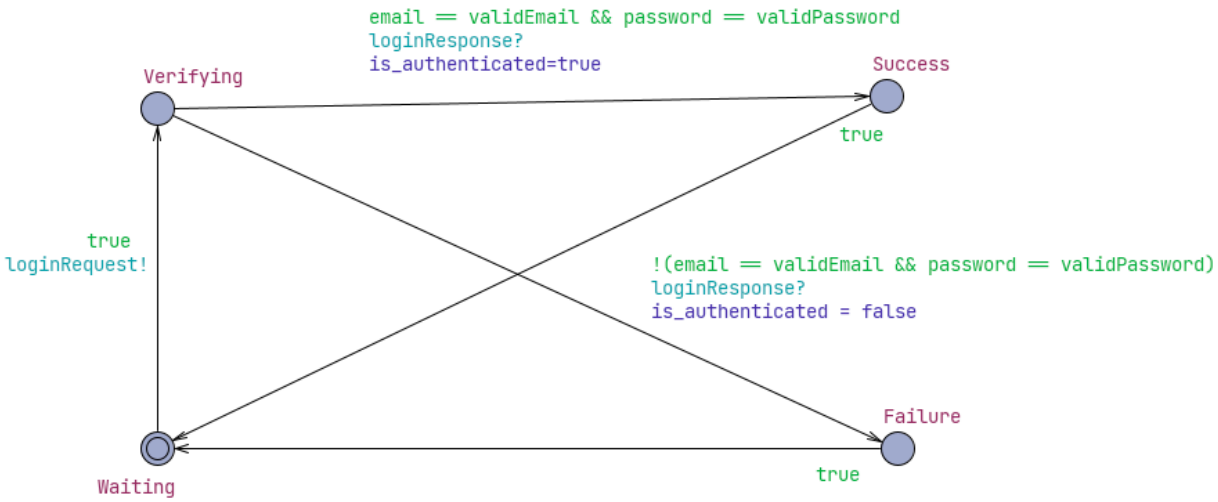
- Handles login, homepage navigation, scholarship application, and logout.
- Separates *Merit Scholarship* and *Merit-Cum-Need Scholarship*.
- Ensures only one type of scholarship is applied for in a session.



---

2. CREDENTIALCHECKER MODEL:

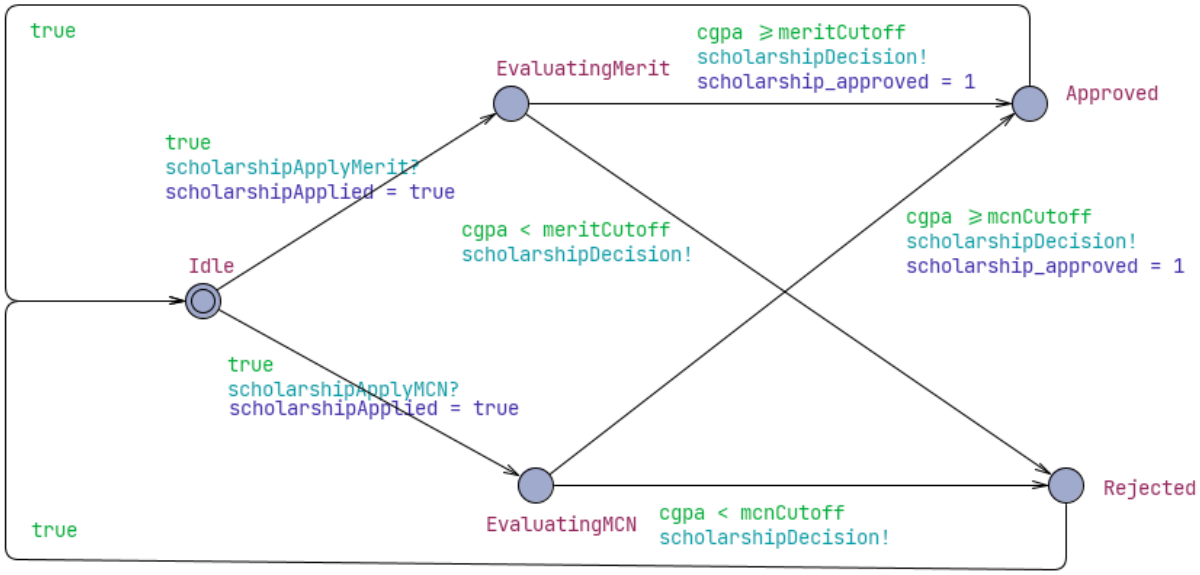
- Authenticates unique users based on their IDs and passwords.
- Tracks authenticated users globally to prevent duplicate logins.
- Key states include **Waiting** and **Verifying**, which handle login requests and responses.
- Input credentials are integers for simplicity during modeling.



---

3. SCHOLARSHIPCOMM MODEL:

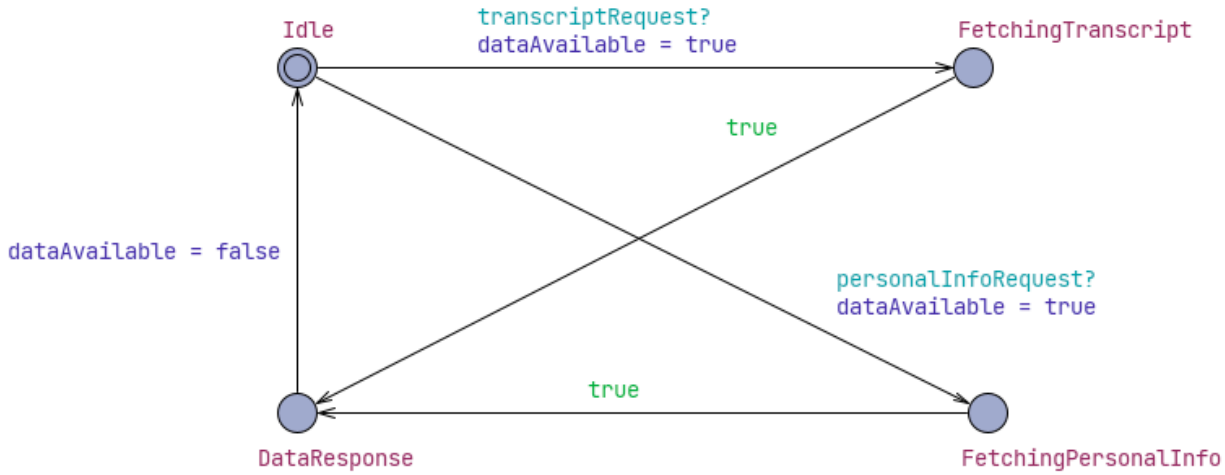
- Evaluates scholarships based on CGPA thresholds:
  i. **MCN Scholarship:** Minimum CGPA of 6.
  ii. **Merit Scholarship:** Minimum CGPA of 7.
- Processes scholarship requests from the website and communicates the decision back.
- SInce we are not dealing with user inputs, a default cgpa = 8 has been entered for checking purposes.
- The synchronisation has been done for the command to be given to the website about if the scholarship(*Merit Scholarship* and *Merit-Cum-Need Scholarship*) has been applied, whether the scholarship has been accepted or not, it ensures that website is not able to redirect to the scholarship page from the homepage login.
- After completing the process, the model automatically redirects to the idle/waiting stage.
- Ensures once a scholarship is applied, further applications are restricted.



---

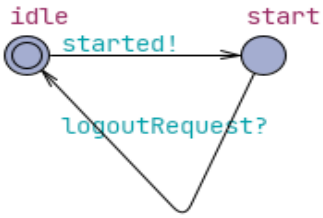
4. TRANSCRIPTINFO MODEL:

- Handles requests for academic transcripts and personal information.
- It is capable of generating the data based on the user's data in the academic server, if required.
- Generates responses based on assumed user data from the academic server.
- Ensures availability of requested data but includes a fallback state for missing information.



---

4. USER MODEL:



Handles requests for accessing the website by a user.

A very simple usage has been incorporated.

ASSUMPTIONS:

1. Every user has a unique ID and valid credentials.
    - Since this is only the model and no test cases have been passed, the assumption is made that only by going through the authentication server the credentials will be verified.
2. A user logging in through multiple devices or multiple times' record has not been ensured by the model. It is treated like a new user every time.
3. Only one student/user will interact with the model at any instance.
4. The portal ensures state synchronization for consistent user actions.
5. We are not making any assumptions on the test case and the test cases if included, must contain name, email-id, password and the cgpa.
6. Immediate server responses are assumed for simplicity; no delays or clocks are modeled.
7. Every successful login has been done by a unique student.
8. The system can support multiple users but can only allow one user at a time.
9. Default values (e.g., email, password, CGPA) are used for modeling purposes.
10. Transcript and personal information are assumed to exist unless specified otherwise.
11. No output has been shown as such on the screen since it is just a simple model to check along the path of states ensuring safety and liveness.

Liveness Properties

1) For the model ScholarshipComm there exists a path where the state of Approved is eventually reached.

E<>ScholarshipComm.Approved - This query evaluates to true.

2) For the model CredentialChecker there exists a path where the state of Success is eventually reached.

E<>CredentialChecker.Success - This query evaluates to true.

3) For all states in CredentialChecker model , if the state of Verifying is true then the state of Waiting will also be true.

A<>CredentialChecker.Verifying imply CredentialChecker.Waiting - This query evaluates to true.

4) In the model of ScholarshipComm if the state of idle is reached then this implies that the condition of scholarship applied will also be true.

E<>ScholarshipComm.Idle imply ScholarshipComm.scholarshipApplied - This query is evaluates to true.

5) If the state of login is true in the model of Website then this implies that the state of Verifying will also be true in the model of CredentialChecker.

Website.login-->CredentialChecker.Verifying - This query is true.

6) There exists a path in the model of ScholarshipComm such that it will eventually reach the state of Approved or Rejected.

E<>(ScholarshipComm.Approved || ScholarshipComm.Rejected) - This query is true.

Safety Properties

1) For all the templates there exists no deadlock.

A[] !deadlock - This query evaluates to true.

2) There exists a path in the template ScholarshipComm such that if EvaluatingMerit state is reached then at every point along the path the condition scholarshipApplied must hold true.

E[]ScholarshipComm.EvaluatingMerit imply ScholarshipComm.scholarshipApplied - This query evaluates to true.

3) Whenever the system is in the Verifying state, it must also be in the Waiting state.

A[]CredentialChecker.Verifying imply CredentialChecker.Waiting - This query evaluates to false.

4) There exists at least one path in the system where the condition scholarshipApplied remains true for every state along that path.

E[]ScholarshipComm.scholarshipApplied - This query evaluates to false.

5) There exists at least one path where, for all states along that path, if the system is in the Verifying state, then the condition is_authenticated must also be true.

E[]CredentialChecker.Verifying imply CredentialChecker.is_authenticated - This query evaluates to true.

6) Thee exists a path where, if the system is in the Verifying state, then either it will be authenticated or not be authenticated.

E[]CredentialChecker.Verifying imply (!CredentialChecker.is_authenticated || CredentialChecker.is_authenticated) - This query evaluates to true

---

Conclusion

This model effectively simulates the Scholarship Application Portal, ensuring robust functionality and adherence to safety and liveness properties. The modular design allows easy extension for additional features, such as handling multiple simultaneous sessions or integrating delay mechanisms for real-world scenarios. The verification of key properties demonstrates the model's correctness, ensuring a reliable and consistent user experience.

Just in case that the declarations are not copied in xml file, please consider adding these and then running them :

```
//System declarations:
Process = User() ;
system Process, TranscriptInfo , Website, ScholarshipComm ,  CredentialChecker;

// Place global declarations here.
chan scholarshipDecision , scholarshipApplyMerit, transcriptRequest, personalInfoRequest, loginRequest, loginResponse, logoutRequest , scholarshipApplyMCN, started;
```
