Seeko

Learning app for parents and children

Abdul Saboor
School of Electrical Engineering and Computer Science
National University of Science and Technology
Islamabad, Pakistan
asaboor.bese19seecs@seecs.edu.pk

Muhammad Jamal
School of Electrical Engineering and Computer Science
National University of Science and Technology
Islamabad, Pakistan
mjamal.bese19seecs@seecs.edu.pk

Taimoor Arshad
School of Electrical Engineering and Computer Science
National University of Science and Technology
Islamabad, Pakistan
tarshad.bese19seecs@seecs.edu.pk

Dr. Sidra Sultana
School of Electrical Engineering and Computer Science
National University of Science and Technology
Islamabad, Pakistan
sidra.sultana@seecs.edu.pk

***Abstract*—Formal modeling and verification helps in ensuring robustness of real time systems. An educative platform is modeled and verified in this paper to ensure smooth and errorless user experience.**

***Keywords*— Uppaal, System Verification, Learning System**

I. Introduction

The app, Seeko, is designed to make learning a fun experience for the parents as well as the kids. With providing a multitude of courses from different levels, the app also provides a fun and interactive learning experience along with all the necessary tests to ensure that the user has fully grasped the most important aspects of the subject. The app works on a monthly subscription system where the user has to pay in advance for the month that they want to use the app for.

In this paper we have modeled a learning application. Payment, Login, and the quiz modules are modeled to verify the safety, deadlock freeness, reachability, liveness, mutual exclusion of the system using Uppaal's verification module.

II. Literature review

Although there were some models available for payment module and log in module, but we could not find any network of automata that matched our requirements. Hence, we created all automata from scratch to help better understand the concepts of Formal Methods in real life systems.

III. SYSTEM MODELLING

In this section we give a system overview and timed automata of the modules is shown in this section.

Keep your text and graphic files separate until after the text has been formatted and styled. Do not use hard tabs, and limit use of hard returns to only one return at the end of a paragraph. Do not add any kind of pagination anywhere in

A. System Overview

As shown in the **Figure 1** our system has 3 main modules. We modeled 3 automata i) Quiz Module, ii) Login Module iii) Payment Module

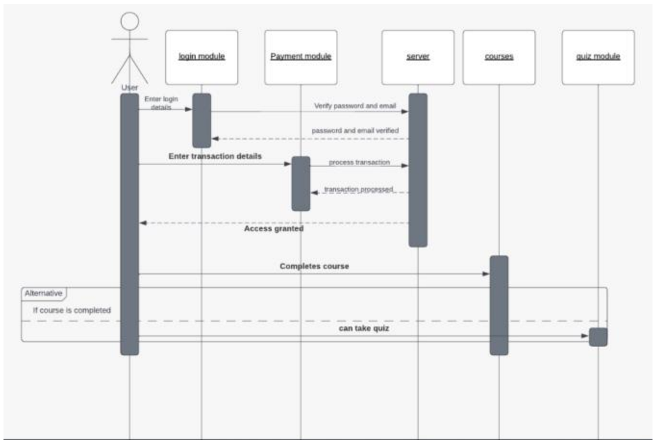

Figure 1

Users can log into the system and proceed to payment. Once their payment is received, they can access the course they bought. During the course users there are some quizzes available for the users where they can test their concepts. Users can then attempt quizzes for each course.

B. Uppaal

We used the Uppaal model checker which is integrated into the Uppaal application. This allows us to verify and model the behavior of system in real time. We can see the sequence diagram of the system as it is being executed which helps us observe every state during the execution of the system.

C. *Timed Auomata*

Uppall model checker was used to develop and test the system. User chooses the course they want to take and they are required to pay for that course. Upon verification of payment from the server user gets access to the course. Once course is finished user is required to attempt a quiz. Once user finishes the course he can move on to the next course after completing the quiz.

We created model for the payment module of the app. **Figure 2** shows the automata for this module. There are 6 states in this automaton namely *start*, *enter_card_details*, *card_verification*,*transaction_in_process*,*transaction_successful*, *access_is_granted.*

When even user wants to start a new course the payment module is started. User is then required to enter the card details and wait for verification from the server. Once verified the user is granted access to the course.

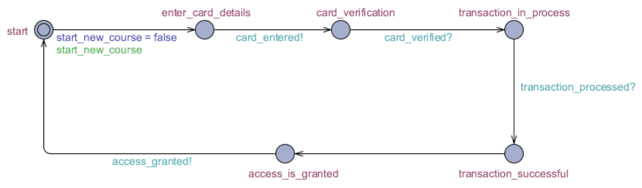

Figure 2

Figure 3 shows the automaton for the server. We simulated the server module for this course. Server automaton has 3 states. These states are *start*, *verify_card*, *verify_payment*. This module is activated once card details entered by the user need to be verified. After that transaction is processed and server approves access to the user.

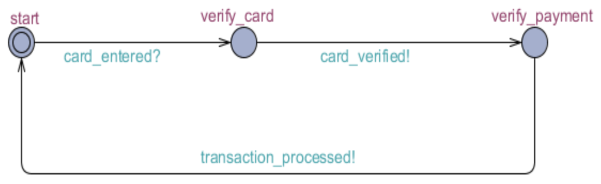

Figure 3

Figure 4 shows the automaton created for the course. This tracks the progress of the user using 4 states which are *start*, *course_started*, *course_in_progress*, *course_ended*. User starts the course when they gain access to the course when their payment is verified. Once course starts user can access the content of course. Once user complete their content, they can access the course quiz at the end of the course.

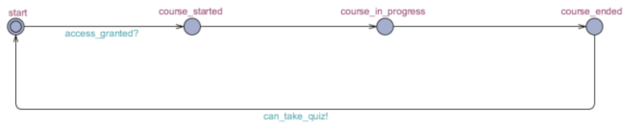

Figure 4

Figure 5 shows the automaton made for quiz module. This automaton comprises of 5 states namely *start*, *star_quiz*, *display_question*, *solve_question*, *end*. User can only start the quiz when they have been granted access to the course and they have completed the course content. Once started questions are displayed to the user. There are a total of 5 questions now in each quiz. System shows 5 questions and users is given fixed time to solve each question. Once 5 questions are passed quiz is ended. A counter variable is introduced which keeps track of the number of questions user has gone through. After the quiz the question counter is reset, and user can start a new course.

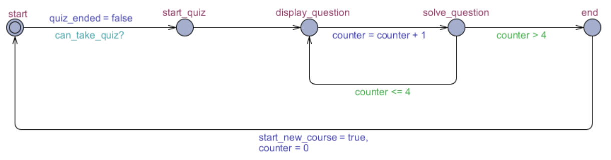

Figure 5

Figure 6.1, 6.2 show the sequence diagram obtained in Uppall. How states change during execution and what triggered the state change. System diagram is a useful tool to track any errors and deadlocks in the system. It was helpful in pointing out the deadlocks and finding out the causes of deadlocks.

Figure 6 shows the network of automata working together. Each automaton is waiting for input from previous automata, and they all work in a loop to avoid any deadlock state for the user. This is done by ensuring that system goes back to payment module after completion of quiz of the course.

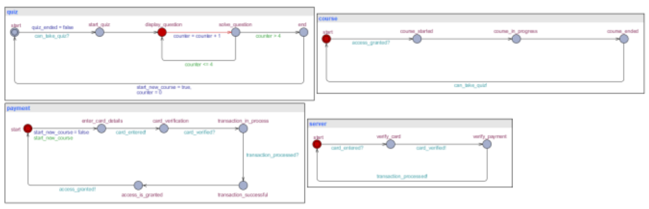

Figure 6

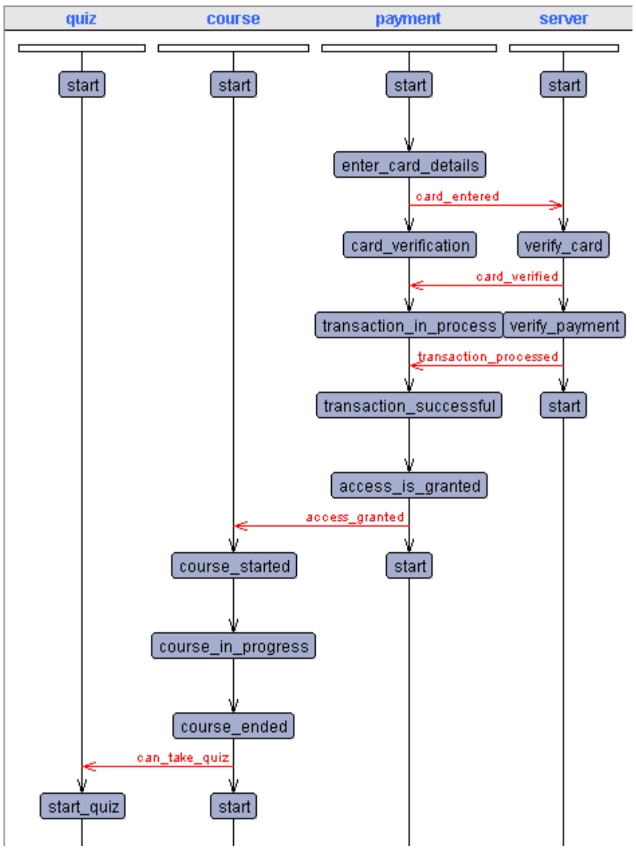

Figure 6.1

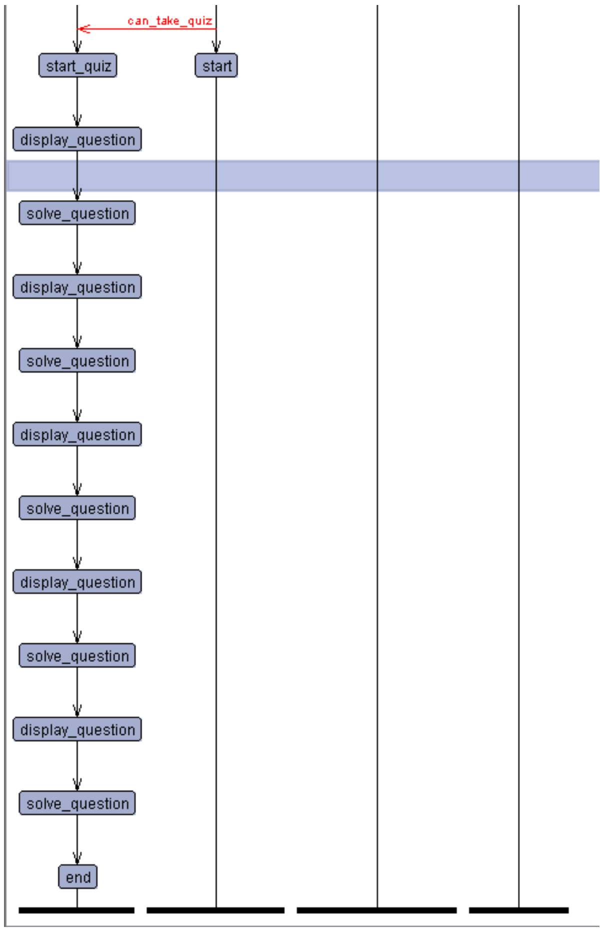

Figure 6.2

IV. System verification

This section covers some of the system properties that were verified using the built in Uppaal verifier.

A. *Deadlock freeness*

- There are no deadlocks in the system at all.
  A[] not deadlock
- This property should not hold. As there should not be any path leading to a deadlock.
  E<> deadlock

B. *Safety*

- Quiz must not start before the course ends.

  E<> (quiz.start_quiz and not(course.course_ended))
- The course should not start unless the transaction is processed.

  E<>(not(payment.transaction_successful)and course.course_started)
- The transaction process should not start unless the card is verified.

  E<>(not(server.verify_card) and payment.transaction_in_process)

C. *Liveness*

- Eventually after quiz time has passed system reaches end state.

  (E<>quiz.end)
- Once payment is verified course user will be granted access to quiz

  (E<>(payment.end) imply payment.access_granted)

D. *Bounded Response*

- The quiz counter remains in between (1,5) question.

  E<>((counter<=0 && counter >5) imply quiz.end)

Future Work

We plan to include clocks in our automata soon. This will help provide a better user experience and reduce the load on server and the application by logging off users who have remained inactive for a long time. We also plan to introduce more automata in this network to get a better model of the system and then we will be able to verify the system in much more detail.

\

Difficulties faced

We faced difficulties in introducing communication channels between the modules. We plan on to learn from our mistakes and better understand the formalism in real time applications.

Conclusion

In this paper we modeled and verified a learning application. We modeled the payment system, log in module and the quiz module. We tried to ensure that there is no chance of dead lock. This model helped us understand and see the application of Formal Methods in real life systems.

Acknowledgment

We would like to thank the startup Edwiz located in National Science and Technology Park, Islamabad who provided us with their project and allowed us to work with them. This way we were able to get firsthand experience on application of formal method on a real time application. We would also like to thank Dr.Sidra for helping and motivating us to do the project so we can understand formal methods in a better way and look into formal methods from a unique perspective.