COMP1216

Coursework 2: Formal Modelling

Group #8

Jury D’Alessio: jd3n18@soton.ac.uk
Alessandro Nerla: an1g19@soton.ac.uk
Giovanni Arcudi: ga1g19@soton.ac.uk
Alberto Tamajo: at2n19@soton.ac.uk

May 2020
Electronics and Computer Science Department
University of Southampton

INTRODUCTION

This report has been produced in order to satisfy the requirements of Coursework 2: Formal Modelling for the module COMP1206: Software Modelling and Design.
The members of the group that have worked on this coursework are the following:

Jury D'Alessio Alessandro Nerla Giovanni Arcudi Alberto Tamajo

All members of this group have equally contributed to the realisation of this report by applying the knowledge gained throughout the semester to accomplish the tasks outlined in the coursework.
The class diagram shown in the following pages has been realised with Visual Paradigm and the event-B model has been developed by using the Rodin tool. Extension refinement has been used extensively, indeed, eight machines and seven contexts have been created.
The model developed does not violate any requirement listed in the coursework description. Furthermore, the model does not violate any additional requirement that we have opted to introduce in our model.
The model verification process has been conducted by using the model checker function of ProB.
Following this, we provide a list of additional requirements that have been introduced in our model.
Afterwards, the Event-B model developed with the Rodin tool will be presented and at the end of this paper the class diagram showing the main entities of the model will be shown.

ADDITIONAL REQUIREMENTS

- **REQ22:** After a user registers, he/she shall be logged out of the platform.
- **REQ23:** Every quiz shall have exactly one creator. The creator of the quiz shall be a registered user.
- **REQ24:** Every question shall belong exactly to one quiz.
- **REQ25:** A quiz can be shared to multiple registered users.
- **REQ26:** Every question shall be matched to a position so that a quiz contains a sequence of questions.
- **REQ27:** Different questions belonging to the same quiz shall be matched to different positions.
- **REQ28:** The creator of a quiz shall not be allowed to share the quiz to (him/her)self.
- **REQ29:** The creator a quiz shall be allowed to share the quiz only if the quiz contains at least one question.
- **REQ30:** A question added to a quiz shall be automatically inserted at the end of the sequence of the quiz's questions.
- **REQ31:** The creator of a quiz shall be allowed to deshare the quiz with a registered user.
- **REQ32:** Every answer shall belong to exactly one question.
- **REQ33:** The correct answer of a question shall belong to the question.
- **REQ34:** A question shall contain at least 2 **different** answers.
- **REQ35:** A quiz instance shall have exactly one host during any of its states .
- **REQ36:** A user (registered or unregistered) shall not be allowed to be playing in two or more quiz instances at a time.
- **REQ37:** A user shall be allowed to be playing in a quiz instance only if the quiz instance is not in the following states: quizCreated, finishedQuiz.
- **REQ38:** A quiz instance which is not in the finishedQuiz state shall be linked to a quiz which contains at least one question.

- **REQ39:** It shall not be necessary for a quiz instance in the finishedQuiz state to be linked to a quiz because if the quiz is deleted, the quiz instance should be deleted as well.
- **REQ40:** The host of an active quiz instance (not in the finishedQuiz state) shall not be allowed to be playing in the quiz instance he/she is hosting.
- **REQ41:** The host of an active quiz instance shall be constantly logged in.
- **REQ42:** A quiz instance in the questioning or questionSummary state shall be matched with the current question being played.
- **REQ43:** The current question being played in a quiz instance shall belong to the quiz linked to the quiz instance.
- **REQ44:** A quiz instance which is not in the quizCreated state shall be linked to the number of questions answered up to now.
- **REQ45:** The number of answered questions of a quiz instance in the questioning or questionSummary state shall not be greater than the number of questions of the quiz the quiz instance is linked to.
- **REQ46:** A player shall be allowed to answer only the current question being played in the quiz instance he/she is playing in.
- **REQ47:** A quiz instance in the questioning state shall being played by at least one player.
- **REQ48:** The host of an active quiz instance shall not be allowed to be playing in another quiz instance.
- **REQ49:** The host of an active quiz instance shall not be allowed to log out of the system.
- **REQ50:** The host of an active quiz instance shall not be allowed to create a quiz at the same time.
- **REQ51:** The host of an active quiz instance shall not be allowed to remove a quiz at the same time.
- **REQ52:** The creator of a quiz shall not be allowed to remove a quiz if there exists an active quiz instance, linked to the quiz, being hosted at the same time.
- **REQ53:** The host of an active quiz instance shall not be allowed to share a quiz at the same time.
- **REQ54:** The host of an active quiz instance shall not be allowed to add a question to a quiz at the same time.
- **REQ55:** The creator of a quiz shall not be allowed to add a question to the quiz if there exists an active quiz instance, linked to the quiz, being hosted at the same time.
- **REQ56:** The host of an active quiz instance shall not be allowed to add an answer to a question of a quiz at the same time.
- **REQ57:** The creator of a quiz shall not be allowed to add an answer to a question of the quiz if there exists an active quiz instance, linked to the quiz, being hosted at the same time.
- **REQ58:** The host of an active quiz instance shall not be allowed to remove an answer from a question of a quiz at the same time.
- **REQ59:** The creator of a quiz shall not be allowed to remove an answer from a question of the quiz if there exists an active quiz instance, linked to the quiz, being hosted at the same time.
- **REQ60:** The host of an active quiz instance shall not be allowed to set the correct answer of a question of a quiz at the same time.
- **REQ61:** The creator of a quiz shall not be allowed to set the correct answer of a question of the quiz if there exists an active quiz instance, linked to the quiz, being hosted at the same time.

- **REQ62:** The host of an active quiz instance shall not be allowed to update an answer of a question of a quiz at the same time.
- **REQ63:** The creator of a quiz shall not be allowed to update an answer of a question of the quiz if there exists an active quiz instance, linked to the quiz, being hosted at the same time.
- **REQ64:** The host of an active quiz instance shall not be allowed to remove a question of a quiz at the same time.
- **REQ65:** The creator of a quiz shall not be allowed to remove a question from the quiz if there exists an active quiz instance, linked to the quiz, being hosted at the same time.
- **REQ66:** The host of an active quiz instance shall not be allowed to update a question of a quiz at the same time.
- **REQ67:** The creator of a quiz shall not be allowed to update a question of the quiz if there exists an active quiz instance, linked to the quiz, being hosted at the same time.
- **REQ68:** A player of a quiz instance shall be allowed to select an answer multiple times while the quiz instance is in the questioning state.
- **REQ69:** A player of a quiz instance shall not be allowed to select an answer when all other players have answered because the quiz instance passes to the questionSummary state.
- **REQ70:** The host of a quiz instance in the questioning state shall not be allowed to end prematurately a question if all players have answered because the quiz instance passes to the questionSummary state.
- **REQ71:** The host of a quiz instance in the questionSummary state shall have the possibility not to show the question summary to the players.
- **REQ72:** The host of a quiz instance in the questionSummary state shall be allowed to show the question summary only if at least 1 player is playing in the quiz instance.
- **REQ73:** The host of a quiz instance in the questionSummary state shall be allowed to proceed with the next question only if at least one player is playing AND there exists a next question.
- **REQ74:** The host of a quiz instance in the questionSummary state shall have the possibility not to show the quiz summary (report) to the players.
- **REQ75:** The host of a quiz instance in the quizSummary state shall be allowed to show the quiz summary (report) only if at least 1 player is playing in the quiz instance.
- **REQ76:** The creator of a quiz shall not be allowed to deshare a quiz with another registered user if the latter is hosting a quiz instance linked to the quiz.
- **REQ77:** The creator of a quiz shall not be allowed to deshare a quiz if he/she is hosting an active quiz instance.
- **REQ78:** The host of a quiz instance in the finishedQuiz state can remove the quiz instance only if he/she is not hosting any active quiz instance.
- **REQ79:** A quiz instance must be linked to exactly one report.
- **REQ80:** The statistics of the answered questions of a quiz instance not in the finishedQuiz state shall be stored in as many question summaries as the number of answered questions.
- **REQ81:** The final report of a quiz instance in the quizSummary state shall be generated by using the question summaries previously generated.
- **REQ82:** The question summaries shall be discarded when the quizInstance passes to the finishedQuiz state.
- **REQ83:** A question shall be matched to a time within which players can select an answer during a quiz instance.
- **REQ84:** The time of a question shall vary between 1 and 10 minutes.

- **REQ85:** A quiz instance in the questioning state shall be matched with the time when it has transitioned into the questioning state.
- **REQ86:** A quiz instance in the questioning state shall be matched with the time when it will transition into the questionSummary state.
- **REQ87:** The time n when a quiz instance in the questioning state transitions into the questionSummary state shall be equal to -> n= startingTime + question time of the current question being played
- **REQ88:** A player shall be allowed to answer a question only within the time of the question.
- **REQ89:** The host of a quiz instance in the questioning state shall be allowed to end a question prematurately only if the question time is not up.
- **REQ90:** A registered user that has lost access to a quiz, because the creator has unhared it, shall be allowed to have access to the quiz instances that has previously run on that quiz.
- **REQ91:** A user shall not be allowed to register into the platform if he/she is playing in a quiz instance.
- **REQ92:** A registered user shall not be allowed to log in the platform if he/she is playing in a quiz instance.
- **REQ93:** A registered user shall not be allowed to log out of the platform if he/she is playing in a quiz instance.

FURTHER EXPLANATIONS

- It is important to notice that while a quiz instance not in the finishedQuiz state must me matched to exactly one quiz, a quiz instance in the finishedQuiz state may be matched to 0 or 1 quiz. This choice has been made because if there existed a composition relation between a quiz and a quiz instance in the finishedQuiz state then the deletion of a quiz would lead to the deletion of all its associated quiz instances. However, this would lead to a loss of precious information as a quiz instance in the finishedQuiz state is the bridge through which a host can have access to the generated report.

- We have opted to split REQ18 into two requirements. Indeed, when a question is finished and its associated quiz instance transitions from the questioning state to the question summary state, a question summary is generated and shown to **only the host of the quiz instance.** We have not created a specific event for showing the question summary to the host as when a quiz instance transitions to the question summary state a question summary is generated and so shown to the host. Additionally, we have added an event that lets the host decide whether to show the question summary to the players or not. In other words, the system is capable of showing the question summary both on the host device and on the players' devices. The same holds for the final quiz summary which is shown when the quiz instance transitions to the quizSummary state.

- As it is possible to notice, a report does not have any direct relation with a quiz and all its components (questions and answers). This choice has been made because if a report was linked to a quiz and all its components then if a quiz or one of its components changed then the report would refer to erratic information. For example, if the answer text of one question changed then the report would refer to an answer that did not exist when the quiz instance was run. Thus, a report must not be dynamically generated when a host needs to analyze it but must be a static file. The decision of what type of file to use for the report should be postponed to later stages of software

development. It is possible to have access to a report by accessing a quiz instance that has been previously run as a report is associated with a quiz instance by mean of a one-to-one relation.

- A report is created when a quiz instance transitions to the quizSummary state. A report file is created by merging together the previously created question summaries which, each of them, collected the performance of the players on a specific question.

- The host of a running quiz instance can terminate the quiz instance at any given time. It is possible to directly transition from the quizCreated or the quizInit state to the finishedQuiz state. On the other hand, if a quiz instance is in the questioning state, the quiz instance cannot transition directly to the finishedQuiz state but must pass through the following states: questionSummary, quizSummary. A quiz instance in the

- questionSummary state can be terminated by the host but must transition to the quizSummary state before being In the finishedQuiz state.
- A quiz instance is in the quizCreated state immediately after a registered user having access to a quiz starts hosting.

- A quiz instance is in the quizInit state when players are allowed to join the quiz instance.

- A quiz instance is in the questioning state when players are allowed to answer a question.

- A quiz instance is in the questionSummary state immediately after a question has ended. A question summary is shown to the host. The host can decide to show the question summary to the players of the quiz instance.

- A quiz instance is in the quizSummary state whenever the host has decided to end the quiz instance or all questions of the quiz have been answered. A quiz summary is shown to the host. The host can decide to show the quiz summary to the players of the quiz instance.

- A quiz instance is in the finishedQuiz state immediately after the quizSummary state. In other words, a quiz instance in the finishedQuiz state is not active.

Contents

CONTEXT **c0** **2**

CONTEXT **c1** **3**

CONTEXT **c2** **4**

CONTEXT **c3** **5**

CONTEXT **c4** **6**

CONTEXT **c5** **7**

CONTEXT **c6** **8**

MACHINE **m1** **9**

MACHINE **m2** **10**

MACHINE **m3** **11**

MACHINE **m4** **13**

MACHINE **m5** **18**

MACHINE **m8** **26**

MACHINE **m9** **44**

MACHINE **m10** **63**

CONTEXT c0
SETS
USER
END

CONTEXT c1
EXTENDS c0
SETS
PASSWORD
END

CONTEXT c2
EXTENDS c1
SETS
QUIZ
QUESTION
CONSTANTS
POSITION
AXIOMS
axm21: $POSITION = \mathbb{N}$
END

CONTEXT c3
EXTENDS c2
SETS
ANSWER
END

CONTEXT c4
EXTENDS c3
SETS
INSTANCE_QUIZ
END

CONTEXT c5
EXTENDS c4
SETS
QUESTION_SUMMARY
REPORT
END

CONTEXT c6
EXTENDS c5
CONSTANTS
QUESTION_TIME
AXIOMS
axm61: $QUESTION\_TIME \subseteq \mathbb{N}_1$
axm62: $QUESTION\_TIME = \{1, 2, 3, 4, 5, 6, 7, 8, 9, 10\}$
The time of a question shall vary between 1 and 10 minutes.
END

MACHINE m1
SEES c0
VARIABLES
registeredUser Represents the set of registered users
INVARIANTS
inv1; : $registeredUser \subseteq USER$
The registeredUser set must be a subset of the carrier set USER
EVENTS
Initialisation
begin
act1; : $registeredUser := \varnothing$
end
Event REGISTER ⟨ordinary⟩ $\widehat{=}$
Event that lets a non-registered user register
any
u A user
where
grd1; : $u \in USER \setminus registeredUser$
The user must not be already registered
then
act1; : $registeredUser := registeredUser \cup \{u\}$
The user is added to the set of registered users
end
END

MACHINE m2
REFINES m1
SEES c1
VARIABLES
registeredUser
password
INVARIANTS
inv21;: $password \in registeredUser \rightarrow PASSWORD$
password is a total function from the set of registered users to the carrier set PASSWORD
EVENTS
Initialisation ⟨extended⟩
begin
act1;: $registeredUser := \varnothing$
act21;: $password := \varnothing$
end
Event REGISTER ⟨ordinary⟩ $\widehat{=}$
extends REGISTER
any
u A user
p
where
grd1;: $u \in USER \setminus registeredUser$
The user must not be already registered
grd21;: $p \in PASSWORD$
then
act1;: $registeredUser := registeredUser \cup \{u\}$
The user is added to the set of registered users
act21;: $password(u) := p$
The password of user u is p
end
END

MACHINE m3
REFINES m2
SEES c1
VARIABLES
registeredUser
password
loggedIn
loggedOut
INVARIANTS
inv31;: $partition(registeredUser, loggedIn, loggedOut)$
All registered users must be either logged in or logged out
EVENTS
Initialisation ⟨extended⟩
begin
act1;: $registeredUser := \varnothing$
act21;: $password := \varnothing$
act31;: $loggedIn := \varnothing$
act32;: $loggedOut := \varnothing$
end
Event REGISTER ⟨ordinary⟩ $\widehat{=}$
extends REGISTER
any
u A user
p
where
grd1;: $u \in USER \setminus registeredUser$
The user must not be already registered
grd21;: $p \in PASSWORD$
then
act1;: $registeredUser := registeredUser \cup \{u\}$
The user is added to the set of registered users
act21;: $password(u) := p$
The password of user u is p
act31;: $loggedOut := loggedOut \cup \{u\}$
When a user u registers, u is added to the set of loggedOut users
end
Event LOGIN ⟨ordinary⟩ $\widehat{=}$
REQ3
any
u
p
where
inv31;: $u \in loggedOut$
inv32;: $p = password(u)$
p must be the pssword of u
then
act31;: $loggedIn := loggedIn \cup \{u\}$
The user u is added to the set of loggedIn users
act32;: $loggedOut := loggedOut \setminus \{u\}$
The user u is removed from the set of loggedOut users
end
Event LOGOUT ⟨ordinary⟩ $\widehat{=}$
REQ4
any
u
where

inv31;: $u \in loggedIn$
The user u mst be logged in

then

act31;: $loggedIn := loggedIn \setminus \{u\}$
The user u is removed from the set of loggedIn users

act32;: $loggedOut := loggedOut \cup \{u\}$
The user u is added to the set of loggedOut users

end

END

MACHINE m4
REFINES m3
SEES c2
VARIABLES

registeredUser
password
loggedIn
loggedOut
quizzes
questions
quizCreator
belongingQuiz
sharedTo
questionPosition

INVARIANTS

inv41;: $quizzes \subseteq QUIZ$
quizzes is a subset of the carrier set QUIZ

inv42;: $questions \subseteq QUESTION$
questions is a subset of the carrier set QUESTION

inv43;: $quizCreator \in quizzes \rightarrow registeredUser$
quizCreator is a total function from quizzes to registeredUser because every quiz must be created by one registered user

inv44;: $belongingQuiz \in questions \rightarrow quizzes$
It is a total function because every question belongs to just one quiz

inv45;: $sharedTo \in quizzes \leftrightarrow registeredUser$
It is a many-to-many relation because a quiz can be shared to many registered users and a registered user can have access to many shared quizzes

inv46;: $questionPosition \in questions \rightarrow POSITION$
It is a total function because every question must have a position that indicates the postion of the question inside the belonging quiz

inv47;: $\forall a, b, c \cdot (a \in questions \land b \in questions \land a \neq b \land a \notin \varnothing \land b \notin \varnothing \land c \in quizzes \land belongingQuiz(a) = c \land belongingQuiz(b) = c) \Rightarrow questionPosition(a) \neq questionPosition(b)$
Different questions belonging to the same quiz shall be matched to different positions.

inv48: $\forall q, u \cdot q \in quizzes \land u \in registeredUser \land q \mapsto u \in sharedTo \Rightarrow u \neq quizCreator(q)$
The creator of a quiz shall not be allowed to share a quiz to (him/her)self

EVENTS

Initialisation ⟨extended⟩

begin

act1;: $registeredUser := \varnothing$
act21;: $password := \varnothing$
act31;: $loggedIn := \varnothing$
act32;: $loggedOut := \varnothing$
act41;: $quizzes := \varnothing$
act42;: $questions := \varnothing$
act43;: $quizCreator := \varnothing$
act44;: $belongingQuiz := \varnothing$
act45;: $sharedTo := \varnothing$
act46;: $questionPosition := \varnothing$

end

Event REGISTER ⟨ordinary⟩ $\widehat{=}$
extends REGISTER

any

u A user
p

where

grd1;: $u \in USER \setminus registeredUser$
The user must not be already registered
grd21;: $p \in PASSWORD$

then

act1;: $registeredUser := registeredUser \cup \{u\}$
The user is added to the set of registered users
act21;: $password(u) := p$
The password of user u is p
act31;: $loggedOut := loggedOut \cup \{u\}$
When a user u registers, u is added to the set of loggedOut users

end

Event LOGIN ⟨ordinary⟩ $\widehat{=}$
extends LOGIN

any

u
p

where

inv31;: $u \in loggedOut$
inv32;: $p = password(u)$
p must be the pssword of u

then

act31;: $loggedIn := loggedIn \cup \{u\}$
The user u is added to the set of loggedIn users
act32;: $loggedOut := loggedOut \setminus \{u\}$
The user u is removed from the set of loggedOut users

end

Event LOGOUT ⟨ordinary⟩ $\widehat{=}$
extends LOGOUT

any

u

where

inv31;: $u \in loggedIn$
The user u mst be logged in

then

act31;: $loggedIn := loggedIn \setminus \{u\}$
The user u is removed from the set of loggedIn users
act32;: $loggedOut := loggedOut \cup \{u\}$
The user u is added to the set of loggedOut users

end

Event CREATEQUIZ ⟨ordinary⟩ $\widehat{=}$
REQ5

any

u
q

where

grd41;: $u \in loggedIn$
A user must be logged in so that to create a quiz
grd42;: $q \in QUIZ \setminus quizzes$
The created quiz must not have been already created

then

act41;: $quizzes := quizzes \cup \{q\}$
The created quiz is added to the dynamic set quizzes
act42;: $quizCreator(q) := u$
the creator of the quiz q is u

end

Event REMOVEQUIZ ⟨ordinary⟩ $\widehat{=}$
REQ10

any

u
q

where

grd41;: $q \in quizzes$
grd42;: $u \in loggedIn$
The user must be logged in so that to remove a quiz
grd43;: $quizCreator(q) = u$
Only the creator of the quiz can remove the quiz

then

act41;: $quizzes := quizzes \setminus \{q\}$
the quiz is removed
act42;: $quizCreator := \{q\} \lhd quizCreator$
The quiz does not have anymore a creator because is removed
act43;: $sharedTo := \{q\} \lhd sharedTo$
All registerd users that had access to the quiz, do not have access to it anymore because the quiz is removed
act44;: $belongingQuiz := belongingQuiz \rhd \{q\}$
The questions of the quiz do not belong anymore to it
act45;: $questions := questions \setminus \{quest | quest \in questions \land belongingQuiz(quest) = q\}$
The questions of the quiz are removed
act46;: $questionPosition := \{quest | quest \in questions \land belongingQuiz(quest) = q\} \lhd questionPosition$

The position of the questions is removed

end

Event SHAREQUIZ ⟨ordinary⟩ $\widehat{=}$

REQ13

any

u1
u2
q

where

grd41;: $u1 \in loggedIn$
the user must be logged in so that to share the quiz
grd42;: $u2 \in registeredUser$
the other user that will have access to the quiz
grd43;: $u1 \neq u2$
the two users cannot be the same person
grd44;: $q \in quizzes$
grd45;: $quizCreator(q) = u1$
u1 must be the creator of the quiz
grd46: $q \in ran(belongingQuiz)$
The quiz must contain at least 1 question in order to be shared
grd47: $q \mapsto u2 \notin sharedTo$
the quiz must not have already been shared with u2

then

act41;: $sharedTo := sharedTo \cup \{q \mapsto u2\}$

end

Event CREATEQUESTION ⟨ordinary⟩ $\widehat{=}$

REQ7

any

quest
q
u
p

where

grd41;: $quest \in QUESTION \setminus questions$
The question must be a new question
grd42;: $q \in quizzes$
grd43;: $u \in loggedIn$
The user must be logged in in order to create a new question

grd44: $quizCreator(q) = u$
The creator of the quiz must be u
grd45: $p \in POSITION$
grd46: $\forall z \cdot z \in questions \land belongingQuiz(z) = q \Rightarrow p > questionPosition(z)$
The question must be added at the end of the question list of the quiz

then

act41: $questions := questions \cup \{quest\}$
act42: $belongingQuiz(quest) := q$
the question quest belongs to q
act43: $questionPosition(quest) := p$
The position of the question quest is p inside the question list of the quiz q

end

Event REMOVEQUESTION ⟨ordinary⟩ $\widehat{=}$

REQ9

any

u
quest
q

where

grd41: $u \in loggedIn$
The user must be logged in order to remove a question
grd42: $q \in quizzes$
grd43: $quest \in questions$
grd45: $belongingQuiz(quest) = q$
q must be the belonging quiz of quest
grd44: $quizCreator(q) = u$
The creator of the quiz q must be u

then

act41: $questions := questions \setminus \{quest\}$
act42: $belongingQuiz := \{quest\} \lhd belongingQuiz$
act43: $questionPosition := \{quest\} \lhd questionPosition$
The position of the question is removed

end

Event UPDATE_QUESTION_OK ⟨ordinary⟩ $\widehat{=}$

REQ8 this event updates just the text of the question

any

u
quest
q
result

where

grd41: $u \in loggedIn$
grd42: $q \in quizzes$
grd43: $quizCreator(q) = u$
Only the quiz creator of q can update the question
grd44: $quest \in questions$
grd45: $result = TRUE$
The update has been succesfully
grd46: $belongingQuiz(quest) = q$
the question must belong to q

then

skip

end

Event UPDATE_QUESTION_NOT_OK ⟨ordinary⟩ $\widehat{=}$

REQ8 this event updates just the text of the question

any

u
quest
q

result

where

grd41;: $u \in loggedIn$

grd42;: $q \in quizzes$

grd43;: $quizCreator(q) = u$

grd1: $quest \in questions$

grd45;: $belongingQuiz(quest) = q$

grd46;: $result = FALSE$

The update has not been successfully. The previous text of the question is restored

then

skip

end

Event UNSHARE_QUIZ ⟨ordinary⟩ $\widehat{=}$

any

u1

u2

q

where

grd41: $u1 \in registeredUser$

grd42: $u2 \in registeredUser$

grd43: $u1 \in loggedIn$

A registerd user can unshare only if he/she is online

grd44: $u1 \neq u2$

grd45: $q \mapsto u2 \in sharedTo$

u2 had to have access to the quiz

grd46: $u1 = quizCreator(q)$

Only the creator of the quiz can unshare a quiz

then

act41: $sharedTo := sharedTo \setminus \{q \mapsto u2\}$

end

END

MACHINE m5
REFINES m4
SEES c3
VARIABLES

registeredUser
password
loggedIn
loggedOut
quizzes
questions
quizCreator
belongingQuiz
sharedTo
questionPosition
answers
correctAnswer
belongingQuestion

INVARIANTS

inv51;: $answers \subseteq ANSWER$

inv52;: $finite(answers)$

inv53;: $belongingQuestion \in answers \rightarrow questions$
Every answer shall belong to exactly one question.

inv54;: $correctAnswer \in questions \rightarrowtail answers$
Every question must have a correct answer and an answer can be the correct answer of at most 1 question (TOTAL INJECTIVITY)

inv55;: $\forall quest \cdot quest \in questions \Rightarrow card(\{a | a \in answers \land belongingQuestion(a) = quest\}) \geq 2 \land card(\{a | a \in answers \land belongingQuestion(a) = quest\}) \leq 4$
Every question must have at least 2 answers and at most 4 answers

inv56: $\forall a, quest \cdot a \in answers \land quest \in questions \land correctAnswer(quest) = a \Rightarrow belongingQuestion(a) = quest$
The correct answer of a question shall belong to the question.

inv58: $correctAnswer \subseteq belongingQuestion^{-1}$
Actually this is a redundant invariant that has the same meaning as the invariant above

EVENTS

Initialisation ⟨extended⟩

begin

act1;: $registeredUser := \varnothing$
act21;: $password := \varnothing$
act31;: $loggedIn := \varnothing$
act32;: $loggedOut := \varnothing$
act41;: $quizzes := \varnothing$
act42;: $questions := \varnothing$
act43;: $quizCreator := \varnothing$
act44;: $belongingQuiz := \varnothing$
act45;: $sharedTo := \varnothing$
act46;: $questionPosition := \varnothing$
act51;: $answers := \varnothing$
act52;: $correctAnswer := \varnothing$
act53;: $belongingQuestion := \varnothing$

end

Event REGISTER ⟨ordinary⟩ $\widehat{=}$
extends REGISTER

any

u A user
p

where

grd1;: $u \in USER \setminus registeredUser$

The user must not be already registered

grd21;: $p \in PASSWORD$

then

act1;: $registeredUser := registeredUser \cup \{u\}$

The user is added to the set of registered users

act21;: $password(u) := p$

The password of user u is p

act31;: $loggedOut := loggedOut \cup \{u\}$

When a user u registers, u is added to the set of loggedOut users

end

Event LOGIN ⟨ordinary⟩ $\widehat{=}$

extends LOGIN

any

u

p

where

inv31;: $u \in loggedOut$

inv32;: $p = password(u)$

p must be the pssword of u

then

act31;: $loggedIn := loggedIn \cup \{u\}$

The user u is added to the set of loggedIn users

act32;: $loggedOut := loggedOut \setminus \{u\}$

The user u is removed from the set of loggedOut users

end

Event LOGOUT ⟨ordinary⟩ $\widehat{=}$

extends LOGOUT

any

u

where

inv31;: $u \in loggedIn$

The user u mst be logged in

then

act31;: $loggedIn := loggedIn \setminus \{u\}$

The user u is removed from the set of loggedIn users

act32;: $loggedOut := loggedOut \cup \{u\}$

The user u is added to the set of loggedOut users

end

Event CREATEQUIZ ⟨ordinary⟩ $\widehat{=}$

extends CREATEQUIZ

any

u

q

where

grd41;: $u \in loggedIn$

A user must be logged in so that to create a quiz

grd42;: $q \in QUIZ \setminus quizzes$

The created quiz must not have been already created

then

act41;: $quizzes := quizzes \cup \{q\}$

The created quiz is added to the dynamic set quizzes

act42;: $quizCreator(q) := u$

the creator of the quiz q is u

end

Event REMOVEQUIZ ⟨ordinary⟩ $\widehat{=}$

extends REMOVEQUIZ

any

u

q

where

grd41;: $q \in quizzes$

grd42;: $u \in loggedIn$

The user must be logged in so that to remove a quiz

grd43;: $quizCreator(q) = u$

Only the creator of the quiz can remove the quiz

then

act41;: $quizzes := quizzes \setminus \{q\}$

the quiz is removed

act42;: $quizCreator := \{q\} \lhd quizCreator$

The quiz does not have anymore a creator because is removed

act43;: $sharedTo := \{q\} \lhd sharedTo$

All registerd users that had access to the quiz, do not have access to it anymore because the quiz is removed

act44;: $belongingQuiz := belongingQuiz \rhd \{q\}$

The questions of the quiz do not belong anymore to it

act45;: $questions := questions \setminus \{quest | quest \in questions \land belongingQuiz(quest) = q\}$

The questions of the quiz are removed

act46;: $questionPosition := \{quest | quest \in questions \land belongingQuiz(quest) = q\} \lhd questionPosition$

The position of the questions is removed

act54;: $correctAnswer := \{q2 | q2 \in questions \land belongingQuiz(q2) = q\} \lhd correctAnswer$

The correct answer of all questions of the quiz must be removed

act55;: $belongingQuestion := belongingQuestion \rhd \{q2 | q2 \in questions \land belongingQuiz(q2) = q\}$

The answers of the questions of the quiz do not belong to the questions anymore

act56: $answers := answers \setminus \{a1 | a1 \in answers \land belongingQuestion(a1) \in \{quest1 | quest1 \in questions \land belongingQuiz(quest1) = q\}\}$

Removing a quiz involves removing all answers of the questions of the quiz

end

Event SHAREQUIZ ⟨ordinary⟩ $\widehat{=}$

extends SHAREQUIZ

any

$u1$

$u2$

q

where

grd41;: $u1 \in loggedIn$

the user must be logged in so that to share the quiz

grd42;: $u2 \in registeredUser$

the other user that will have access to the quiz

grd43;: $u1 \neq u2$

the two users cannot be the same person

grd44;: $q \in quizzes$

grd45;: $quizCreator(q) = u1$

u1 must be the creator of the quiz

grd46: $q \in ran(belongingQuiz)$

The quiz must contain at least 1 question in order to be shared

grd47: $q \mapsto u2 \notin sharedTo$

the quiz must not have already been shared with u2

then

act41;: $sharedTo := sharedTo \cup \{q \mapsto u2\}$

end

Event CREATEQUESTION ⟨ordinary⟩ $\widehat{=}$

extends CREATEQUESTION

any

$quest$

q
u
p
a
b

where

grd41;: $quest \in QUESTION \setminus questions$
The question must be a new question
grd42;: $q \in quizzes$
grd43;: $u \in loggedIn$
The user must be logged in in order to create a new question
grd44;: $quizCreator(q) = u$
The creator of the quiz must be u
grd45;: $p \in POSITION$
grd46;: $\forall z \cdot z \in questions \land belongingQuiz(z) = q \Rightarrow p > questionPosition(z)$
The question must be added at the end of the question list of the quiz
grd51;: $(a \in ANSWER \setminus answers) \land (b \in ANSWER \setminus answers)$
The 2 answers must be new ones
grd52: $a \neq b$
The 2 answers must be different

then

act41;: $questions := questions \cup \{quest\}$
act42;: $belongingQuiz(quest) := q$
the question quest belongs to q
act43;: $questionPosition(quest) := p$
The position of the question quest is p inside the question list of the quiz q
act51;: $answers := answers \cup \{a, b\}$
act53;: $correctAnswer(quest) := a$
The correct answer is a
act54;: $belongingQuestion := belongingQuestion \cup \{a \mapsto quest, b \mapsto quest\}$
The 2 new answers belong to the question quest

end

Event ADDANSWER ⟨ordinary⟩ $\widehat{=}$
REQ7

any

quest
q
u
a

where

grd51;: $u \in loggedIn$
The user us must be logged in
grd52;: $q \in quizzes$
grd53;: $a \in ANSWER \setminus answers$
grd54;: $quest \in questions$
grd55;: $quizCreator(q) = u$
The user u must be the creator of the quiz
grd56;: $belongingQuiz(quest) = q$
grd57;: $card(\{a1 | a1 \in answers \land belongingQuestion(a1) = quest\}) < 4$
The current answers of the question must be less than 4

then

act51;: $answers := answers \cup \{a\}$
act52;: $belongingQuestion(a) := quest$

end

Event REMOVENOTCORRECTANSWER ⟨ordinary⟩ $\widehat{=}$

any

quest
q
u

a

where

grd51;: $u \in loggedIn$

The user u must be logged in

grd52;: $q \in quizzes$

grd53;: $quest \in questions$

grd54;: $a \in answers$

grd55;: $card(\{a1 | a1 \in answers \land belongingQuestion(a1) = quest\}) > 2$

The number of current answers of the question must be more than 2

grd56;: $correctAnswer(quest) \neq a$

The answer to be removed must not be the correct answer of the question

grd57: $quizCreator(q) = u$

The user u must be the creator of the quiz

grd58: $belongingQuiz(quest) = q$

grd59: $belongingQuestion(a) = quest$

The anser must belong to the quiz q

then

act51;: $answers := answers \setminus \{a\}$

act52;: $belongingQuestion := \{a\} \lhd belongingQuestion$

end

Event REMOVECORRECTANSWER ⟨ordinary⟩ $\widehat{=}$

any

quest

q

u

a1

a2

where

grd51;: $u \in loggedIn$

The user u must be logged in

grd52;: $quest \in questions$

grd53;: $q \in quizzes$

grd54;: $a1 \in answers$

grd55;: $a2 \in answers$

grd56;: $correctAnswer(quest) = a1$

The answer to be removed must be the correct answer of the question

grd57;: $belongingQuestion(a2) = quest$

The new correct answer must belong to the question

grd58: $quizCreator(q) = u$

The user u must be the creator of the quiz

grd59: $belongingQuiz(quest) = q$

grd60: $belongingQuestion(a1) = quest$

grd61: $card(\{a | a \in answers \land belongingQuestion(a) = quest\}) > 2$

The current number of answers of the question must be greater than 2

grd62: $a1 \neq a2$

then

act51;: $answers := answers \setminus \{a1\}$

act52;: $belongingQuestion := \{a1\} \lhd belongingQuestion$

act53;: $correctAnswer := correctAnswer \lhd\!\!+ \{quest \mapsto a2\}$

the new coorect answer is a2

end

Event SETCORRECTANSWER ⟨ordinary⟩ $\widehat{=}$

any

u

q

quest

a1

a2

where

grd51;: $u \in loggedIn$
The user u must be loggged in
grd52;: $q \in quizzes$
grd53;: $quest \in questions$
grd54;: $a1 \in answers$
grd55;: $a2 \in answers$
grd56;: $a1 \neq a2$
grd57: $quizCreator(q) = u$
The user u must be the creator of the quiz
grd58: $belongingQuiz(quest) = q$
grd59: $correctAnswer(quest) = a1$
a1 must be the current correct answer of the question
grd60: $belongingQuestion(a2) = quest$
a2 must belong to the question

then

act51;: $correctAnswer := correctAnswer \lhd\!\!+ \{quest \mapsto a2\}$

end

Event UPDATE_ANSWER_OK ⟨ordinary⟩ $\widehat{=}$

any

a
q
quest
result
u

where

grd51;: $u \in loggedIn$
grd52;: $a \in answers$
grd53;: $q \in quizzes$
grd54;: $quest \in questions$
grd55;: $belongingQuestion(a) = quest$
grd56: $quizCreator(q) = u$
grd57: $belongingQuiz(quest) = q$
grd58: $result = TRUE$
The update has been successful

then

skip

end

Event UPDATE_ANSWER_NOT_OK ⟨ordinary⟩ $\widehat{=}$

The event updates only the text of an answer

any

a
q
quest
result
u

where

grd51;: $u \in loggedIn$
the user u must be logged in
grd52;: $a \in answers$
grd53;: $quest \in questions$
grd54;: $q \in quizzes$
grd55;: $belongingQuestion(a) = quest$
grd56;: $quizCreator(q) = u$
u must be the creator of the quiz
grd58;: $belongingQuiz(quest) = q$
grd57;: $result = FALSE$
The update has not been successful. The previous text is restored

then

skip

end

Event REMOVEQUESTION ⟨ordinary⟩ ≙

extends REMOVEQUESTION

any

u

$quest$

q

where

grd41;: $u \in loggedIn$

The user must be logged in order to remove a question

grd42;: $q \in quizzes$

grd43;: $quest \in questions$

grd45;: $belongingQuiz(quest) = q$

q must be the belonging quiz of quest

grd44;: $quizCreator(q) = u$

The creator of the quiz q must be u

then

act41;: $questions := questions \setminus \{quest\}$

act42;: $belongingQuiz := \{quest\} \lhd belongingQuiz$

act43;: $questionPosition := \{quest\} \lhd questionPosition$

The position of the question is removed

act51;: $answers := answers \setminus \{a | a \in answers \land belongingQuestion(a) = quest\}$

the answers of the question must be removed when the question is removed

act52;: $correctAnswer := \{quest\} \lhd correctAnswer$

The correct answer of quest is not the correct answer anymore

act53;: $belongingQuestion := belongingQuestion \rhd \{quest\}$

All answers of the question do not belong to it anymore

end

Event UPDATE_QUESTION_OK ⟨ordinary⟩ ≙

extends UPDATE_QUESTION_OK

any

u

$quest$

q

$result$

where

grd41;: $u \in loggedIn$

grd42;: $q \in quizzes$

grd43;: $quizCreator(q) = u$

Only the quiz creator of q can update the question

grd44;: $quest \in questions$

grd45;: $result = TRUE$

The update has been succesfully

grd46;: $belongingQuiz(quest) = q$

the question must belong to q

then

$skip$

end

Event UPDATE_QUESTION_NOT_OK ⟨ordinary⟩ ≙

extends UPDATE_QUESTION_NOT_OK

any

u

$quest$

q

$result$

where

grd41;: $u \in loggedIn$

grd42;: $q \in quizzes$

grd43;: $quizCreator(q) = u$
grd1: $quest \in questions$
grd45;: $belongingQuiz(quest) = q$
grd46;: $result = FALSE$
The update has not been successfully. The previous text of the question is restored

then
skip
end

Event UNSHARE_QUIZ ⟨ordinary⟩ $\widehat{=}$
extends UNSHARE_QUIZ

any
u1
u2
q
where
grd41: $u1 \in registeredUser$
grd42: $u2 \in registeredUser$
grd43: $u1 \in loggedIn$
A registerd user can unshare only if he/she is online
grd44: $u1 \neq u2$
grd45: $q \mapsto u2 \in sharedTo$
u2 had to have access to the quiz
grd46: $u1 = quizCreator(q)$
Only the creator of the quiz can unshare a quiz
then
act41: $sharedTo := sharedTo \setminus \{q \mapsto u2\}$
end
END

MACHINE m8
REFINES m5
SEES c4
VARIABLES

registeredUser
password
loggedIn
loggedOut
quizzes
questions
quizCreator
belongingQuiz
sharedTo
questionPosition
answers
correctAnswer
belongingQuestion
quizInstances
host
playingIn
instanceOfQuiz
currentQuestion
quizCreated
quizInit
questioning
questionSummary
quizSummary
finishedQuiz
answeredQuestions
peopleAnswers
peopleSelectedAnswer

INVARIANTS

inv61;: $quizInstances \subseteq INSTANCE\_QUIZ$

inv62;: $partition(quizInstances, quizCreated, quizInit, questioning, questionSummary, quizSummary, finishedQuiz)$

It describes all the possible states of every quiz instance

inv63;: $host \in quizInstances \rightarrow registeredUser$

Every quiz instance must be hosted by exactly one registered user. Notice that this relations must exist for every possible state of a quiz instance

inv64;: $playingIn \in USER \nrightarrow quizInstances \setminus (quizCreated \cup finishedQuiz)$

A USER (registerd or not) can play in at most one quiz instance at a time (PARTIAL FUNCTION). This relation must exist for all quiz instances that are not in the following states: quizCreated,finishedQuiz

inv65;: $instanceOfQuiz \in quizInstances \nrightarrow quizzes$

It shall not be necessary for a quiz instance in the finishedQuiz state to be linked to a quiz because if the quiz is deleted, the quiz instance should be deleted as well.

inv66;: $\forall qi \cdot qi \in (quizInstances \setminus finishedQuiz) \Rightarrow host(qi) \notin playingIn^{-1}[\{qi\}]$

The host of an active quiz instance (not in the finishedQuiz state) shall not be allowed to be playing in the quiz instance he/she is hosting.

inv67;: $\forall qi, q \cdot q \in quizzes \wedge qi \in quizInstances \wedge qi \notin finishedQuiz \wedge instanceOfQuiz(qi) = q \Rightarrow host(qi) \in (\{quizCreator(q)\} \cup sharedTo[\{q\}]) \wedge host(qi) \in loggedIn$

The host of an active quiz instance, which is linked to a quiz q, must be the creator of q or must have access to q. The host of an active quiz instance must always be logged in

inv68;: $currentQuestion \in (questioning \cup questionSummary) \rightarrow questions$
A quiz instance in the questioning or questionSummary state shall be matched with the current question being played.

inv69;: $answeredQuestions \in (quizInstances \setminus quizCreated) \rightarrow \mathbb{N}$
A quiz instance which is not in the quizCreated state shall be linked to the number of questions answered up to now.

inv610;: $\forall qi, quest \cdot qi \in quizInstances \wedge qi \in (questioning \cup questionSummary) \wedge quest \in questions \wedge (qi \mapsto quest) \in currentQuestion \Rightarrow belongingQuiz(quest) = instanceOfQuiz(qi)$
The current question of any quiz instance in the questioning
and questionSummary states shall be a question of the quiz the quiz instance is linked to

inv611;: $\forall qi, q \cdot qi \in (questioning \cup questionSummary) \wedge q \in quizzes \wedge instanceOfQuiz(qi) = q \Rightarrow answeredQuestions(qi) \leq card(belongingQuiz^{-1}[\{q\}])$
The number of answered questions of a quiz instance in the questioning or questionSummary state shall not be greater than the number of questions of the quiz the quiz instance is linked to.

inv612;: $peopleAnswers \in dom(playingIn) \nrightarrow questioning$
It is a partial function from the domain of playingIn to the set questioning.
This function indicates which players have answered the current question of a quiz instance in the questioning state.

inv613;: $peopleAnswers \subseteq playingIn$
every player can answer only a question of the quiz instance he/she is playing in

inv614;: $\forall qi \cdot qi \in quizInstances \wedge qi \in questioning \Rightarrow (card(playingIn^{-1}[\{qi\}]) > 0)$
A quiz instance in the questioning state shall being played by at least one player.

inv615: $\forall qi, q \cdot qi \in quizInstances \wedge qi \notin finishedQuiz \wedge q \in quizzes \wedge instanceOfQuiz(qi) = q \Rightarrow belongingQuiz^{-1}[\{q\}] \neq \varnothing$
Every active (not finished) quiz instance must be linked to a quiz which must contain at least 1 question

inv616: $quizInstances \setminus finishedQuiz \subseteq dom(instanceOfQuiz)$

inv617: $\forall u \cdot u \in registeredUser \wedge u \in loggedIn \wedge u \in host[quizInstances \setminus finishedQuiz] \Rightarrow u \notin dom(playingIn)$
The host of an active quiz instance shall not be allowed to be playing in another quiz instance.

inv618: $peopleSelectedAnswer \in dom(playingIn) \nrightarrow answers$
It is a partial function from the domain of playinIn to answers. It indicates what answers the players of an active quiz instance have selected

inv619: $(\forall qi \cdot qi \in quizInstances \Rightarrow qi \notin questioning) \Rightarrow peopleSelectedAnswer = \varnothing$
The function is defined only when a quiz instance is in the questioning state.

inv620: $\forall p, a \cdot p \in USER \wedge p \in dom(playingIn) \wedge a \in answers \wedge p \mapsto a \in peopleSelectedAnswer \Rightarrow a \in belongingQuestion^{-1}[\{currentQuestion(playingIn(p))\}]$
The players must select an answer of the current question of the quiz instance

EVENTS

Initialisation ⟨extended⟩

begin

act1;: $registeredUser := \varnothing$
act21;: $password := \varnothing$
act31;: $loggedIn := \varnothing$
act32;: $loggedOut := \varnothing$
act41;: $quizzes := \varnothing$
act42;: $questions := \varnothing$
act43;: $quizCreator := \varnothing$
act44;: $belongingQuiz := \varnothing$
act45;: $sharedTo := \varnothing$
act46;: $questionPosition := \varnothing$
act51;: $answers := \varnothing$
act52;: $correctAnswer := \varnothing$
act53;: $belongingQuestion := \varnothing$
act61;: $quizInstances := \varnothing$
act62;: $host := \varnothing$
act63;: $playingIn := \varnothing$

act64;: $instanceOfQuiz := \varnothing$
act65;: $peopleAnswers := \varnothing$
act66: $answeredQuestions := \varnothing$
act67: $currentQuestion := \varnothing$
act68: $finishedQuiz := \varnothing$
act69: $questioning := \varnothing$
act611: $questionSummary := \varnothing$
act612: $quizCreated := \varnothing$
act613: $quizInit := \varnothing$
act614: $quizSummary := \varnothing$
act615: $peopleSelectedAnswer := \varnothing$

end

Event REGISTER ⟨ordinary⟩ $\widehat{=}$
extends REGISTER

any

u A user
p

where

grd1;: $u \in USER \setminus registeredUser$
The user must not be already registered
grd21;: $p \in PASSWORD$
grd81: $u \notin dom(playingIn)$

then

act1;: $registeredUser := registeredUser \cup \{u\}$
The user is added to the set of registered users
act21;: $password(u) := p$
The password of user u is p
act31;: $loggedOut := loggedOut \cup \{u\}$
When a user u registers, u is added to the set of loggedOut users

end

Event LOGIN ⟨ordinary⟩ $\widehat{=}$
extends LOGIN

any

u
p

where

inv31;: $u \in loggedOut$
inv32;: $p = password(u)$
p must be the pssword of u
grd81: $u \notin dom(playingIn)$

then

act31;: $loggedIn := loggedIn \cup \{u\}$
The user u is added to the set of loggedIn users
act32;: $loggedOut := loggedOut \setminus \{u\}$
The user u is removed from the set of loggedOut users

end

Event LOGOUT ⟨ordinary⟩ $\widehat{=}$
extends LOGOUT

any

u

where

inv31;: $u \in loggedIn$
The user u mst be logged in
grd81: $u \notin host[quizInstances \setminus finishedQuiz]$
The host of an active quiz instance shall not be allowed to log out of the system.
grd82: $u \notin dom(playingIn)$

then

act31;: $loggedIn := loggedIn \setminus \{u\}$
The user u is removed from the set of loggedIn users

act32;: $loggedOut := loggedOut \cup \{u\}$
The user u is added to the set of loggedOut users

end

Event CREATEQUIZ ⟨ordinary⟩ $\widehat{=}$
extends CREATEQUIZ

any
u
q

where
grd41;: $u \in loggedIn$
A user must be logged in so that to create a quiz
grd42;: $q \in QUIZ \setminus quizzes$
The created quiz must not have been already created
grd61;: $u \notin host[quizInstances \setminus finishedQuiz]$
The host of an active quiz instance shall not be allowed to create a quiz at the same time.

then
act41;: $quizzes := quizzes \cup \{q\}$
The created quiz is added to the dynamic set quizzes
act42;: $quizCreator(q) := u$
the creator of the quiz q is u

end

Event REMOVEQUIZ ⟨ordinary⟩ $\widehat{=}$
extends REMOVEQUIZ

any
u
q

where
grd41;: $q \in quizzes$
grd42;: $u \in loggedIn$
The user must be logged in so that to remove a quiz
grd43;: $quizCreator(q) = u$
Only the creator of the quiz can remove the quiz
grd61;: $u \notin host[quizInstances \setminus finishedQuiz]$
The host of an active quiz instance shall not be allowed to remove a quiz at the same time.
grd62: $\forall qi \cdot qi \in quizInstances \wedge qi \in instanceOfQuiz^{-1}[\{q\}] \Rightarrow qi \notin (quizInstances \setminus finishedQuiz)$

The creator of a quiz shall not be allowed to remove a quiz if there exists an active quiz instance, linked to the quiz, being hosted at the same time.

then
act41;: $quizzes := quizzes \setminus \{q\}$
the quiz is removed
act42;: $quizCreator := \{q\} \lhd quizCreator$
The quiz does not have anymore a creator because is removed
act43;: $sharedTo := \{q\} \lhd sharedTo$
All registerd users that had access to the quiz, do not have access to it anymore because the quiz is removed
act44;: $belongingQuiz := belongingQuiz \rhd \{q\}$
The questions of the quiz do not belong anymore to it
act45;: $questions := questions \setminus \{quest | quest \in questions \wedge belongingQuiz(quest) = q\}$
The questions of the quiz are removed
act46;: $questionPosition := \{quest | quest \in questions \wedge belongingQuiz(quest) = q\} \lhd questionPosition$

The position of the questions is removed
act54;: $correctAnswer := \{q2 | q2 \in questions \wedge belongingQuiz(q2) = q\} \lhd correctAnswer$
The correct answer of all questions of the quiz must be removed
act55;: $belongingQuestion := belongingQuestion \rhd \{q2 | q2 \in questions \wedge belongingQuiz(q2) = q\}$
The answers of the questions of the quiz do not belong to the questions anymore
act56: $answers := answers \setminus \{a1 | a1 \in answers \wedge belongingQuestion(a1) \in \{quest1 | quest1 \in questions \wedge belongingQuiz(quest1) = q\}\}$
Removing a quiz involves removing all answers of the questions of the quiz

act61: $instanceOfQuiz := instanceOfQuiz \rhd \{q\}$

end

Event SHAREQUIZ ⟨ordinary⟩ $\widehat{=}$

extends SHAREQUIZ

any

$u1$

$u2$

q

where

grd41;: $u1 \in loggedIn$

the user must be logged in so that to share the quiz

grd42;: $u2 \in registeredUser$

the other user that will have access to the quiz

grd43;: $u1 \neq u2$

the two users cannot be the same person

grd44;: $q \in quizzes$

grd45;: $quizCreator(q) = u1$

u1 must be the creator of the quiz

grd46: $q \in ran(belongingQuiz)$

The quiz must contain at least 1 question in order to be shared

grd47: $q \mapsto u2 \notin sharedTo$

the quiz must not have already been shared with u2

grd61;: $u1 \notin host[quizInstances \setminus finishedQuiz]$

The host of an active quiz instance shall not be allowed to share a quiz at the same time.

then

act41;: $sharedTo := sharedTo \cup \{q \mapsto u2\}$

end

Event CREATEQUESTION ⟨ordinary⟩ $\widehat{=}$

extends CREATEQUESTION

any

$quest$

q

u

p

a

b

where

grd41;: $quest \in QUESTION \setminus questions$

The question must be a new question

grd42;: $q \in quizzes$

grd43;: $u \in loggedIn$

The user must be logged in in order to create a new question

grd44;: $quizCreator(q) = u$

The creator of the quiz must be u

grd45;: $p \in POSITION$

grd46;: $\forall z \cdot z \in questions \land belongingQuiz(z) = q \Rightarrow p > questionPosition(z)$

The question must be added at the end of the question list of the quiz

grd51;: $(a \in ANSWER \setminus answers) \land (b \in ANSWER \setminus answers)$

The 2 answers must be new ones

grd52: $a \neq b$

The 2 answers must be different

grd61;: $u \notin host[quizInstances \setminus finishedQuiz]$

The host of an active quiz instance shall not be allowed to add a question to a quiz at the same time.

grd62: $\forall qi \cdot qi \in quizInstances \land qi \in instanceOfQuiz^{-1}[\{q\}] \Rightarrow qi \notin (quizInstances \setminus finishedQuiz)$

The creator of a quiz shall not be allowed to add a question to a quiz if there exists an active quiz instance, linked to the quiz, being hosted at the same time.

then

act41;: $questions := questions \cup \{quest\}$
act42;: $belongingQuiz(quest) := q$
the question quest belongs to q
act43;: $questionPosition(quest) := p$
The position of the question quest is p inside the question list of the quiz q
act51;: $answers := answers \cup \{a, b\}$
act53;: $correctAnswer(quest) := a$
The correct answer is a
act54;: $belongingQuestion := belongingQuestion \cup \{a \mapsto quest, b \mapsto quest\}$
The 2 new answers belong to the question quest

end

Event ADDANSWER ⟨ordinary⟩ $\widehat{=}$
REQ7
extends ADDANSWER
any
$quest$
q
u
a
where
grd51;: $u \in loggedIn$
The user us must be logged in
grd52;: $q \in quizzes$
grd53;: $a \in ANSWER \setminus answers$
grd54;: $quest \in questions$
grd55;: $quizCreator(q) = u$
The user u must be the creator of the quiz
grd56;: $belongingQuiz(quest) = q$
grd57;: $card(\{a1 | a1 \in answers \wedge belongingQuestion(a1) = quest\}) < 4$
The current answers of the question must be less than 4
grd61;: $u \notin host[quizInstances \setminus finishedQuiz]$
The host of an active quiz instance shall not be allowed to add an answer to a question of a quiz at the same time.
grd62: $\forall qi \cdot qi \in quizInstances \wedge qi \in instanceOfQuiz^{-1}[\{q\}] \Rightarrow qi \notin (quizInstances \setminus finishedQuiz)$
The creator of a quiz shall not be allowed to add an answer to a question of a quiz if there exists an active quiz instance, linked to the quiz, being hosted at the same time.
then
act51;: $answers := answers \cup \{a\}$
act52;: $belongingQuestion(a) := quest$

end

Event REMOVENOTCORRECTANSWER ⟨ordinary⟩ $\widehat{=}$
extends REMOVENOTCORRECTANSWER
any
$quest$
q
u
a
where
grd51;: $u \in loggedIn$
The user u must be logged in
grd52;: $q \in quizzes$
grd53;: $quest \in questions$
grd54;: $a \in answers$
grd55;: $card(\{a1 | a1 \in answers \wedge belongingQuestion(a1) = quest\}) > 2$
The number of current answers of the question must be more than 2
grd56;: $correctAnswer(quest) \neq a$
The answer to be removed must not be the correct answer of the question

grd57: $quizCreator(q) = u$
The user u must be the creator of the quiz
grd58: $belongingQuiz(quest) = q$
grd59: $belongingQuestion(a) = quest$
The anser must belong to the quiz q
grd61;: $u \notin host[quizInstances \setminus finishedQuiz]$
The host of an active quiz instance shall not be allowed to remove a non-correct answer from a question of a quiz at the same time.
grd62: $\forall qi \cdot qi \in quizInstances \land qi \in instanceOfQuiz^{-1}[\{q\}] \Rightarrow qi \notin (quizInstances \setminus finishedQuiz)$

The creator of a quiz shall not be allowed to remove a non-correct answer from a question of the quiz if there exists an active quiz instance, linked to the quiz, being hosted at the same time.

then

act51;: $answers := answers \setminus \{a\}$
act52;: $belongingQuestion := \{a\} \lhd belongingQuestion$

end

Event REMOVECORRECTANSWER ⟨ordinary⟩ $\widehat{=}$
extends REMOVECORRECTANSWER

any

$quest$
q
u
$a1$
$a2$

where

grd51;: $u \in loggedIn$
The user u must be logged in
grd52;: $quest \in questions$
grd53;: $q \in quizzes$
grd54;: $a1 \in answers$
grd55;: $a2 \in answers$
grd56;: $correctAnswer(quest) = a1$
The answer to be removed must be the correct answer of the question
grd57;: $belongingQuestion(a2) = quest$
The new correct answer must belong to the question
grd58: $quizCreator(q) = u$
The user u must be the creator of the quiz
grd59: $belongingQuiz(quest) = q$
grd60: $belongingQuestion(a1) = quest$
grd61: $card(\{a | a \in answers \land belongingQuestion(a) = quest\}) > 2$
The current number of answers of the question must be greater than 2
grd62: $a1 \neq a2$
grd61;: $u \notin host[quizInstances \setminus finishedQuiz]$
The host of an active quiz instance shall not be allowed to remove the correct answer from a question of a quiz at the same time.
grd63: $\forall qi \cdot qi \in quizInstances \land qi \in instanceOfQuiz^{-1}[\{q\}] \Rightarrow qi \notin (quizInstances \setminus finishedQuiz)$

The creator of a quiz shall not be allowed to remove the correct answer from a question of the quiz if there exists an active quiz instance, linked to the quiz, being hosted at the same time

then

act51;: $answers := answers \setminus \{a1\}$
act52;: $belongingQuestion := \{a1\} \lhd belongingQuestion$
act53;: $correctAnswer := correctAnswer \lhd\!\!+ \{quest \mapsto a2\}$
the new coorect answer is a2

end

Event SETCORRECTANSWER ⟨ordinary⟩ $\widehat{=}$
extends SETCORRECTANSWER

any

u
q
$quest$
$a1$
$a2$

where

grd51;: $u \in loggedIn$
The user u must be loggged in
grd52;: $q \in quizzes$
grd53;: $quest \in questions$
grd54;: $a1 \in answers$
grd55;: $a2 \in answers$
grd56;: $a1 \neq a2$
grd57: $quizCreator(q) = u$
The user u must be the creator of the quiz
grd58: $belongingQuiz(quest) = q$
grd59: $correctAnswer(quest) = a1$
a1 must be the current correct answer of the question
grd60: $belongingQuestion(a2) = quest$
a2 must belong to the question
grd61;: $u \notin host[quizInstances \setminus finishedQuiz]$
The host of an active quiz instance shall not be allowed to set the correct answer of a question of a quiz at the same time.
grd62: $\forall qi \cdot qi \in quizInstances \wedge qi \in instanceOfQuiz^{-1}[\{q\}] \Rightarrow qi \notin (quizInstances \setminus finishedQuiz)$

The creator of a quiz shall not be allowed to set the correct answer
of a question of the quiz if there exists an active quiz instance, linked to the quiz,
being hosted at the same time.

then

act51;: $correctAnswer := correctAnswer \lhd\!\!\!\!- \{quest \mapsto a2\}$

end

Event UPDATE_ANSWER_OK ⟨ordinary⟩ $\widehat{=}$
extends UPDATE_ANSWER_OK

any

a
q
$quest$
$result$
u

where

grd51;: $u \in loggedIn$
grd52;: $a \in answers$
grd53;: $q \in quizzes$
grd54;: $quest \in questions$
grd55;: $belongingQuestion(a) = quest$
grd56: $quizCreator(q) = u$
grd57: $belongingQuiz(quest) = q$
grd58: $result = TRUE$
The update has been successful
grd61;: $u \notin host[quizInstances \setminus finishedQuiz]$
The host of an active quiz instance shall not be allowed to update an answer of a question of a quiz at the same time.
grd62: $\forall qi \cdot qi \in quizInstances \wedge qi \in instanceOfQuiz^{-1}[\{q\}] \Rightarrow qi \notin (quizInstances \setminus finishedQuiz)$

The creator of a quiz shall not be allowed to update an answer of a
question of the quiz if there exists an active quiz instance, linked to the quiz, being hosted at the same time.

then

skip

end

Event UPDATE_ANSWER_NOT_OK ⟨ordinary⟩ ≙

extends UPDATE_ANSWER_NOT_OK

any

a

q

quest

result

u

where

grd51;: $u \in loggedIn$

the user u must be logged in

grd52;: $a \in answers$

grd53;: $quest \in questions$

grd54;: $q \in quizzes$

grd55;: $belongingQuestion(a) = quest$

grd56;: $quizCreator(q) = u$

u must be the creator of the quiz

grd58;: $belongingQuiz(quest) = q$

grd57;: $result = FALSE$

The update has not been successful. The previous text is restored

grd61: $u \notin host[quizInstances \setminus finishedQuiz]$

The host of an active quiz instance shall not be allowed to update an answer of a question of a quiz at the same time.

grd62: $\forall qi \cdot qi \in quizInstances \land qi \in instanceOfQuiz^{-1}[\{q\}] \Rightarrow qi \notin (quizInstances \setminus finishedQuiz)$

The creator of a quiz shall not be allowed to update an answer of a question of the quiz if there exists an active quiz instance, linked to the quiz, being hosted at the same time.

then

skip

end

Event REMOVEQUESTION ⟨ordinary⟩ ≙

extends REMOVEQUESTION

any

u

quest

q

where

grd41;: $u \in loggedIn$

The user must be logged in order to remove a question

grd42;: $q \in quizzes$

grd43;: $quest \in questions$

grd45;: $belongingQuiz(quest) = q$

q must be the belonging quiz of quest

grd44;: $quizCreator(q) = u$

The creator of the quiz q must be u

grd61: $u \notin host[quizInstances \setminus finishedQuiz]$

The host of an active quiz instance shall not be allowed to remove a question from a quiz at the same time.

grd62: $\forall qi \cdot qi \in quizInstances \land qi \in instanceOfQuiz^{-1}[\{q\}] \Rightarrow qi \notin (quizInstances \setminus finishedQuiz)$

The creator of a quiz shall not be allowed to remove a question from the quiz if there exists an active quiz instance, linked to the quiz, being hosted at the same time.

then

act41;: $questions := questions \setminus \{quest\}$

act42;: $belongingQuiz := \{quest\} \lhd belongingQuiz$

act43;: $questionPosition := \{quest\} \lhd questionPosition$
The position of the question is removed
act51;: $answers := answers \setminus \{a | a \in answers \land belongingQuestion(a) = quest\}$
the answers of the question must be removed when the question is removed
act52;: $correctAnswer := \{quest\} \lhd correctAnswer$
The correct answer of quest is not the correct answer anymore
act53;: $belongingQuestion := belongingQuestion \rhd \{quest\}$
All answers of the question do not belong to it anymore

end

Event UPDATE_QUESTION_OK ⟨ordinary⟩ $\widehat{=}$
extends UPDATE_QUESTION_OK

any
u
$quest$
q
$result$

where
grd41;: $u \in loggedIn$
grd42;: $q \in quizzes$
grd43;: $quizCreator(q) = u$
Only the quiz creator of q can update the question
grd44;: $quest \in questions$
grd45;: $result = TRUE$
The update has been succesfully
grd46;: $belongingQuiz(quest) = q$
the question must belong to q
grd61;: $u \notin host[quizInstances \setminus finishedQuiz]$
The host of an active quiz instance shall not be allowed to update a question of a quiz at the same time.
grd62: $\forall qi \cdot qi \in quizInstances \land qi \in instanceOfQuiz^{-1}[\{q\}] \Rightarrow qi \notin (quizInstances \setminus finishedQuiz)$

The creator of a quiz shall not be allowed to update a question of the quiz if there exists an active quiz instance, linked to the quiz, being hosted at the same time.

then
skip

end

Event UPDATE_QUESTION_NOT_OK ⟨ordinary⟩ $\widehat{=}$
extends UPDATE_QUESTION_NOT_OK

any
u
$quest$
q
$result$

where
grd41;: $u \in loggedIn$
grd42;: $q \in quizzes$
grd43;: $quizCreator(q) = u$
grd1: $quest \in questions$
grd45;: $belongingQuiz(quest) = q$
grd46;: $result = FALSE$
The update has not been successfully. The previous text of the question is restored
grd61;: $u \notin host[quizInstances \setminus finishedQuiz]$
The host of an active quiz instance shall not be allowed to update a question of a quiz at the same time.
grd62: $\forall qi \cdot qi \in quizInstances \land qi \in instanceOfQuiz^{-1}[\{q\}] \Rightarrow qi \notin (quizInstances \setminus finishedQuiz)$

The creator of a quiz shall not be allowed to update a question of the quiz if there exists an active quiz instance, linked to the quiz, being hosted at the same time.

then

skip

end

Event HOST_QUIZ_INSTANCE ⟨ordinary⟩ $\widehat{=}$

REQ14

any

u

q

qi

where

grd61;: $u \in loggedIn$

The user u must be logged in

grd62;: $qi \in INSTANCE\_QUIZ \setminus quizInstances$

grd63;: $q \in quizzes$

grd65;: $u \notin ran((finishedQuiz \lhd host))$

The user u must not be hosting any other active quiz instance

grd66: $u \in (\{quizCreator(q)\} \cup sharedTo[\{q\}])$

The user u must be the creator or have access to the quiz

grd67: $belongingQuiz^{-1}[\{q\}] \neq \varnothing$

The quiz must have at least 1 question

then

act61;: $quizInstances := quizInstances \cup \{qi\}$

act62;: $host(qi) := u$

The host of the quiz instance is u

act63;: $instanceOfQuiz(qi) := q$

The quiz instance is linked to the quiz q

act64;: $quizCreated := quizCreated \cup \{qi\}$

The quiz instance is in the quizCreated state

end

Event BEGIN_QUIZ_INSTANCE ⟨ordinary⟩ $\widehat{=}$

any

u

qi

where

grd61;: $u \in loggedIn$

The user u must be logged in

grd62;: $qi \in quizCreated$

The quiz instance must be in the quizCreated state

grd63;: $host(qi) = u$

The host of the quiz instance must be u

then

act61;: $quizCreated := quizCreated \setminus \{qi\}$

The quiz instance is not in the quizCreated state anymore

act62;: $quizInit := quizInit \cup \{qi\}$

The quiz instance becomes part of the quizInit state set

act63;: $answeredQuestions(qi) := 0$

No question has been answered so far

end

Event START_QUIZ_INSTANCE ⟨ordinary⟩ $\widehat{=}$

any

u

qi

quest

where

grd61;: $u \in loggedIn$

The user u must be logged in

grd62;: $qi \in quizInit$

The quiz instance must be in the quizInit state

grd63;: $host(qi) = u$

The host of the quiz instance must be u

grd64;: $\forall k \cdot belongingQuiz(k) = instanceOfQuiz(qi) \Rightarrow questionPosition(k) \geq questionPosition(quest)$

The question of the quiz that will become the first question to be answered in the quiz instance must be the first one in the question list of the quiz

grd65: $qi \in ran(playingIn)$

Some players must have joined the quiz during the quizInit state

then

act61;: $quizInit := quizInit \setminus \{qi\}$

The quiz is not in the quizInit state anymore

act62;: $questioning := questioning \cup \{qi\}$

The quiz instance is in the questioning state

act63;: $currentQuestion(qi) := quest$

The current question of the quiz instance is the first question of the question list of the quiz

end

Event JOIN_QUIZ_INSTANCE ⟨ordinary⟩ $\widehat{=}$

any

u

qi

where

grd61;: $u \in USER$

The player can be either regitered or unregistered

grd62;: $qi \in quizInit$

It is possible to join a quiz instanec only if it is in the quizInit state

grd63;: $u \neq host(qi)$

The player must not be the host of the quiz instance

grd64;: $u \notin dom(playingIn)$

The player must not be playing in another quiz instance

grd1: $u \notin host[quizInstances \setminus finishedQuiz]$

The player cannot be any registered user that is hosting any active quiz instance

then

act64;: $playingIn := playingIn \cup \{u \mapsto qi\}$

end

Event ANSWER_QUESTION ⟨ordinary⟩ $\widehat{=}$

any

p

qi

quest

a

where

grd61;: $p \in USER$

grd62;: $qi \in quizInstances$

grd63;: $p \in playingIn^{-1}[\{qi\}]$

The player must be playing in the quiz instance

grd64;: $qi \in questioning$

The quiz instance must be in the questioning state

grd65: $quest \in questions$

grd66: $currentQuestion(qi) = quest$

The question the player is answering must be the current question

grd67: $a \in answers$

grd68: $a \in belongingQuestion^{-1}[\{quest\}]$

The answer being selected must belong to the current question of the quiz instance

grd69: $card(peopleAnswers^{-1}[\{qi\}]) < card(playingIn^{-1}[\{qi\}])$

A player of a quiz instance shall not be allowed to select an answer when all other players have answered because the quiz instance passes to the questionSummary state.

then

act61;: $peopleAnswers := peopleAnswers \cup \{p \mapsto qi\}$

act62: $peopleSelectedAnswer := peopleSelectedAnswer \Leftarrow \{p \mapsto a\}$

end

Event END_QUESTION_ALL_ANSWERED ⟨ordinary⟩ $\widehat{=}$

any
qi
where
grd61;: $qi \in questioning$
The quiz instance must be in the questionig state
grd62;: $card(peopleAnswers^{-1}[\{qi\}]) = card(playingIn^{-1}[\{qi\}])$
All players must have answered
then
act61;: $questioning := questioning \setminus \{qi\}$
The quiz instance is not part of the questioning state anymore
act62;: $questionSummary := questionSummary \cup \{qi\}$
The quiz instance passes to the questionSummary state
act63;: $answeredQuestions(qi) := answeredQuestions(qi) + 1$
The number of answered questions is incremented by 1
act64;: $peopleAnswers := peopleAnswers \rhd \{qi\}$
The players that have answered the question of the quiz instance are removed
act65: $peopleSelectedAnswer := playingIn^{-1}[\{qi\}] \lhd peopleSelectedAnswer$
The answers select by the players of the quiz instance are removed
end

Event END_QUESTION_PREMATURATELY ⟨ordinary⟩ $\widehat{=}$
any
u
qi
where
grd61;: $qi \in questioning$
The quiz instance must be in the question state
grd62;: $u \in loggedIn$
The user u must be logged in
grd63;: $host(qi) = u$
The host of the quiz instance must be u
grd1: $card(peopleAnswers^{-1}[\{qi\}]) \neq card(playingIn^{-1}[\{qi\}])$
The host of a quiz instance in the questioning state
shall not be allowed to end prematurately a question
if all players have answered because the quiz instance passes to the questionSummary state.
then
act61;: $questioning := questioning \setminus \{qi\}$
The quiz instance is not in the questioning state anymore
act62;: $questionSummary := questionSummary \cup \{qi\}$
The quiz instance passes to the questionSummary state
act63;: $answeredQuestions(qi) := answeredQuestions(qi) + 1$
The number of answered questions of the quiz instance is incremebted by 1
act64;: $peopleAnswers := peopleAnswers \rhd \{qi\}$
The players that have answered the question of the quiz instance are removed
act65: $peopleSelectedAnswer := playingIn^{-1}[\{qi\}] \lhd peopleSelectedAnswer$
The answers select by the players of the quiz instance are removed
end

Event SHOW_QUESTION_SUMMARY_TO_PLAYERS ⟨ordinary⟩ $\widehat{=}$
any
u
qi
where
grd61;: $u \in loggedIn$
The user u must be logged in
grd62;: $qi \in quizInstances$
grd63;: $host(qi) = u$
The user u must be the host of the quiz instance
grd63: $qi \in questionSummary$
The quiz instance must be in the questionSummary state

grd64: $qi \in ran(playingIn)$
The host of a quiz instance in the questionSummary state shall be allowed to show the question summary only if at least 1 player is playing in the quiz instance.

then

skip

end

Event NEXT_QUESTION ⟨ordinary⟩ $\widehat{=}$

any

u
qi
quest

where

grd61;: $u \in loggedIn$
grd62;: $qi \in questionSummary$
The quiz instance must be in the questionSummary state
grd63;: $host(qi) = u$
grd64;: $answeredQuestions(qi) < card(belongingQuiz^{-1}[\{instanceOfQuiz(qi)\}])$
There must be other questions to be answered
grd65: $qi \in ran(playingIn)$
At least 1 player must be playing in the quiz instance
grd66: $quest \in questions$
grd67: $card(\{quests | quests \in questions \wedge quests \in belongingQuiz^{-1}[\{instanceOfQuiz(qi)\}] \wedge questionPosition(quests) < questionPosition(quest)\}) = answeredQuestions(qi)$
The next question to be answered must be the successive one

then

act61;: $questionSummary := questionSummary \setminus \{qi\}$
The quiz instance is not in the questionSummary state anymore
act62;: $questioning := questioning \cup \{qi\}$
The quiz instance passes to the questioning state
act63: $currentQuestion(qi) := quest$
The current qyestion is set to the successive question

end

Event FINISH_QUIZ_INSTANCE ⟨ordinary⟩ $\widehat{=}$

any

u
qi

where

grd61;: $u \in loggedIn$
The user must be logged in
grd62;: $qi \in questionSummary$
The quiz instance must be in the questionSummary state
grd63;: $host(qi) = u$
The user u must be the host of the quiz instance

then

act61;: $questionSummary := questionSummary \setminus \{qi\}$
The quiz instance is not in the questionSummary state anymore
act62;: $quizSummary := quizSummary \cup \{qi\}$
The quiz instance passes to the quizSummary state
act63;: $currentQuestion := \{qi\} \lhd currentQuestion$
The current question of the quiz instance is removed as it is not anymore in the questionSummary state

end

Event SHOW_QUIZ_INSTANCE_SUMMARY_TO_PLAYERS ⟨ordinary⟩ $\widehat{=}$

any

qi
u

where

grd61;: $qi \in quizInstances$

grd62: $qi \in quizSummary$
The quiz instance must be in the quizSummary state
grd63: $u \in registeredUser$
grd64: $u \in loggedIn$
The user u must be logged in
grd65: $host(qi) = u$
The user u must be the host of the quiz instance
grd66: $qi \in ran(playingIn)$
there must at least 1 player playing in the quiz instance

then

skip

end

Event TERMINATE_QUIZ_INSTANCE_QUIZ_SUMMARY_STATE ⟨ordinary⟩ $\widehat{=}$

any

u
qi

where

grd61;: $u \in loggedIn$
The user u must be loggedin
grd62;: $qi \in quizSummary$
The quiz instance must be in the quizSummary state
grd63;: $host(qi) = u$
The user u must be the host of the quiz instance

then

act61;: $quizSummary := quizSummary \setminus \{qi\}$
The quiz instance is not in the quizSummary state anymore
act62;: $finishedQuiz := finishedQuiz \cup \{qi\}$
The quiz instance passes to the finishedQuiz state
act63: $playingIn := playingIn \rhd \{qi\}$
The players of the quiz instance do not play anymore in the quiz instance

end

Event LEAVE_QUIZ_INSTANCE_NOT_IN_QUESTIONING_STATE ⟨ordinary⟩ $\widehat{=}$

any

p
qi

where

grd61: $p \in USER$
grd62: $qi \in quizInstances \setminus (quizCreated \cup finishedQuiz \cup questioning)$
The quiz instance must not be in the quiz created state, finishedQuiz or questioning state
grd63: $p \in playingIn^{-1}[\{qi\}]$
The player must be playing in the quiz instance

then

act61: $playingIn := \{p\} \lhd playingIn$
The player does not play anymore in the quiz instance

end

Event LEAVE_QUIZ_INSTANCE_LAST_ONE_IN_QUESTIONING_STATE ⟨ordinary⟩ $\widehat{=}$

any

p
qi

where

grd61: $p \in USER$
grd62: $qi \in questioning$
The quiz instance must be in the questioning state
grd63: $p \in playingIn^{-1}[\{qi\}]$
The player must be playing in the quiz instance
grd64: $card(playingIn^{-1}[\{qi\}]) = 1$
The player must be the only player be playing in the quiz instance

then

act61: $playingIn := \{p\} \lhd playingIn$
The player does not play anymore in the quiz instance
act62: $peopleAnswers := \{p\} \lhd peopleAnswers$
It is removed whether the player has answered the question or not
act63: $questioning := questioning \setminus \{qi\}$
act64: $questionSummary := questionSummary \cup \{qi\}$
The quiz instance passes to the questionSummary state
act65: $peopleSelectedAnswer := \{p\} \lhd peopleSelectedAnswer$
The answer selected by the player is removed (if he/she has naswered)

end

Event LEAVE_QUIZ_INSTANCE_NOT_LAST_ONE_IN_QUESTIONING_STATE ⟨ordinary⟩ $\widehat{=}$

any

p
qi

where

grd61: $p \in USER$
grd62: $qi \in questioning$
grd63: $p \in playingIn^{-1}[\{qi\}]$
grd64: $card(playingIn^{-1}[\{qi\}]) > 1$
There must be at least 2 players be playing

then

act61: $playingIn := \{p\} \lhd playingIn$
act62: $peopleAnswers := \{p\} \lhd peopleAnswers$
act63: $peopleSelectedAnswer := \{p\} \lhd peopleSelectedAnswer$

end

Event UNSHARE_QUIZ ⟨ordinary⟩ $\widehat{=}$

extends UNSHARE_QUIZ

any

$u1$
$u2$
q

where

grd41: $u1 \in registeredUser$
grd42: $u2 \in registeredUser$
grd43: $u1 \in loggedIn$
A registerd user can unshare only if he/she is online
grd44: $u1 \neq u2$
grd45: $q \mapsto u2 \in sharedTo$
u2 had to have access to the quiz
grd46: $u1 = quizCreator(q)$
Only the creator of the quiz can unshare a quiz
grd81: $u2 \notin host[instanceOfQuiz^{-1}[\{q\}] \setminus finishedQuiz]$
The creator of a quiz shall not be allowed to deshare a quiz
with another registered user if the latter is hosting a quiz instance linked to the quiz.
grd82: $u1 \notin host[quizInstances \setminus finishedQuiz]$
The creator of a quiz shall not be allowed to deshare a
quiz if he/she is hosting an active quiz instance.

then

act41: $sharedTo := sharedTo \setminus \{q \mapsto u2\}$

end

Event REMOVE_QUIZ_INSTANCE ⟨ordinary⟩ $\widehat{=}$

any

u
qi

where

grd81: $u \in registeredUser$
grd82: $qi \in quizInstances$
grd83: $qi \in finishedQuiz$
It is possible to remove a quiz instance only when it is in the finishedQuiz state

grd84: $u \in loggedIn$
The user must be logged in
grd85: $host(qi) = u$
The user u must be the host of the quiz instance
grd86: $u \notin host[quizInstances \setminus finishedQuiz]$
The user must not be hosting any active quiz instance

then

act81: $quizInstances := quizInstances \setminus \{qi\}$
The quiz instance is removed
act82: $finishedQuiz := finishedQuiz \setminus \{qi\}$
The quiz instance is removed from the finishedQuiz state
act83: $host := \{qi\} \lhd host$
The user is not the host of the quiz instance anymore
act84: $instanceOfQuiz := \{qi\} \lhd instanceOfQuiz$
The quiz instance is not linked anymore to a quiz
act85: $answeredQuestions := \{qi\} \lhd answeredQuestions$
The number of answered questions of the quiz instance is removed

end

Event TERMINATE_QUIZ_INSTANCE_QUIZ_CREATED_STATE ⟨ordinary⟩ $\widehat{=}$

any

u
qi

where

grd91: $u \in registeredUser$
grd92: $u \in loggedIn$
The user u must be loggedIn
grd93: $qi \in quizInstances$
grd94: $qi \in quizCreated$
The quiz instance must be in the quizCreated state
grd95: $host(qi) = u$
The user u must be the host of the quiz instance

then

act91: $quizCreated := quizCreated \setminus \{qi\}$
The quiz instance is removed from the quizCreated state
act92: $finishedQuiz := finishedQuiz \cup \{qi\}$
The quiz instance passes to the finishedQuiz state
act93: $answeredQuestions(qi) := 0$
The number of answered questions is set to 0

end

Event TERMINATE_QUIZ_INSTANCE_QUIZ_INIT_STATE ⟨ordinary⟩ $\widehat{=}$

any

u
qi

where

grd81: $u \in registeredUser$
grd82: $u \in loggedIn$
The user u must be loggedin
grd83: $qi \in quizInstances$
grd84: $qi \in quizInit$
The quiz instance must be in the quizInit state
grd85: $host(qi) = u$
The user u must be the host of the quiz instance

then

act81: $quizInit := quizInit \setminus \{qi\}$
The quiz instance is not anymore in the quizInit state
act82: $finishedQuiz := finishedQuiz \cup \{qi\}$
The quiz instance passes to the finishedQuiz state
act83: $playingIn := playingIn \rhd \{qi\}$
The players of the quiz instance do not play anymore in the quiz instance

end

Event END_QUESTION_TIME_IS_UP ⟨ordinary⟩ $\widehat{=}$

any

qi

where

grd61: $qi \in quizInstances$

grd62: $qi \in questioning$

The quiz instance must be in the questioning state

then

act61: $questioning := questioning \setminus \{qi\}$

act62: $questionSummary := questionSummary \cup \{qi\}$

The quiz instance passes to the questionSummary state

act63: $answeredQuestions(qi) := answeredQuestions(qi) + 1$

The number of answered questions of the quiz instance is incremented by 1

act64: $peopleAnswers := peopleAnswers \rhd \{qi\}$

act65: $peopleSelectedAnswer := playingIn^{-1}[\{qi\}] \lhd peopleSelectedAnswer$

end

END

MACHINE m9
REFINES m8
SEES c5
VARIABLES

registeredUser
password
loggedIn
loggedOut
quizzes
questions
quizCreator
belongingQuiz
sharedTo
questionPosition
answers
correctAnswer
belongingQuestion
quizInstances
host
playingIn
instanceOfQuiz
currentQuestion
quizInit
questioning
questionSummary
quizSummary
finishedQuiz
answeredQuestions
peopleAnswers
quizCreated
embeddedQuestions
reports
summaryFor
reportedBy
qSummary
qReportedBy
peopleSelectedAnswer

INVARIANTS

inv91: $embeddedQuestions \in quizInstances \setminus finishedQuiz \nleftrightarrow questions$
Every quiz instance (not finished) must keep track of the questions
the quiz they are linked to contains

inv92: $qSummary \subseteq QUESTION\_SUMMARY$
The set of summary of the questions

inv93: $reports \subseteq REPORT$
The set of reports

inv94: $summaryFor \in qSummary \rightarrowtail embeddedQuestions$
every question summary must be the summary of a question in a certain quiz instance

inv95: $qReportedBy \in qSummary \rightarrow reports$
every question summary is linked to exactly one report

inv96: $reportedBy \in quizInstances \twoheadrightarrow reports$
(TOTAL SURJECTION)Every quiz instance must have exactly one report and a report must be reporting
exactly one quiz instance

inv97: $(\forall qi \cdot qi \in quizInstances \Rightarrow qi \in finishedQuiz) \Rightarrow qSummary = \varnothing$
The summaries of the questions of a quiz instance must exist only when the quiz instance is active (not finished). These summaries will help construct the report of the whole quiz instance

EVENTS

Initialisation ⟨extended⟩

begin

act1;: $registeredUser := \varnothing$
act21;: $password := \varnothing$
act31;: $loggedIn := \varnothing$
act32;: $loggedOut := \varnothing$
act41;: $quizzes := \varnothing$
act42;: $questions := \varnothing$
act43;: $quizCreator := \varnothing$
act44;: $belongingQuiz := \varnothing$
act45;: $sharedTo := \varnothing$
act46;: $questionPosition := \varnothing$
act51;: $answers := \varnothing$
act52;: $correctAnswer := \varnothing$
act53;: $belongingQuestion := \varnothing$
act61;: $quizInstances := \varnothing$
act62;: $host := \varnothing$
act63;: $playingIn := \varnothing$
act64;: $instanceOfQuiz := \varnothing$
act65;: $peopleAnswers := \varnothing$
act66: $answeredQuestions := \varnothing$
act67: $currentQuestion := \varnothing$
act68: $finishedQuiz := \varnothing$
act69: $questioning := \varnothing$
act611: $questionSummary := \varnothing$
act612: $quizCreated := \varnothing$
act613: $quizInit := \varnothing$
act614: $quizSummary := \varnothing$
act615: $peopleSelectedAnswer := \varnothing$
act91: $embeddedQuestions := \varnothing$
act92: $reports := \varnothing$
act93: $summaryFor := \varnothing$
act94: $reportedBy := \varnothing$
act95: $qSummary := \varnothing$
act96: $qReportedBy := \varnothing$

end

Event REGISTER ⟨ordinary⟩ $\widehat{=}$
extends REGISTER

any

u A user
p

where

grd1;: $u \in USER \setminus registeredUser$
The user must not be already registered
grd21;: $p \in PASSWORD$
grd81: $u \notin dom(playingIn)$

then

act1;: $registeredUser := registeredUser \cup \{u\}$
The user is added to the set of registered users
act21;: $password(u) := p$
The password of user u is p
act31;: $loggedOut := loggedOut \cup \{u\}$
When a user u registers, u is added to the set of loggedOut users

end

Event LOGIN ⟨ordinary⟩ ≙

extends LOGIN

any

u

p

where

inv31;: $u \in loggedOut$

inv32;: $p = password(u)$

p must be the pssword of u

grd81: $u \notin dom(playingIn)$

then

act31;: $loggedIn := loggedIn \cup \{u\}$

The user u is added to the set of loggedIn users

act32;: $loggedOut := loggedOut \setminus \{u\}$

The user u is removed from the set of loggedOut users

end

Event LOGOUT ⟨ordinary⟩ ≙

extends LOGOUT

any

u

where

inv31;: $u \in loggedIn$

The user u mst be logged in

grd81: $u \notin host[quizInstances \setminus finishedQuiz]$

The host of an active quiz instance shall not be allowed to log out of the system.

grd82: $u \notin dom(playingIn)$

then

act31;: $loggedIn := loggedIn \setminus \{u\}$

The user u is removed from the set of loggedIn users

act32;: $loggedOut := loggedOut \cup \{u\}$

The user u is added to the set of loggedOut users

end

Event CREATEQUIZ ⟨ordinary⟩ ≙

extends CREATEQUIZ

any

u

q

where

grd41;: $u \in loggedIn$

A user must be logged in so that to create a quiz

grd42;: $q \in QUIZ \setminus quizzes$

The created quiz must not have been already created

grd61;: $u \notin host[quizInstances \setminus finishedQuiz]$

The host of an active quiz instance shall not be allowed to create a quiz at the same time.

then

act41;: $quizzes := quizzes \cup \{q\}$

The created quiz is added to the dynamic set quizzes

act42;: $quizCreator(q) := u$

the creator of the quiz q is u

end

Event REMOVEQUIZ ⟨ordinary⟩ ≙

extends REMOVEQUIZ

any

u

q

where

grd41;: $q \in quizzes$

grd42;: $u \in loggedIn$
The user must be logged in so that to remove a quiz
grd43;: $quizCreator(q) = u$
Only the creator of the quiz can remove the quiz
grd61;: $u \notin host[quizInstances \setminus finishedQuiz]$
The host of an active quiz instance shall not be allowed to remove a quiz at the same time.
grd62: $\forall qi \cdot qi \in quizInstances \land qi \in instanceOfQuiz^{-1}[\{q\}] \Rightarrow qi \notin (quizInstances \setminus finishedQuiz)$

The creator of a quiz shall not be allowed to remove a quiz if there exists an active quiz instance, linked to the quiz, being hosted at the same time.

then

act41;: $quizzes := quizzes \setminus \{q\}$
the quiz is removed
act42;: $quizCreator := \{q\} \lhd quizCreator$
The quiz does not have anymore a creator because is removed
act43;: $sharedTo := \{q\} \lhd sharedTo$
All registerd users that had access to the quiz, do not have access to it anymore because the quiz is removed
act44;: $belongingQuiz := belongingQuiz \rhd \{q\}$
The questions of the quiz do not belong anymore to it
act45;: $questions := questions \setminus \{quest | quest \in questions \land belongingQuiz(quest) = q\}$
The questions of the quiz are removed
act46;: $questionPosition := \{quest | quest \in questions \land belongingQuiz(quest) = q\} \lhd questionPosition$

The position of the questions is removed
act54;: $correctAnswer := \{q2 | q2 \in questions \land belongingQuiz(q2) = q\} \lhd correctAnswer$
The correct answer of all questions of the quiz must be removed
act55;: $belongingQuestion := belongingQuestion \rhd \{q2 | q2 \in questions \land belongingQuiz(q2) = q\}$
The answers of the questions of the quiz do not belong to the questions anymore
act56: $answers := answers \setminus \{a1 | a1 \in answers \land belongingQuestion(a1) \in \{quest1 | quest1 \in questions \land belongingQuiz(quest1) = q\}\}$
Removing a quiz involves removing all answers of the questions of the quiz
act61: $instanceOfQuiz := instanceOfQuiz \rhd \{q\}$

end

Event SHAREQUIZ ⟨ordinary⟩ $\widehat{=}$
extends SHAREQUIZ

any

$u1$
$u2$
q

where

grd41;: $u1 \in loggedIn$
the user must be logged in so that to share the quiz
grd42;: $u2 \in registeredUser$
the other user that will have access to the quiz
grd43;: $u1 \neq u2$
the two users cannot be the same person
grd44;: $q \in quizzes$
grd45;: $quizCreator(q) = u1$
u1 must be the creator of the quiz
grd46: $q \in ran(belongingQuiz)$
The quiz must contain at least 1 question in order to be shared
grd47: $q \mapsto u2 \notin sharedTo$
the quiz must not have already been shared with u2
grd61;: $u1 \notin host[quizInstances \setminus finishedQuiz]$
The host of an active quiz instance shall not be allowed to share a quiz at the same time.

then

act41;: $sharedTo := sharedTo \cup \{q \mapsto u2\}$

end

Event CREATEQUESTION ⟨ordinary⟩ $\widehat{=}$
extends CREATEQUESTION

any

$quest$
q
u
p
a
b

where

grd41;: $quest \in QUESTION \setminus questions$
The question must be a new question
grd42;: $q \in quizzes$
grd43;: $u \in loggedIn$
The user must be logged in in order to create a new question
grd44;: $quizCreator(q) = u$
The creator of the quiz must be u
grd45;: $p \in POSITION$
grd46;: $\forall z \cdot z \in questions \land belongingQuiz(z) = q \Rightarrow p > questionPosition(z)$
The question must be added at the end of the question list of the quiz
grd51;: $(a \in ANSWER \setminus answers) \land (b \in ANSWER \setminus answers)$
The 2 answers must be new ones
grd52: $a \neq b$
The 2 answers must be different
grd61;: $u \notin host[quizInstances \setminus finishedQuiz]$
The host of an active quiz instance shall not be allowed to add a question to a quiz at the same time.
grd62: $\forall qi \cdot qi \in quizInstances \land qi \in instanceOfQuiz^{-1}[\{q\}] \Rightarrow qi \notin (quizInstances \setminus finishedQuiz)$

The creator of a quiz shall not be allowed to add a question to a quiz if there exists an active quiz instance, linked to the quiz, being hosted at the same time.

then

act41;: $questions := questions \cup \{quest\}$
act42;: $belongingQuiz(quest) := q$
the question quest belongs to q
act43;: $questionPosition(quest) := p$
The position of the question quest is p inside the question list of the quiz q
act51;: $answers := answers \cup \{a, b\}$
act53;: $correctAnswer(quest) := a$
The correct answer is a
act54;: $belongingQuestion := belongingQuestion \cup \{a \mapsto quest, b \mapsto quest\}$
The 2 new answers belong to the question quest

end

Event ADDANSWER ⟨ordinary⟩ $\widehat{=}$
REQ7
extends ADDANSWER

any

$quest$
q
u
a

where

grd51;: $u \in loggedIn$
The user us must be logged in
grd52;: $q \in quizzes$
grd53;: $a \in ANSWER \setminus answers$
grd54;: $quest \in questions$
grd55;: $quizCreator(q) = u$
The user u must be the creator of the quiz

grd56;: $belongingQuiz(quest) = q$
grd57;: $card(\{a1 | a1 \in answers \land belongingQuestion(a1) = quest\}) < 4$
The current answers of the question must be less than 4
grd61;: $u \notin host[quizInstances \setminus finishedQuiz]$
The host of an active quiz instance shall not be allowed to add an answer to a question of a quiz at the same time.
grd62: $\forall qi \cdot qi \in quizInstances \land qi \in instanceOfQuiz^{-1}[\{q\}] \Rightarrow qi \notin (quizInstances \setminus finishedQuiz)$

The creator of a quiz shall not be allowed to add an answer to a question of a quiz if there exists an active quiz instance, linked to the quiz, being hosted at the same time.

then

act51;: $answers := answers \cup \{a\}$
act52;: $belongingQuestion(a) := quest$

end

Event REMOVENOTCORRECTANSWER ⟨ordinary⟩ $\widehat{=}$
extends REMOVENOTCORRECTANSWER

any

$quest$
q
u
a

where

grd51;: $u \in loggedIn$
The user u must be logged in
grd52;: $q \in quizzes$
grd53;: $quest \in questions$
grd54;: $a \in answers$
grd55;: $card(\{a1 | a1 \in answers \land belongingQuestion(a1) = quest\}) > 2$
The number of current answers of the question must be more than 2
grd56;: $correctAnswer(quest) \neq a$
The answer to be removed must not be the correct answer of the question
grd57: $quizCreator(q) = u$
The user u must be the creator of the quiz
grd58: $belongingQuiz(quest) = q$
grd59: $belongingQuestion(a) = quest$
The anser must belong to the quiz q
grd61;: $u \notin host[quizInstances \setminus finishedQuiz]$
The host of an active quiz instance shall not be allowed to remove a non-correct answer from a question of a quiz at the same time.
grd62: $\forall qi \cdot qi \in quizInstances \land qi \in instanceOfQuiz^{-1}[\{q\}] \Rightarrow qi \notin (quizInstances \setminus finishedQuiz)$

The creator of a quiz shall not be allowed to remove a non-correct answer from a question of the quiz if there exists an active quiz instance, linked to the quiz, being hosted at the same time.

then

act51;: $answers := answers \setminus \{a\}$
act52;: $belongingQuestion := \{a\} \lhd belongingQuestion$

end

Event REMOVECORRECTANSWER ⟨ordinary⟩ $\widehat{=}$
extends REMOVECORRECTANSWER

any

$quest$
q
u
$a1$
$a2$

where

grd51;: $u \in loggedIn$
The user u must be logged in

grd52;: $quest \in questions$
grd53;: $q \in quizzes$
grd54;: $a1 \in answers$
grd55;: $a2 \in answers$
grd56;: $correctAnswer(quest) = a1$
The answer to be removed must be the correct answer of the question
grd57;: $belongingQuestion(a2) = quest$
The new correct answer must belong to the question
grd58: $quizCreator(q) = u$
The user u must be the creator of the quiz
grd59: $belongingQuiz(quest) = q$
grd60: $belongingQuestion(a1) = quest$
grd61: $card(\{a|a \in answers \land belongingQuestion(a) = quest\}) > 2$
The current number of answers of the question must be greater than 2
grd62: $a1 \neq a2$
grd61;: $u \notin host[quizInstances \setminus finishedQuiz]$
The host of an active quiz instance shall not be allowed to remove the correct answer from a question of a quiz at the same time.
grd63: $\forall qi \cdot qi \in quizInstances \land qi \in instanceOfQuiz^{-1}[\{q\}] \Rightarrow qi \notin (quizInstances \setminus finishedQuiz)$

The creator of a quiz shall not be allowed to remove the correct answer from a question of the quiz if there exists an active quiz instance, linked to the quiz, being hosted at the same time

then

act51;: $answers := answers \setminus \{a1\}$
act52;: $belongingQuestion := \{a1\} \lhd belongingQuestion$
act53;: $correctAnswer := correctAnswer \lhd \{quest \mapsto a2\}$
the new coorect answer is a2

end

Event SETCORRECTANSWER ⟨ordinary⟩ $\widehat{=}$
extends SETCORRECTANSWER

any

u
q
$quest$
$a1$
$a2$

where

grd51;: $u \in loggedIn$
The user u must be loggged in
grd52;: $q \in quizzes$
grd53;: $quest \in questions$
grd54;: $a1 \in answers$
grd55;: $a2 \in answers$
grd56;: $a1 \neq a2$
grd57: $quizCreator(q) = u$
The user u must be the creator of the quiz
grd58: $belongingQuiz(quest) = q$
grd59: $correctAnswer(quest) = a1$
a1 must be the current correct answer of the question
grd60: $belongingQuestion(a2) = quest$
a2 must belong to the question
grd61;: $u \notin host[quizInstances \setminus finishedQuiz]$
The host of an active quiz instance shall not be allowed to set the correct answer of a question of a quiz at the same time.
grd62: $\forall qi \cdot qi \in quizInstances \land qi \in instanceOfQuiz^{-1}[\{q\}] \Rightarrow qi \notin (quizInstances \setminus finishedQuiz)$

The creator of a quiz shall not be allowed to set the correct answer of a question of the quiz if there exists an active quiz instance, linked to the quiz, being hosted at the same time.

then
act51;: $correctAnswer := correctAnswer \lhd \{quest \mapsto a2\}$
end
Event UPDATE_ANSWER_OK ⟨ordinary⟩ $\widehat{=}$
extends UPDATE_ANSWER_OK
any
a
q
$quest$
$result$
u
where
grd51;: $u \in loggedIn$
grd52;: $a \in answers$
grd53;: $q \in quizzes$
grd54;: $quest \in questions$
grd55;: $belongingQuestion(a) = quest$
grd56: $quizCreator(q) = u$
grd57: $belongingQuiz(quest) = q$
grd58: $result = TRUE$
The update has been successful
grd61;: $u \notin host[quizInstances \setminus finishedQuiz]$
The host of an active quiz instance shall not be allowed to update an answer of a question of a quiz at the same time.
grd62: $\forall qi \cdot qi \in quizInstances \land qi \in instanceOfQuiz^{-1}[\{q\}] \Rightarrow qi \notin (quizInstances \setminus finishedQuiz)$

The creator of a quiz shall not be allowed to update an answer of a question of the quiz if there exists an active quiz instance, linked to the quiz, being hosted at the same time.
then
skip
end
Event UPDATE_ANSWER_NOT_OK ⟨ordinary⟩ $\widehat{=}$
extends UPDATE_ANSWER_NOT_OK
any
a
q
$quest$
$result$
u
where
grd51;: $u \in loggedIn$
the user u must be logged in
grd52;: $a \in answers$
grd53;: $quest \in questions$
grd54;: $q \in quizzes$
grd55;: $belongingQuestion(a) = quest$
grd56;: $quizCreator(q) = u$
u must be the creator of the quiz
grd58;: $belongingQuiz(quest) = q$
grd57;: $result = FALSE$
The update has not been successful. The previous text is restored
grd61: $u \notin host[quizInstances \setminus finishedQuiz]$
The host of an active quiz instance shall not be allowed to update an answer of a question of a quiz at the same time.
grd62: $\forall qi \cdot qi \in quizInstances \land qi \in instanceOfQuiz^{-1}[\{q\}] \Rightarrow qi \notin (quizInstances \setminus finishedQuiz)$

The creator of a quiz shall not be allowed to update an answer of a question of the quiz if there exists an active quiz instance, linked to the quiz, being hosted at the same time.

then
skip
end
Event REMOVEQUESTION ⟨ordinary⟩ ≙
extends REMOVEQUESTION
any
u
$quest$
q
where
grd41;: $u \in loggedIn$
The user must be logged in order to remove a question
grd42;: $q \in quizzes$
grd43;: $quest \in questions$
grd45;: $belongingQuiz(quest) = q$
q must be the belonging quiz of quest
grd44;: $quizCreator(q) = u$
The creator of the quiz q must be u
grd61: $u \notin host[quizInstances \setminus finishedQuiz]$
The host of an active quiz instance shall not be allowed to remove a question from a quiz at the same time.
grd62: $\forall qi \cdot qi \in quizInstances \land qi \in instanceOfQuiz^{-1}[\{q\}] \Rightarrow qi \notin (quizInstances \setminus finishedQuiz)$

The creator of a quiz shall not be allowed to remove a question from the quiz if there exists an active quiz instance, linked to the quiz, being hosted at the same time.
then
act41;: $questions := questions \setminus \{quest\}$
act42;: $belongingQuiz := \{quest\} \lhd belongingQuiz$
act43;: $questionPosition := \{quest\} \lhd questionPosition$
The position of the question is removed
act51;: $answers := answers \setminus \{a | a \in answers \land belongingQuestion(a) = quest\}$
the answers of the question must be removed when the question is removed
act52;: $correctAnswer := \{quest\} \lhd correctAnswer$
The correct answer of quest is not the correct answer anymore
act53;: $belongingQuestion := belongingQuestion \rhd \{quest\}$
All answers of the question do not belong to it anymore
end
Event UPDATE_QUESTION_OK ⟨ordinary⟩ ≙
extends UPDATE_QUESTION_OK
any
u
$quest$
q
$result$
where
grd41;: $u \in loggedIn$
grd42;: $q \in quizzes$
grd43;: $quizCreator(q) = u$
Only the quiz creator of q can update the question
grd44;: $quest \in questions$
grd45;: $result = TRUE$
The update has been succesfully
grd46;: $belongingQuiz(quest) = q$
the question must belong to q
grd61;: $u \notin host[quizInstances \setminus finishedQuiz]$
The host of an active quiz instance shall not be allowed to update a question of a quiz at the same time.
grd62: $\forall qi \cdot qi \in quizInstances \land qi \in instanceOfQuiz^{-1}[\{q\}] \Rightarrow qi \notin (quizInstances \setminus finishedQuiz)$

The creator of a quiz shall not be allowed to update a question of the quiz if there exists an active quiz instance, linked to the quiz, being hosted at the same time.

then

skip

end

Event UPDATE_QUESTION_NOT_OK ⟨ordinary⟩ ≙

extends UPDATE_QUESTION_NOT_OK

any

u

$quest$

q

$result$

where

grd41;: $u \in loggedIn$

grd42;: $q \in quizzes$

grd43;: $quizCreator(q) = u$

grd1: $quest \in questions$

grd45;: $belongingQuiz(quest) = q$

grd46;: $result = FALSE$

The update has not been successfully. The previous text of the question is restored

grd61;: $u \notin host[quizInstances \setminus finishedQuiz]$

The host of an active quiz instance shall not be allowed to update a question of a quiz at the same time.

grd62: $\forall qi \cdot qi \in quizInstances \land qi \in instanceOfQuiz^{-1}[\{q\}] \Rightarrow qi \notin (quizInstances \setminus finishedQuiz)$

The creator of a quiz shall not be allowed to update a question of the quiz if there exists an active quiz instance, linked to the quiz, being hosted at the same time.

then

skip

end

Event HOST_QUIZ_INSTANCE ⟨ordinary⟩ ≙

extends HOST_QUIZ_INSTANCE

any

u

q

qi

r

where

grd61;: $u \in loggedIn$

The user u must be logged in

grd62;: $qi \in INSTANCE\_QUIZ \setminus quizInstances$

grd63;: $q \in quizzes$

grd65;: $u \notin ran((finishedQuiz \lhd host))$

The user u must not be hosting any other active quiz instance

grd66: $u \in (\{quizCreator(q)\} \cup sharedTo[\{q\}])$

The user u must be the creator or have access to the quiz

grd67: $belongingQuiz^{-1}[\{q\}] \neq \varnothing$

The quiz must have at least 1 question

grd91: $r \in REPORT \setminus reports$

A new report is created. Initially the report is an empty file. When the quiz instance terminates the report

will be populated with the information resulting from the content of the question summaries

then

act61;: $quizInstances := quizInstances \cup \{qi\}$

act62;: $host(qi) := u$

The host of the quiz instance is u

act63;: $instanceOfQuiz(qi) := q$

The quiz instance is linked to the quiz q

act64;: $quizCreated := quizCreated \cup \{qi\}$
The quiz instance is in the quizCreated state
act91: $embeddedQuestions := embeddedQuestions \cup (\{qi\} \times \{quests | quests \in questions \wedge belongingQuiz(quests) = q\})$
The quiz instance is linked to the questions of the quiz
act92: $reports := reports \cup \{r\}$
The report is added to the set of reports
act93: $reportedBy(qi) := r$
The quiz instance is linked to the report

end

Event BEGIN_QUIZ_INSTANCE ⟨ordinary⟩ $\widehat{=}$
extends BEGIN_QUIZ_INSTANCE

any
u
qi

where
grd61;: $u \in loggedIn$
The user u must be logged in
grd62;: $qi \in quizCreated$
The quiz instance must be in the quizCreated state
grd63;: $host(qi) = u$
The host of the quiz instance must be u

then
act61;: $quizCreated := quizCreated \setminus \{qi\}$
The quiz instance is not in the quizCreated state anymore
act62;: $quizInit := quizInit \cup \{qi\}$
The quiz instance becomes part of the quizInit state set
act63;: $answeredQuestions(qi) := 0$
No question has been answered so far

end

Event START_QUIZ_INSTANCE ⟨ordinary⟩ $\widehat{=}$
extends START_QUIZ_INSTANCE

any
u
qi
$quest$

where
grd61;: $u \in loggedIn$
The user u must be logged in
grd62;: $qi \in quizInit$
The quiz instance must be in the quizInit state
grd63;: $host(qi) = u$
The host of the quiz instance must be u
grd64;: $\forall k \cdot belongingQuiz(k) = instanceOfQuiz(qi) \Rightarrow questionPosition(k) \geq questionPosition(quest)$
The question of the quiz that will become the first question to be answered in the quiz instance must be the first one in the question list of the quiz
grd65: $qi \in ran(playingIn)$
Some players must have joined the quiz during the quizInit state

then
act61;: $quizInit := quizInit \setminus \{qi\}$
The quiz is not in the quizInit state anymore
act62;: $questioning := questioning \cup \{qi\}$
The quiz instance is in the questioning state
act63;: $currentQuestion(qi) := quest$
The current question of the quiz instance is the first question of the question list of the quiz

end

Event JOIN_QUIZ_INSTANCE ⟨ordinary⟩ $\widehat{=}$
extends JOIN_QUIZ_INSTANCE

any

u

qi

where

grd61;: $u \in USER$

The player can be either regitered or unregistered

grd62;: $qi \in quizInit$

It is possible to join a quiz instanec only if it is in the quizInit state

grd63;: $u \neq host(qi)$

The player must not be the host of the quiz instance

grd64;: $u \notin dom(playingIn)$

The player must not be playing in another quiz instance

grd1: $u \notin host[quizInstances \setminus finishedQuiz]$

The player cannot be any registered user that is hosting any active quiz instance

then

act64;: $playingIn := playingIn \cup \{u \mapsto qi\}$

end

Event ANSWER_QUESTION ⟨ordinary⟩ $\widehat{=}$

extends ANSWER_QUESTION

any

p

qi

$quest$

a

where

grd61;: $p \in USER$

grd62;: $qi \in quizInstances$

grd63;: $p \in playingIn^{-1}[\{qi\}]$

The player must be playing in the quiz instance

grd64;: $qi \in questioning$

The quiz instance must be in the questioning state

grd65: $quest \in questions$

grd66: $currentQuestion(qi) = quest$

The question the player is answering must be the current question

grd67: $a \in answers$

grd68: $a \in belongingQuestion^{-1}[\{quest\}]$

The answer being selected must belong to the current question of the quiz instance

grd69: $card(peopleAnswers^{-1}[\{qi\}]) < card(playingIn^{-1}[\{qi\}])$

A player of a quiz instance shall not be allowed to select an answer when all other players have answered because the quiz instance passes to the questionSummary state.

then

act61;: $peopleAnswers := peopleAnswers \cup \{p \mapsto qi\}$

act62: $peopleSelectedAnswer := peopleSelectedAnswer \lhd\!\!+ \{p \mapsto a\}$

end

Event END_QUESTION_ALL_ANSWERED ⟨ordinary⟩ $\widehat{=}$

extends END_QUESTION_ALL_ANSWERED

any

qi

qs

r

where

grd61;: $qi \in questioning$

The quiz instance must be in the questionig state

grd62;: $card(peopleAnswers^{-1}[\{qi\}]) = card(playingIn^{-1}[\{qi\}])$

All players must have answered

grd91: $qs \in QUESTION\_SUMMARY \setminus qSummary$

A question summary is generated. The question summary is populated with information about the performance of the players by using the information resulting from the relation peopleSelectedAnswer

grd92: $r \in reports$
grd93: $reportedBy(qi) = r$
r must be the report of the quiz instance

then

act61;: $questioning := questioning \setminus \{qi\}$
The quiz instance is not part of the questioning state anymore
act62;: $questionSummary := questionSummary \cup \{qi\}$
The quiz instance passes to the questionSummary state
act63;: $answeredQuestions(qi) := answeredQuestions(qi) + 1$
The number of answered questions is incremented by 1
act64;: $peopleAnswers := peopleAnswers \rhd \{qi\}$
The players that have answered the question of the quiz instance are removed
act65: $peopleSelectedAnswer := playingIn^{-1}[\{qi\}] \lhd peopleSelectedAnswer$
The answers select by the players of the quiz instance are removed
act91: $qSummary := qSummary \cup \{qs\}$
act92: $summaryFor(qs) := (qi \mapsto currentQuestion(qi))$
The summary is linked to the quiz instance and current question of the quiz instance
act93: $qReportedBy(qs) := r$
The question summary is linked to the report of the quiz instance

end

Event END_QUESTION_PREMATURATELY ⟨ordinary⟩ $\widehat{=}$
extends END_QUESTION_PREMATURATELY

any

u
qi
qs
r

where

grd61;: $qi \in questioning$
The quiz instance must be in the question state
grd62;: $u \in loggedIn$
The user u must be logged in
grd63;: $host(qi) = u$
The host of the quiz instance must be u
grd1: $card(peopleAnswers^{-1}[\{qi\}]) \neq card(playingIn^{-1}[\{qi\}])$
The host of a quiz instance in the questioning state
shall not be allowed to end prematurately a question
if all players have answered because the quiz instance passes to the questionSummary state.
grd91: $qs \in QUESTION\_SUMMARY \setminus qSummary$
grd92: $r \in reports$
grd93: $reportedBy(qi) = r$

then

act61;: $questioning := questioning \setminus \{qi\}$
The quiz instance is not in the questioning state anymore
act62;: $questionSummary := questionSummary \cup \{qi\}$
The quiz instance passes to the questionSummary state
act63;: $answeredQuestions(qi) := answeredQuestions(qi) + 1$
The number of answered questions of the quiz instance is incremebted by 1
act64;: $peopleAnswers := peopleAnswers \rhd \{qi\}$
The players that have answered the question of the quiz instance are removed
act65: $peopleSelectedAnswer := playingIn^{-1}[\{qi\}] \lhd peopleSelectedAnswer$
The answers select by the players of the quiz instance are removed
act91: $qSummary := qSummary \cup \{qs\}$
act92: $summaryFor(qs) := qi \mapsto currentQuestion(qi)$
act93: $qReportedBy(qs) := r$

end

Event SHOW_QUESTION_SUMMARY_TO_PLAYERS ⟨ordinary⟩ $\widehat{=}$
extends SHOW_QUESTION_SUMMARY_TO_PLAYERS

any

u
qi
qs

where

grd61;: $u \in loggedIn$
The user u must be logged in
grd62;: $qi \in quizInstances$
grd63;: $host(qi) = u$
The user u must be the host of the quiz instance
grd63: $qi \in questionSummary$
The quiz instance must be in the questionSummary state
grd64: $qi \in ran(playingIn)$
The host of a quiz instance in the questionSummary state shall
be allowed to show the question summary only if at least 1 player is playing in the quiz instance.
grd91: $summaryFor(qs) = qi \mapsto currentQuestion(qi)$
The summary to show must be the
summary linked to the quiz instance and current question
of the quiz instance

then

skip

end

Event NEXT_QUESTION ⟨ordinary⟩ $\widehat{=}$
extends NEXT_QUESTION

any

u
qi
$quest$

where

grd61;: $u \in loggedIn$
grd62;: $qi \in questionSummary$
The quiz instance must be in the questionSummary state
grd63;: $host(qi) = u$
grd64;: $answeredQuestions(qi) < card(belongingQuiz^{-1}[\{instanceOfQuiz(qi)\}])$
There must be other questions to be answered
grd65: $qi \in ran(playingIn)$
At least 1 player must be playing in the quiz instance
grd66: $quest \in questions$
grd67: $card(\{quests | quests \in questions \wedge quests \in belongingQuiz^{-1}[\{instanceOfQuiz(qi)\}] \wedge questionPosition(quests) < questionPosition(quest)\}) = answeredQuestions(qi)$
The next question to be answered must be the successive one

then

act61;: $questionSummary := questionSummary \setminus \{qi\}$
The quiz instance is not in the questionSummary state anymore
act62;: $questioning := questioning \cup \{qi\}$
The quiz instance passes to the questioning state
act63: $currentQuestion(qi) := quest$
The current qyestion is set to the successive question

end

Event FINISH_QUIZ_INSTANCE ⟨ordinary⟩ $\widehat{=}$
extends FINISH_QUIZ_INSTANCE

any

u
qi

where

grd61;: $u \in loggedIn$
The user must be logged in
grd62;: $qi \in questionSummary$
The quiz instance must be in the questionSummary state

grd63;: $host(qi) = u$
The user u must be the host of the quiz instance

then

act61;: $questionSummary := questionSummary \setminus \{qi\}$
The quiz instance is not in the questionSummary state anymore
act62;: $quizSummary := quizSummary \cup \{qi\}$
The quiz instance passes to the quizSummary state
act63;: $currentQuestion := \{qi\} \lhd currentQuestion$
The current question of the quiz instance is removed as it is not anymore in the questionSummary state

end

Event SHOW_QUIZ_INSTANCE_SUMMARY_TO_PLAYERS ⟨ordinary⟩ $\widehat{=}$
extends SHOW_QUIZ_INSTANCE_SUMMARY_TO_PLAYERS

any

qi
u
r

where

grd61;: $qi \in quizInstances$
grd62: $qi \in quizSummary$
The quiz instance must be in the quizSummary state
grd63: $u \in registeredUser$
grd64: $u \in loggedIn$
The user u must be logged in
grd65: $host(qi) = u$
The user u must be the host of the quiz instance
grd66: $qi \in ran(playingIn)$
there must at least 1 player playing in the quiz instance
grd91: $r \in reports$
grd92: $reportedBy(qi) = r$
the report to show must be the report of the quiz instance

then

skip

end

Event TERMINATE_QUIZ_INSTANCE_QUIZ_SUMMARY_STATE ⟨ordinary⟩ $\widehat{=}$
extends TERMINATE_QUIZ_INSTANCE_QUIZ_SUMMARY_STATE

any

u
qi

where

grd61;: $u \in loggedIn$
The user u must be loggedin
grd62;: $qi \in quizSummary$
The quiz instance must be in the quizSummary state
grd63;: $host(qi) = u$
The user u must be the host of the quiz instance

then

act61;: $quizSummary := quizSummary \setminus \{qi\}$
The quiz instance is not in the quizSummary state anymore
act62;: $finishedQuiz := finishedQuiz \cup \{qi\}$
The quiz instance passes to the finishedQuiz state
act63: $playingIn := playingIn \rhd \{qi\}$
The players of the quiz instance do not play anymore in the quiz instance
act91: $embeddedQuestions := \{qi\} \lhd embeddedQuestions$
The quiz instance is not linked anymore to the questions of the quiz instance
act92: $summaryFor := summaryFor \rhd (\{qi\} \lhd embeddedQuestions)$
The question summaries do not report anymore a specific question of the quiz instance
act93: $qSummary := qSummary \setminus summaryFor^{-1}[\{qi\} \lhd embeddedQuestions]$
All the summaries of the quiz instance built so far must be deleted

act94: $qReportedBy := qReportedBy \rhd reportedBy[\{qi\}]$
The question summaries are not linked anymore to the report of the quiz instance

end

Event LEAVE_QUIZ_INSTANCE_LAST_ONE_IN_QUESTIONING_STATE ⟨ordinary⟩ $\widehat{=}$
extends LEAVE_QUIZ_INSTANCE_LAST_ONE_IN_QUESTIONING_STATE

any

p
qi
qs
r

where

grd61: $p \in USER$
grd62: $qi \in questioning$
The quiz instance must be in the questioning state
grd63: $p \in playingIn^{-1}[\{qi\}]$
The player must be playing in the quiz instance
grd64: $card(playingIn^{-1}[\{qi\}]) = 1$
The player must be the only player be playing in the quiz instance
grd91: $qs \in QUESTION\_SUMMARY \setminus qSummary$
A question summary is generated
grd92: $r \in reports$
grd93: $reportedBy(qi) = r$
Thw question summary is linked to the report of the quiz instance

then

act61: $playingIn := \{p\} \lhd playingIn$
The player does not play anymore in the quiz instance
act62: $peopleAnswers := \{p\} \lhd peopleAnswers$
It is removed whether the player has answered the question or not
act63: $questioning := questioning \setminus \{qi\}$
act64: $questionSummary := questionSummary \cup \{qi\}$
The quiz instance passes to the questionSummary state
act65: $peopleSelectedAnswer := \{p\} \lhd peopleSelectedAnswer$
The answer selected by the player is removed (if he/she has naswered)
act91: $qSummary := qSummary \cup \{qs\}$
act92: $summaryFor(qs) := qi \mapsto currentQuestion(qi)$
act93: $qReportedBy(qs) := r$

end

Event LEAVE_QUIZ_INSTANCE_NOT_IN_QUESTIONING_STATE ⟨ordinary⟩ $\widehat{=}$
extends LEAVE_QUIZ_INSTANCE_NOT_IN_QUESTIONING_STATE

any

p
qi

where

grd61: $p \in USER$
grd62: $qi \in quizInstances \setminus (quizCreated \cup finishedQuiz \cup questioning)$
The quiz instance must not be in the quiz created state, finishedQuiz or questioning state
grd63: $p \in playingIn^{-1}[\{qi\}]$
The player must be playing in the quiz instance

then

act61: $playingIn := \{p\} \lhd playingIn$
The player does not play anymore in the quiz instance

end

Event LEAVE_QUIZ_INSTANCE__NOT_LAST_ONE_IN_QUESTIONING_STATE ⟨ordinary⟩ $\widehat{=}$
extends LEAVE_QUIZ_INSTANCE_NOT_LAST_ONE_IN_QUESTIONING_STATE

any

p
qi

where

grd61: $p \in USER$
grd62: $qi \in questioning$
grd63: $p \in playingIn^{-1}[\{qi\}]$
grd64: $card(playingIn^{-1}[\{qi\}]) > 1$
There must be at least 2 players be playing

then

act61: $playingIn := \{p\} \lhd playingIn$
act62: $peopleAnswers := \{p\} \lhd peopleAnswers$
act63: $peopleSelectedAnswer := \{p\} \lhd peopleSelectedAnswer$

end

Event UNSHARE_QUIZ ⟨ordinary⟩ $\widehat{=}$
extends UNSHARE_QUIZ

any

$u1$
$u2$
q

where

grd41: $u1 \in registeredUser$
grd42: $u2 \in registeredUser$
grd43: $u1 \in loggedIn$
A registerd user can unshare only if he/she is online
grd44: $u1 \neq u2$
grd45: $q \mapsto u2 \in sharedTo$
u2 had to have access to the quiz
grd46: $u1 = quizCreator(q)$
Only the creator of the quiz can unshare a quiz
grd81: $u2 \notin host[instanceOfQuiz^{-1}[\{q\}] \setminus finishedQuiz]$
The creator of a quiz shall not be allowed to deshare a quiz
with another registered user if the latter is hosting a quiz instance linked to the quiz.
grd82: $u1 \notin host[quizInstances \setminus finishedQuiz]$
The creator of a quiz shall not be allowed to deshare a
quiz if he/she is hosting an active quiz instance.

then

act41: $sharedTo := sharedTo \setminus \{q \mapsto u2\}$

end

Event REMOVE_QUIZ_INSTANCE ⟨ordinary⟩ $\widehat{=}$
extends REMOVE_QUIZ_INSTANCE

any

u
qi

where

grd81: $u \in registeredUser$
grd82: $qi \in quizInstances$
grd83: $qi \in finishedQuiz$
It is possible to remove a quiz instance only when it is in the finishedQuiz state
grd84: $u \in loggedIn$
The user must be logged in
grd85: $host(qi) = u$
The user u must be the host of the quiz instance
grd86: $u \notin host[quizInstances \setminus finishedQuiz]$
The user must not be hosting any active quiz instance

then

act81: $quizInstances := quizInstances \setminus \{qi\}$
The quiz instance is removed
act82: $finishedQuiz := finishedQuiz \setminus \{qi\}$
The quiz instance is removed from the finishedQuiz state
act83: $host := \{qi\} \lhd host$
The user is not the host of the quiz instance anymore

act84: $instanceOfQuiz := \{qi\} \lhd instanceOfQuiz$
The quiz instance is not linked anymore to a quiz
act85: $answeredQuestions := \{qi\} \lhd answeredQuestions$
The number of answered questions of the quiz instance is removed
act91: $reportedBy := \{qi\} \lhd reportedBy$
The report of the quiz instance is removed
act92: $reports := reports \setminus \{reportedBy(qi)\}$
The report is not macthed anymore to the quiz instance

end

Event TERMINATE_QUIZ_INSTANCE_QUIZ_CREATED_STATE ⟨ordinary⟩ $\widehat{=}$
extends TERMINATE_QUIZ_INSTANCE_QUIZ_CREATED_STATE

any
u
qi
where
grd91: $u \in registeredUser$
grd92: $u \in loggedIn$
The user u must be loggedIn
grd93: $qi \in quizInstances$
grd94: $qi \in quizCreated$
The quiz instance must be in the quizCreated state
grd95: $host(qi) = u$
The user u must be the host of the quiz instance
then
act91: $quizCreated := quizCreated \setminus \{qi\}$
The quiz instance is removed from the quizCreated state
act92: $finishedQuiz := finishedQuiz \cup \{qi\}$
The quiz instance passes to the finishedQuiz state
act93: $answeredQuestions(qi) := 0$
The number of answered questions is set to 0
act94: $embeddedQuestions := \{qi\} \lhd embeddedQuestions$
The quiz instance is not linked anymore to the questions of the quiz instance
end

Event TERMINATE_QUIZ_INSTANCE_QUIZ_INIT_STATE ⟨ordinary⟩ $\widehat{=}$
extends TERMINATE_QUIZ_INSTANCE_QUIZ_INIT_STATE

any
u
qi
where
grd81: $u \in registeredUser$
grd82: $u \in loggedIn$
The user u must be loggedin
grd83: $qi \in quizInstances$
grd84: $qi \in quizInit$
The quiz instance must be in the quizInit state
grd85: $host(qi) = u$
The user u must be the host of the quiz instance
then
act81: $quizInit := quizInit \setminus \{qi\}$
The quiz instance is not anymore in the quizInit state
act82: $finishedQuiz := finishedQuiz \cup \{qi\}$
The quiz instance passes to the finishedQuiz state
act83: $playingIn := playingIn \rhd \{qi\}$
The players of the quiz instance do not play anymore in the quiz instance
act91: $embeddedQuestions := \{qi\} \lhd embeddedQuestions$
The quiz instance is not linked anymore to the questions of the quiz instance

end

Event END_QUESTION_TIME_IS_UP ⟨ordinary⟩ $\widehat{=}$
extends END_QUESTION_TIME_IS_UP

any

qi

qs

r

where

grd61: $qi \in quizInstances$

grd62: $qi \in questioning$

The quiz instance must be in the questioning state

grd91: $qs \in QUESTION\_SUMMARY \setminus qSummary$

grd92: $r \in reports$

grd93: $reportedBy(qi) = r$

then

act61: $questioning := questioning \setminus \{qi\}$

act62: $questionSummary := questionSummary \cup \{qi\}$

The quiz instance passes to the questionSummary state

act63: $answeredQuestions(qi) := answeredQuestions(qi) + 1$

The number of answered questions of the quiz instance is incremented by 1

act64: $peopleAnswers := peopleAnswers \rhd \{qi\}$

act65: $peopleSelectedAnswer := playingIn^{-1}[\{qi\}] \lhd peopleSelectedAnswer$

act91: $qSummary := qSummary \cup \{qs\}$

act92: $summaryFor(qs) := (qi \mapsto currentQuestion(qi))$

act93: $qReportedBy(qs) := r$

end

END

MACHINE m10
REFINES m9
SEES c6
VARIABLES

registeredUser
password
loggedIn
loggedOut
quizzes
questions
quizCreator
belongingQuiz
sharedTo
questionPosition
answers
correctAnswer
belongingQuestion
quizInstances
host
playingIn
instanceOfQuiz
currentQuestion
quizInit
questioning
questionSummary
quizSummary
finishedQuiz
answeredQuestions
peopleAnswers
quizCreated
embeddedQuestions
reports
summaryFor
reportedBy
qSummary
qReportedBy
peopleSelectedAnswer
time
questionTime
questionEndsAt
questionStartedAt

INVARIANTS

inv101: $time \in \mathbb{N}$
This variables indicates the time expressed in minutes. For example, if time = 1 then 1 minute is passed from time 0

inv103: $questionTime \in questions \rightarrow QUESTION\_TIME$
A question shall be matched to a time within which players can select an answer during a quiz instance.

inv104: $questionEndsAt \in questioning \rightarrow \mathbb{N}$
A quiz instance in the questioning state shall be matched with the time when it will transition into the questionSummary state.

inv105: $questionStartedAt \in questioning \rightarrow \mathbb{N}$
A quiz instance in the questioning state shall be matched with the time when it has transitioned into the questioning state

inv106: $\forall qi, t, n \cdot qi \in quizInstances \land qi \in questioning \land t \in \mathbb{N} \land qi \mapsto t \in questionEndsAt \land n \in \mathbb{N} \land n = questionTime(currentQuestion(qi)) \Rightarrow t = questionStartedAt(qi) + n$
A quiz instance passes from the questionig state to the questionSummary state
after time n from when it has started where n=question time of the current question
of the quiz instance

EVENTS

Initialisation ⟨extended⟩

begin

act1;: $registeredUser := \varnothing$
act21;: $password := \varnothing$
act31;: $loggedIn := \varnothing$
act32;: $loggedOut := \varnothing$
act41;: $quizzes := \varnothing$
act42;: $questions := \varnothing$
act43;: $quizCreator := \varnothing$
act44;: $belongingQuiz := \varnothing$
act45;: $sharedTo := \varnothing$
act46;: $questionPosition := \varnothing$
act51;: $answers := \varnothing$
act52;: $correctAnswer := \varnothing$
act53;: $belongingQuestion := \varnothing$
act61;: $quizInstances := \varnothing$
act62;: $host := \varnothing$
act63;: $playingIn := \varnothing$
act64;: $instanceOfQuiz := \varnothing$
act65;: $peopleAnswers := \varnothing$
act66: $answeredQuestions := \varnothing$
act67: $currentQuestion := \varnothing$
act68: $finishedQuiz := \varnothing$
act69: $questioning := \varnothing$
act611: $questionSummary := \varnothing$
act612: $quizCreated := \varnothing$
act613: $quizInit := \varnothing$
act614: $quizSummary := \varnothing$
act615: $peopleSelectedAnswer := \varnothing$
act91: $embeddedQuestions := \varnothing$
act92: $reports := \varnothing$
act93: $summaryFor := \varnothing$
act94: $reportedBy := \varnothing$
act95: $qSummary := \varnothing$
act96: $qReportedBy := \varnothing$
act616: $time := 0$
act617: $questionTime := \varnothing$
act618: $questionEndsAt := \varnothing$
act619: $questionStartedAt := \varnothing$

end

Event REGISTER ⟨ordinary⟩ $\widehat{=}$
extends REGISTER

any

u A user
p

where

grd1;: $u \in USER \setminus registeredUser$
The user must not be already registered
grd21;: $p \in PASSWORD$
grd81: $u \notin dom(playingIn)$

then

act1;: $registeredUser := registeredUser \cup \{u\}$
The user is added to the set of registered users

act21;: $password(u) := p$
The password of user u is p
act31;: $loggedOut := loggedOut \cup \{u\}$
When a user u registers, u is added to the set of loggedOut users

end

Event LOGIN ⟨ordinary⟩ $\widehat{=}$
extends LOGIN

any
u
p
where
inv31;: $u \in loggedOut$
inv32;: $p = password(u)$
p must be the pssword of u
grd81: $u \notin dom(playingIn)$
then
act31;: $loggedIn := loggedIn \cup \{u\}$
The user u is added to the set of loggedIn users
act32;: $loggedOut := loggedOut \setminus \{u\}$
The user u is removed from the set of loggedOut users

end

Event LOGOUT ⟨ordinary⟩ $\widehat{=}$
extends LOGOUT

any
u
where
inv31;: $u \in loggedIn$
The user u mst be logged in
grd81: $u \notin host[quizInstances \setminus finishedQuiz]$
The host of an active quiz instance shall not be allowed to log out of the system.
grd82: $u \notin dom(playingIn)$
then
act31;: $loggedIn := loggedIn \setminus \{u\}$
The user u is removed from the set of loggedIn users
act32;: $loggedOut := loggedOut \cup \{u\}$
The user u is added to the set of loggedOut users

end

Event CREATEQUIZ ⟨ordinary⟩ $\widehat{=}$
extends CREATEQUIZ

any
u
q
where
grd41;: $u \in loggedIn$
A user must be logged in so that to create a quiz
grd42;: $q \in QUIZ \setminus quizzes$
The created quiz must not have been already created
grd61;: $u \notin host[quizInstances \setminus finishedQuiz]$
The host of an active quiz instance shall not be allowed to create a quiz at the same time.
then
act41;: $quizzes := quizzes \cup \{q\}$
The created quiz is added to the dynamic set quizzes
act42;: $quizCreator(q) := u$
the creator of the quiz q is u

end

Event REMOVEQUIZ ⟨ordinary⟩ $\widehat{=}$
extends REMOVEQUIZ

any

u

q

where

grd41;: $q \in quizzes$

grd42;: $u \in loggedIn$

The user must be logged in so that to remove a quiz

grd43;: $quizCreator(q) = u$

Only the creator of the quiz can remove the quiz

grd61;: $u \notin host[quizInstances \setminus finishedQuiz]$

The host of an active quiz instance shall not be allowed to remove a quiz at the same time.

grd62: $\forall qi \cdot qi \in quizInstances \land qi \in instanceOfQuiz^{-1}[\{q\}] \Rightarrow qi \notin (quizInstances \setminus finishedQuiz)$

The creator of a quiz shall not be allowed to remove a quiz if there exists an active quiz instance, linked to the quiz, being hosted at the same time.

then

act41;: $quizzes := quizzes \setminus \{q\}$

the quiz is removed

act42;: $quizCreator := \{q\} \lhd quizCreator$

The quiz does not have anymore a creator because is removed

act43;: $sharedTo := \{q\} \lhd sharedTo$

All registerd users that had access to the quiz, do not have access to it anymore because the quiz is removed

act44;: $belongingQuiz := belongingQuiz \rhd \{q\}$

The questions of the quiz do not belong anymore to it

act45;: $questions := questions \setminus \{quest | quest \in questions \land belongingQuiz(quest) = q\}$

The questions of the quiz are removed

act46;: $questionPosition := \{quest | quest \in questions \land belongingQuiz(quest) = q\} \lhd questionPosition$

The position of the questions is removed

act54;: $correctAnswer := \{q2 | q2 \in questions \land belongingQuiz(q2) = q\} \lhd correctAnswer$

The correct answer of all questions of the quiz must be removed

act55;: $belongingQuestion := belongingQuestion \rhd \{q2 | q2 \in questions \land belongingQuiz(q2) = q\}$

The answers of the questions of the quiz do not belong to the questions anymore

act56: $answers := answers \setminus \{a1 | a1 \in answers \land belongingQuestion(a1) \in \{quest1 | quest1 \in questions \land belongingQuiz(quest1) = q\}\}$

Removing a quiz involves removing all answers of the questions of the quiz

act61: $instanceOfQuiz := instanceOfQuiz \rhd \{q\}$

act101: $questionTime := belongingQuiz^{-1}[\{q\}] \lhd questionTime$

When removing a quiz, it is necessary to remove the question time of each of its questions

end

Event SHAREQUIZ ⟨ordinary⟩ $\widehat{=}$

extends SHAREQUIZ

any

$u1$

$u2$

q

where

grd41;: $u1 \in loggedIn$

the user must be logged in so that to share the quiz

grd42;: $u2 \in registeredUser$

the other user that will have access to the quiz

grd43;: $u1 \neq u2$

the two users cannot be the same person

grd44;: $q \in quizzes$

grd45;: $quizCreator(q) = u1$

u1 must be the creator of the quiz

grd46: $q \in ran(belongingQuiz)$

The quiz must contain at least 1 question in order to be shared

grd47: $q \mapsto u2 \notin sharedTo$

the quiz must not have already been shared with u2

grd61;: $u1 \notin host[quizInstances \setminus finishedQuiz]$
The host of an active quiz instance shall not be allowed to share a quiz at the same time.

then

act41;: $sharedTo := sharedTo \cup \{q \mapsto u2\}$

end

Event CREATEQUESTION ⟨ordinary⟩ $\widehat{=}$

extends CREATEQUESTION

any

$quest$
q
u
p
a
b
t

where

grd41;: $quest \in QUESTION \setminus questions$
The question must be a new question

grd42;: $q \in quizzes$

grd43;: $u \in loggedIn$
The user must be logged in in order to create a new question

grd44;: $quizCreator(q) = u$
The creator of the quiz must be u

grd45;: $p \in POSITION$

grd46;: $\forall z \cdot z \in questions \land belongingQuiz(z) = q \Rightarrow p > questionPosition(z)$
The question must be added at the end of the question list of the quiz

grd51;: $(a \in ANSWER \setminus answers) \land (b \in ANSWER \setminus answers)$
The 2 answers must be new ones

grd52: $a \neq b$
The 2 answers must be different

grd61;: $u \notin host[quizInstances \setminus finishedQuiz]$
The host of an active quiz instance shall not be allowed to add a question to a quiz at the same time.

grd62: $\forall qi \cdot qi \in quizInstances \land qi \in instanceOfQuiz^{-1}[\{q\}] \Rightarrow qi \notin (quizInstances \setminus finishedQuiz)$

The creator of a quiz shall not be allowed to add a question to a quiz if there exists an active quiz instance, linked to the quiz, being hosted at the same time.

grd101: $t \in QUESTION\_TIME$

then

act41;: $questions := questions \cup \{quest\}$

act42;: $belongingQuiz(quest) := q$
the question quest belongs to q

act43;: $questionPosition(quest) := p$
The position of the question quest is p inside the question list of the quiz q

act51;: $answers := answers \cup \{a, b\}$

act53;: $correctAnswer(quest) := a$
The correct answer is a

act54;: $belongingQuestion := belongingQuestion \cup \{a \mapsto quest, b \mapsto quest\}$
The 2 new answers belong to the question quest

act101: $questionTime(quest) := t$
When creatig a question, it is neccìessary to match it with a time

end

Event ADDANSWER ⟨ordinary⟩ $\widehat{=}$

REQ7

extends ADDANSWER

any

$quest$
q
u

a

where

grd51;: $u \in loggedIn$

The user us must be logged in

grd52;: $q \in quizzes$

grd53;: $a \in ANSWER \setminus answers$

grd54;: $quest \in questions$

grd55;: $quizCreator(q) = u$

The user u must be the creator of the quiz

grd56;: $belongingQuiz(quest) = q$

grd57;: $card(\{a1 | a1 \in answers \land belongingQuestion(a1) = quest\}) < 4$

The current answers of the question must be less than 4

grd61;: $u \notin host[quizInstances \setminus finishedQuiz]$

The host of an active quiz instance shall not be allowed to add an answer to a question of a quiz at the same time.

grd62: $\forall qi \cdot qi \in quizInstances \land qi \in instanceOfQuiz^{-1}[\{q\}] \Rightarrow qi \notin (quizInstances \setminus finishedQuiz)$

The creator of a quiz shall not be allowed to add an answer to a question of a quiz if there exists an active quiz instance, linked to the quiz, being hosted at the same time.

then

act51;: $answers := answers \cup \{a\}$

act52;: $belongingQuestion(a) := quest$

end

Event REMOVENOTCORRECTANSWER ⟨ordinary⟩ $\widehat{=}$

extends REMOVENOTCORRECTANSWER

any

$quest$

q

u

a

where

grd51;: $u \in loggedIn$

The user u must be logged in

grd52;: $q \in quizzes$

grd53;: $quest \in questions$

grd54;: $a \in answers$

grd55;: $card(\{a1 | a1 \in answers \land belongingQuestion(a1) = quest\}) > 2$

The number of current answers of the question must be more than 2

grd56;: $correctAnswer(quest) \neq a$

The answer to be removed must not be the correct answer of the question

grd57: $quizCreator(q) = u$

The user u must be the creator of the quiz

grd58: $belongingQuiz(quest) = q$

grd59: $belongingQuestion(a) = quest$

The anser must belong to the quiz q

grd61;: $u \notin host[quizInstances \setminus finishedQuiz]$

The host of an active quiz instance shall not be allowed to remove a non-correct answer from a question of a quiz at the same time.

grd62: $\forall qi \cdot qi \in quizInstances \land qi \in instanceOfQuiz^{-1}[\{q\}] \Rightarrow qi \notin (quizInstances \setminus finishedQuiz)$

The creator of a quiz shall not be allowed to remove a non-correct answer from a question of the quiz if there exists an active quiz instance, linked to the quiz, being hosted at the same time.

then

act51;: $answers := answers \setminus \{a\}$

act52;: $belongingQuestion := \{a\} \lhd belongingQuestion$

end

Event REMOVECORRECTANSWER ⟨ordinary⟩ $\widehat{=}$

extends REMOVECORRECTANSWER

any

$quest$
q
u
$a1$
$a2$

where

grd51;: $u \in loggedIn$
The user u must be logged in
grd52;: $quest \in questions$
grd53;: $q \in quizzes$
grd54;: $a1 \in answers$
grd55;: $a2 \in answers$
grd56;: $correctAnswer(quest) = a1$
The answer to be removed must be the correct answer of the question
grd57;: $belongingQuestion(a2) = quest$
The new correct answer must belong to the question
grd58: $quizCreator(q) = u$
The user u must be the creator of the quiz
grd59: $belongingQuiz(quest) = q$
grd60: $belongingQuestion(a1) = quest$
grd61: $card(\{a|a \in answers \land belongingQuestion(a) = quest\}) > 2$
The current number of answers of the question must be greater than 2
grd62: $a1 \neq a2$
grd61;: $u \notin host[quizInstances \setminus finishedQuiz]$
The host of an active quiz instance shall not be allowed to remove the correct answer from a question of a quiz at the same time.
grd63: $\forall qi \cdot qi \in quizInstances \land qi \in instanceOfQuiz^{-1}[\{q\}] \Rightarrow qi \notin (quizInstances \setminus finishedQuiz)$

The creator of a quiz shall not be allowed to remove the correct answer from a question of the quiz if there exists an active quiz instance, linked to the quiz, being hosted at the same time

then

act51;: $answers := answers \setminus \{a1\}$
act52;: $belongingQuestion := \{a1\} \lhd belongingQuestion$
act53;: $correctAnswer := correctAnswer \lhd\!\!+ \{quest \mapsto a2\}$
the new coorect answer is a2

end

Event SETCORRECTANSWER ⟨ordinary⟩ $\widehat{=}$
extends SETCORRECTANSWER

any

u
q
$quest$
$a1$
$a2$

where

grd51;: $u \in loggedIn$
The user u must be loggged in
grd52;: $q \in quizzes$
grd53;: $quest \in questions$
grd54;: $a1 \in answers$
grd55;: $a2 \in answers$
grd56;: $a1 \neq a2$
grd57: $quizCreator(q) = u$
The user u must be the creator of the quiz
grd58: $belongingQuiz(quest) = q$
grd59: $correctAnswer(quest) = a1$
a1 must be the current correct answer of the question

grd60: $belongingQuestion(a2) = quest$
a2 must belong to the question
grd61;: $u \notin host[quizInstances \setminus finishedQuiz]$
The host of an active quiz instance shall not be allowed to set the correct answer of a question of a quiz at the same time.
grd62: $\forall qi \cdot qi \in quizInstances \land qi \in instanceOfQuiz^{-1}[\{q\}] \Rightarrow qi \notin (quizInstances \setminus finishedQuiz)$

The creator of a quiz shall not be allowed to set the correct answer of a question of the quiz if there exists an active quiz instance, linked to the quiz, being hosted at the same time.

then

act51;: $correctAnswer := correctAnswer \lhd\!\!+ \{quest \mapsto a2\}$

end

Event UPDATE_ANSWER_OK ⟨ordinary⟩ $\widehat{=}$

extends UPDATE_ANSWER_OK

any

a
q
$quest$
$result$
u

where

grd51;: $u \in loggedIn$
grd52;: $a \in answers$
grd53;: $q \in quizzes$
grd54;: $quest \in questions$
grd55;: $belongingQuestion(a) = quest$
grd56: $quizCreator(q) = u$
grd57: $belongingQuiz(quest) = q$
grd58: $result = TRUE$
The update has been successful
grd61;: $u \notin host[quizInstances \setminus finishedQuiz]$
The host of an active quiz instance shall not be allowed to update an answer of a question of a quiz at the same time.
grd62: $\forall qi \cdot qi \in quizInstances \land qi \in instanceOfQuiz^{-1}[\{q\}] \Rightarrow qi \notin (quizInstances \setminus finishedQuiz)$

The creator of a quiz shall not be allowed to update an answer of a question of the quiz if there exists an active quiz instance, linked to the quiz, being hosted at the same time.

then

$skip$

end

Event UPDATE_ANSWER_NOT_OK ⟨ordinary⟩ $\widehat{=}$

extends UPDATE_ANSWER_NOT_OK

any

a
q
$quest$
$result$
u

where

grd51;: $u \in loggedIn$
the user u must be logged in
grd52;: $a \in answers$
grd53;: $quest \in questions$
grd54;: $q \in quizzes$
grd55;: $belongingQuestion(a) = quest$
grd56;: $quizCreator(q) = u$
u must be the creator of the quiz

grd58;: $belongingQuiz(quest) = q$

grd57;: $result = FALSE$

The update has not been successful. The previous text is restored

grd61: $u \notin host[quizInstances \setminus finishedQuiz]$

The host of an active quiz instance shall not be allowed to update an answer of a question of a quiz at the same time.

grd62: $\forall qi \cdot qi \in quizInstances \land qi \in instanceOfQuiz^{-1}[\{q\}] \Rightarrow qi \notin (quizInstances \setminus finishedQuiz)$

The creator of a quiz shall not be allowed to update an answer of a question of the quiz if there exists an active quiz instance, linked to the quiz, being hosted at the same time.

then

skip

end

Event REMOVEQUESTION ⟨ordinary⟩ $\widehat{=}$

extends REMOVEQUESTION

any

u

$quest$

q

where

grd41;: $u \in loggedIn$

The user must be logged in order to remove a question

grd42;: $q \in quizzes$

grd43;: $quest \in questions$

grd45;: $belongingQuiz(quest) = q$

q must be the belonging quiz of quest

grd44;: $quizCreator(q) = u$

The creator of the quiz q must be u

grd61: $u \notin host[quizInstances \setminus finishedQuiz]$

The host of an active quiz instance shall not be allowed to remove a question from a quiz at the same time.

grd62: $\forall qi \cdot qi \in quizInstances \land qi \in instanceOfQuiz^{-1}[\{q\}] \Rightarrow qi \notin (quizInstances \setminus finishedQuiz)$

The creator of a quiz shall not be allowed to remove a question from the quiz if there exists an active quiz instance, linked to the quiz, being hosted at the same time.

then

act41;: $questions := questions \setminus \{quest\}$

act42;: $belongingQuiz := \{quest\} \lhd belongingQuiz$

act43;: $questionPosition := \{quest\} \lhd questionPosition$

The position of the question is removed

act51;: $answers := answers \setminus \{a | a \in answers \land belongingQuestion(a) = quest\}$

the answers of the question must be removed when the question is removed

act52;: $correctAnswer := \{quest\} \lhd correctAnswer$

The correct answer of quest is not the correct answer anymore

act53;: $belongingQuestion := belongingQuestion \rhd \{quest\}$

All answers of the question do not belong to it anymore

act101: $questionTime := \{quest\} \lhd questionTime$

When removing a question it is necessary to remove its time

end

Event UPDATE_QUESTION_OK ⟨ordinary⟩ $\widehat{=}$

extends UPDATE_QUESTION_OK

any

u

$quest$

q

$result$

where

grd41;: $u \in loggedIn$

grd42: $q \in quizzes$
grd43: $quizCreator(q) = u$
Only the quiz creator of q can update the question
grd44: $quest \in questions$
grd45: $result = TRUE$
The update has been succesfully
grd46: $belongingQuiz(quest) = q$
the question must belong to q
grd61: $u \notin host[quizInstances \setminus finishedQuiz]$
The host of an active quiz instance shall not be allowed to update a question of a quiz at the same time.
grd62: $\forall qi \cdot qi \in quizInstances \land qi \in instanceOfQuiz^{-1}[\{q\}] \Rightarrow qi \notin (quizInstances \setminus finishedQuiz)$

The creator of a quiz shall not be allowed to update a question of the quiz if there exists an active quiz instance, linked to the quiz, being hosted at the same time.

then

skip

end

Event UPDATE_QUESTION_NOT_OK ⟨ordinary⟩ $\widehat{=}$

extends UPDATE_QUESTION_NOT_OK

any

u
$quest$
q
$result$

where

grd41: $u \in loggedIn$
grd42: $q \in quizzes$
grd43: $quizCreator(q) = u$
grd1: $quest \in questions$
grd45: $belongingQuiz(quest) = q$
grd46: $result = FALSE$
The update has not been successfully. The previous text of the question is restored
grd61: $u \notin host[quizInstances \setminus finishedQuiz]$
The host of an active quiz instance shall not be allowed to update a question of a quiz at the same time.
grd62: $\forall qi \cdot qi \in quizInstances \land qi \in instanceOfQuiz^{-1}[\{q\}] \Rightarrow qi \notin (quizInstances \setminus finishedQuiz)$

The creator of a quiz shall not be allowed to update a question of the quiz if there exists an active quiz instance, linked to the quiz, being hosted at the same time.

then

skip

end

Event HOST_QUIZ_INSTANCE ⟨ordinary⟩ $\widehat{=}$

extends HOST_QUIZ_INSTANCE

any

u
q
qi
r

where

grd61: $u \in loggedIn$
The user u must be logged in
grd62: $qi \in INSTANCE\_QUIZ \setminus quizInstances$
grd63: $q \in quizzes$
grd65: $u \notin ran((finishedQuiz \lhd host))$
The user u must not be hosting any other active quiz instance
grd66: $u \in (\{quizCreator(q)\} \cup sharedTo[\{q\}])$
The user u must be the creator or have access to the quiz

grd67: $belongingQuiz^{-1}[\{q\}] \neq \varnothing$
The quiz must have at least 1 question
grd91: $r \in REPORT \setminus reports$
A new report is created. Initially the report is an empty file. When the quiz instance terminates the report
will be populated with the information resulting from the content of the question summaries

then

act61;: $quizInstances := quizInstances \cup \{qi\}$
act62;: $host(qi) := u$
The host of the quiz instance is u
act63;: $instanceOfQuiz(qi) := q$
The quiz instance is linked to the quiz q
act64;: $quizCreated := quizCreated \cup \{qi\}$
The quiz instance is in the quizCreated state
act91: $embeddedQuestions := embeddedQuestions \cup (\{qi\} \times \{quests | quests \in questions \wedge belongingQuiz(quests) = q\})$
The quiz instance is linked to the questions of the quiz
act92: $reports := reports \cup \{r\}$
The report is added to the set of reports
act93: $reportedBy(qi) := r$
The quiz instance is linked to the report

end

Event BEGIN_QUIZ_INSTANCE ⟨ordinary⟩ $\widehat{=}$
extends BEGIN_QUIZ_INSTANCE

any

u
qi

where

grd61;: $u \in loggedIn$
The user u must be logged in
grd62;: $qi \in quizCreated$
The quiz instance must be in the quizCreated state
grd63;: $host(qi) = u$
The host of the quiz instance must be u

then

act61;: $quizCreated := quizCreated \setminus \{qi\}$
The quiz instance is not in the quizCreated state anymore
act62;: $quizInit := quizInit \cup \{qi\}$
The quiz instance becomes part of the quizInit state set
act63;: $answeredQuestions(qi) := 0$
No question has been answered so far

end

Event START_QUIZ_INSTANCE ⟨ordinary⟩ $\widehat{=}$
extends START_QUIZ_INSTANCE

any

u
qi
$quest$

where

grd61;: $u \in loggedIn$
The user u must be logged in
grd62;: $qi \in quizInit$
The quiz instance must be in the quizInit state
grd63;: $host(qi) = u$
The host of the quiz instance must be u
grd64;: $\forall k \cdot belongingQuiz(k) = instanceOfQuiz(qi) \Rightarrow questionPosition(k) \geq questionPosition(quest)$

The question of the quiz that will become the first question to be answered in the quiz instance must be the first one in the question list of the quiz

grd65: $qi \in ran(playingIn)$
Some players must have joined the quiz during the quizInit state

then

act61;: $quizInit := quizInit \setminus \{qi\}$
The quiz is not in the quizInit state anymore
act62;: $questioning := questioning \cup \{qi\}$
The quiz instance is in the questioning state
act63;: $currentQuestion(qi) := quest$
The current question of the quiz instance is the first question of the question list of the quiz
act101: $questionStartedAt(qi) := time$
act102: $questionEndsAt(qi) := time + questionTime(quest)$

end

Event JOIN_QUIZ_INSTANCE ⟨ordinary⟩ $\widehat{=}$
extends JOIN_QUIZ_INSTANCE

any

u
qi

where

grd61;: $u \in USER$
The player can be either regitered or unregistered
grd62;: $qi \in quizInit$
It is possible to join a quiz instanec only if it is in the quizInit state
grd63;: $u \neq host(qi)$
The player must not be the host of the quiz instance
grd64;: $u \notin dom(playingIn)$
The player must not be playing in another quiz instance
grd1: $u \notin host[quizInstances \setminus finishedQuiz]$
The player cannot be any registered user that is hosting any active quiz instance

then

act64;: $playingIn := playingIn \cup \{u \mapsto qi\}$

end

Event ANSWER_QUESTION ⟨ordinary⟩ $\widehat{=}$
extends ANSWER_QUESTION

any

p
qi
$quest$
a

where

grd61;: $p \in USER$
grd62;: $qi \in quizInstances$
grd63;: $p \in playingIn^{-1}[\{qi\}]$
The player must be playing in the quiz instance
grd64;: $qi \in questioning$
The quiz instance must be in the questioning state
grd65: $quest \in questions$
grd66: $currentQuestion(qi) = quest$
The question the player is answering must be the current question
grd67: $a \in answers$
grd68: $a \in belongingQuestion^{-1}[\{quest\}]$
The answer being selected must belong to the current question of the quiz instance
grd69: $card(peopleAnswers^{-1}[\{qi\}]) < card(playingIn^{-1}[\{qi\}])$
A player of a quiz instance shall not be allowed to select an answer when all other players have answered because the quiz instance passes to the questionSummary state.
grd101: $time < questionEndsAt(qi)$
A player shall be allowed to answer a question only within the time of the question.

then

act61;: $peopleAnswers := peopleAnswers \cup \{p \mapsto qi\}$
act62: $peopleSelectedAnswer := peopleSelectedAnswer \lhd\!\!+ \{p \mapsto a\}$

end

Event END_QUESTION_ALL_ANSWERED ⟨ordinary⟩ $\widehat{=}$

extends END_QUESTION_ALL_ANSWERED

any

qi

qs

r

where

grd61;: $qi \in questioning$

The quiz instance must be in the questionig state

grd62;: $card(peopleAnswers^{-1}[\{qi\}]) = card(playingIn^{-1}[\{qi\}])$

All players must have answered

grd91: $qs \in QUESTION\_SUMMARY \setminus qSummary$

A question summary is generated. The question summary is populated with information about the performance of the players by using the information resulting from the relation peopleSelectedAnswer

grd92: $r \in reports$

grd93: $reportedBy(qi) = r$

r must be the report of the quiz instance

grd101: $time < questionEndsAt(qi)$

this event can be used only if all players have answered the question within its time. Otherwise, the even END_QUESTION_TIME_IS_UP must be used

then

act61;: $questioning := questioning \setminus \{qi\}$

The quiz instance is not part of the questioning state anymore

act62;: $questionSummary := questionSummary \cup \{qi\}$

The quiz instance passes to the questionSummary state

act63;: $answeredQuestions(qi) := answeredQuestions(qi) + 1$

The number of answered questions is incremented by 1

act64;: $peopleAnswers := peopleAnswers \rhd \{qi\}$

The players that have answered the question of the quiz instance are removed

act65: $peopleSelectedAnswer := playingIn^{-1}[\{qi\}] \lhd peopleSelectedAnswer$

The answers select by the players of the quiz instance are removed

act91: $qSummary := qSummary \cup \{qs\}$

act92: $summaryFor(qs) := (qi \mapsto currentQuestion(qi))$

The summary is linked to the quiz instance and current question of the quiz instance

act93: $qReportedBy(qs) := r$

The question summary is linked to the report of the quiz instance

act101: $questionStartedAt := \{qi\} \lhd questionStartedAt$

act102: $questionEndsAt := \{qi\} \lhd questionEndsAt$

end

Event END_QUESTION_PREMATURATELY ⟨ordinary⟩ $\widehat{=}$

extends END_QUESTION_PREMATURATELY

any

u

qi

qs

r

where

grd61;: $qi \in questioning$

The quiz instance must be in the question state

grd62;: $u \in loggedIn$

The user u must be logged in

grd63;: $host(qi) = u$

The host of the quiz instance must be u

grd1: $card(peopleAnswers^{-1}[\{qi\}]) \neq card(playingIn^{-1}[\{qi\}])$

The host of a quiz instance in the questioning state shall not be allowed to end prematurately a question if all players have answered because the quiz instance passes to the questionSummary state.

grd91: $qs \in QUESTION\_SUMMARY \setminus qSummary$
grd92: $r \in reports$
grd93: $reportedBy(qi) = r$
grd101: $time < questionEndsAt(qi)$
The host of a quiz instance in the questioning state shall be allowed to end a question prematurely only if the question time is not up.

then

act61;: $questioning := questioning \setminus \{qi\}$
The quiz instance is not in the questioning state anymore
act62;: $questionSummary := questionSummary \cup \{qi\}$
The quiz instance passes to the questionSummary state
act63;: $answeredQuestions(qi) := answeredQuestions(qi) + 1$
The number of answered questions of the quiz instance is incremebted by 1
act64;: $peopleAnswers := peopleAnswers \rhd \{qi\}$
The players that have answered the question of the quiz instance are removed
act65: $peopleSelectedAnswer := playingIn^{-1}[\{qi\}] \lhd peopleSelectedAnswer$
The answers select by the players of the quiz instance are removed
act91: $qSummary := qSummary \cup \{qs\}$
act92: $summaryFor(qs) := qi \mapsto currentQuestion(qi)$
act93: $qReportedBy(qs) := r$
act101: $questionStartedAt := \{qi\} \lhd questionStartedAt$
act102: $questionEndsAt := \{qi\} \lhd questionEndsAt$

end

Event SHOW_QUESTION_SUMMARY_TO_PLAYERS ⟨ordinary⟩ $\widehat{=}$
extends SHOW_QUESTION_SUMMARY_TO_PLAYERS

any

u
qi
qs

where

grd61;: $u \in loggedIn$
The user u must be logged in
grd62;: $qi \in quizInstances$
grd63;: $host(qi) = u$
The user u must be the host of the quiz instance
grd63: $qi \in questionSummary$
The quiz instance must be in the questionSummary state
grd64: $qi \in ran(playingIn)$
The host of a quiz instance in the questionSummary state shall
be allowed to show the question summary only if at least 1 player is playing in the quiz instance.
grd91: $summaryFor(qs) = qi \mapsto currentQuestion(qi)$
The summary to show must be the
summary linked to the quiz instance and current question
of the quiz instance

then

$skip$

end

Event NEXT_QUESTION ⟨ordinary⟩ $\widehat{=}$
extends NEXT_QUESTION

any

u
qi
$quest$

where

grd61;: $u \in loggedIn$
grd62;: $qi \in questionSummary$
The quiz instance must be in the questionSummary state
grd63;: $host(qi) = u$

grd64;: $answeredQuestions(qi) < card(belongingQuiz^{-1}[\{instanceOfQuiz(qi)\}])$
There must be other questions to be answered
grd65: $qi \in ran(playingIn)$
At least 1 player must be playing in the quiz instance
grd66: $quest \in questions$
grd67: $card(\{quests | quests \in questions \land quests \in belongingQuiz^{-1}[\{instanceOfQuiz(qi)\}] \land questionPosition(quests) < questionPosition(quest)\}) = answeredQuestions(qi)$
The next question to be answered must be the successive one

then

act61;: $questionSummary := questionSummary \setminus \{qi\}$
The quiz instance is not in the questionSummary state anymore
act62;: $questioning := questioning \cup \{qi\}$
The quiz instance passes to the questioning state
act63: $currentQuestion(qi) := quest$
The current qyestion is set to the successive question
act101: $questionStartedAt(qi) := time$
act102: $questionEndsAt(qi) := time + questionTime(quest)$

end

Event FINISH_QUIZ_INSTANCE ⟨ordinary⟩ $\widehat{=}$
extends FINISH_QUIZ_INSTANCE

any

u
qi

where

grd61;: $u \in loggedIn$
The user must be logged in
grd62;: $qi \in questionSummary$
The quiz instance must be in the questionSummary state
grd63;: $host(qi) = u$
The user u must be the host of the quiz instance

then

act61;: $questionSummary := questionSummary \setminus \{qi\}$
The quiz instance is not in the questionSummary state anymore
act62;: $quizSummary := quizSummary \cup \{qi\}$
The quiz instance passes to the quizSummary state
act63;: $currentQuestion := \{qi\} \lhd currentQuestion$
The current question of the quiz instance is removed as it is not anymore in the questionSummary state

end

Event SHOW_QUIZ_INSTANCE_SUMMARY_TO_PLAYERS ⟨ordinary⟩ $\widehat{=}$
extends SHOW_QUIZ_INSTANCE_SUMMARY_TO_PLAYERS

any

qi
u
r

where

grd61;: $qi \in quizInstances$
grd62: $qi \in quizSummary$
The quiz instance must be in the quizSummary state
grd63: $u \in registeredUser$
grd64: $u \in loggedIn$
The user u must be logged in
grd65: $host(qi) = u$
The user u must be the host of the quiz instance
grd66: $qi \in ran(playingIn)$
there must at least 1 player playing in the quiz instance
grd91: $r \in reports$
grd92: $reportedBy(qi) = r$
the report to show must be the report of the quiz instance

then
skip
end

Event TERMINATE_QUIZ_INSTANCE_QUIZ_SUMMARY_STATE ⟨ordinary⟩ ≙
extends TERMINATE_QUIZ_INSTANCE_QUIZ_SUMMARY_STATE
any
u
qi
where
grd61;: $u \in loggedIn$
The user u must be loggedin
grd62;: $qi \in quizSummary$
The quiz instance must be in the quizSummary state
grd63;: $host(qi) = u$
The user u must be the host of the quiz instance
then
act61;: $quizSummary := quizSummary \setminus \{qi\}$
The quiz instance is not in the quizSummary state anymore
act62;: $finishedQuiz := finishedQuiz \cup \{qi\}$
The quiz instance passes to the finishedQuiz state
act63: $playingIn := playingIn \rhd \{qi\}$
The players of the quiz instance do not play anymore in the quiz instance
act91: $embeddedQuestions := \{qi\} \lhd embeddedQuestions$
The quiz instance is not linked anymore to the questions of the quiz instance
act92: $summaryFor := summaryFor \rhd (\{qi\} \lhd embeddedQuestions)$
The question summaries do not report anymore a specific question of the quiz instance
act93: $qSummary := qSummary \setminus summaryFor^{-1}[\{qi\} \lhd embeddedQuestions]$
All the summaries of the quiz instance built so far must be deleted
act94: $qReportedBy := qReportedBy \rhd reportedBy[\{qi\}]$
The question summaries are not linked anymore to the report of the quiz instance
end

Event LEAVE_QUIZ_INSTANCE_LAST_ONE_IN_QUESTIONING_STATE ⟨ordinary⟩ ≙
extends LEAVE_QUIZ_INSTANCE_LAST_ONE_IN_QUESTIONING_STATE
any
p
qi
qs
r
where
grd61: $p \in USER$
grd62: $qi \in questioning$
The quiz instance must be in the questioning state
grd63: $p \in playingIn^{-1}[\{qi\}]$
The player must be playing in the quiz instance
grd64: $card(playingIn^{-1}[\{qi\}]) = 1$
The player must be the only player be playing in the quiz instance
grd91: $qs \in QUESTION\_SUMMARY \setminus qSummary$
A question summary is generated
grd92: $r \in reports$
grd93: $reportedBy(qi) = r$
Thw question summary is linked to the report of the quiz instance
grd101: $time < questionEndsAt(qi)$
A player can leave during the questioning state only if the question time is not up
grd102: $card(peopleAnswers^{-1}[\{qi\}]) < card(playingIn^{-1}[\{qi\}])$
A player can only leave if not all other players have already answered
then
act61: $playingIn := \{p\} \lhd playingIn$
The player does not play anymore in the quiz instance

act62: $peopleAnswers := \{p\} \lhd peopleAnswers$
It is removed whether the player has answered the question or not
act63: $questioning := questioning \setminus \{qi\}$
act64: $questionSummary := questionSummary \cup \{qi\}$
The quiz instance passes to the questionSummary state
act65: $peopleSelectedAnswer := \{p\} \lhd peopleSelectedAnswer$
The answer selected by the player is removed (if he/she has naswered)
act91: $qSummary := qSummary \cup \{qs\}$
act92: $summaryFor(qs) := qi \mapsto currentQuestion(qi)$
act93: $qReportedBy(qs) := r$
act101: $questionStartedAt := \{qi\} \lhd questionStartedAt$
act102: $questionEndsAt := \{qi\} \lhd questionEndsAt$

end

Event LEAVE_QUIZ_INSTANCE_NOT_IN_QUESTIONING_STATE ⟨ordinary⟩ $\widehat{=}$
extends LEAVE_QUIZ_INSTANCE_NOT_IN_QUESTIONING_STATE

any
p
qi
where
grd61: $p \in USER$
grd62: $qi \in quizInstances \setminus (quizCreated \cup finishedQuiz \cup questioning)$
The quiz instance must not be in the quiz created state, finishedQuiz or questioning state
grd63: $p \in playingIn^{-1}[\{qi\}]$
The player must be playing in the quiz instance
then
act61: $playingIn := \{p\} \lhd playingIn$
The player does not play anymore in the quiz instance
end

Event LEAVE_QUIZ_INSTANCE__NOT_LAST_ONE_IN_QUESTIONING_STATE ⟨ordinary⟩ $\widehat{=}$
extends LEAVE_QUIZ_INSTANCE__NOT_LAST_ONE_IN_QUESTIONING_STATE

any
p
qi
where
grd61: $p \in USER$
grd62: $qi \in questioning$
grd63: $p \in playingIn^{-1}[\{qi\}]$
grd64: $card(playingIn^{-1}[\{qi\}]) > 1$
There must be at least 2 players be playing
grd101: $time < questionEndsAt(qi)$
A player can leave during the questioning state only if the question time is not up
grd102: $card(peopleAnswers^{-1}[\{qi\}]) < card(playingIn^{-1}[\{qi\}])$
A player can only leave if not all other players have already answered
then
act61: $playingIn := \{p\} \lhd playingIn$
act62: $peopleAnswers := \{p\} \lhd peopleAnswers$
act63: $peopleSelectedAnswer := \{p\} \lhd peopleSelectedAnswer$
end

Event UNSHARE_QUIZ ⟨ordinary⟩ $\widehat{=}$
extends UNSHARE_QUIZ

any
$u1$
$u2$
q
where
grd41: $u1 \in registeredUser$
grd42: $u2 \in registeredUser$
grd43: $u1 \in loggedIn$
A registerd user can unshare only if he/she is online

grd44: $u1 \neq u2$
grd45: $q \mapsto u2 \in sharedTo$
u2 had to have access to the quiz
grd46: $u1 = quizCreator(q)$
Only the creator of the quiz can unshare a quiz
grd81: $u2 \notin host[instanceOfQuiz^{-1}[\{q\}] \setminus finishedQuiz]$
The creator of a quiz shall not be allowed to deshare a quiz
with another registered user if the latter is hosting a quiz instance linked to the quiz.
grd82: $u1 \notin host[quizInstances \setminus finishedQuiz]$
The creator of a quiz shall not be allowed to deshare a
quiz if he/she is hosting an active quiz instance.

then

act41: $sharedTo := sharedTo \setminus \{q \mapsto u2\}$

end

Event REMOVE_QUIZ_INSTANCE ⟨ordinary⟩ $\widehat{=}$
extends REMOVE_QUIZ_INSTANCE

any

u
qi

where

grd81: $u \in registeredUser$
grd82: $qi \in quizInstances$
grd83: $qi \in finishedQuiz$
It is possible to remove a quiz instance only when it is in the finishedQuiz state
grd84: $u \in loggedIn$
The user must be logged in
grd85: $host(qi) = u$
The user u must be the host of the quiz instance
grd86: $u \notin host[quizInstances \setminus finishedQuiz]$
The user must not be hosting any active quiz instance

then

act81: $quizInstances := quizInstances \setminus \{qi\}$
The quiz instance is removed
act82: $finishedQuiz := finishedQuiz \setminus \{qi\}$
The quiz instance is removed from the finishedQuiz state
act83: $host := \{qi\} \lhd host$
The user is not the host of the quiz instance anymore
act84: $instanceOfQuiz := \{qi\} \lhd instanceOfQuiz$
The quiz instance is not linked anymore to a quiz
act85: $answeredQuestions := \{qi\} \lhd answeredQuestions$
The number of answered questions of the quiz instance is removed
act91: $reportedBy := \{qi\} \lhd reportedBy$
The report of the quiz instance is removed
act92: $reports := reports \setminus \{reportedBy(qi)\}$
The report is not macthed anymore to the quiz instance

end

Event TERMINATE_QUIZ_INSTANCE_QUIZ_CREATED_STATE ⟨ordinary⟩ $\widehat{=}$
extends TERMINATE_QUIZ_INSTANCE_QUIZ_CREATED_STATE

any

u
qi

where

grd91: $u \in registeredUser$
grd92: $u \in loggedIn$
The user u must be loggedIn
grd93: $qi \in quizInstances$
grd94: $qi \in quizCreated$
The quiz instance must be in the quizCreated state

grd95: $host(qi) = u$
The user u must be the host of the quiz instance

then

act91: $quizCreated := quizCreated \setminus \{qi\}$
The quiz instance is removed from the quizCreated state

act92: $finishedQuiz := finishedQuiz \cup \{qi\}$
The quiz instance passes to the finishedQuiz state

act93: $answeredQuestions(qi) := 0$
The number of answered questions is set to 0

act94: $embeddedQuestions := \{qi\} \lhd embeddedQuestions$
The quiz instance is not linked anymore to the questions of the quiz instance

end

Event TERMINATE_QUIZ_INSTANCE_QUIZ_INIT_STATE ⟨ordinary⟩ $\widehat{=}$

extends TERMINATE_QUIZ_INSTANCE_QUIZ_INIT_STATE

any

u

qi

where

grd81: $u \in registeredUser$

grd82: $u \in loggedIn$
The user u must be loggedin

grd83: $qi \in quizInstances$

grd84: $qi \in quizInit$
The quiz instance must be in the quizInit state

grd85: $host(qi) = u$
The user u must be the host of the quiz instance

then

act81: $quizInit := quizInit \setminus \{qi\}$
The quiz instance is not anymore in the quizInit state

act82: $finishedQuiz := finishedQuiz \cup \{qi\}$
The quiz instance passes to the finishedQuiz state

act83: $playingIn := playingIn \rhd \{qi\}$
The players of the quiz instance do not play anymore in the quiz instance

act91: $embeddedQuestions := \{qi\} \lhd embeddedQuestions$
The quiz instance is not linked anymore to the questions of the quiz instance

end

Event INCREASE_TIME ⟨ordinary⟩ $\widehat{=}$

when

grd101: $\forall qi \cdot qi \in quizInstances \land qi \in questioning \Rightarrow card(peopleAnswers^{-1}[\{qi\}]) < card(playingIn^{-1}[\{qi\}])$

It is possible to increase the time only if in all quiz instances in the questioning state not all players have answered

grd102: $\forall qi \cdot qi \in quizInstances \land qi \in questioning \Rightarrow time < questionEndsAt(qi)$
It is possible to increase the time only if the ending time of each quiz instance in the questioning state is strictly greater than the current time

then

act101: $time := time + 1$
Time is increased by 1

end

Event END_QUESTION_TIME_IS_UP ⟨ordinary⟩ $\widehat{=}$

extends END_QUESTION_TIME_IS_UP

any

qi

qs

r

where

grd61: $qi \in quizInstances$

grd62: $qi \in questioning$
The quiz instance must be in the questioning state

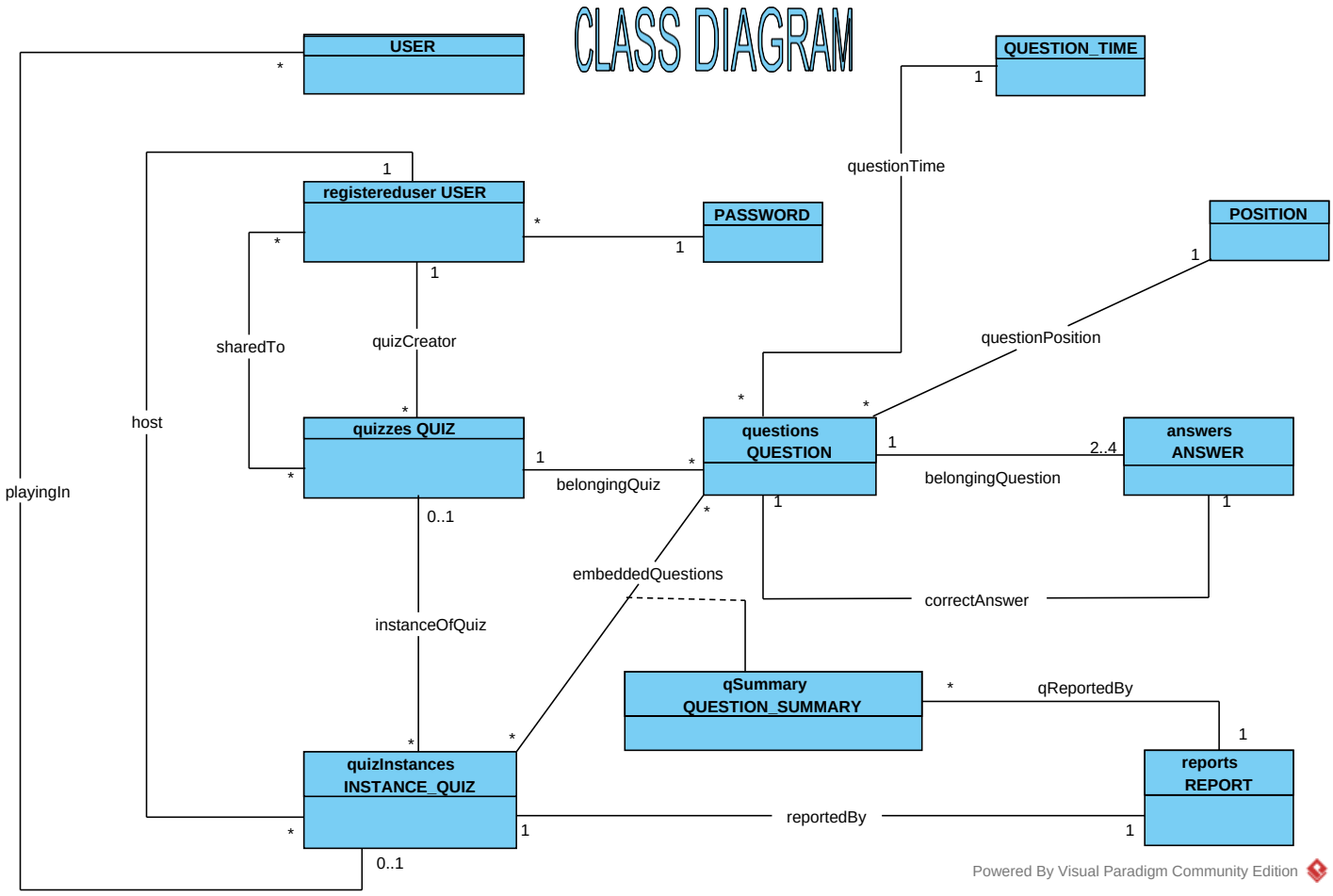
grd91: $qs \in QUESTION\_SUMMARY \setminus qSummary$
grd92: $r \in reports$
grd93: $reportedBy(qi) = r$
grd101: $time = questionEndsAt(qi)$
The time must be equal to the ending time of the quiz instance

then

act61: $questioning := questioning \setminus \{qi\}$
act62: $questionSummary := questionSummary \cup \{qi\}$
The quiz instance passes to the questionSummary state
act63: $answeredQuestions(qi) := answeredQuestions(qi) + 1$
The number of answered questions of the quiz instance is incremented by 1
act64: $peopleAnswers := peopleAnswers \rhd \{qi\}$
act65: $peopleSelectedAnswer := playingIn^{-1}[\{qi\}] \lhd peopleSelectedAnswer$
act91: $qSummary := qSummary \cup \{qs\}$
act92: $summaryFor(qs) := (qi \mapsto currentQuestion(qi))$
act93: $qReportedBy(qs) := r$
act101: $questionStartedAt := \{qi\} \lhd questionStartedAt$
act102: $questionEndsAt := \{qi\} \lhd questionEndsAt$

end

END


CLASS DIAGRAM
USER
QUESTION_TIME
registereduser USER
PASSWORD
POSITION
questionTime
sharedTo
quizCreator
questionPosition
host
quizzes QUIZ
questions
QUESTION
answers
ANSWER
playingIn
belongingQuiz
belongingQuestion
0..1
embeddedQuestions
correctAnswer
instanceOfQuiz
qSummary
QUESTION_SUMMARY
qReportedBy
quizInstances
INSTANCE_QUIZ
reports
REPORT
reportedBy
0..1
Powered By Visual Paradigm Community Edition