Software Design Document

Group #3

Yazied Allahim Abdullah Alnahdi

Abdullah Algarni Mansor Aldandor

Contents

Deployment Architecture

Introduction

We will be illustrating the candidate choices for the technologies to be used for deploying our application, from both perspectives of the server and client sides. First, we talk about the server side by presenting two approaches: Modern approach using Google Mobile App Backend Services, and traditional approach. Second, we talk about the client side. Finally, we summarize our choices of technology in a diagram that illustrates the final deployment architecture of the application.

Server-Side Architecture

Modern Approach: Google Mobile App Backend Services.

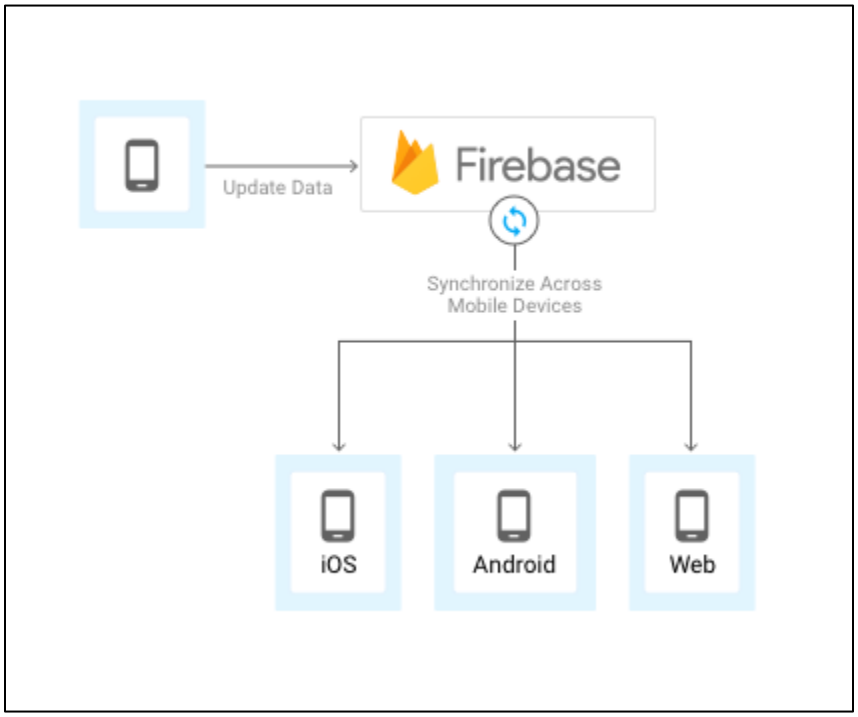
It is a Backend-as-a-Service (BaaS) that provides easy-to-use yet powerful tools for handling the backend work. Features provided include Realtime Database, Storage, Cloud Messaging, Hosting, and many others. All of which run on Google Cloud Platform.

Google backend services come in different design patterns, as follows:

- Firebase
- Firebase & App Engine standard environment
- Firebase & App Engine flexible environment

Details of each pattern will be listed next.

Firebase



Recommended for:

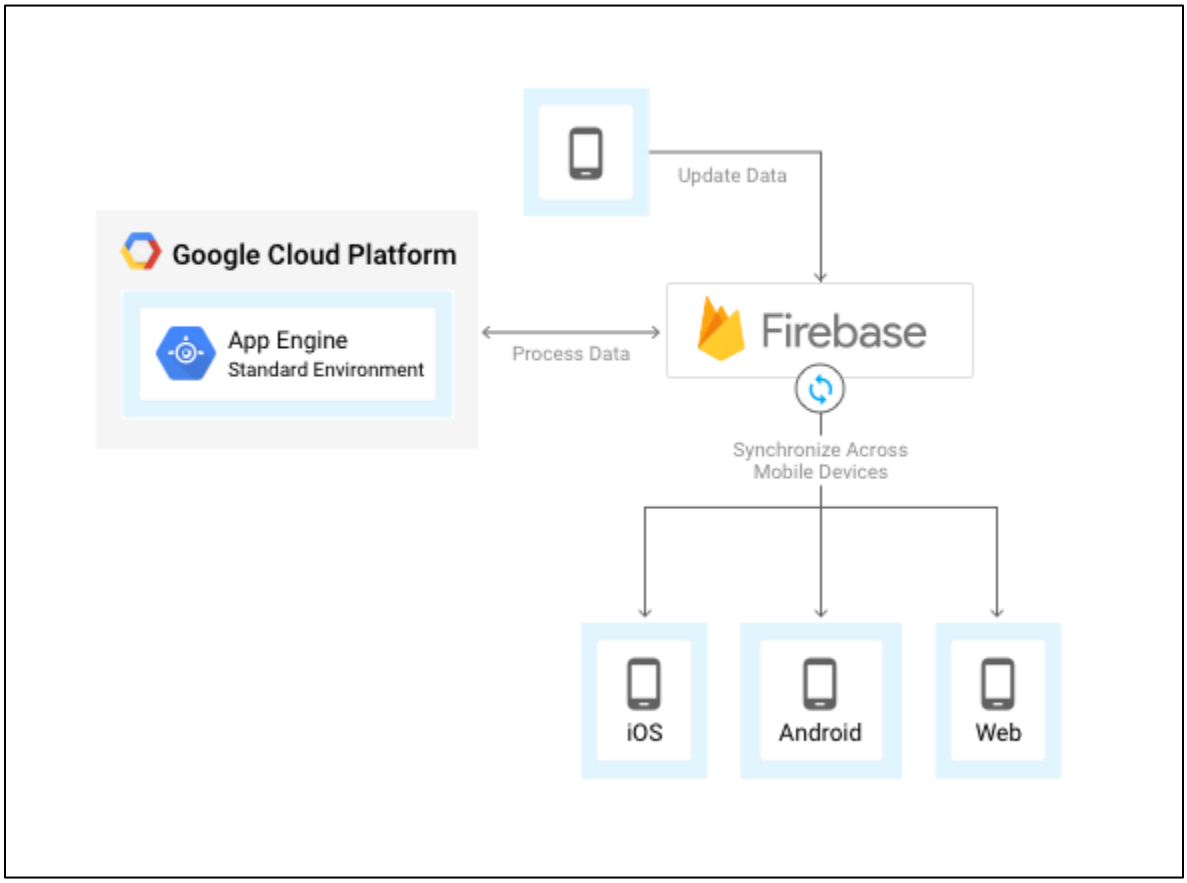
- Limiting on-device data storage by storing JSON data in the Firebase Realtime Database and files in Firebase Storage.
- Sending notifications with Firebase Cloud Messaging.
- Automated real-time data synchronization across multiple devices.
- Gracefully handling the offline case.
- Authenticating users through a variety of identity providers.
- Rapid development of a backend service.

Not recommended for:

- Apps that need a backend service to modify the synchronized data.

Note: We could use Google-Cloud Functions. (JS)

Firebase and Google App Engine standard environment



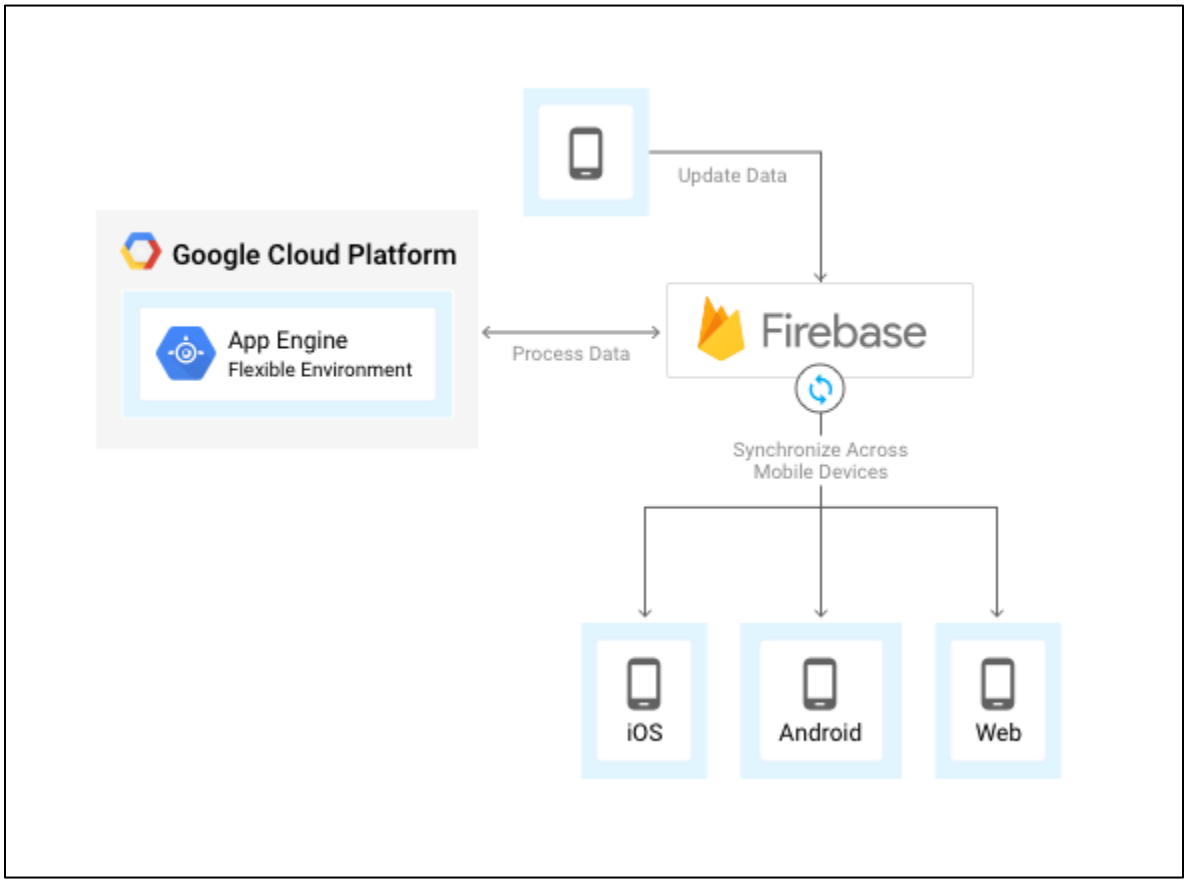
Recommended for:

- Firebase apps that need a backend service to modify the synchronized data.
- Backend services that run periodically to process or analyze Firebase data.

Not recommended for:

- Backend services that call native binaries, write to the file system, or make other system calls.
- Persistent connections to Firebase. App Engine standard environment reclaims socket connections after 2 minutes.

Firebase and App Engine flexible environment



Recommended for:

- Firebase apps that need a backend service to modify the synchronized data, and that service requires a custom server configuration or third-party libraries not supported by App Engine standard environment.
- Backend services that require a persistent connection to Firebase to receive notifications of data changes. Flexible environment can hold a connection open for 24 hours.

Firebase Approaches Summary

Feature	Firebase	Firebase & App Engine standard environment	Firebase & App Engine flexible environment
Automatic capacity scaling	✓	✓	✓
Automatic real-time data synchronization	✓	✓	✓
Automatic server maintenance	✓	✓	✓
Backend logic		✓	✓
Call native binaries, write to the file system, or make other system calls.			✓
Data storage	✓	✓	✓
File storage	✓	✓	✓
Easy user authentication	✓	✓	✓
Language support for backend service logic	N/A	Java, Python, Go, PHP	Any
Messages and notifications, such as push notifications	✓	✓	✓
Platform support	iOS, Android, Web	iOS, Android, Web	iOS, Android, Web
Requires code to run within a sandbox.	N/A	✓	
Requires SSL		✓	

Traditional Approach.

Primarily, four aspects of server-side technology need to be addressed: Server type, serve-side programming language choice, database type, and 3rd party services.

Server Type

We will be using Apache Server for the application deployment. Apache is a free and open-source cross-platform web server

Serve-side Programming Language

We have 5 programming language candidates: PHP, Python, Node.js, Java, Asp.net. We will evaluate each based on the following criteria:

- Support and Community
- Performance
- Elasticity, the ease with which new features can be added
- Time to Production

The following table compares the languages with respect to the above criteria, where 20 denotes the highest, and 1 denotes the lowest.

	Support & Community (35%)	Performance (35%)	Elasticity (20%)	Time to production (10%)	Total Out of 20
PHP	20	12	20	20	17.2
Python	8	4	16	16	9
Node.js	4	16	20	16	12.6
Java	20	16	12	12	16.2
Asp.net	12	20	12	8	14.4

According to the total points accumulated, PHP has the highest.

Database Type

Two database paradigms are available: Relational Database and Non-relational database.

Different criteria are listed below, and the two paradigms are compared against each. Note that 5 denotes the highest, whereas 1 denotes the lowest.

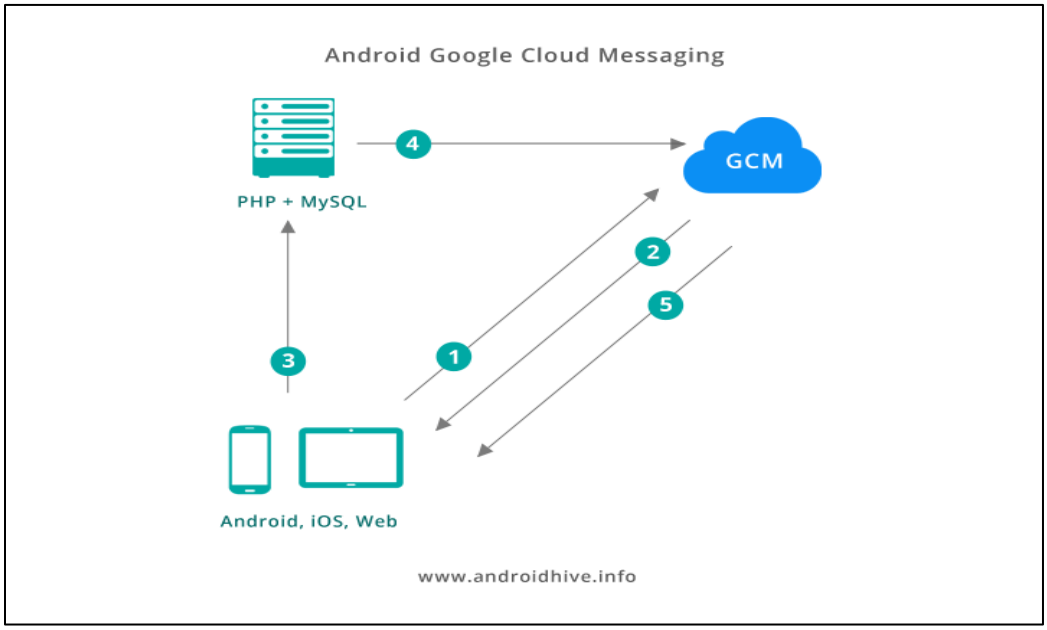
Feature	NoSQL Databases	Relational Databases
Performance	5	2
Reliability	1	5
Availability	4	4
Consistency	1	5
Data Storage	5	3
Scalability	5	3
Total out of 30	21	22

According to the table, Relational Database has the highest total.

3rd Party Services

We will use Google Cloud Messaging (GCM) for push messages.

An example of how such architecture works with GCM is given below:



Client: Android

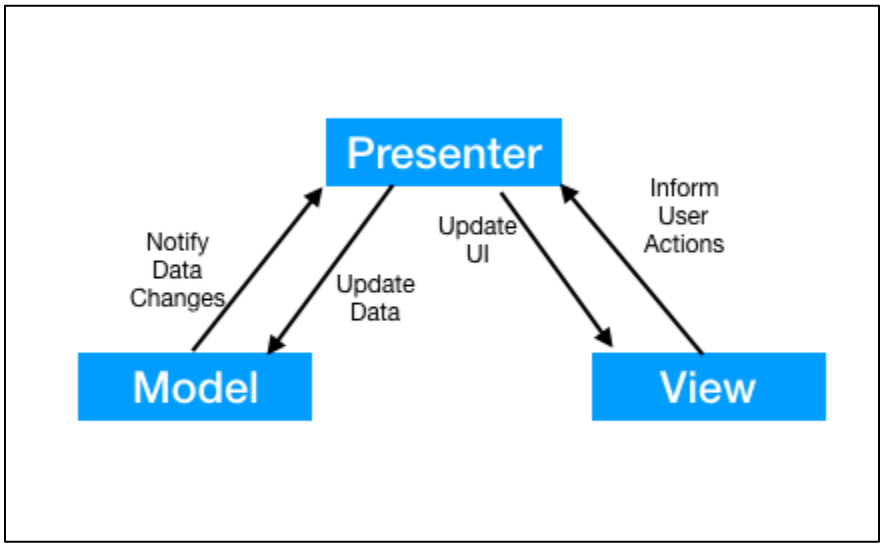
Language:

Java.

Design Strategy:

MPV (Model-Presenter-View).

An illustration of how such strategy works is as follows:



Conclusion

Although the *Backend Modern Approach* offers more manageable and powerful environment for deploying the application, we will select the *Backend Traditional Approach*. As by using it, we will be exposed to the relatively low details of client-server communication, and we will have a direct insight of how such modern approaches are implemented.

Selected Technology.

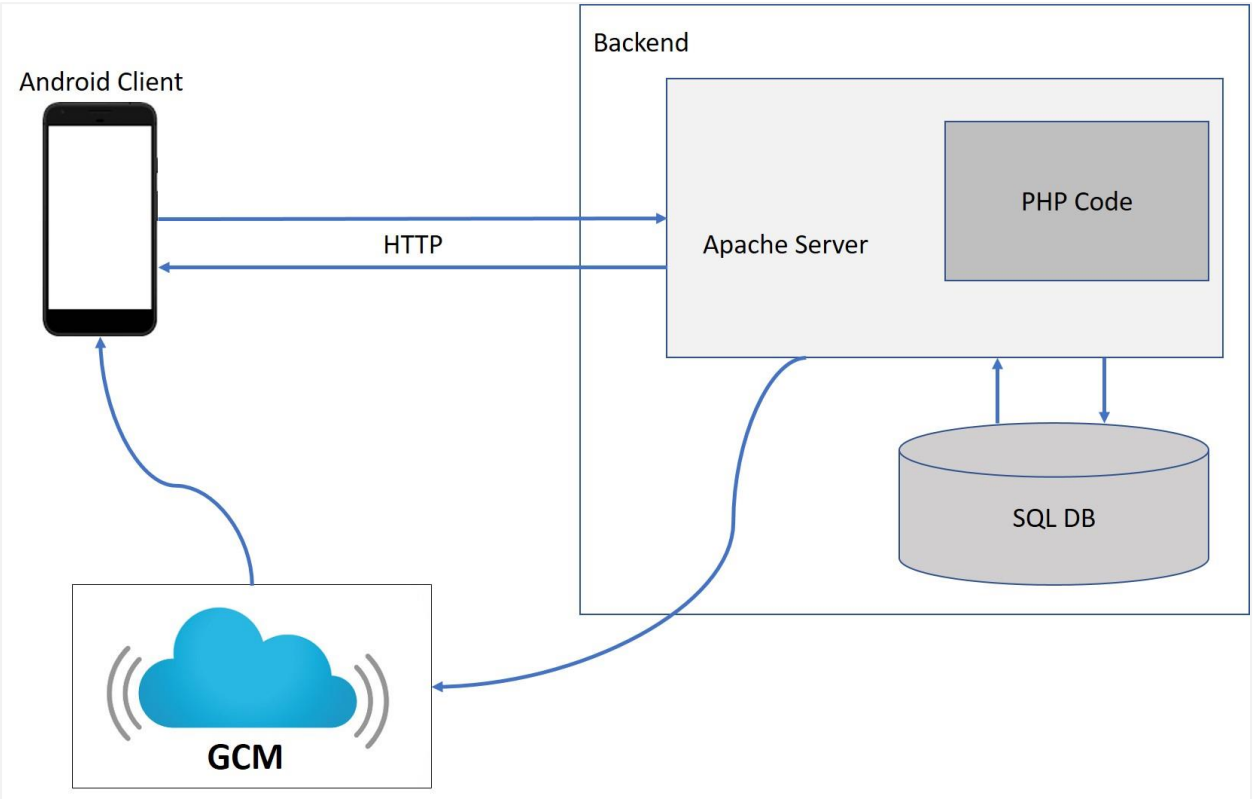
Server: Apache.

Server-side programming language: PHP.

Database type: Relation database (SQL).

Push Messages: Google Cloud Messages (GCM).

Architecture Diagram



References

- Google Mobile Backend Services:
  https://cloud.google.com/solutions/mobile/mobile-app-backend-services

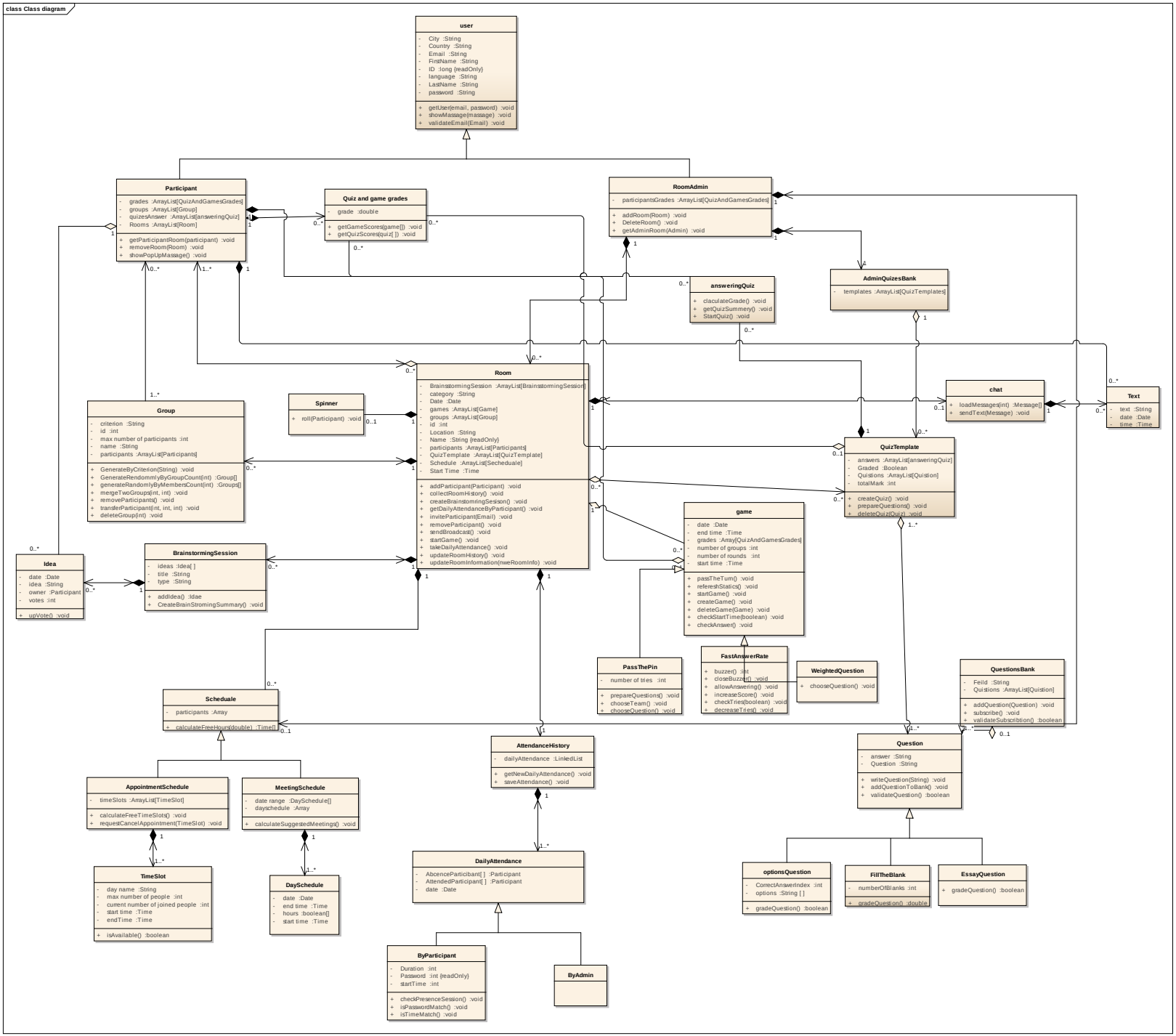
- “Selecting the optimal programming language” By IBM:
  https://www.ibm.com/developerworks/library/wa-optimal/

- Different Programming Languages benchmarks:
  https://github.com/kostya/benchmarks
  https://insights.stackoverflow.com/survey/2016

- “NoSQL Databases: The Definitive Guide”:

  https://blog.pandorafms.org/nosql-databases-the-definitive-guide/

Refined Class Diagram

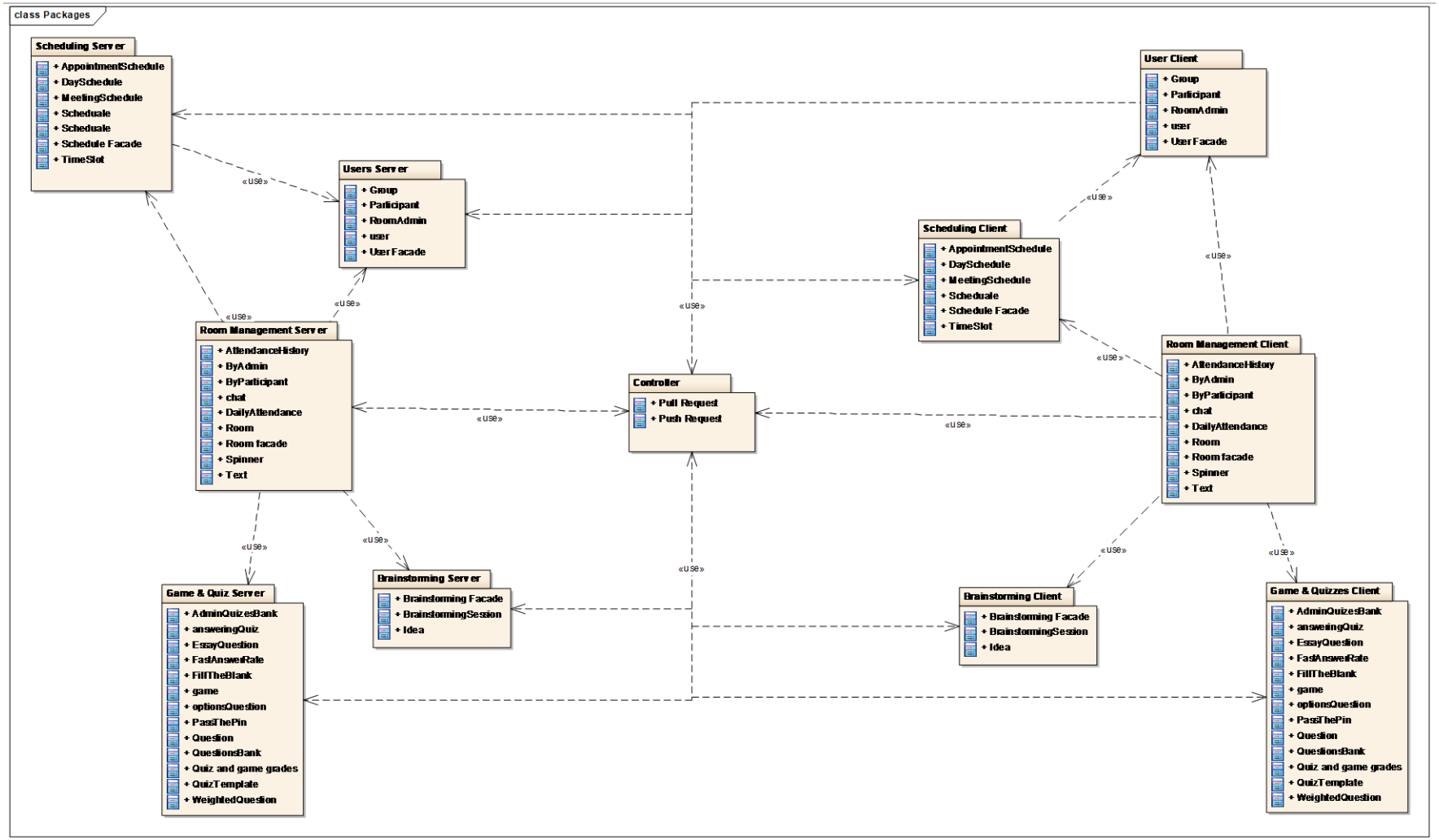

class Class diagram
user
- City :String
- Country :String
- Email :String
- FirstName :String
- ID :long {readOnly}
- language :String
- LastName :String
- password :String
+ getUser(email, password) :void
+ showMassage(massage) :void
+ validateEmail(Email) :void
Participant
- grades :ArrayList[QuizAndGamesGrades]
- groups :ArrayList[Group]
- quizesAnswer :ArrayList[answeringQuiz]
- Rooms :ArrayList[Room]
+ getParticipantRoom(participant) :void
+ removeRoom(Room) :void
+ showPopUpMassage() :void
Quiz and game grades
- grade :double
+ getGameScores(game[]) :void
+ getQuizScores(quiz[ ]) :void
RoomAdmin
- participantsGrades :ArrayList[QuizAndGamesGrades]
+ addRoom(Room) :void
+ DeleteRoom() :void
+ getAdminRoom(Admin) :void
answeringQuiz
+ claculateGrade() :void
+ getQuizSummery() :void
+ StartQuiz() :void
AdminQuizesBank
- templates :ArrayList[QuizTemplates]
Room
- BrainstormingSession :ArrayList[BrainstormingSession]
- category :String
- Date :Date
- games :ArrayList[Game]
- groups :ArrayList[Group]
- id :int
- Location :String
- Name :String {readOnly}
- participants :ArrayList[Participants]
QuizTemplate :ArrayList[QuizTemplate]
- Schedule :ArrayList[Secheduale]
Start Time :Time
+ addParticipant(Participant) :void
+ collectRoomHistory() :void
+ createBrainstormingSession() :void
+ getDailyAttendanceByParticipant() :void
+ inviteParticipant(Email) :void
+ removeParticipant() :void
+ sendBroadcast() :void
+ startGame() :void
+ takeDailyAttendance() :void
+ updateRoomHistory() :void
+ updateRoomInformation(nweRoomInfo) :void
Group
- criterion :String
- id :int
- max number of participants :int
- name :String
- participants :ArrayList[Participants]
+ GenerateByCriterion(String) :void
+ GenerateRendommlyByGroupCount(int) :Group[]
+ generateRandomlyByMembersCount(int) :Groups[]
+ mergeTwoGroups(int, int) :void
+ removeParticipants() :void
+ transferParticipant(int, int, int) :void
+ deleteGroup(int) :void
Spinner
+ roll(Participant) :void
chat
+ loadMessages(int) :Message[]
+ sendText(Message) :void
Text
- text :String
- date :Date
- time :Time
QuizTemplate
- answers :ArrayList[answeringQuiz]
- Graded :Boolean
- Quistions :ArrayList[Quistion]
- totalMark :int
+ createQuiz() :void
+ prepareQuestions() :void
+ deleteQuiz(Quiz) :void
game
- date :Date
- end time :Time
- grades :Array[QuizAndGamesGrades]
- number of groups :int
- number of rounds :int
- start time :Time
+ passTheTurn() :void
+ refereshStatics() :void
+ startGame() :void
+ createGame() :void
+ deleteGame(Game) :void
+ checkStartTime(boolean) :void
+ checkAnswer() :void
Idea
- date :Date
- idea :String
- owner :Participant
- votes :int
+ upVote() :void
BrainstormingSession
- ideas :Idea[ ]
- title :String
- type :String
+ addIdea() :Idae
+ CreateBrainStromingSummary() :void
PassThePin
- number of tries :int
+ prepareQuestions() :void
+ chooseTeam() :void
+ chooseQuestion() :void
FastAnswerRate
+ buzzer() :int
+ closeBuzzer() :void
+ allowAnswering() :void
+ increaseScore() :void
+ checkTries(boolean) :void
+ decreaseTries() :void
WeightedQuestion
+ chooseQuestion() :void
QuestionsBank
- Feild :String
- Quistions :ArrayList[Quistion]
+ addQuestion(Question) :void
+ subscribe() :void
+ validateSubscribtion() :boolean
Scheduale
- participants :Array
+ calculateFreeHours(double) :Time[]
AttendanceHistory
- dailyAttendance :LinkedList
+ getNewDailyAttendance() :void
+ saveAttendance() :void
Question
- answer :String
- Question :String
+ writeQuestion(String) :void
+ addQuestionToBank() :void
+ validateQuestion() :boolean
AppointmentSchedule
- timeSlots :ArrayList[TimeSlot]
+ calculateFreeTimeSlots() :void
+ requestCancelAppointment(TimeSlot) :void
MeetingSchedule
- date range :DaySchedule[]
- dayschedule :Array
+ calculateSuggestedMeetings() :void
DailyAttendance
- AbcenceParticibant[ ] :Participant
- AttendedParticipant[ ] :Participant
- date :Date
optionsQuestion
- CorrectAnswerIndex :int
- options :String [ ]
+ gradeQuestion() :boolean
FillTheBlank
- numberOfBlanks :int
+ gradeQuestion() :double
EssayQuestion
+ gradeQuestion() :boolean
TimeSlot
- day name :String
- max number of people :int
- current number of joined people :int
- start time :Time
- endTime :Time
+ isAvailable() :boolean
DaySchedule
- date :Date
- end time :Time
- hours :boolean[]
- start time :Time
ByParticipant
- Duration :int
- Password :int {readOnly}
- startTime :int
+ checkPresenceSession() :void
+ isPasswordMatch() :void
+ isTimeMatch() :void
ByAdmin

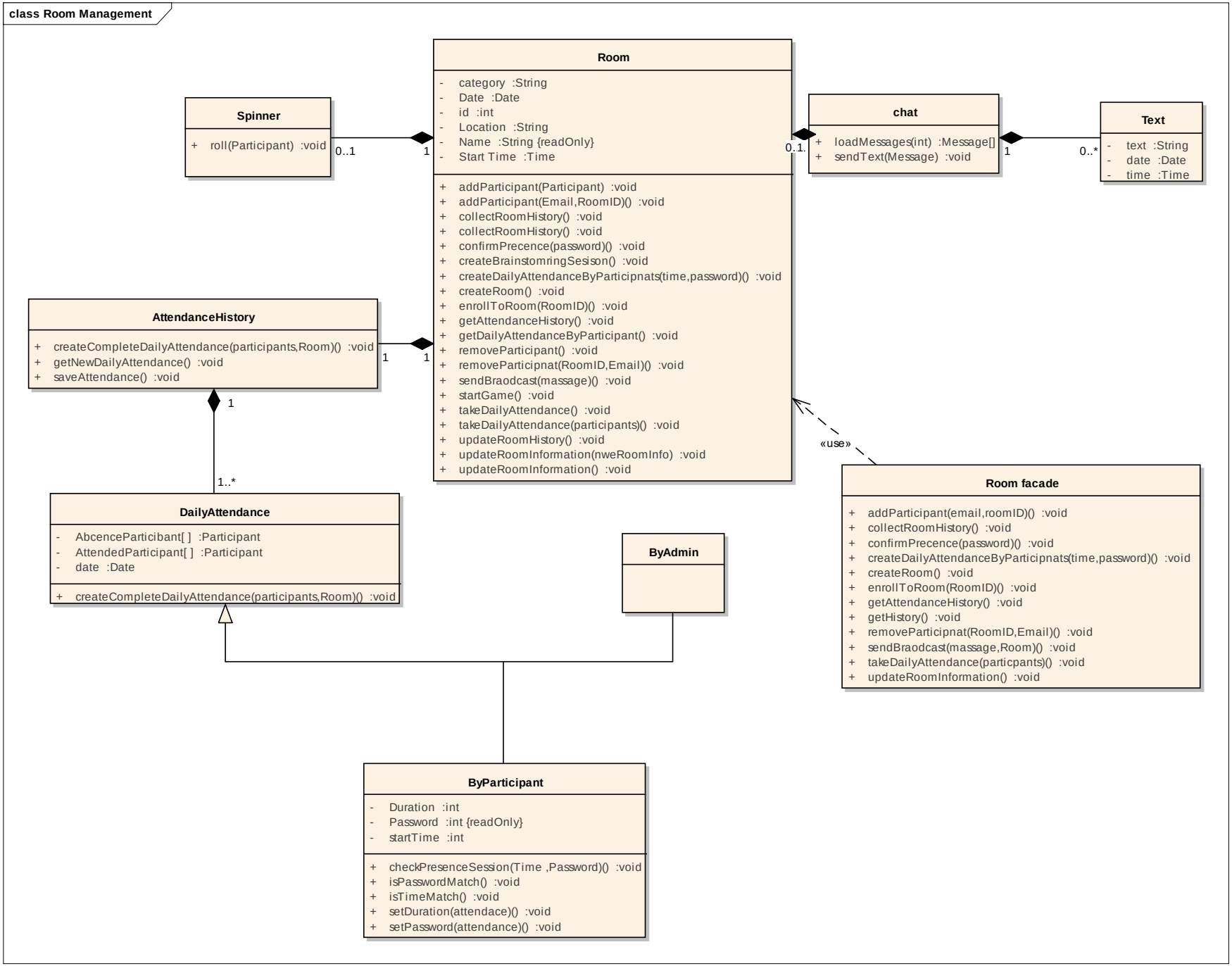
Packaging Through Use Cases

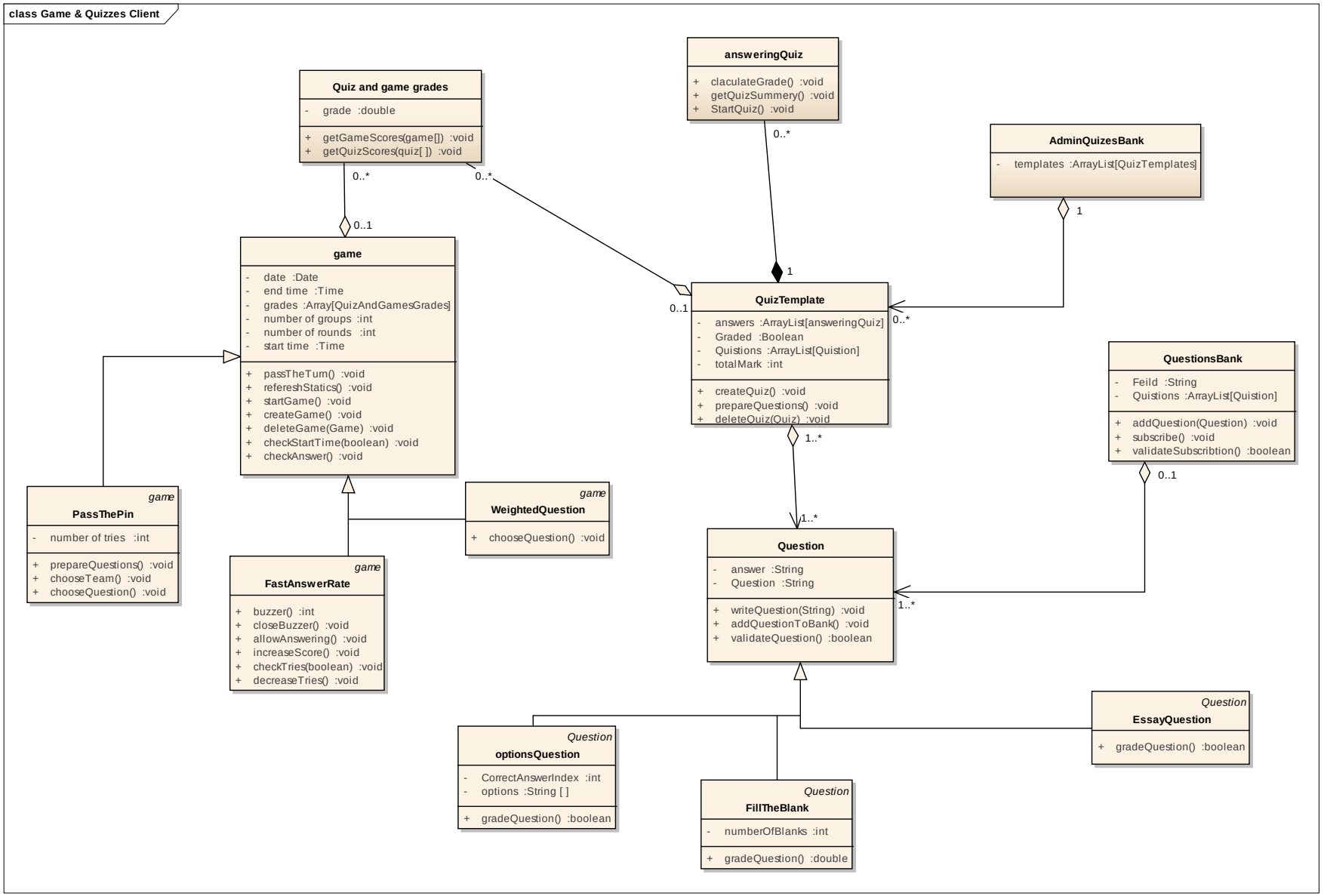
Use Case	Classes Used
[UC 01 -01] – [Login]	User, Admin , Participant
[UC 01 -02] – [Register New account]	User
[UC 01 -03] – [Create Room]	Admin , Room
[UC 01 -04] – [Modify Room]	Admin , Room
[UC 01 -05] – [Invite Participants to Room]	Admin , room , Participant
[UC 01 -06] – [Enroll in Room]	Participant, Room
[UC 01 -07] – [Leave Room]	Participant , Room
[UC 01 -08] – [Take Attendance of Participants]	Admin , Room , AttendanceHistory , DailyAttendance
[UC 01 -09] – [Confirm Presence]	Participant, Room, DailyAttendacneByParticipant
[UC 01 -10] – [Send Broadcast Message	Admin , room , Participant
[UC 01 -11] – [View Participants Records History]	Room , AttendaceHistory,QuizGamesGrades
[UC 01 -12] – [Participant views his History]	Room , AttendaceHistory,QuizGamesGrades
[UC 02 -01] – [Create Schedule]	Room, AppointmetnSchedule, Timeslot
[UC 02 -02] – [Update Schedule]	Room, AppointmetnSchedule, Timeslot
[UC 02 -03] – [Reserve Appointment with Admin]	Room, Admin, AppointmentSchedule
[UC 02 -04] – [Request to Cancel Appointment Reservation]	Room, Admin, AppointmentSchedule
[UC 02 -05] – [Approve Cancel Appointment Request]	Room, AppointmentSchedule
[UC 02 -06] – [Propose a Meeting]	Room, MeetingSchedule, DaySchedule
[UC 02 -07] – [Confirm a Meeting]	Room, Admin, MeetingSchedule, DaySchedule
[UC 03 -01] – [Generate Group]	Room, Group, Participant
[UC 03 -02] – [Modify Generated Group]	Room, Group
[UC 04 -01] – [Create Brainstorming Session]	Room , BrainstormingSession
[UC 04 -02] – [Change Brainstorming Session State]	Room , BrainstormingSession
[UC 04 -03] – [Enter Brainstorming Session]	Room , BrainstormingSession , Idea
[UC 04 -04] – [Talk in Chat]	Room , BrainstormingSession
[UC 04 -05] – [View Summary of Brainstorming Session]	Room , Brainstorming
[UC 05 -01] – [Create Game]	Games, Questions , QuestionBank , Groups , Chat
[UC 05 -02] – [Delete Game]	Games
[UC 05 -03] – [Play “Pass the Pin” Game]	PassThePin , Question
[UC 05 -04] – [Play “Weighted Questions” Game]	WeightedQuestion
[UC 05 -05] – [Play “Fast Answer Race” Game]	WeightedQuestion
[UC 05 -06] – [Prepare Questions by Admin]	Question , QuestionBank
[UC 05 -07] – [Prepare Questions by Participants]	Question , QuestionBank
[UC 05 -08] – [Create Quiz]	Quiz , Question , QuestionBank , chat
[UC 05 -09] – [Modify Quiz]	Quiz , Question , QuestionBank , chat
[UC 05 -10] – [Delete Quiz]	Quiz

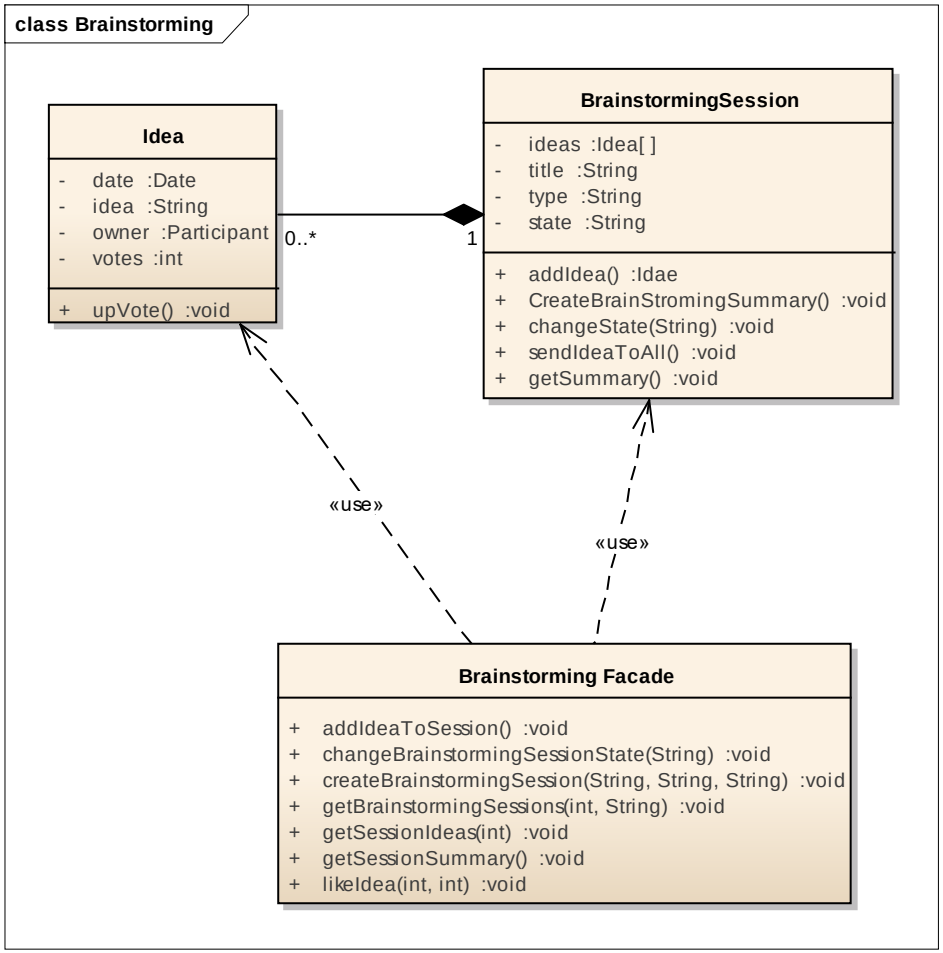
[UC 05 -11] – [Add Questions to the Bank]	Question , QuestionBank
[UC 05 -12] – [Get Questions from the Bank]	QuestionBank
[UC 05 -13] – [Subscribe to the Bank]	QuestionBank
[UC 05 -14] – [Selects Participants Randomly]	Room , Spinner , Participant
[UC 05 -15] – [Answer Quiz]	Room , Participant , Quiz , Question

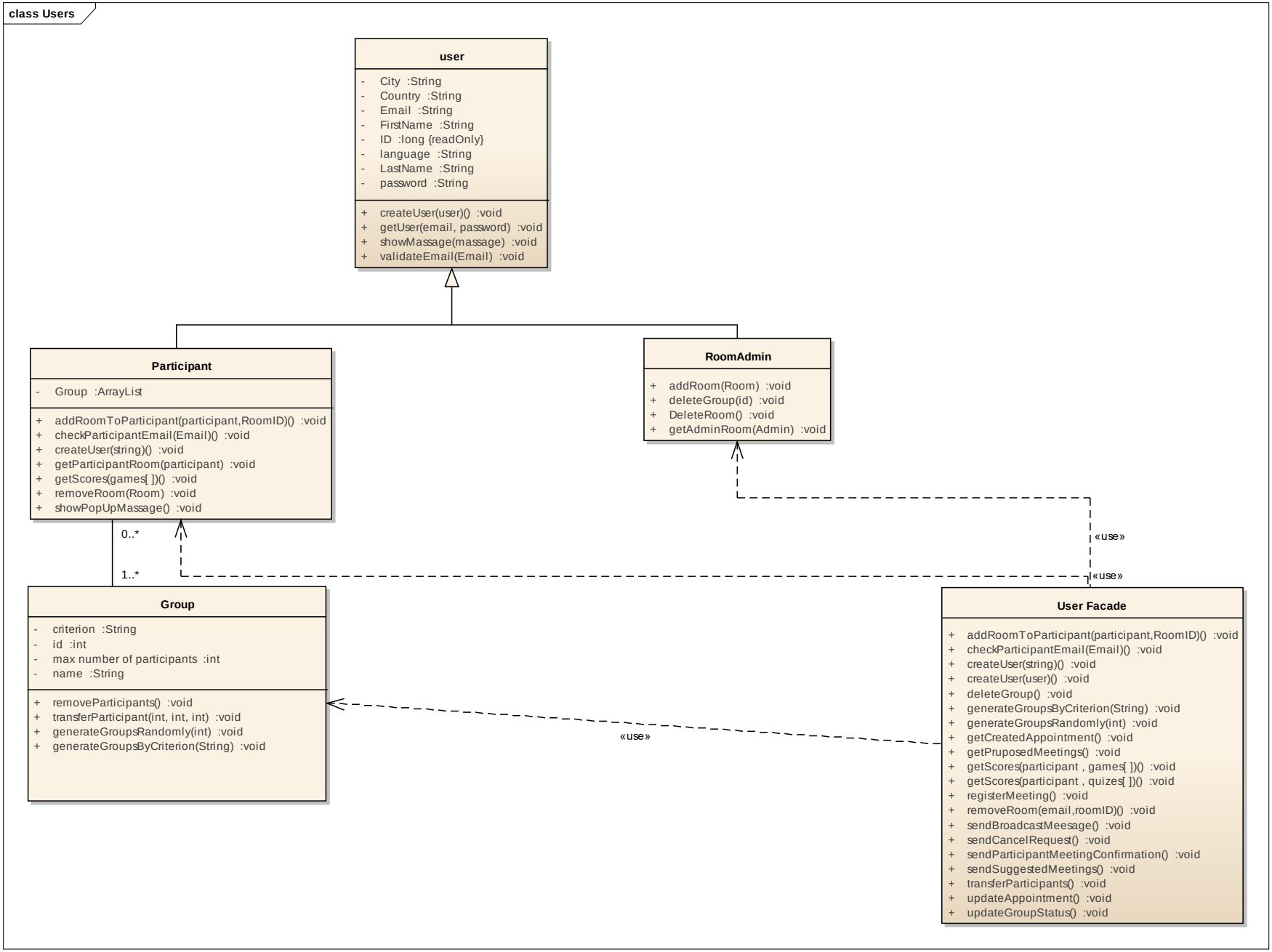
Package Diagram

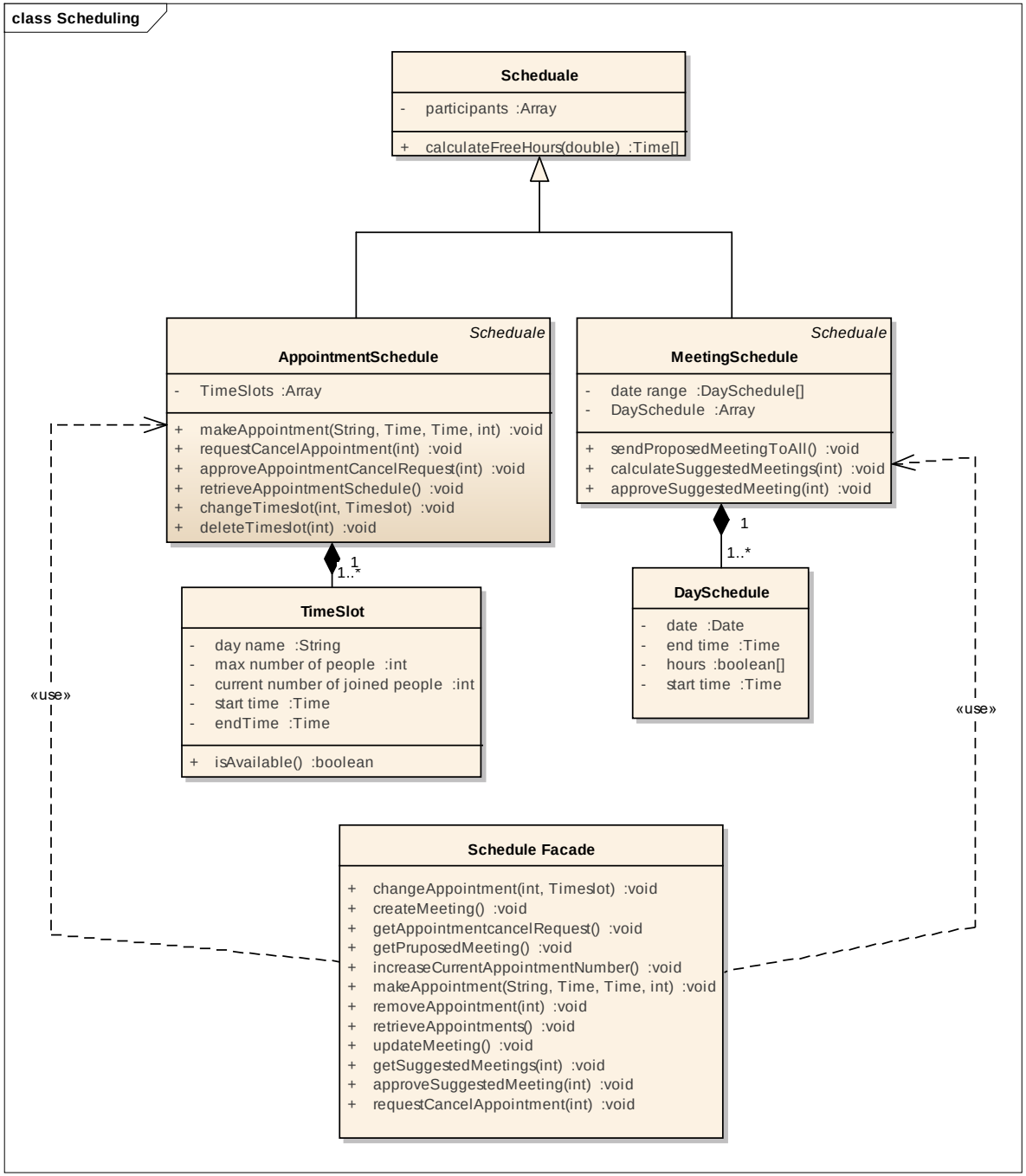



class Room Management
Room
- category :String
- Date :Date
- id :int
- Location :String
- Name :String {readOnly}
- Start Time :Time
+ addParticipant(Participant) :void
+ addParticipant(Email,RoomID)() :void
+ collectRoomHistory() :void
+ collectRoomHistory() :void
+ confirmPrecence(password)() :void
+ createBrainstomringSesison() :void
+ createDailyAttendanceByParticipnats(time,password)() :void
+ createRoom() :void
+ enrollToRoom(RoomID)() :void
+ getAttendanceHistory() :void
+ getDailyAttendanceByParticipant() :void
+ removeParticipant() :void
+ removeParticipnat(RoomID,Email)() :void
+ sendBraodcast(massage)() :void
+ startGame() :void
+ takeDailyAttendance() :void
+ takeDailyAttendance(participants)() :void
+ updateRoomHistory() :void
+ updateRoomInformation(nweRoomInfo) :void
+ updateRoomInformation() :void
Spinner
+ roll(Participant) :void
0..1
1
chat
+ loadMessages(int) :Message[]
+ sendText(Message) :void
0..1
1
Text
- text :String
- date :Date
- time :Time
0..*
AttendanceHistory
+ createCompleteDailyAttendance(participants,Room)() :void
+ getNewDailyAttendance() :void
+ saveAttendance() :void
1
1
1
1..*
«use»
DailyAttendance
- AbcenceParticibant[ ] :Participant
- AttendedParticipant[ ] :Participant
- date :Date
+ createCompleteDailyAttendance(participants,Room)() :void
ByAdmin
Room facade
+ addParticipant(email,roomID)() :void
+ collectRoomHistory() :void
+ confirmPrecence(password)() :void
+ createDailyAttendanceByParticipnats(time,password)() :void
+ createRoom() :void
+ enrollToRoom(RoomID)() :void
+ getAttendanceHistory() :void
+ getHistory() :void
+ removeParticipnat(RoomID,Email)() :void
+ sendBraodcast(massage,Room)() :void
+ takeDailyAttendance(particpants)() :void
+ updateRoomInformation() :void
ByParticipant
- Duration :int
- Password :int {readOnly}
- startTime :int
+ checkPresenceSession(Time ,Password)() :void
+ isPasswordMatch() :void
+ isTimeMatch() :void
+ setDuration(attendace)() :void
+ setPassword(attendance)() :void


class Game & Quizzes Client
Quiz and game grades
- grade :double
+ getGameScores(game[]) :void
+ getQuizScores(quiz[ ]) :void
answeringQuiz
+ claculateGrade() :void
+ getQuizSummery() :void
+ StartQuiz() :void
AdminQuizesBank
- templates :ArrayList[QuizTemplates]
game
- date :Date
- end time :Time
- grades :Array[QuizAndGamesGrades]
- number of groups :int
- number of rounds :int
- start time :Time
+ passTheTum() :void
+ refereshStatics() :void
+ startGame() :void
+ createGame() :void
+ deleteGame(Game) :void
+ checkStartTime(boolean) :void
+ checkAnswer() :void
QuizTemplate
- answers :ArrayList[answeringQuiz]
- Graded :Boolean
- Quistions :ArrayList[Quistion]
- totalMark :int
+ createQuiz() :void
+ prepareQuestions() :void
+ deleteQuiz(Quiz) :void
QuestionsBank
- Feild :String
- Quistions :ArrayList[Quistion]
+ addQuestion(Question) :void
+ subscribe() :void
+ validateSubscribtion() :boolean
game
PassThePin
- number of tries :int
+ prepareQuestions() :void
+ chooseTeam() :void
+ chooseQuestion() :void
game
WeightedQuestion
+ chooseQuestion() :void
game
FastAnswerRate
+ buzzer() :int
+ closeBuzzer() :void
+ allowAnswering() :void
+ increaseScore() :void
+ checkTries(boolean) :void
+ decreaseTries() :void
Question
- answer :String
- Question :String
+ writeQuestion(String) :void
+ addQuestionToBank() :void
+ validateQuestion() :boolean
Question
optionsQuestion
- CorrectAnswerIndex :int
- options :String [ ]
+ gradeQuestion() :boolean
Question
FillTheBlank
- numberOfBlanks :int
+ gradeQuestion() :double
Question
EssayQuestion
+ gradeQuestion() :boolean
0..*
0..*
0..1
1
0..1
0..*
1
1..*
1..*
0..1
1..*

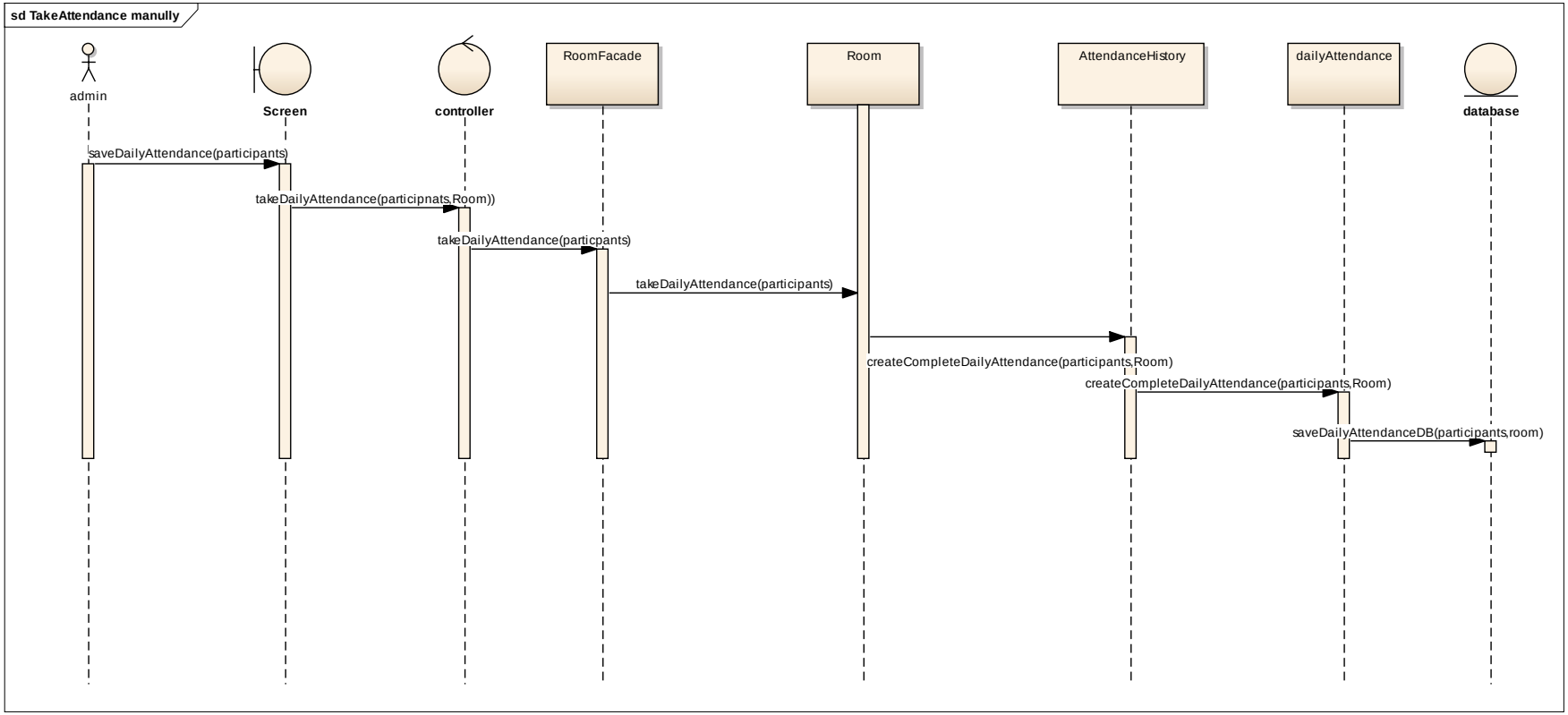

class Brainstorming
Idea
- date :Date
- idea :String
- owner :Participant
- votes :int
+ upVote() :void
0..*
1
BrainstormingSession
- ideas :Idea[ ]
- title :String
- type :String
- state :String
+ addIdea() :Idae
+ CreateBrainStromingSummary() :void
+ changeState(String) :void
+ sendIdeaToAll() :void
+ getSummary() :void
«use»
«use»
Brainstorming Facade
+ addIdeaToSession() :void
+ changeBrainstormingSessionState(String) :void
+ createBrainstormingSession(String, String, String) :void
+ getBrainstormingSessions(int, String) :void
+ getSessionIdeas(int) :void
+ getSessionSummary() :void
+ likeIdea(int, int) :void

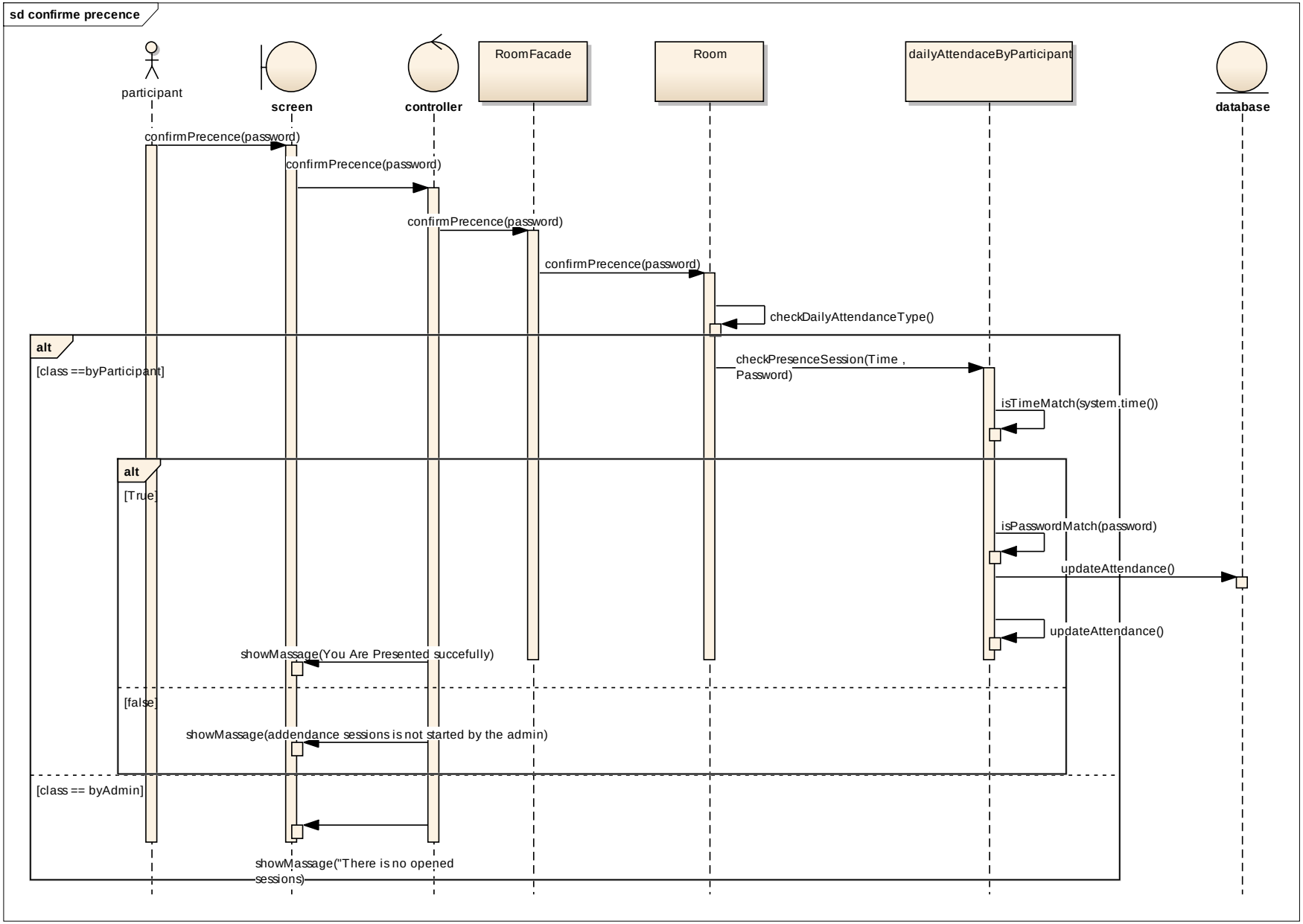

class Users
user
- City :String
- Country :String
- Email :String
- FirstName :String
- ID :long {readOnly}
- language :String
- LastName :String
- password :String
+ createUser(user)() :void
+ getUser(email, password) :void
+ showMassage(massage) :void
+ validateEmail(Email) :void
Participant
- Group :ArrayList
+ addRoomToParticipant(participant,RoomID)() :void
+ checkParticipantEmail(Email)() :void
+ createUser(string)() :void
+ getParticipantRoom(participant) :void
+ getScores(games[ ])() :void
+ removeRoom(Room) :void
+ showPopUpMassage() :void
RoomAdmin
+ addRoom(Room) :void
+ deleteGroup(id) :void
+ DeleteRoom() :void
+ getAdminRoom(Admin) :void
0..*
1..*
«use»
«use»
Group
- criterion :String
- id :int
- max number of participants :int
- name :String
+ removeParticipants() :void
+ transferParticipant(int, int, int) :void
+ generateGroupsRandomly(int) :void
+ generateGroupsByCriterion(String) :void
«use»
User Facade
+ addRoomToParticipant(participant,RoomID)() :void
+ checkParticipantEmail(Email)() :void
+ createUser(string)() :void
+ createUser(user)() :void
+ deleteGroup() :void
+ generateGroupsByCriterion(String) :void
+ generateGroupsRandomly(int) :void
+ getCreatedAppointment() :void
+ getPruposedMeetings() :void
+ getScores(participant , games[ ])() :void
+ getScores(participant , quizes[ ])() :void
+ registerMeeting() :void
+ removeRoom(email,roomID)() :void
+ sendBroadcastMeesage() :void
+ sendCancelRequest() :void
+ sendParticipantMeetingConfirmation() :void
+ sendSuggestedMeetings() :void
+ transferParticipants() :void
+ updateAppointment() :void
+ updateGroupStatus() :void

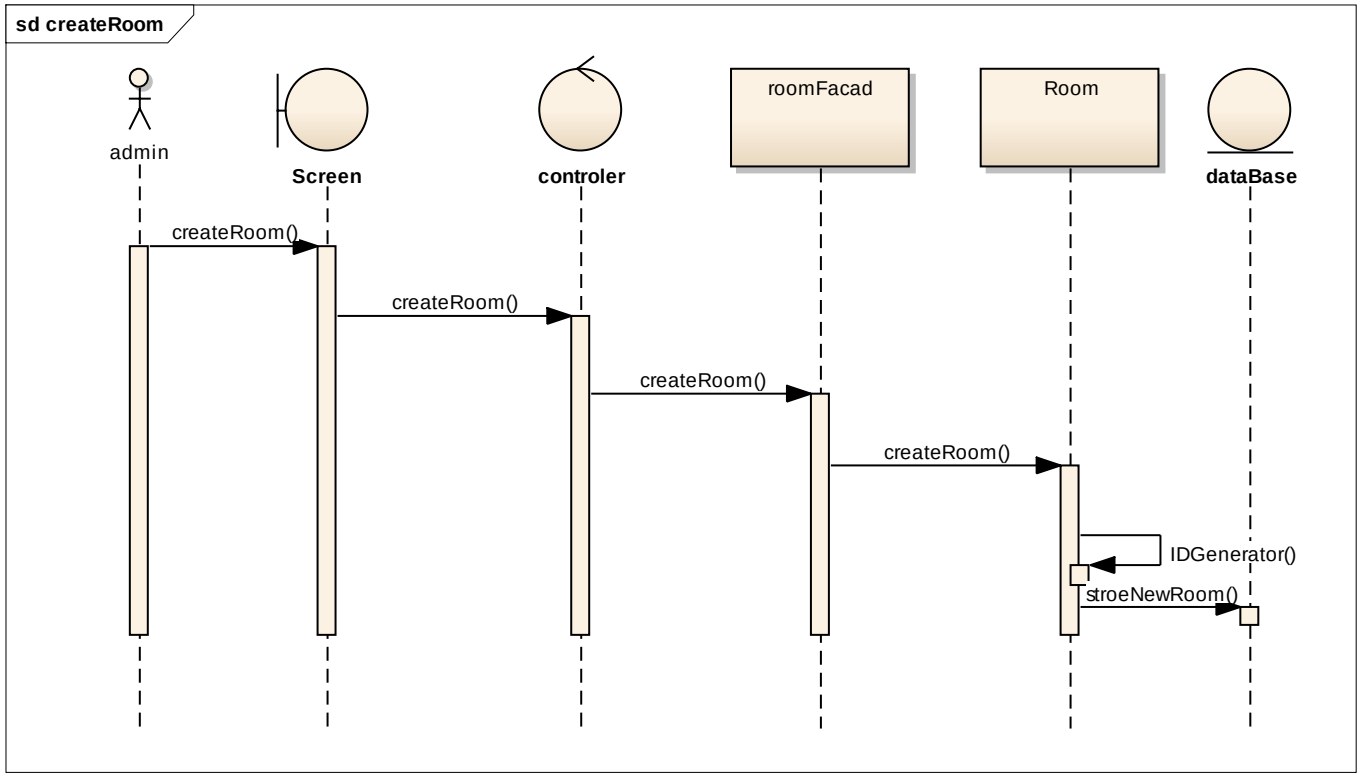

class Scheduling
Scheduale
- participants :Array
+ calculateFreeHours(double) :Time[]
Scheduale
AppointmentSchedule
- TimeSlots :Array
+ makeAppointment(String, Time, Time, int) :void
+ requestCancelAppointment(int) :void
+ approveAppointmentCancelRequest(int) :void
+ retrieveAppointmentSchedule() :void
+ changeTimeslot(int, Timeslot) :void
+ deleteTimeslot(int) :void
Scheduale
MeetingSchedule
- date range :DaySchedule[]
- DaySchedule :Array
+ sendProposedMeetingToAll() :void
+ calculateSuggestedMeetings(int) :void
+ approveSuggestedMeeting(int) :void
1
1..*
1
1..*
TimeSlot
- day name :String
- max number of people :int
- current number of joined people :int
- start time :Time
- endTime :Time
+ isAvailable() :boolean
DaySchedule
- date :Date
- end time :Time
- hours :boolean[]
- start time :Time
«use»
«use»
Schedule Facade
+ changeAppointment(int, Timeslot) :void
+ createMeeting() :void
+ getAppointmentcancelRequest() :void
+ getPruposedMeeting() :void
+ increaseCurrentAppointmentNumber() :void
+ makeAppointment(String, Time, Time, int) :void
+ removeAppointment(int) :void
+ retrieveAppointments() :void
+ updateMeeting() :void
+ getSuggestedMeetings(int) :void
+ approveSuggestedMeeting(int) :void
+ requestCancelAppointment(int) :void

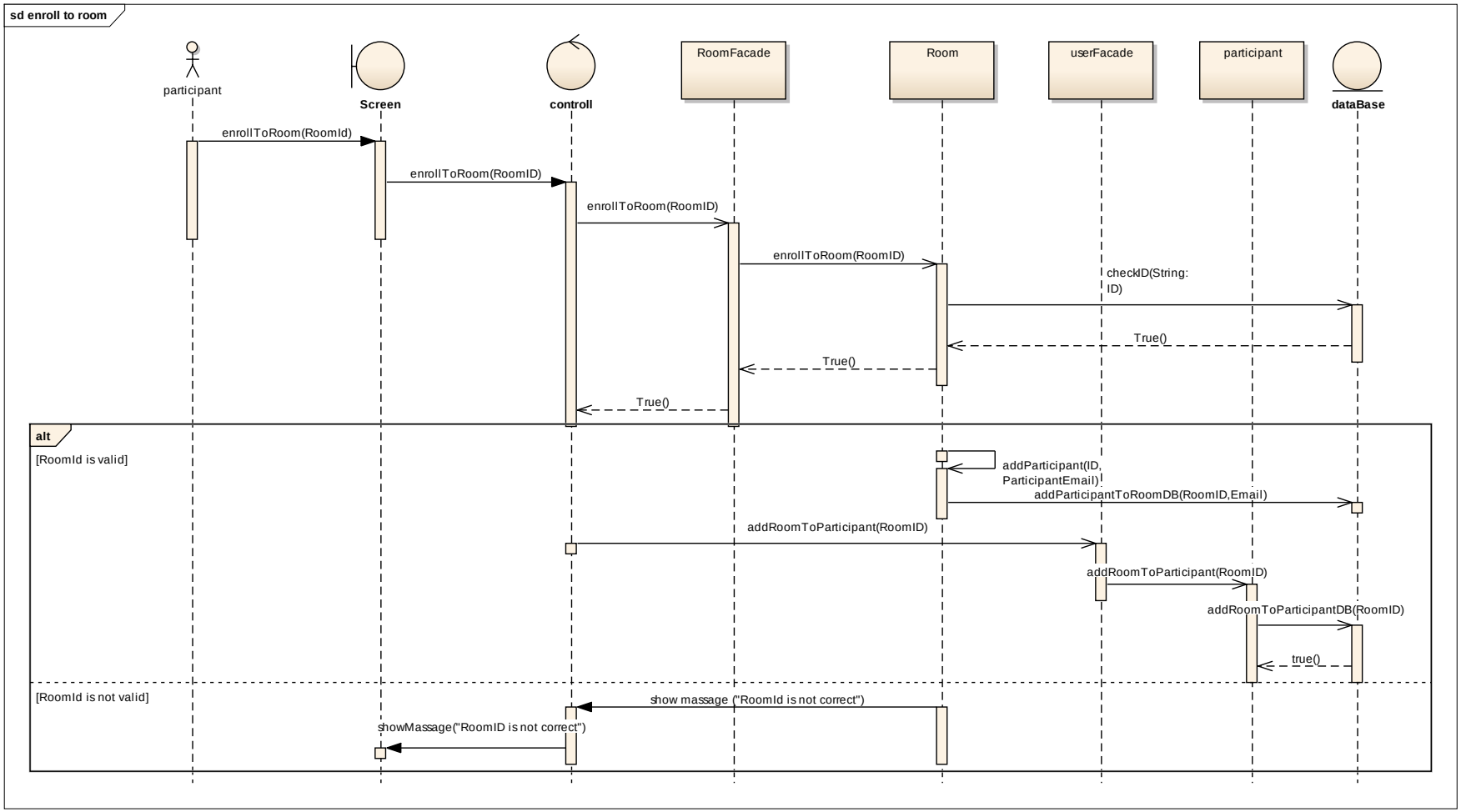
Sequence Diagrams

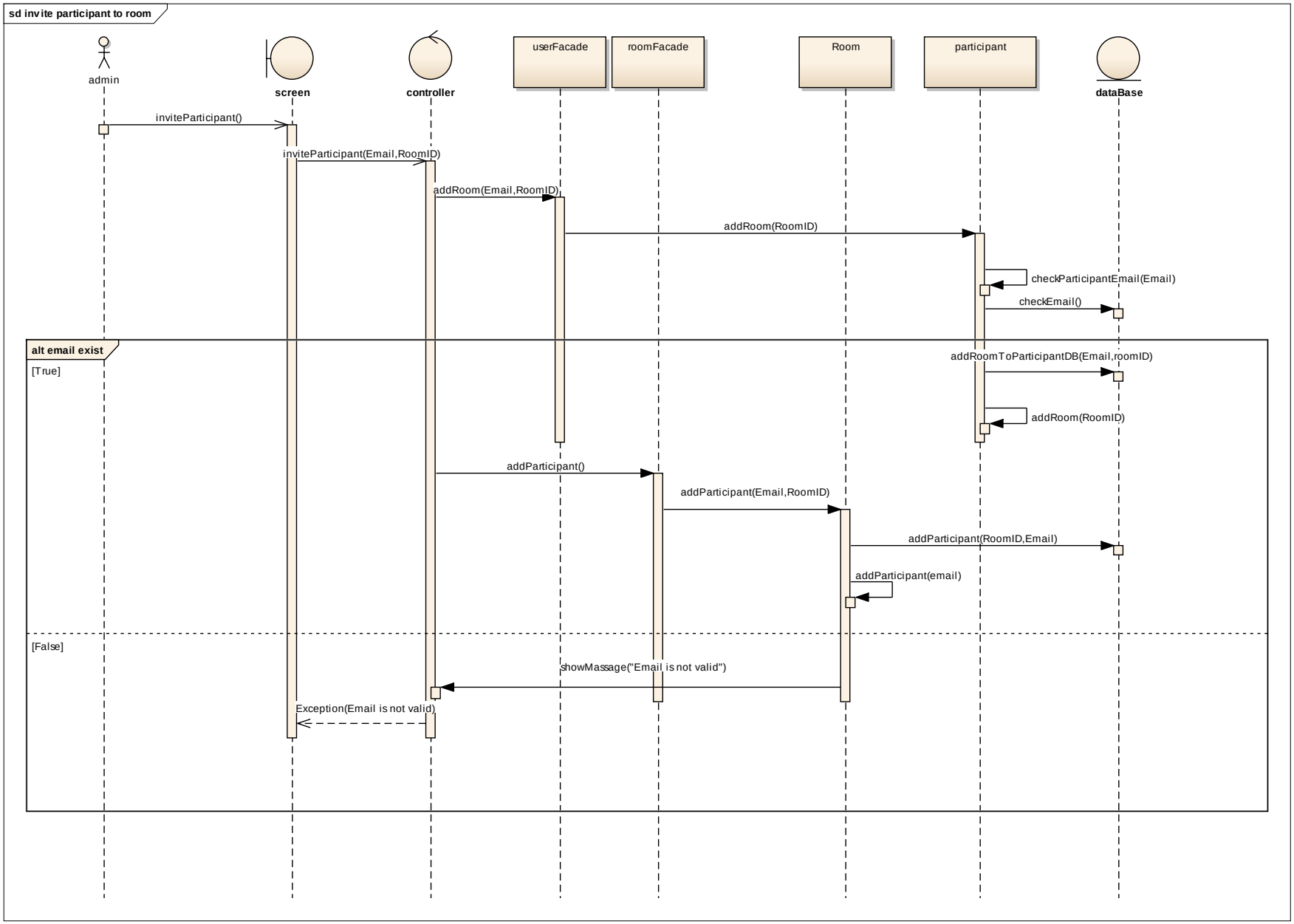
Room Management

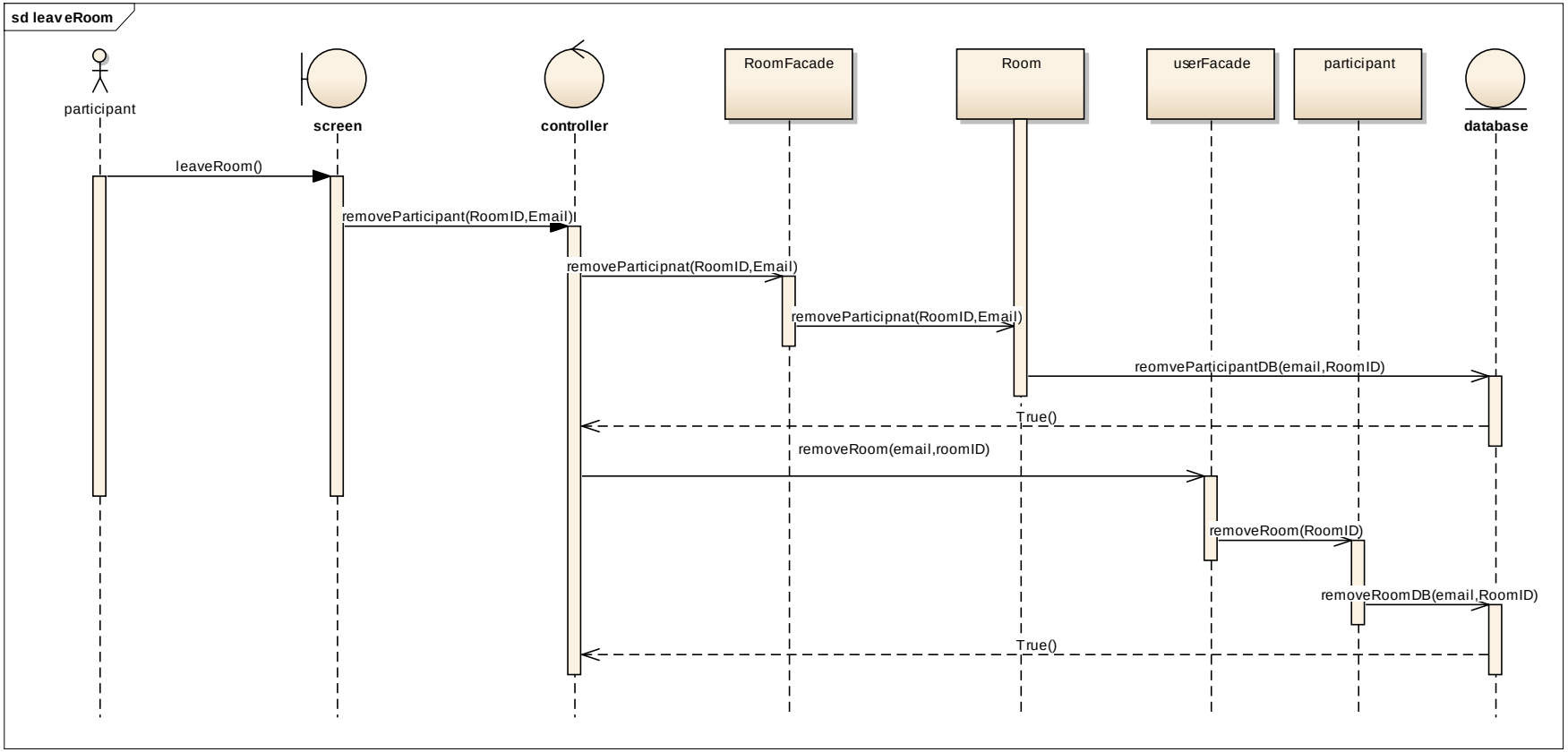


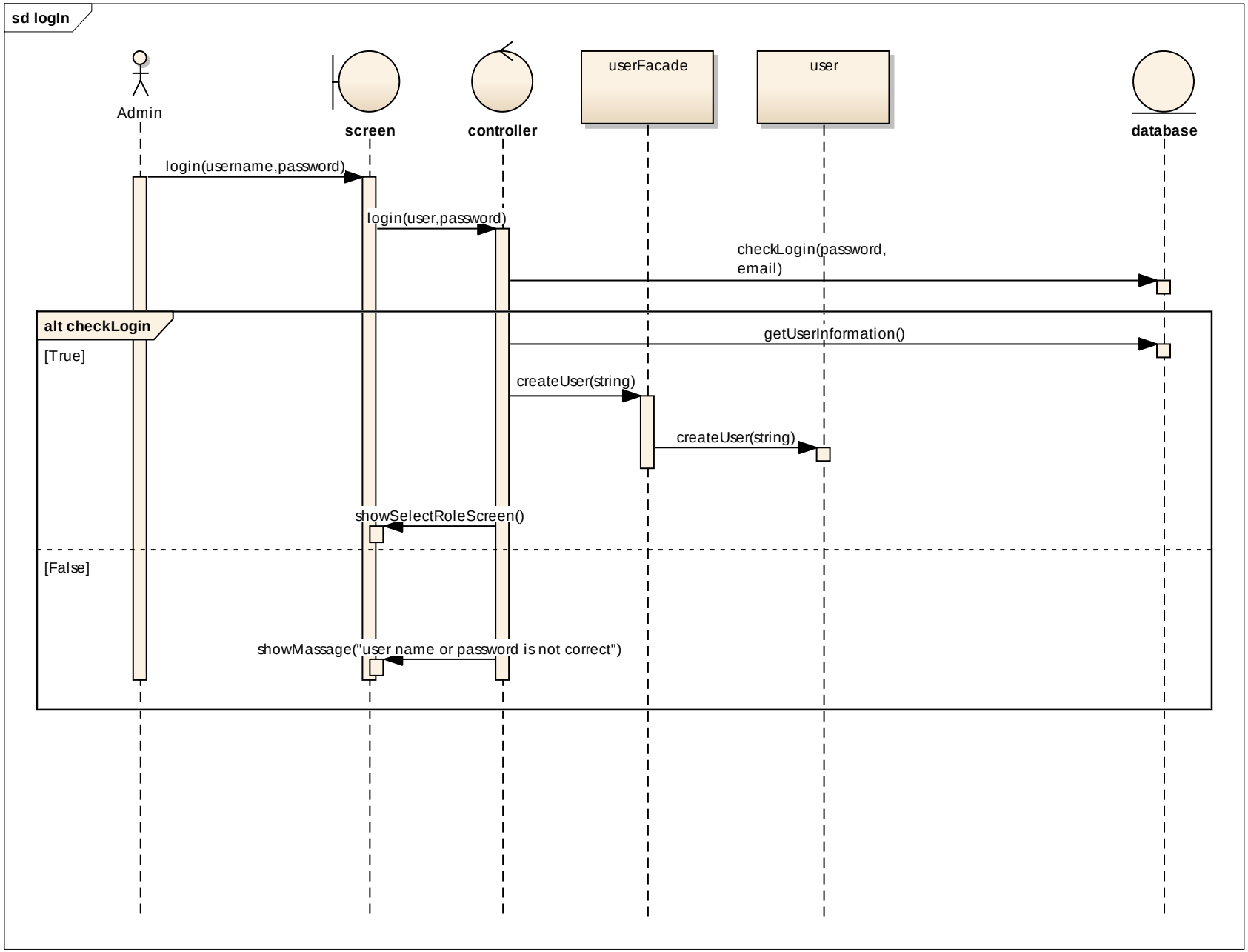

sd confirme precence
participant
screen
controller
RoomFacade
Room
dailyAttendaceByParticipant
database
confirmPrecence(password)
confirmPrecence(password)
confirmPrecence(password)
confirmPrecence(password)
checkDailyAttendanceType()
alt
[class ==byParticipant]
checkPresenceSession(Time , Password)
isTimeMatch(system.time())
alt
[True]
isPasswordMatch(password)
updateAttendance()
updateAttendance()
showMassage(You Are Presented succefully)
[false]
showMassage(addendance sessions is not started by the admin)
[class == byAdmin]
showMassage("There is no opened sessions)

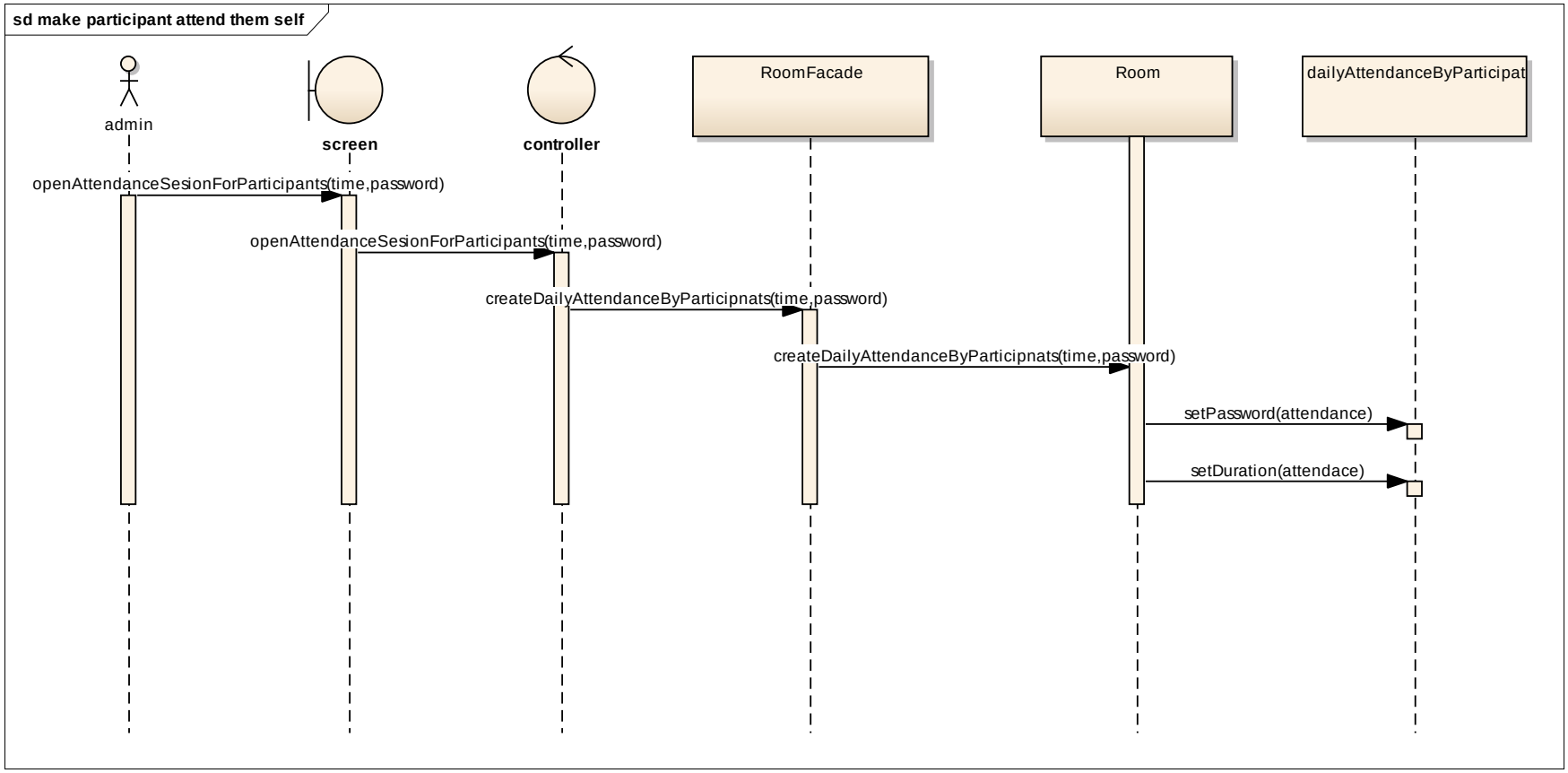

sd createRoom
admin
Screen
controler
roomFacad
Room
dataBase
createRoom()
createRoom()
createRoom()
createRoom()
IDGenerator()
stroeNewRoom()

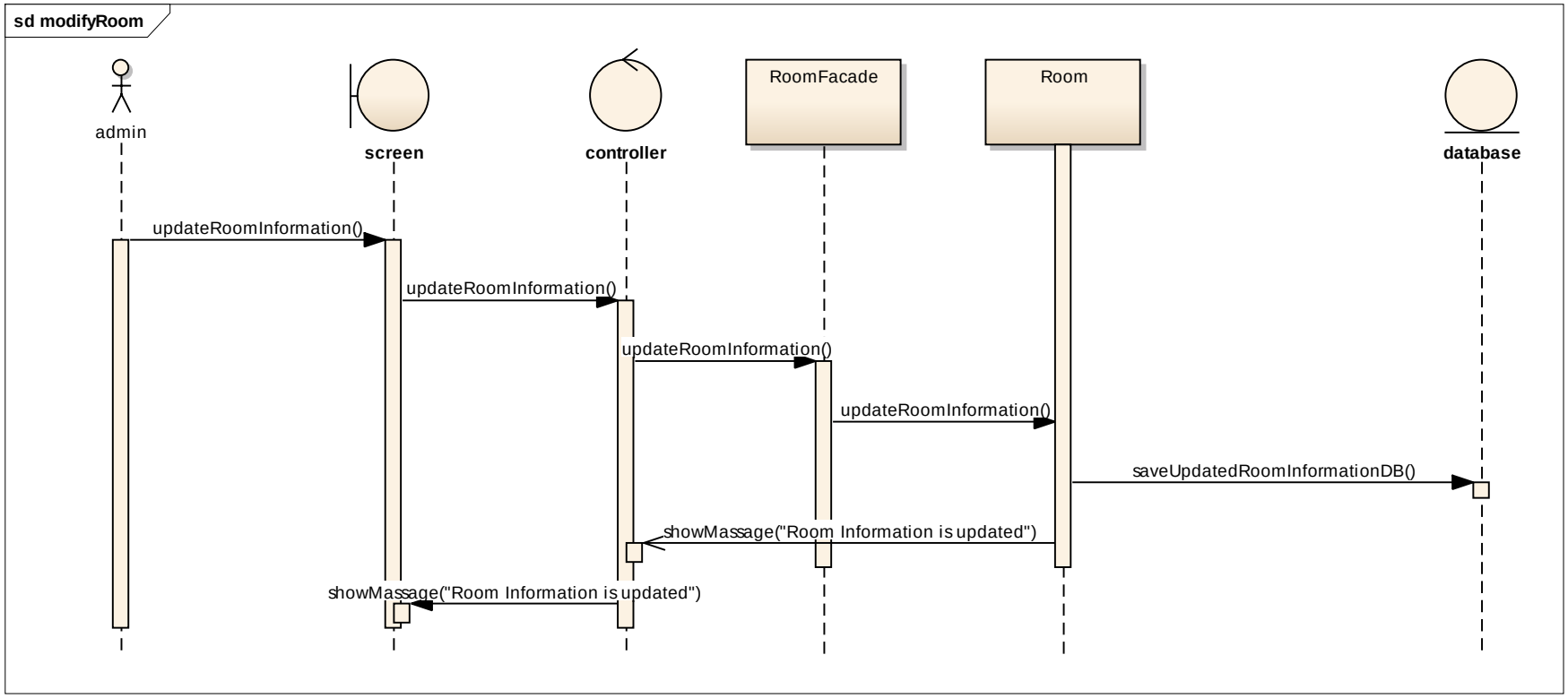

sd enroll to room
participant
Screen
controll
RoomFacade
Room
userFacade
participant
dataBase
enrollToRoom(RoomId)
enrollToRoom(RoomID)
enrollToRoom(RoomID)
enrollToRoom(RoomID)
checkID(String: ID)
True()
True()
True()
alt
[RoomId is valid]
addParticipant(ID, ParticipantEmail)
addParticipantToRoomDB(RoomID,Email)
addRoomToParticipant(RoomID)
addRoomToParticipant(RoomID)
addRoomToParticipantDB(RoomID)
true()
[RoomId is not valid]
show massage ("RoomId is not correct")
showMassage("RoomID is not correct")

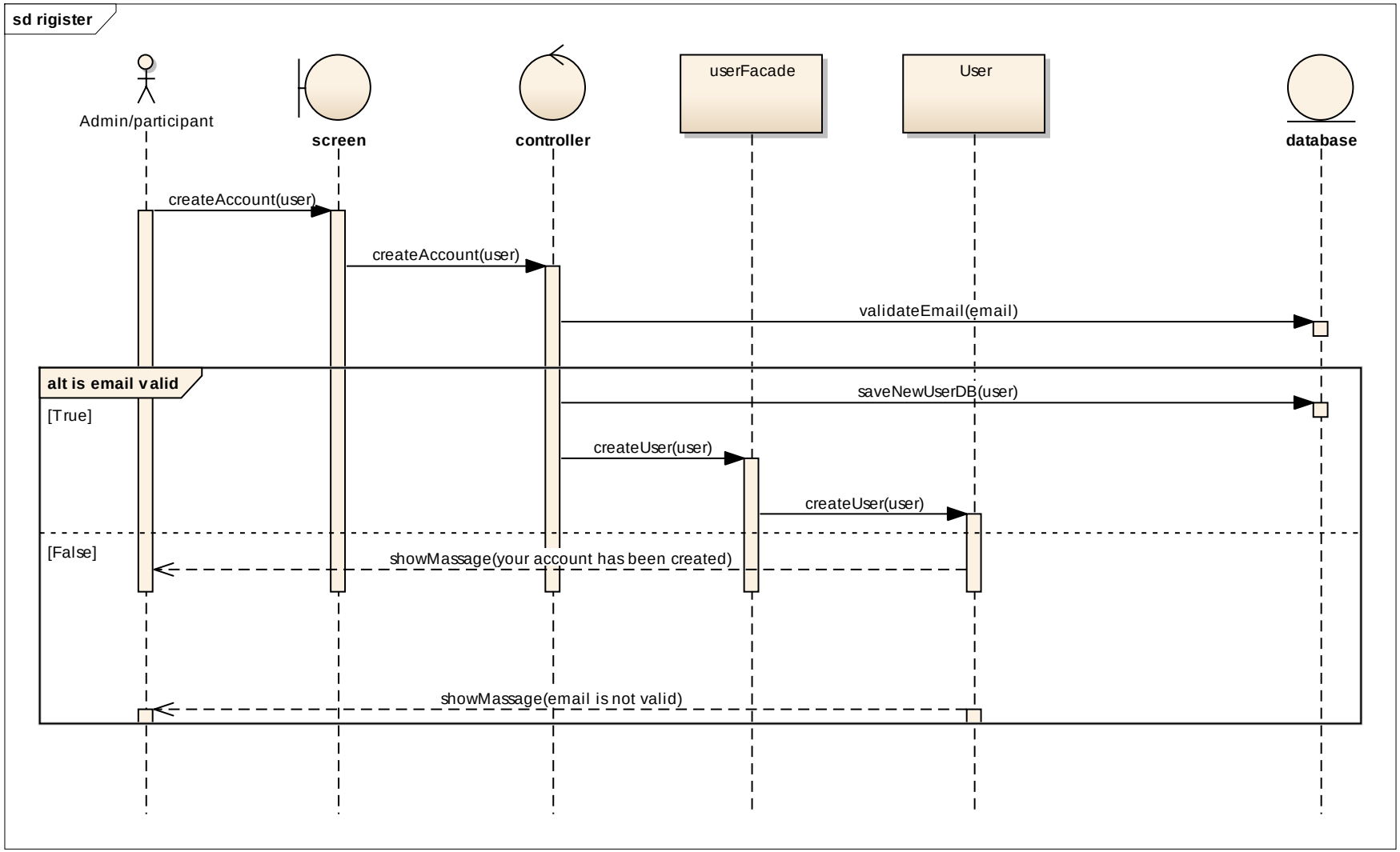

sd invite participant to room
admin
screen
controller
userFacade
roomFacade
Room
participant
dataBase
inviteParticipant()
inviteParticipant(Email,RoomID)
addRoom(Email,RoomID)
addRoom(RoomID)
checkParticipantEmail(Email)
checkEmail()
alt email exist
[True]
addRoomToParticipantDB(Email,roomID)
addRoom(RoomID)
addParticipant()
addParticipant(Email,RoomID)
addParticipant(RoomID,Email)
addParticipant(email)
[False]
showMassage("Email is not valid")
Exception(Email is not valid)

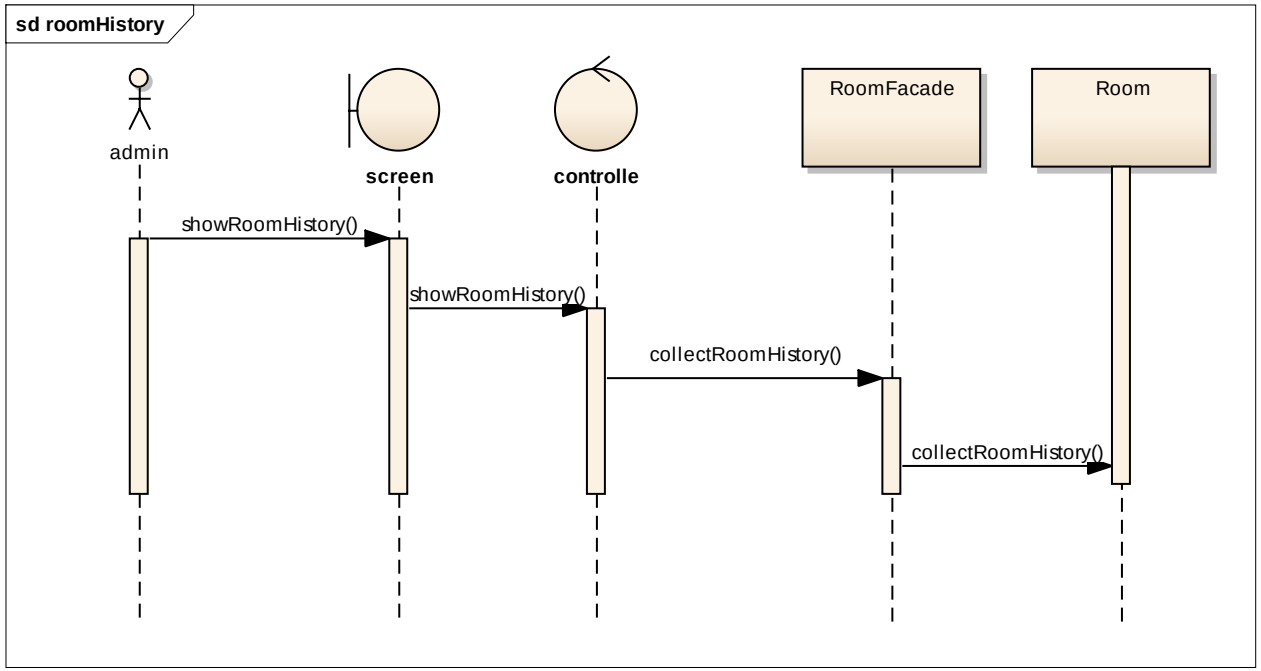

sd leaveRoom
participant
screen
controller
RoomFacade
Room
userFacade
participant
database
leaveRoom()
removeParticipant(RoomID,Email)
removeParticipnat(RoomID,Email)
removeParticipnat(RoomID,Email)
reomveParticipantDB(email,RoomID)
True()
removeRoom(email,roomID)
removeRoom(RoomID)
removeRoomDB(email,RoomID)
True()

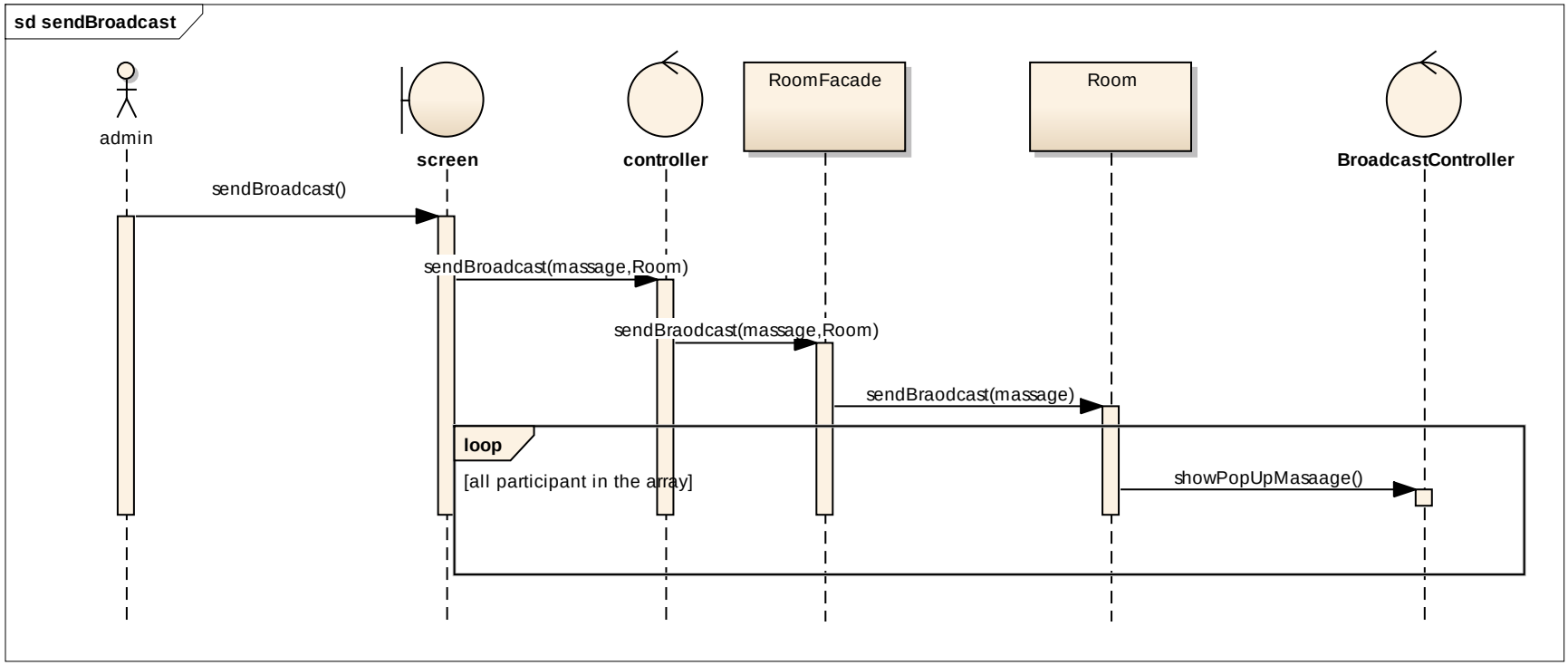

sd logIn
Admin
screen
controller
userFacade
user
database
login(username,password)
login(user,password)
checkLogin(password,
email)
alt checkLogin
[True]
getUserInformation()
createUser(string)
createUser(string)
showSelectRoleScreen()
[False]
showMassage("user name or password is not correct")

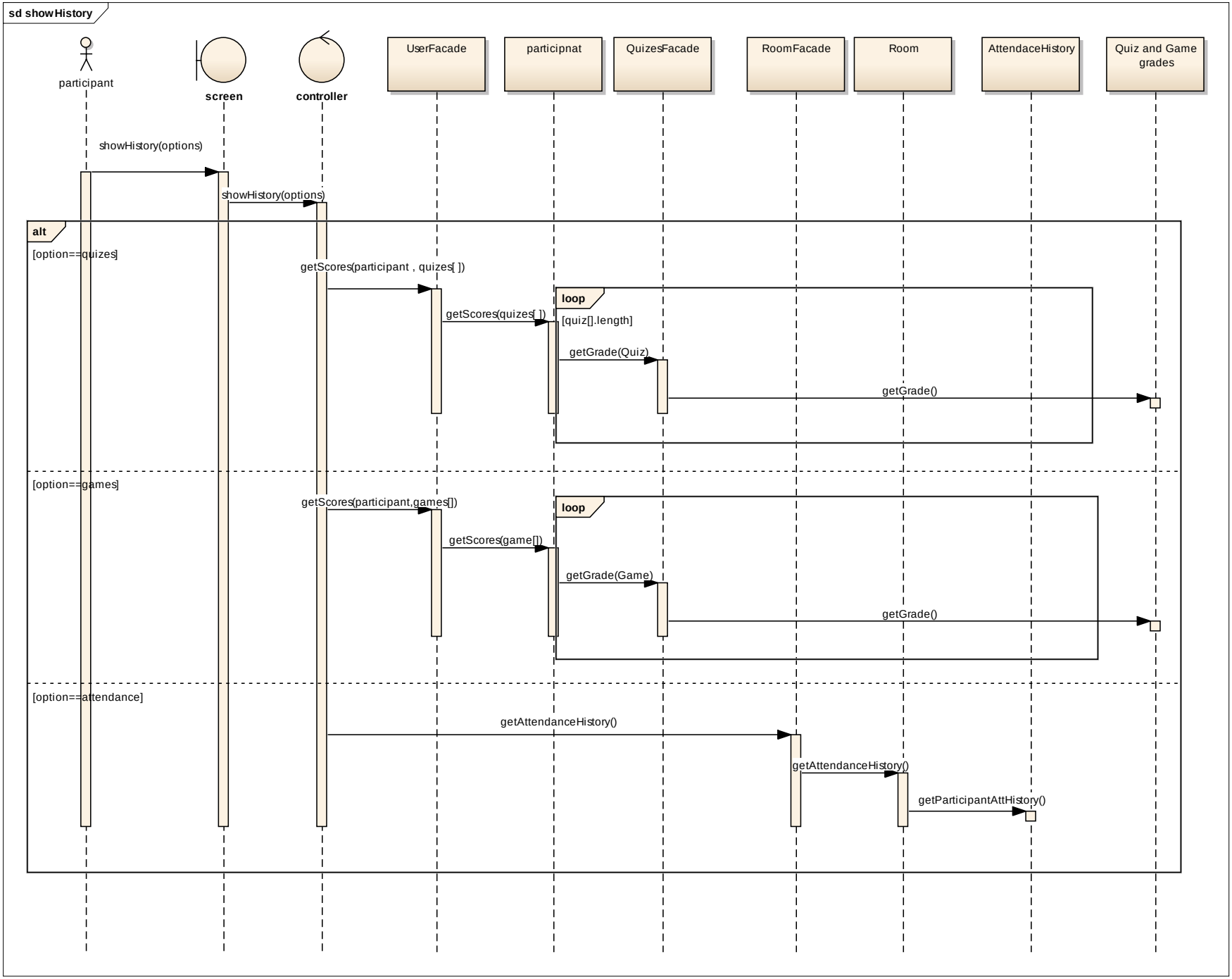

sd make participant attend them self
admin
screen
controller
RoomFacade
Room
dailyAttendanceByParticipat
openAttendanceSesionForParticipants(time,password)
openAttendanceSesionForParticipants(time,password)
createDailyAttendanceByParticipnats(time,password)
createDailyAttendanceByParticipnats(time,password)
setPassword(attendance)
setDuration(attendace)


sd modifyRoom
admin
screen
controller
RoomFacade
Room
database
updateRoomInformation()
updateRoomInformation()
updateRoomInformation()
updateRoomInformation()
saveUpdatedRoomInformationDB()
showMassage("Room Information is updated")
showMassage("Room Information is updated")

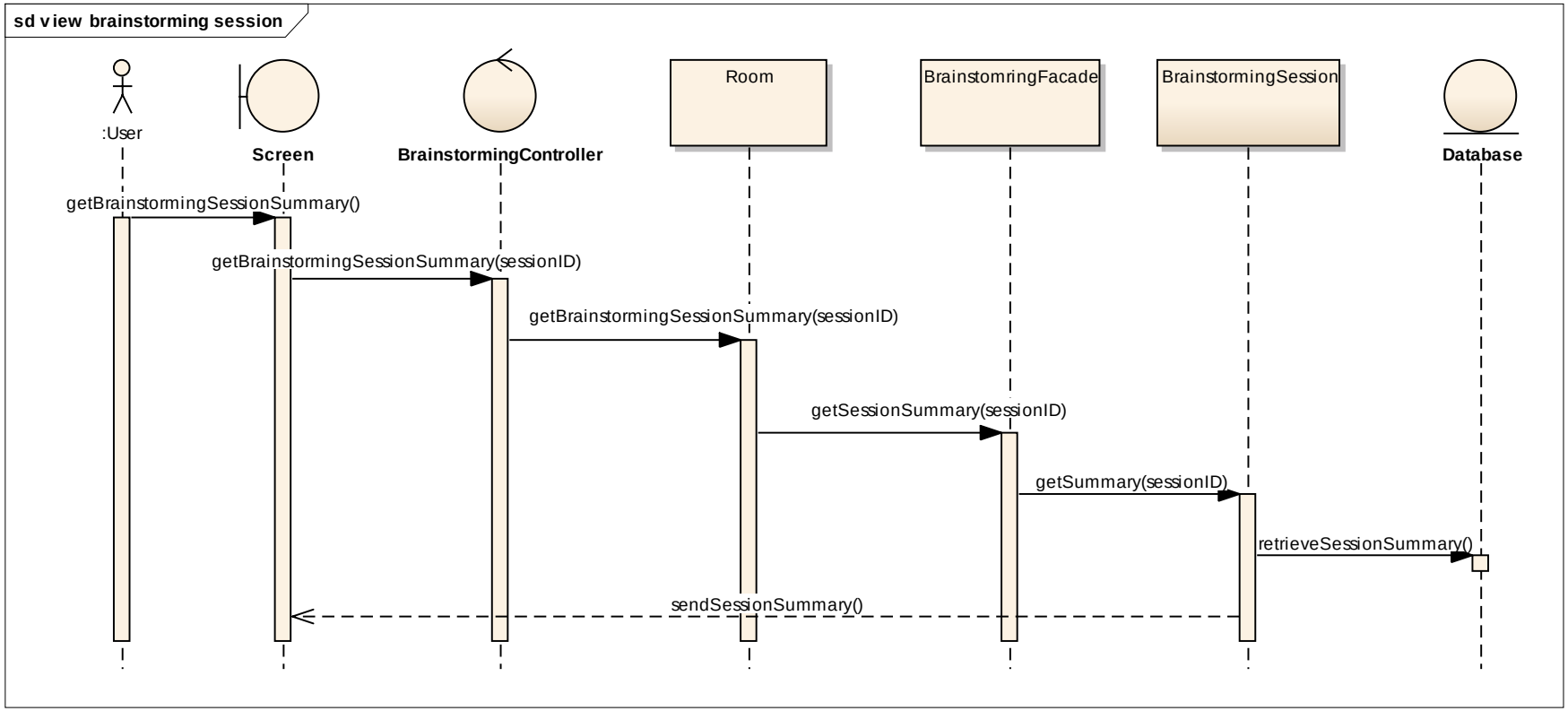

sd rigister
Admin/participant
screen
controller
userFacade
User
database
createAccount(user)
createAccount(user)
validateEmail(email)
alt is email valid
[True]
saveNewUserDB(user)
createUser(user)
createUser(user)
[False]
showMassage(your account has been created)
showMassage(email is not valid)

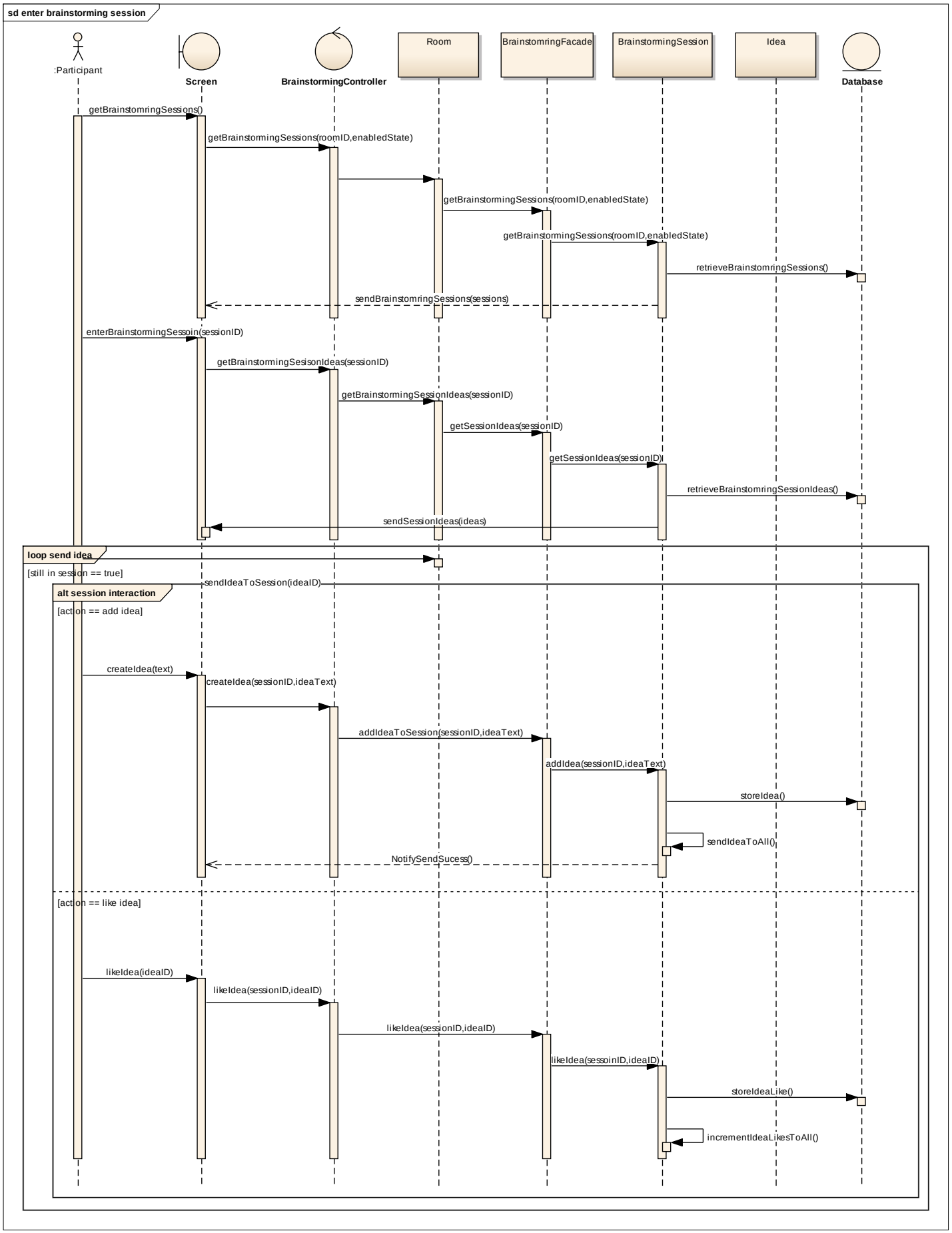

sd roomHistory
admin
screen
controlle
RoomFacade
Room
showRoomHistory()
showRoomHistory()
collectRoomHistory()
collectRoomHistory()

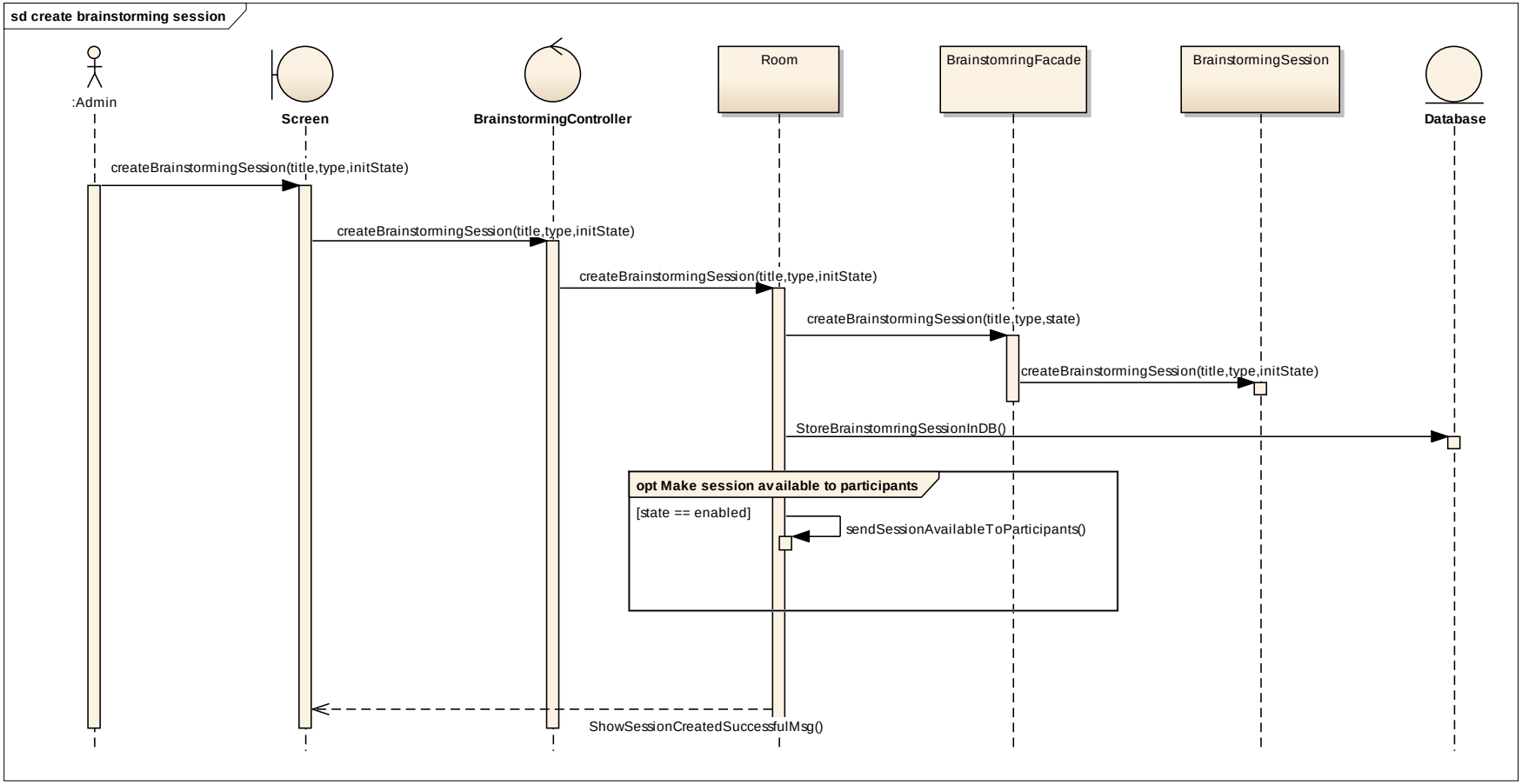

sd sendBroadcast
admin
screen
controller
RoomFacade
Room
BroadcastController
sendBroadcast()
sendBroadcast(massage,Room)
sendBraodcast(massage,Room)
sendBraodcast(massage)
loop
[all participant in the array]
showPopUpMasaage()

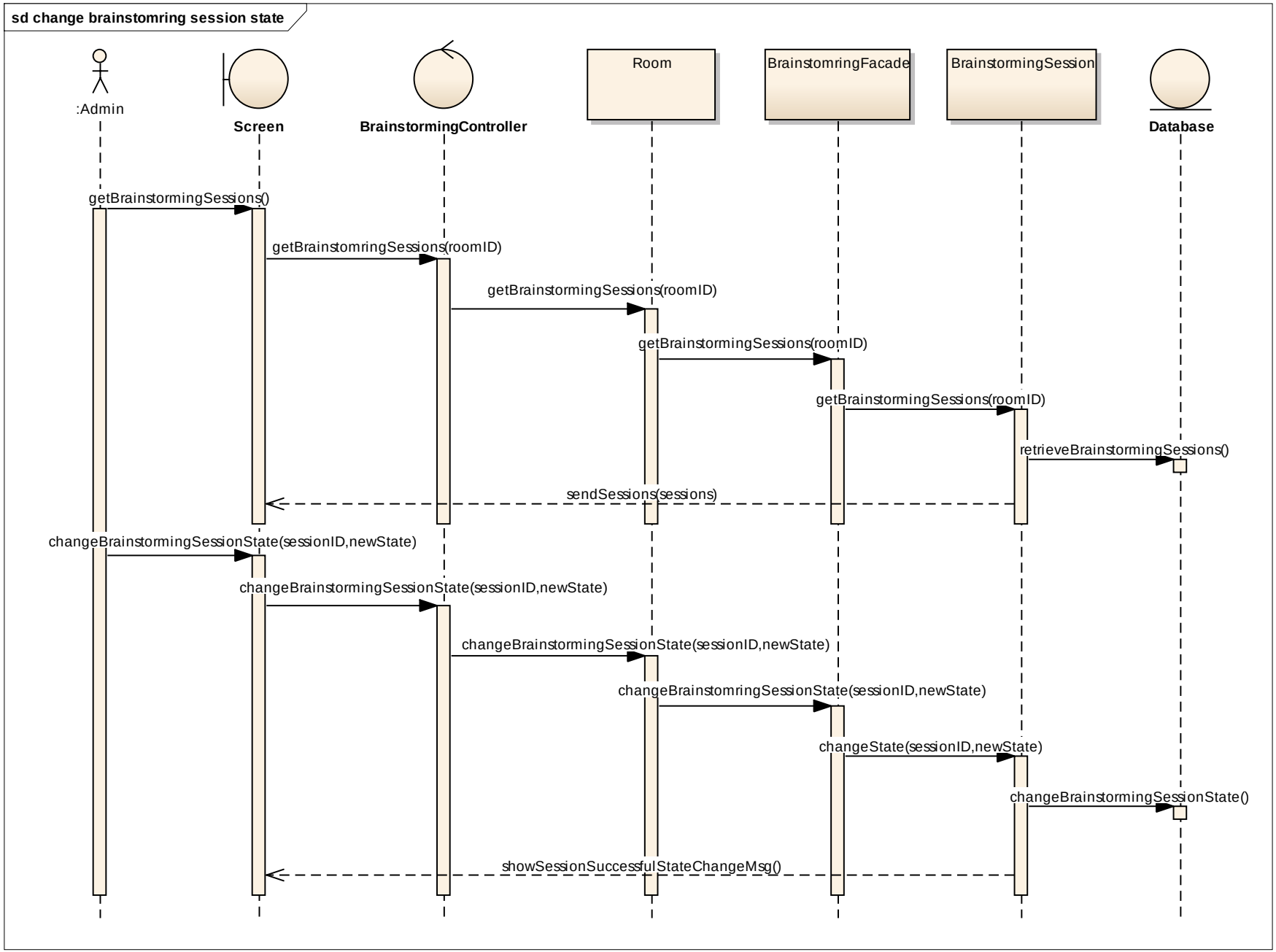

sd showHistory
participant
screen
controller
UserFacade
participnat
QuizesFacade
RoomFacade
Room
AttendaceHistory
Quiz and Game grades
showHistory(options)
showHistory(options)
alt
[option==quizes]
getScores(participant , quizes[ ])
loop
getScores(quizes[ ])
[quiz[].length]
getGrade(Quiz)
getGrade()
[option==games]
getScores(participant,games[])
loop
getScores(game[])
getGrade(Game)
getGrade()
[option==attendance]
getAttendanceHistory()
getAttendanceHistory()
getParticipantAttHistory()

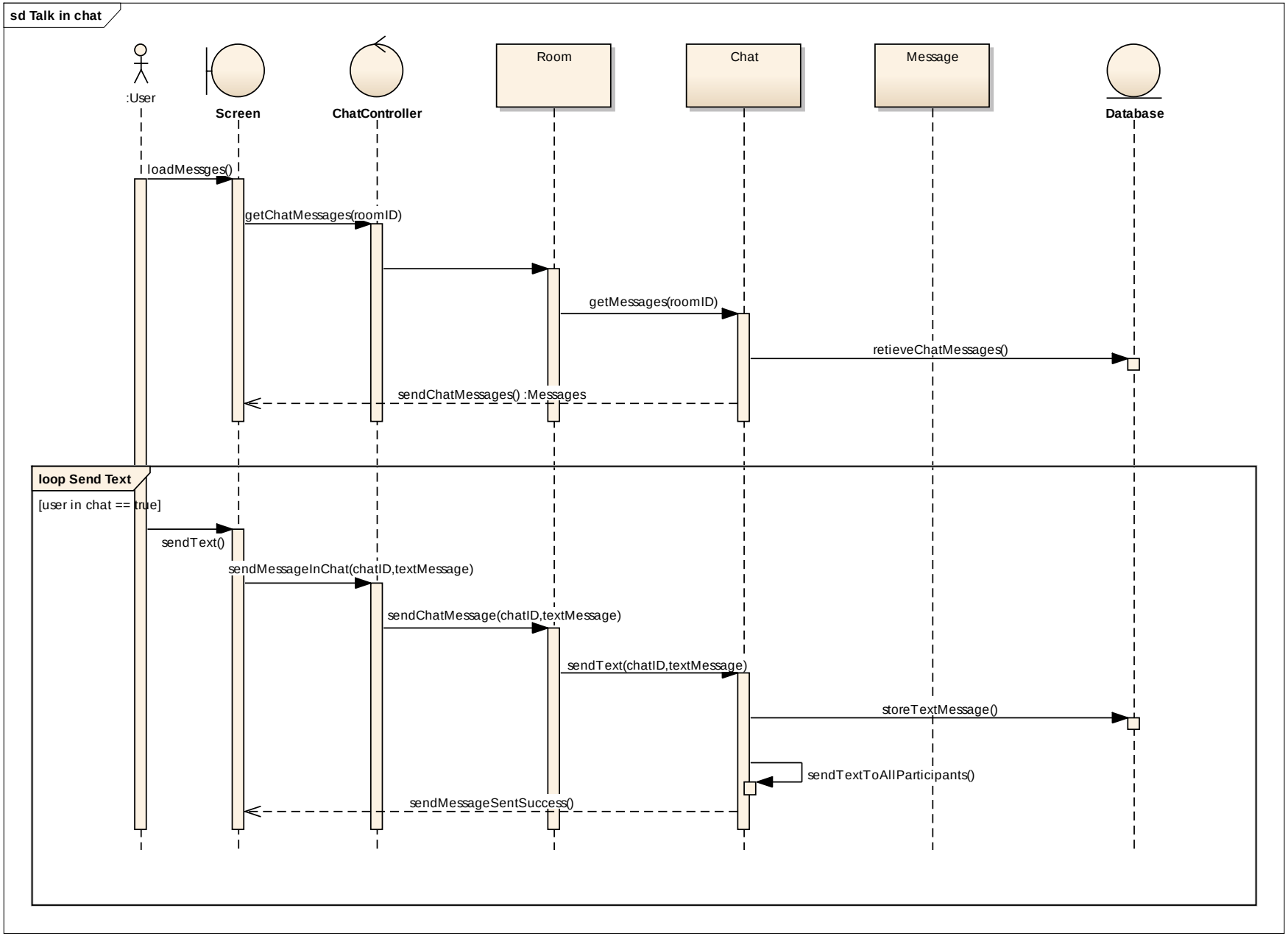
Brainstorming



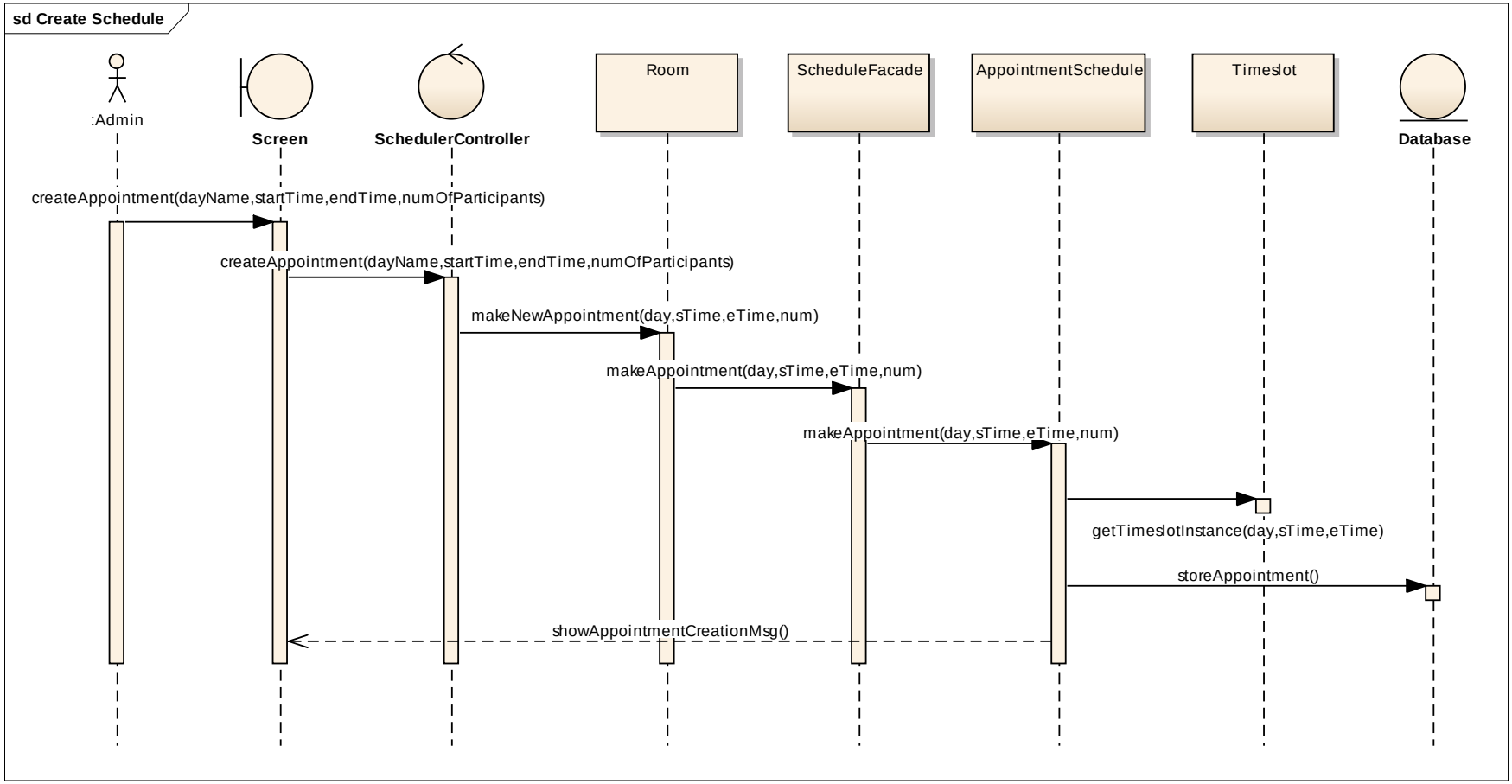

sd enter brainstorming session
:Participant
Screen
BrainstormingController
Room
BrainstomringFacade
BrainstormingSession
Idea
Database
getBrainstomringSessions()
getBrainstormingSessions(roomID,enabledState)
getBrainstormingSessions(roomID,enabledState)
getBrainstormingSessions(roomID,enabledState)
retrieveBrainstomringSessions()
sendBrainstomringSessions(sessions)
enterBrainstormingSessoin(sessionID)
getBrainstormingSesisonIdeas(sessionID)
getBrainstormingSessionIdeas(sessionID)
getSessionIdeas(sessionID)
getSessionIdeas(sessionID)
retrieveBrainstomringSessionIdeas()
sendSessionIdeas(ideas)
loop send idea
[still in session == true]
sendIdeaToSession(idealD)
alt session interaction
[action == add idea]
createIdea(text)
createIdea(sessionID,ideaText)
addIdeaToSession(sessionID,ideaText)
addIdea(sessionID,ideaText)
storeIdea()
sendIdeaToAll()
NotifySendSucess()
[action == like idea]
likeIdea(idealD)
likeIdea(sessionID,idealD)
likeIdea(sessionID,idealD)
likeIdea(sessoinID,idealD)
storeIdeaLike()
incrementIdeaLikesToAll()

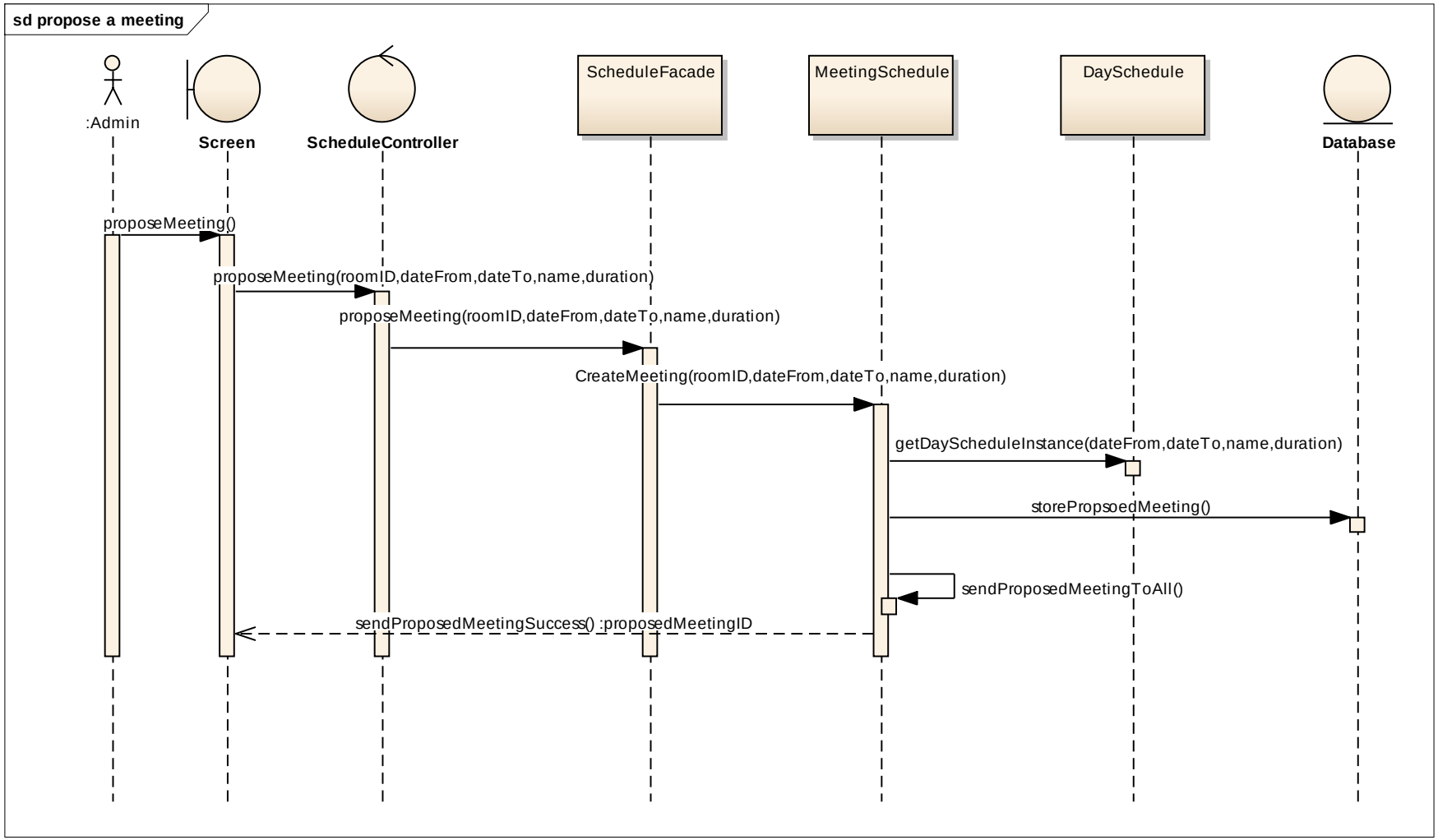

sd create brainstorming session
:Admin
Screen
BrainstormingController
Room
BrainstomringFacade
BrainstormingSession
Database
createBrainstormingSession(title,type,initState)
createBrainstormingSession(title,type,initState)
createBrainstormingSession(title,type,initState)
createBrainstormingSession(title,type,state)
createBrainstormingSession(title,type,initState)
StoreBrainstomringSessionInDB()
opt Make session available to participants
[state == enabled]
sendSessionAvailableToParticipants()
ShowSessionCreatedSuccessfulMsg()

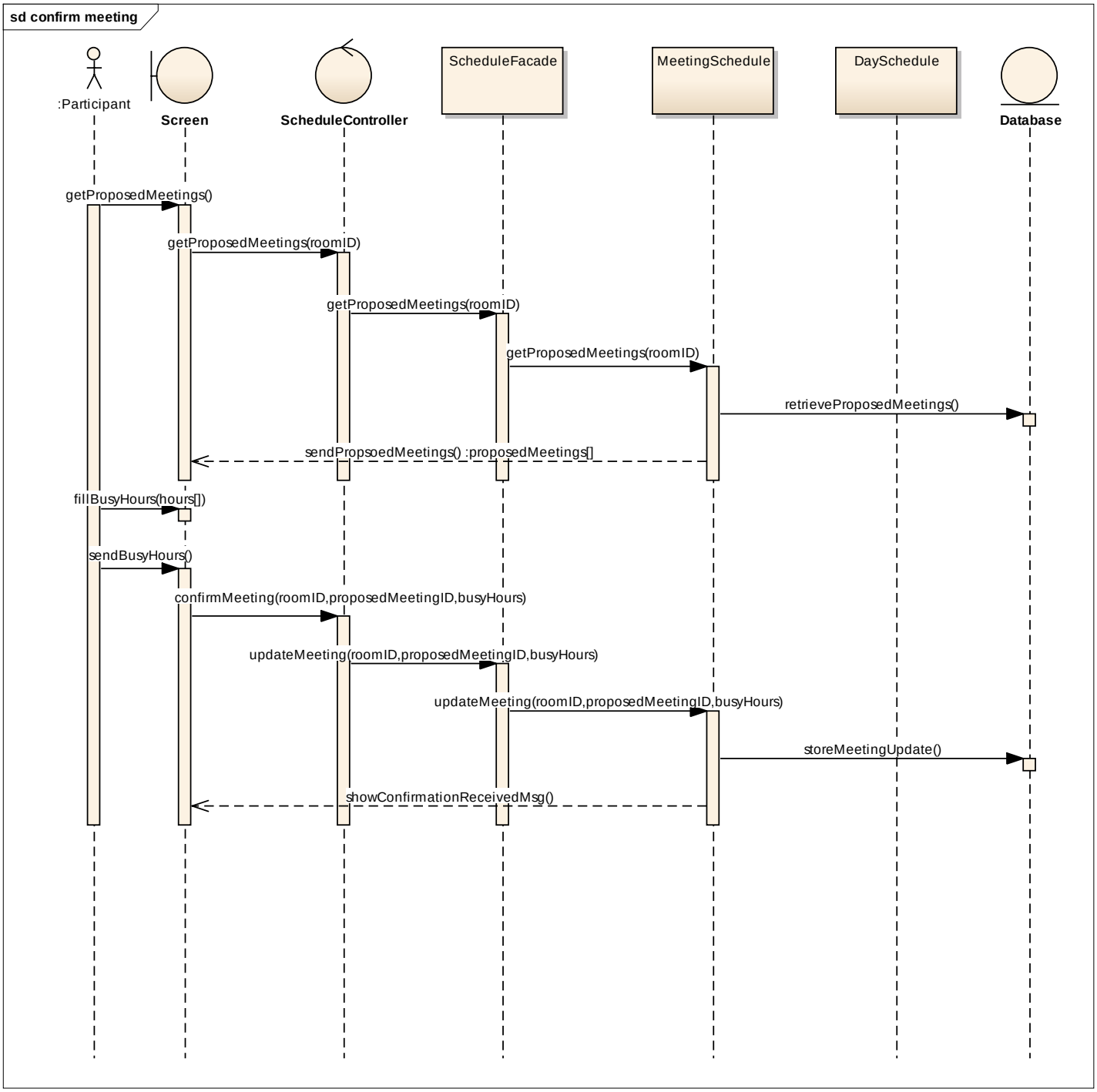

sd change brainstomring session state
:Admin
Screen
BrainstormingController
Room
BrainstomringFacade
BrainstormingSession
Database
getBrainstormingSessions()
getBrainstomringSessions(roomID)
getBrainstormingSessions(roomID)
getBrainstormingSessions(roomID)
getBrainstormingSessions(roomID)
retrieveBrainstormingSessions()
sendSessions(sessions)
changeBrainstormingSessionState(sessionID,newState)
changeBrainstormingSessionState(sessionID,newState)
changeBrainstormingSessionState(sessionID,newState)
changeBrainstomringSessionState(sessionID,newState)
changeState(sessionID,newState)
changeBrainstormingSessionState()
showSessionSuccessfulStateChangeMsg()

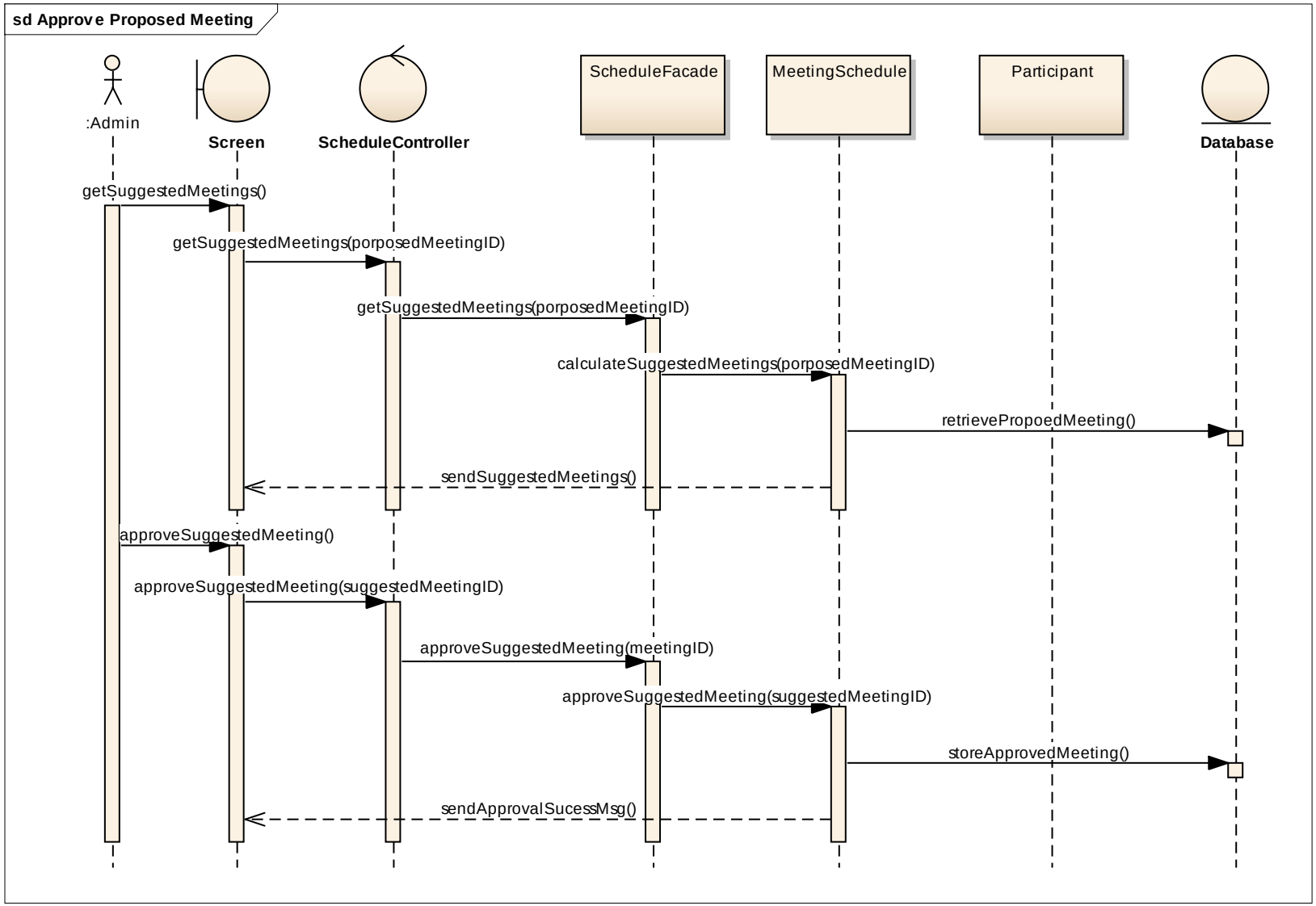

sd Talk in chat
:User
Screen
ChatController
Room
Chat
Message
Database
loadMessges()
getChatMessages(roomID)
getMessages(roomID)
retieveChatMessages()
sendChatMessages() :Messages
loop Send Text
[user in chat == true]
sendText()
sendMessageInChat(chatID,textMessage)
sendChatMessage(chatID,textMessage)
sendText(chatID,textMessage)
storeTextMessage()
sendTextToAllParticipants()
sendMessageSentSuccess()

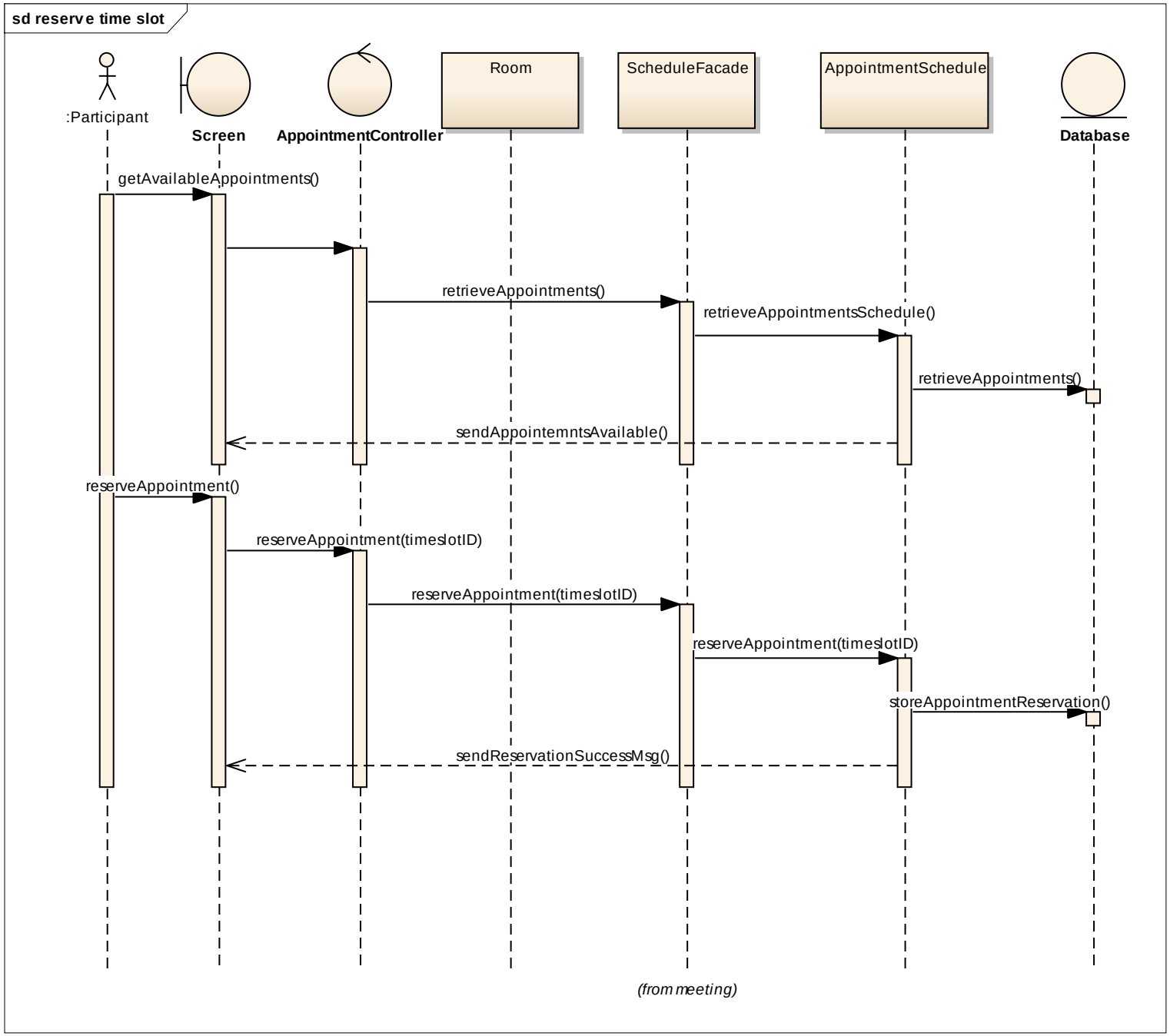
Scheduling

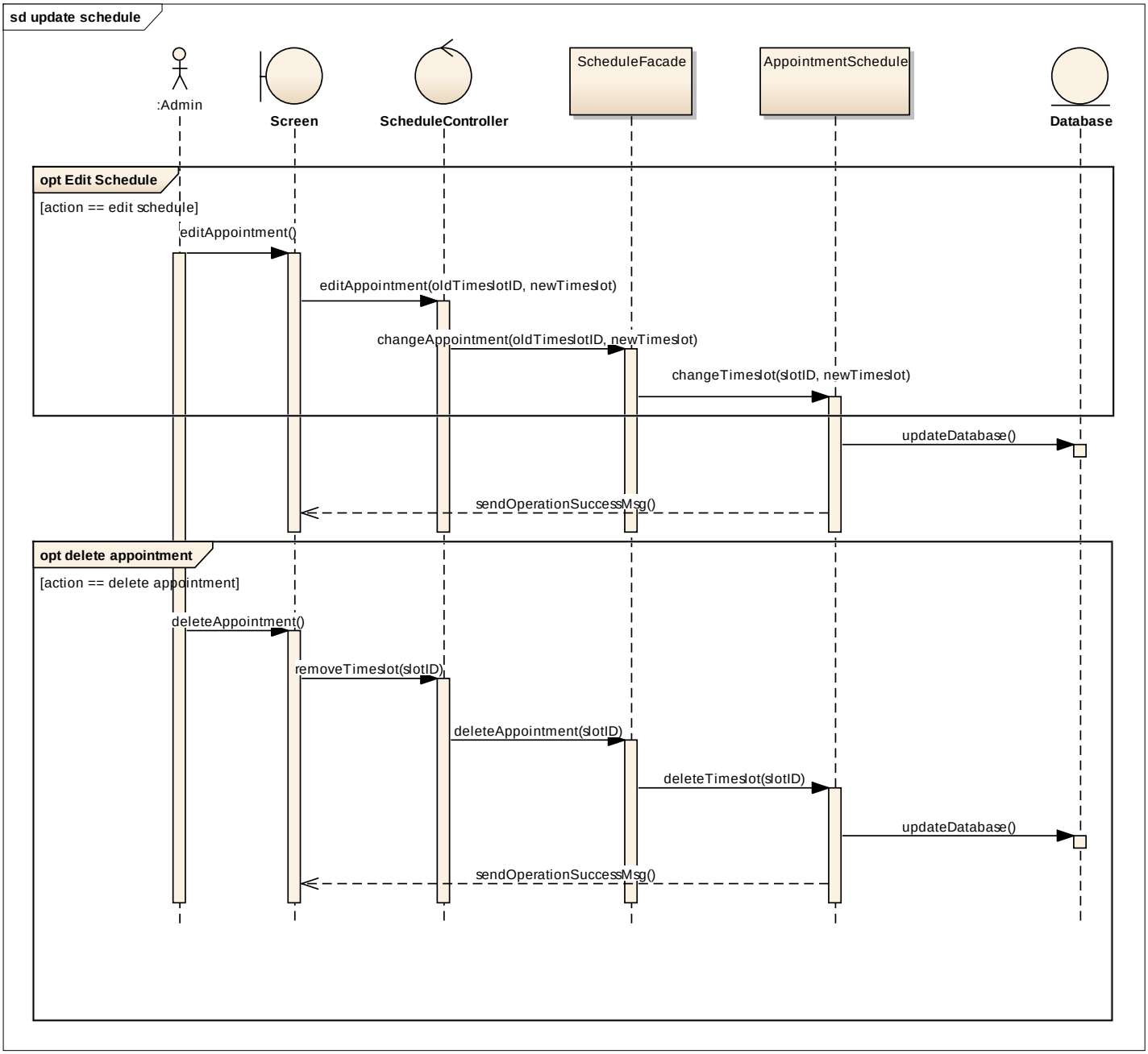

sd Create Schedule
:Admin
Screen
SchedulerController
Room
ScheduleFacade
AppointmentSchedule
Timeslot
Database
createAppointment(dayName,startTime,endTime,numOfParticipants)
createAppointment(dayName,startTime,endTime,numOfParticipants)
makeNewAppointment(day,sTime,eTime,num)
makeAppointment(day,sTime,eTime,num)
makeAppointment(day,sTime,eTime,num)
getTimeslotInstance(day,sTime,eTime)
storeAppointment()
showAppointmentCreationMsg()

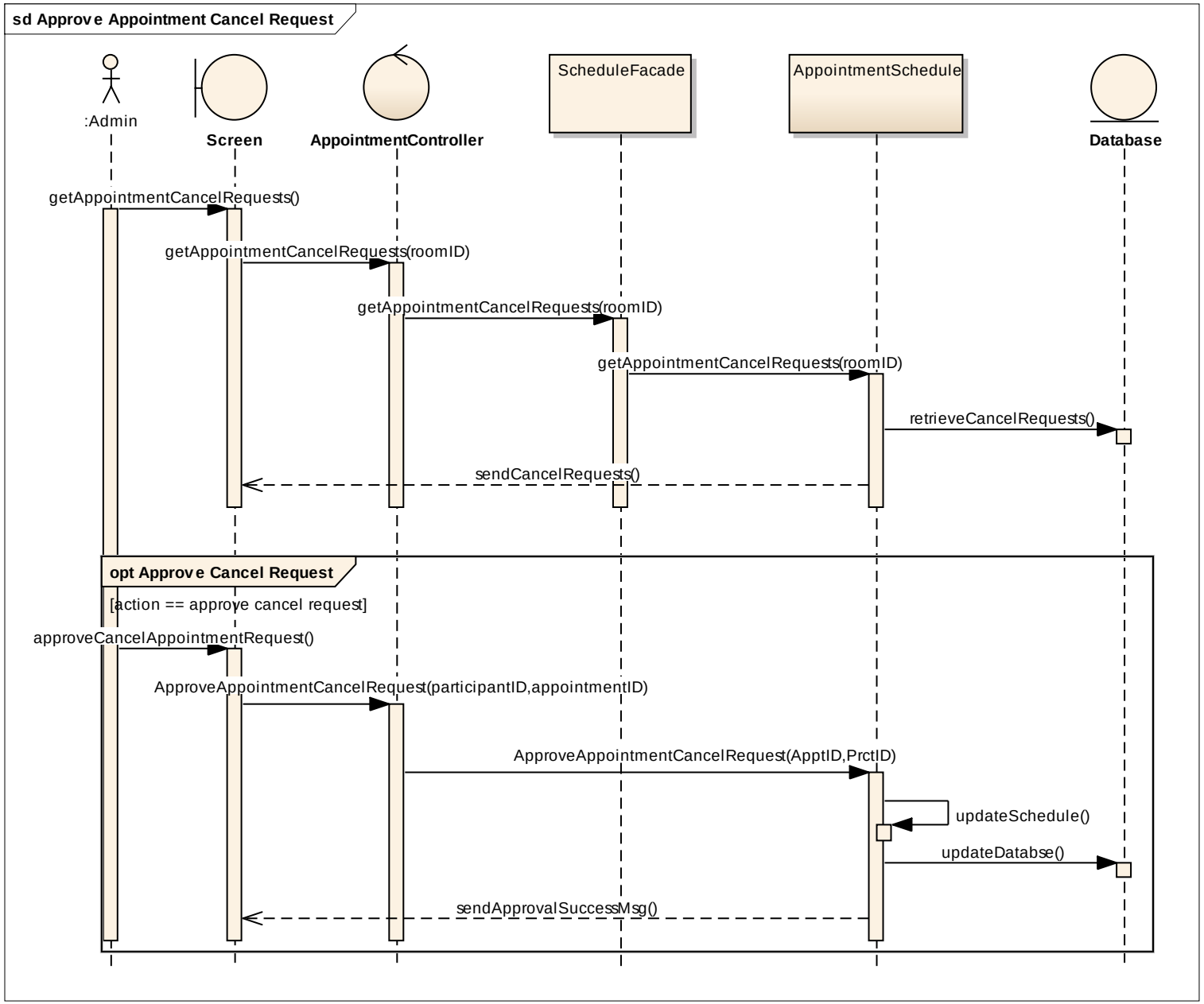

sd propose a meeting
:Admin
Screen
ScheduleController
ScheduleFacade
MeetingSchedule
DaySchedule
Database
proposeMeeting()
proposeMeeting(roomID,dateFrom,dateTo,name,duration)
proposeMeeting(roomID,dateFrom,dateTo,name,duration)
CreateMeeting(roomID,dateFrom,dateTo,name,duration)
getDayScheduleInstance(dateFrom,dateTo,name,duration)
storePropsoedMeeting()
sendProposedMeetingToAll()
sendProposedMeetingSuccess() :proposedMeetingID

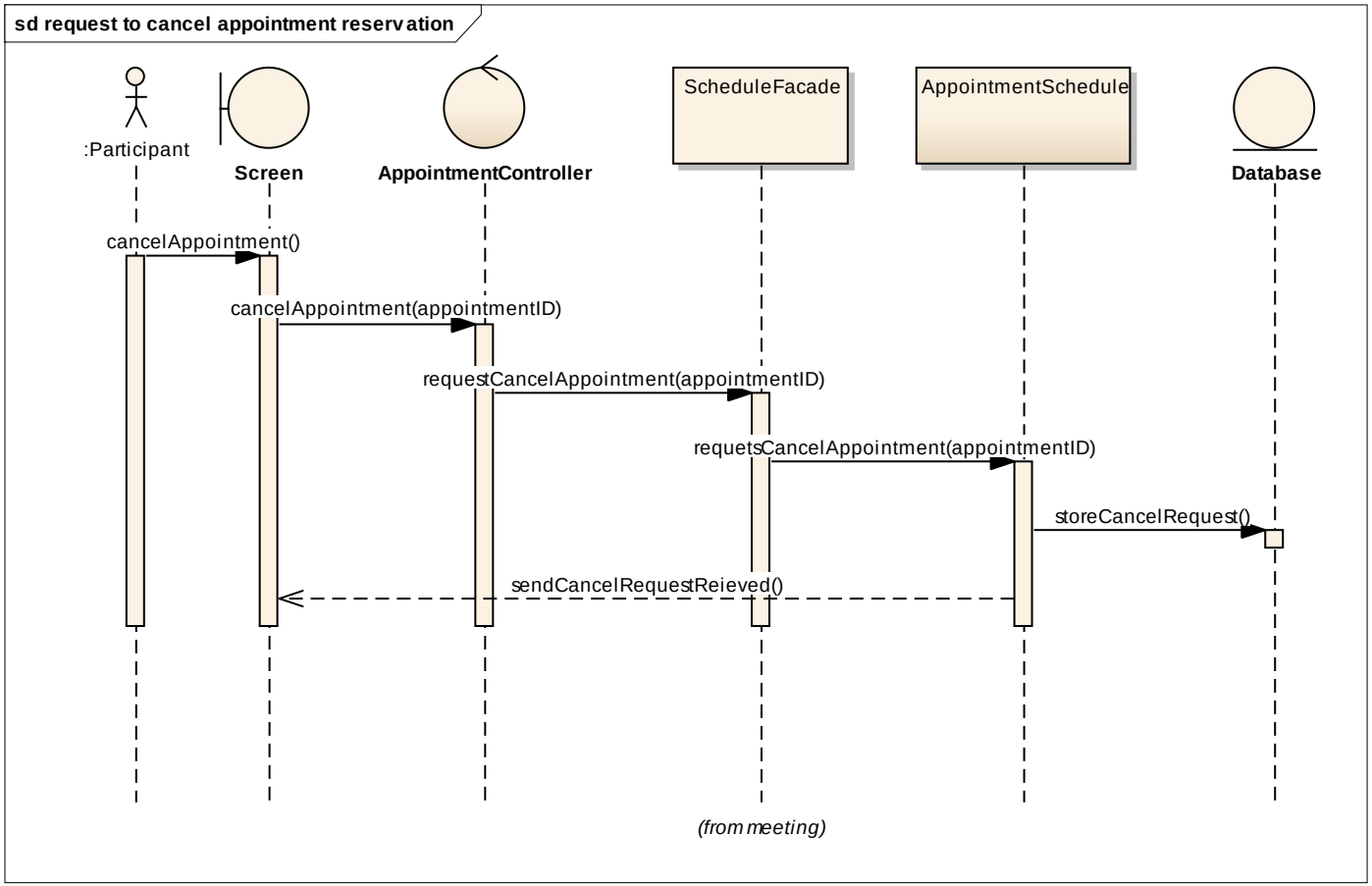

sd confirm meeting
:Participant
Screen
ScheduleController
ScheduleFacade
MeetingSchedule
DaySchedule
Database
getProposedMeetings()
getProposedMeetings(roomID)
getProposedMeetings(roomID)
getProposedMeetings(roomID)
retrieveProposedMeetings()
sendPropsoedMeetings() :proposedMeetings[]
fillBusyHours(hours[])
sendBusyHours()
confirmMeeting(roomID,proposedMeetingID,busyHours)
updateMeeting(roomID,proposedMeetingID,busyHours)
updateMeeting(roomID,proposedMeetingID,busyHours)
storeMeetingUpdate()
showConfirmationReceivedMsg()


sd Approve Proposed Meeting
:Admin
Screen
ScheduleController
ScheduleFacade
MeetingSchedule
Participant
Database
getSuggestedMeetings()
getSuggestedMeetings(porposedMeetingID)
getSuggestedMeetings(porposedMeetingID)
calculateSuggestedMeetings(porposedMeetingID)
retrieveProposedMeeting()
sendSuggestedMeetings()
approveSuggestedMeeting()
approveSuggestedMeeting(suggestedMeetingID)
approveSuggestedMeeting(meetingID)
approveSuggestedMeeting(suggestedMeetingID)
storeApprovedMeeting()
sendApprovalSucessMsg()

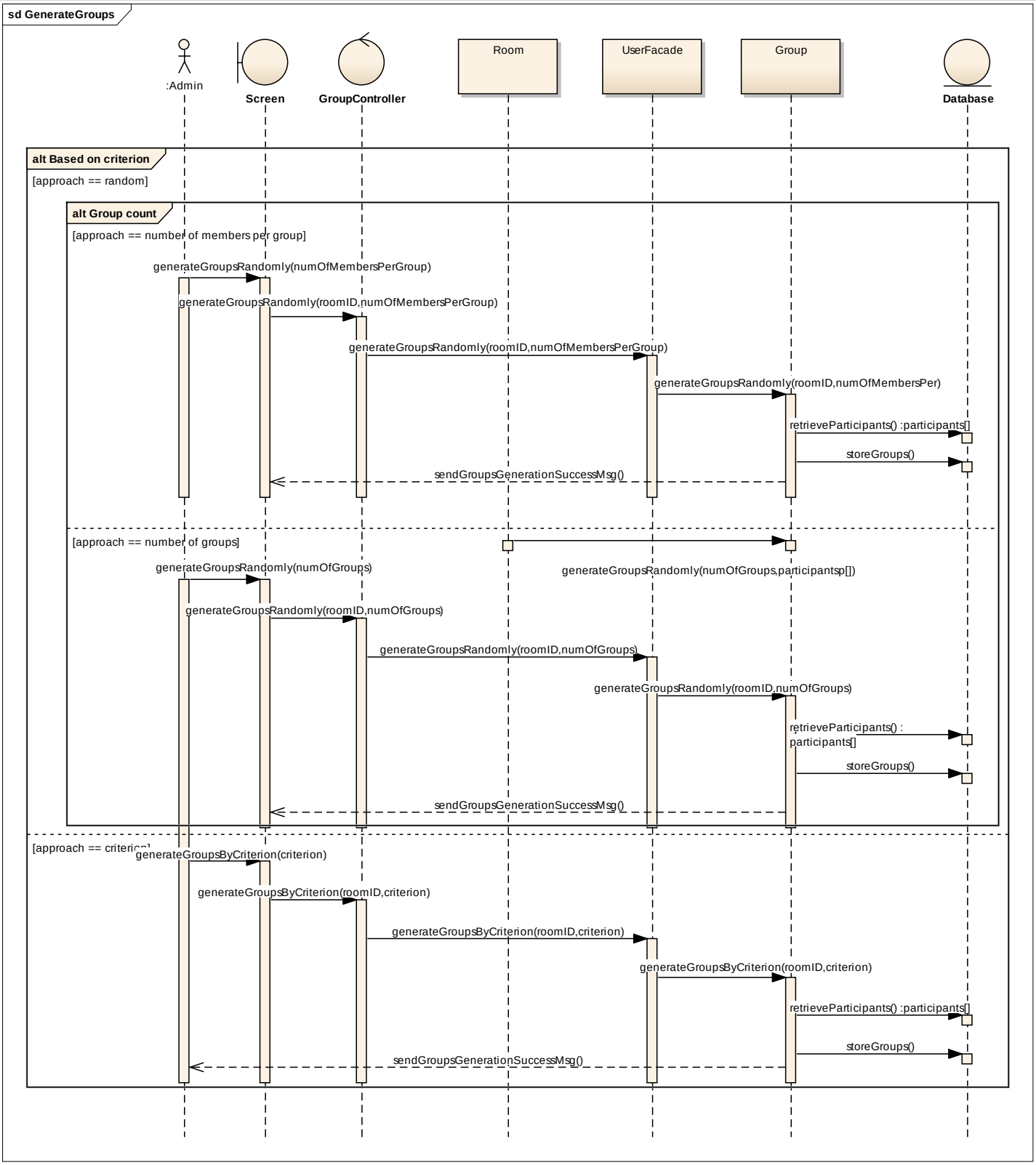

sd reserve time slot
:Participant
Screen
AppointmentController
Room
ScheduleFacade
AppointmentSchedule
Database
getAvailableAppointments()
retrieveAppointments()
retrieveAppointmentsSchedule()
retrieveAppointments()
sendAppointemntsAvailable()
reserveAppointment()
reserveAppointment(timeslotID)
reserveAppointment(timeslotID)
reserveAppointment(timeslotID)
storeAppointmentReservation()
sendReservationSuccessMsg()
(from meeting)

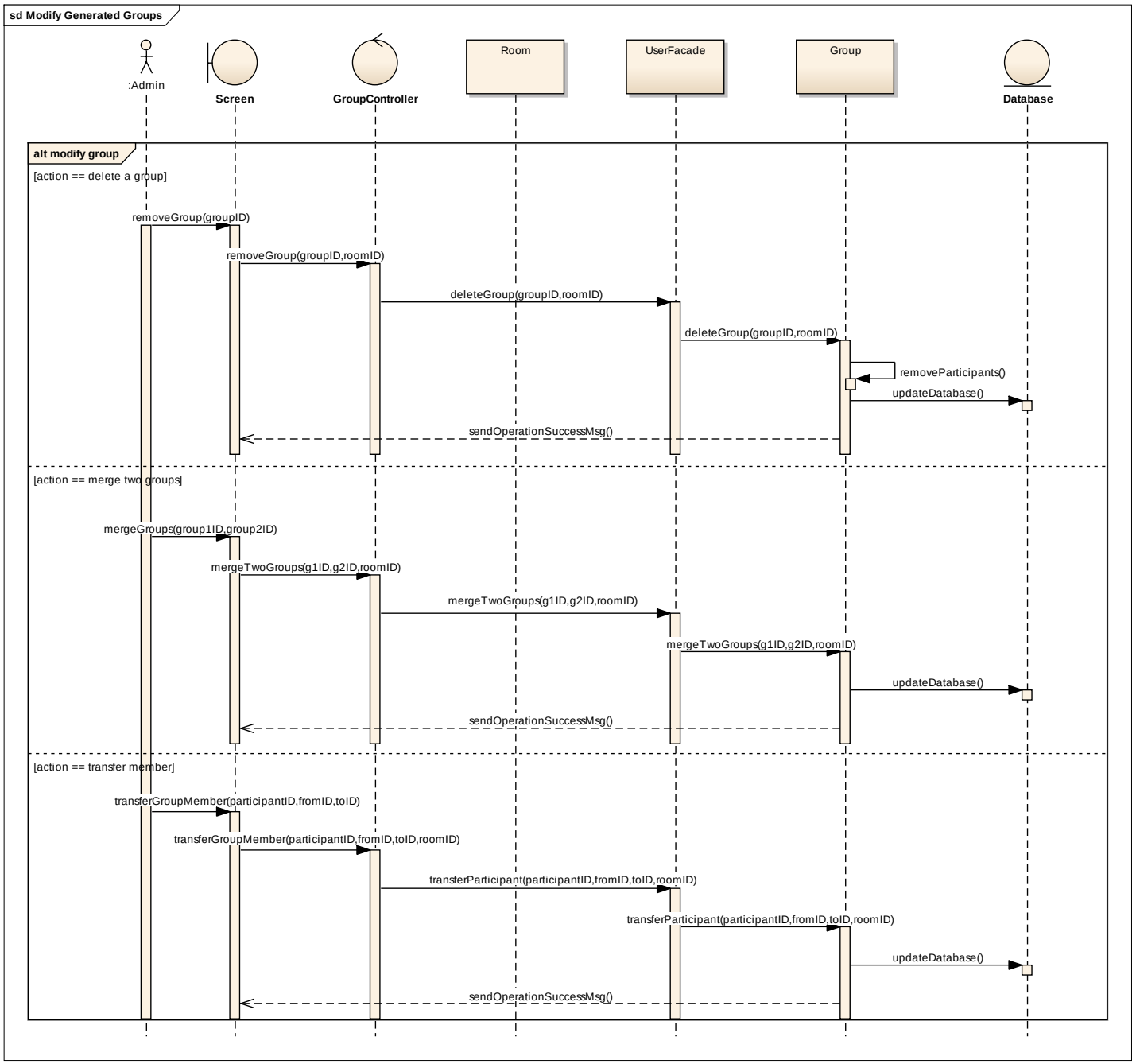

sd update schedule
:Admin
Screen
ScheduleController
ScheduleFacade
AppointmentSchedule
Database
opt Edit Schedule
[action == edit schedule]
editAppointment()
editAppointment(oldTimeslotID, newTimeslot)
changeAppointment(oldTimeslotID, newTimeslot)
changeTimeslot(slotID, newTimeslot)
updateDatabase()
sendOperationSuccessMsg()
opt delete appointment
[action == delete appointment]
deleteAppointment()
removeTimeslot(slotID)
deleteAppointment(slotID)
deleteTimeslot(slotID)
updateDatabase()
sendOperationSuccessMsg()


sd Approve Appointment Cancel Request
:Admin
Screen
AppointmentController
ScheduleFacade
AppointmentSchedule
Database
getAppointmentCancelRequests()
getAppointmentCancelRequests(roomID)
getAppointmentCancelRequests(roomID)
getAppointmentCancelRequests(roomID)
retrieveCancelRequests()
sendCancelRequests()
opt Approve Cancel Request
[action == approve cancel request]
approveCancelAppointmentRequest()
ApproveAppointmentCancelRequest(participantID,appointmentID)
ApproveAppointmentCancelRequest(ApptID,PrctID)
updateSchedule()
updateDatabse()
sendApprovalSuccessMsg()

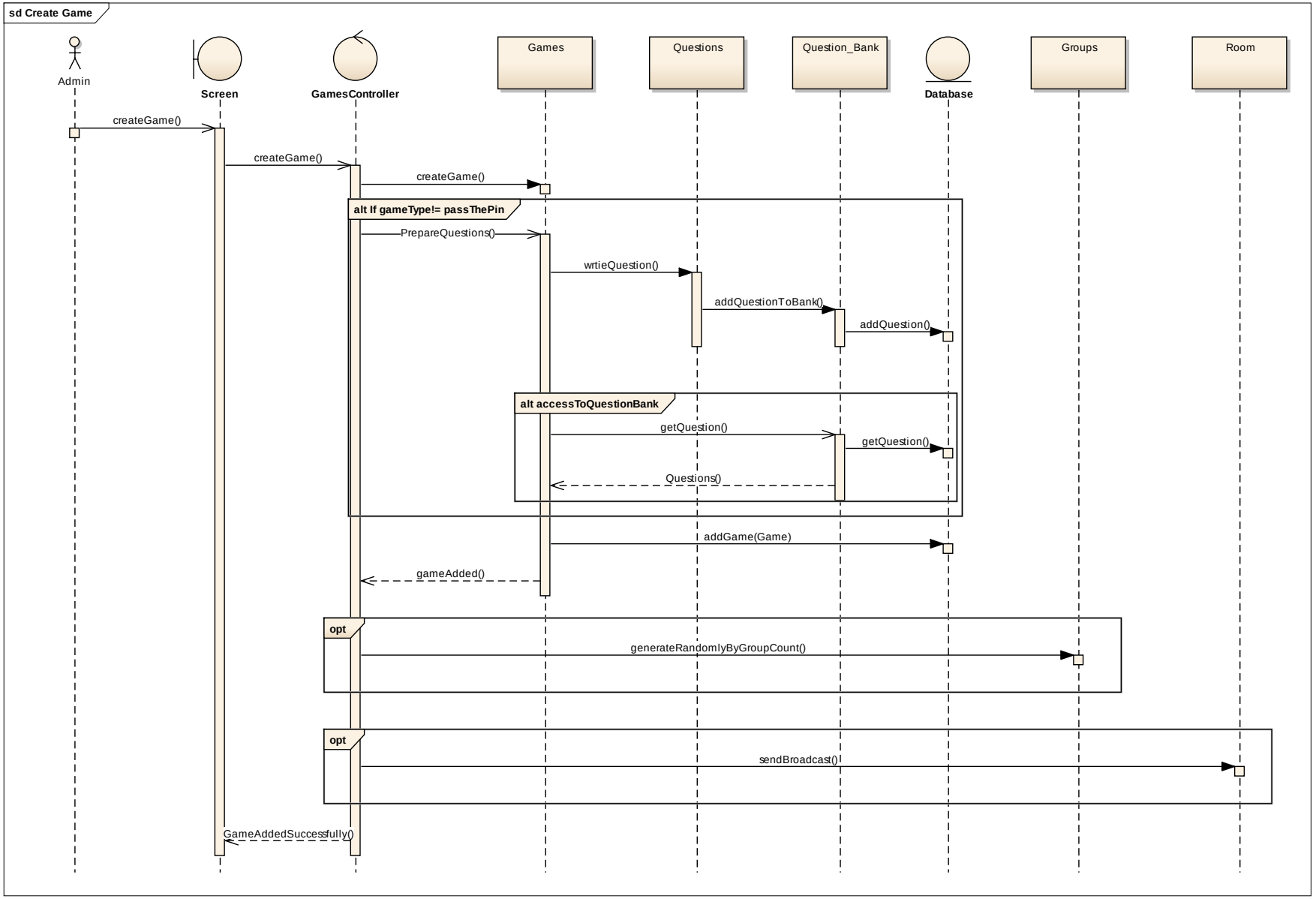

sd request to cancel appointment reservation
:Participant
Screen
AppointmentController
ScheduleFacade
AppointmentSchedule
Database
cancelAppointment()
cancelAppointment(appointmentID)
requestCancelAppointment(appointmentID)
requetsCancelAppointment(appointmentID)
storeCancelRequest()
sendCancelRequestReieved()
(from meeting)

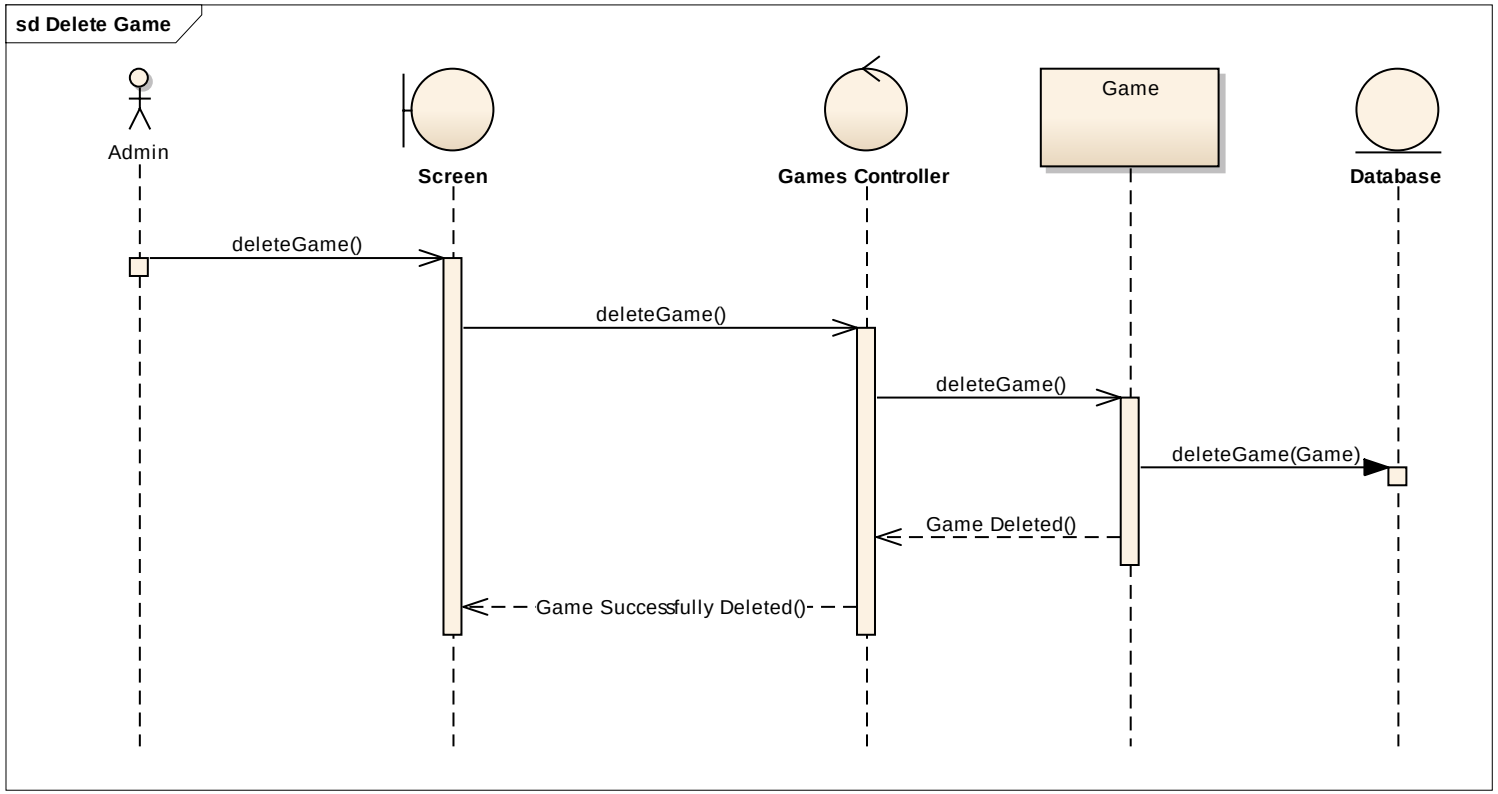
Group Generation

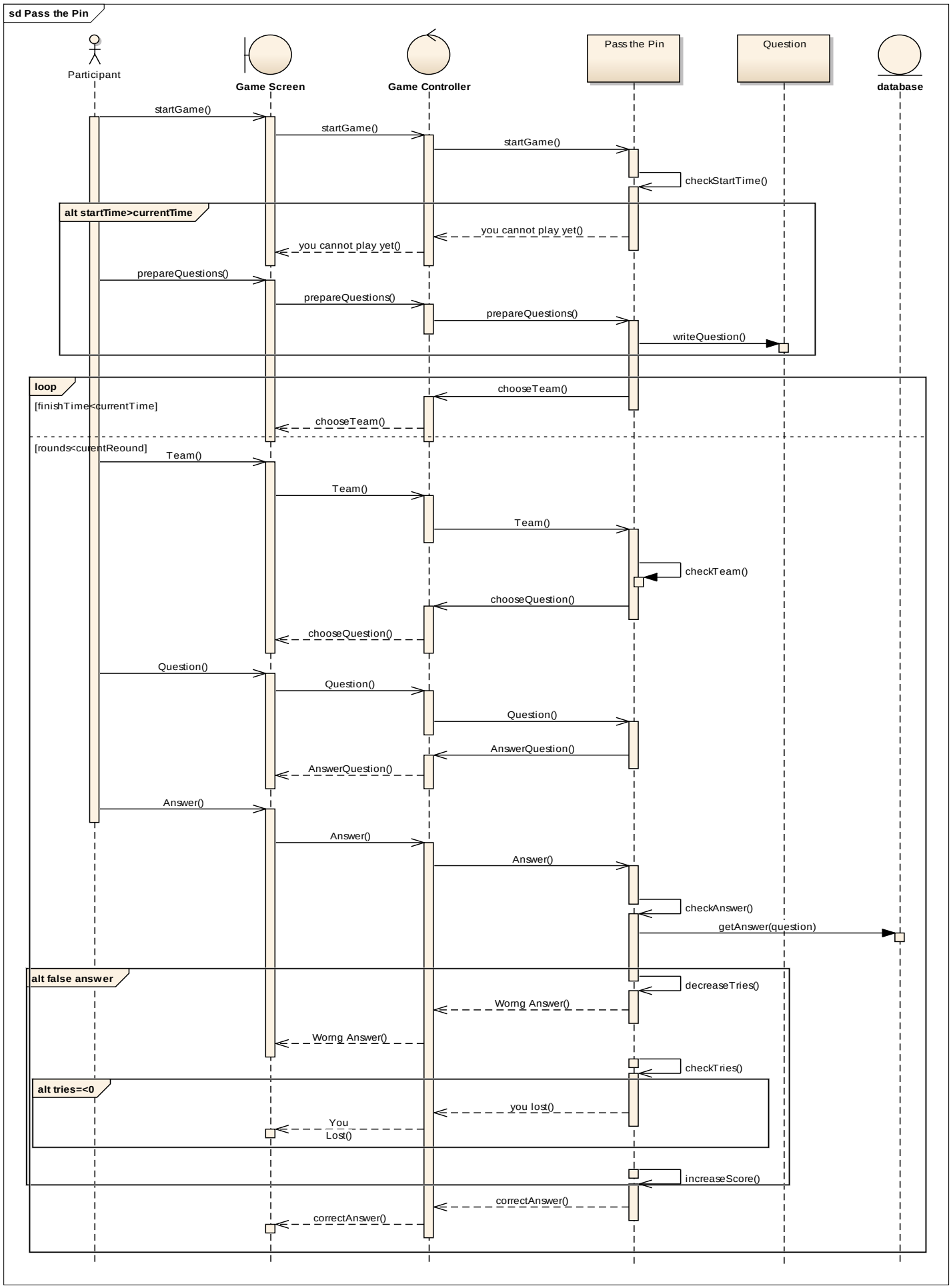


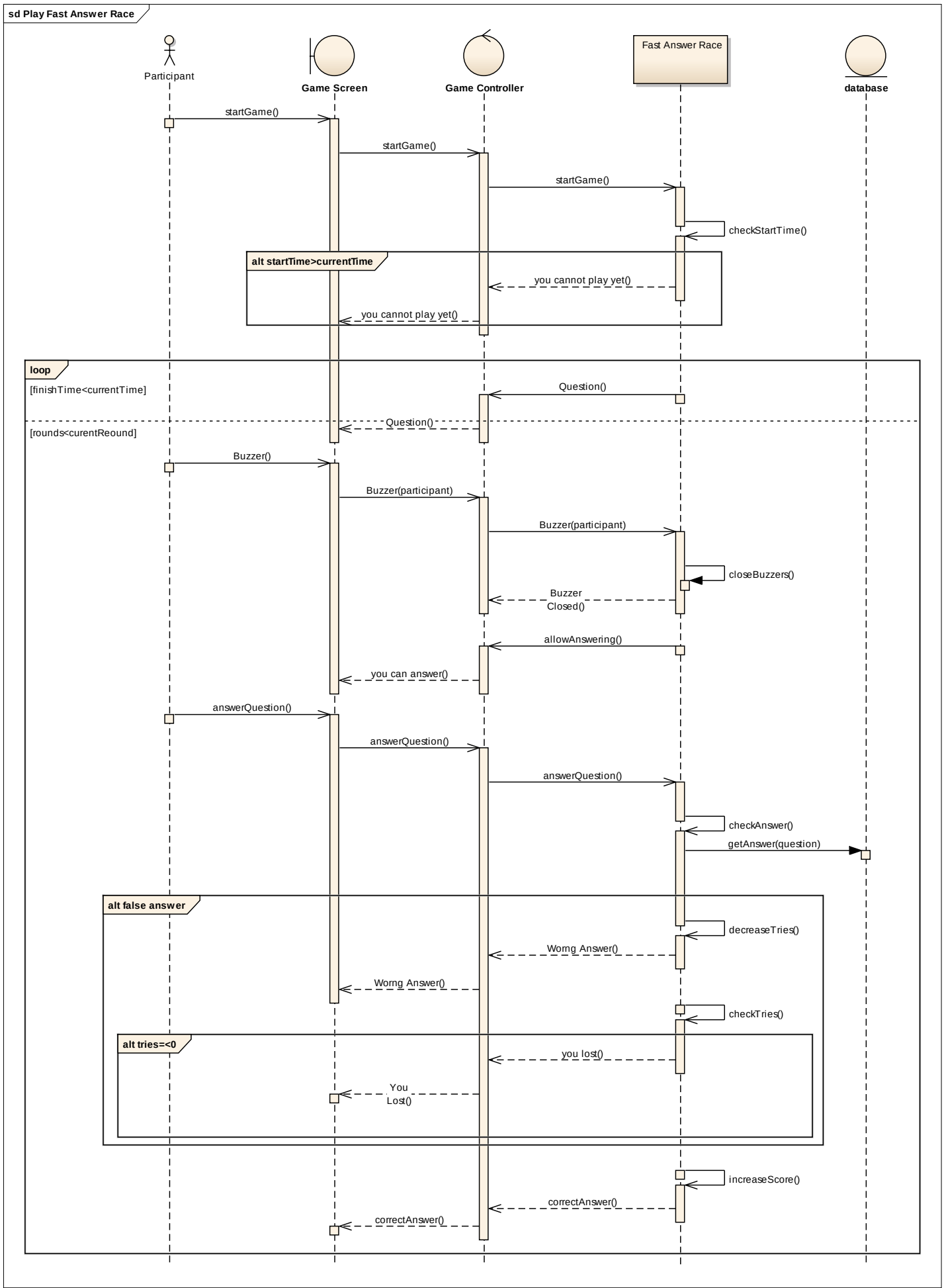

sd Modify Generated Groups
:Admin
Screen
GroupController
Room
UserFacade
Group
Database
alt modify group
[action == delete a group]
removeGroup(groupID)
removeGroup(groupID,roomID)
deleteGroup(groupID,roomID)
deleteGroup(groupID,roomID)
removeParticipants()
updateDatabase()
sendOperationSuccessMsg()
[action == merge two groups]
mergeGroups(group1ID,group2ID)
mergeTwoGroups(g1ID,g2ID,roomID)
mergeTwoGroups(g1ID,g2ID,roomID)
mergeTwoGroups(g1ID,g2ID,roomID)
updateDatabase()
sendOperationSuccessMsg()
[action == transfer member]
transferGroupMember(participantID,fromID,toID)
transferGroupMember(participantID,fromID,toID,roomID)
transferParticipant(participantID,fromID,toID,roomID)
transferParticipant(participantID,fromID,toID,roomID)
updateDatabase()
sendOperationSuccessMsg()

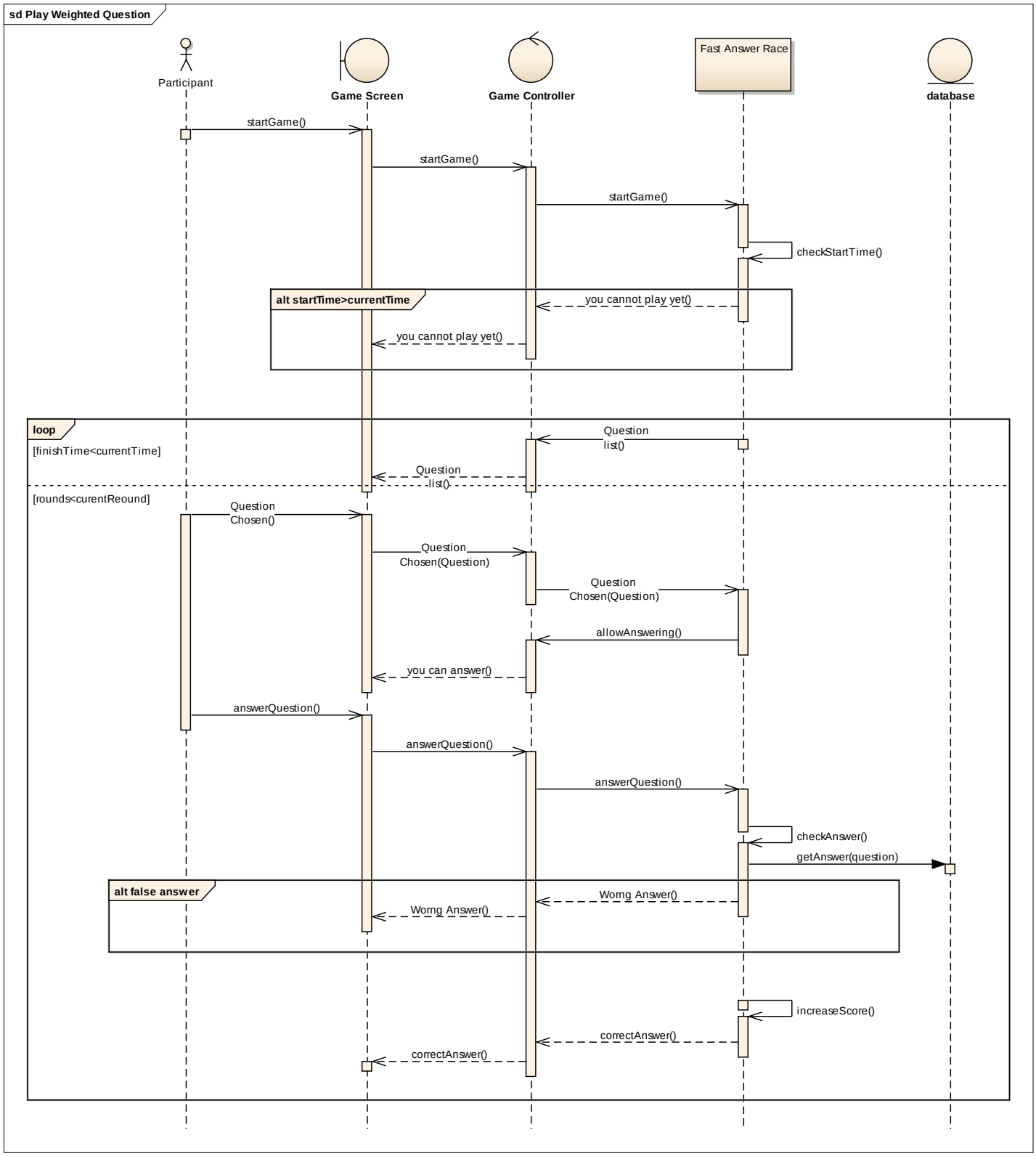
Games and Quizzes

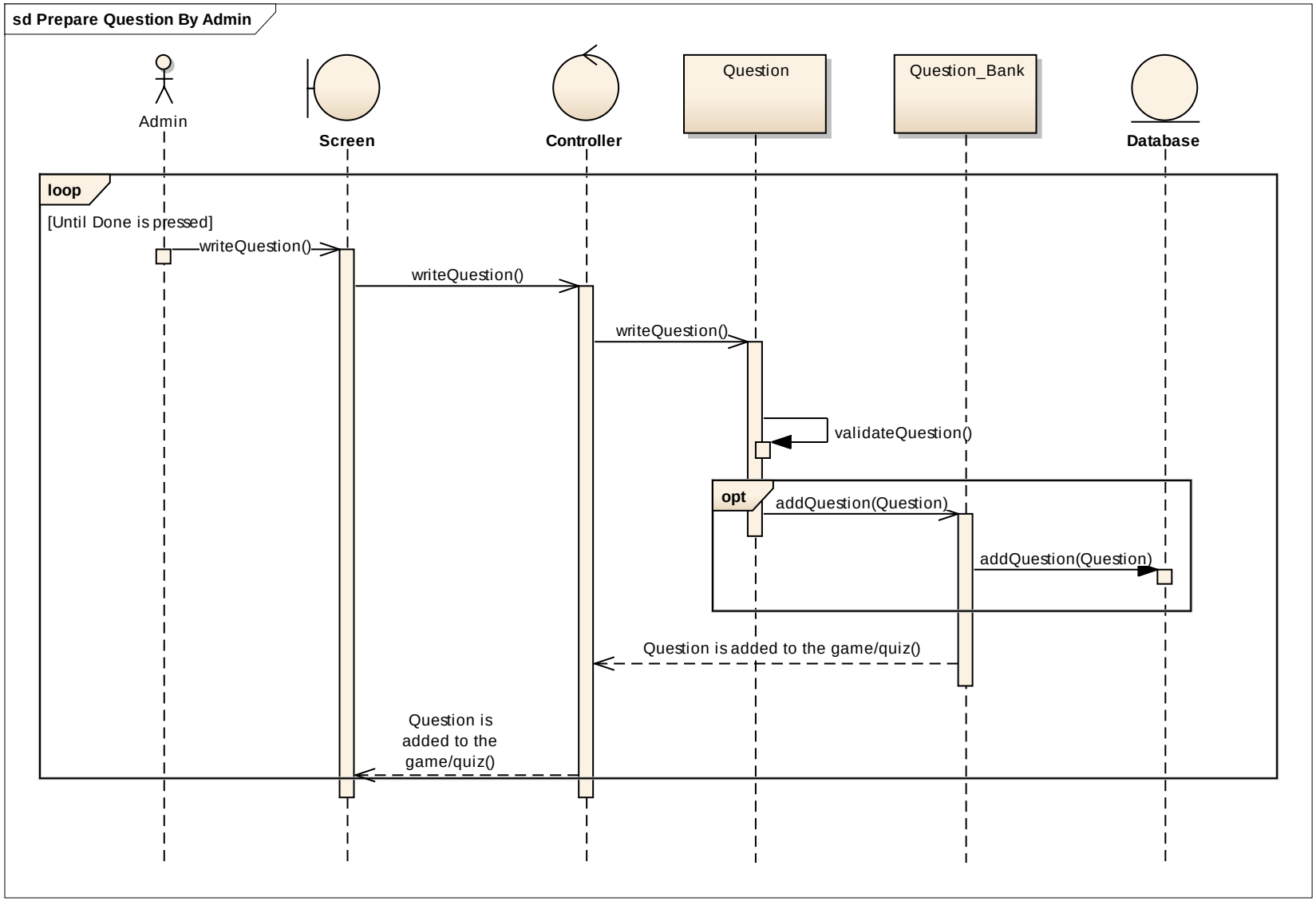


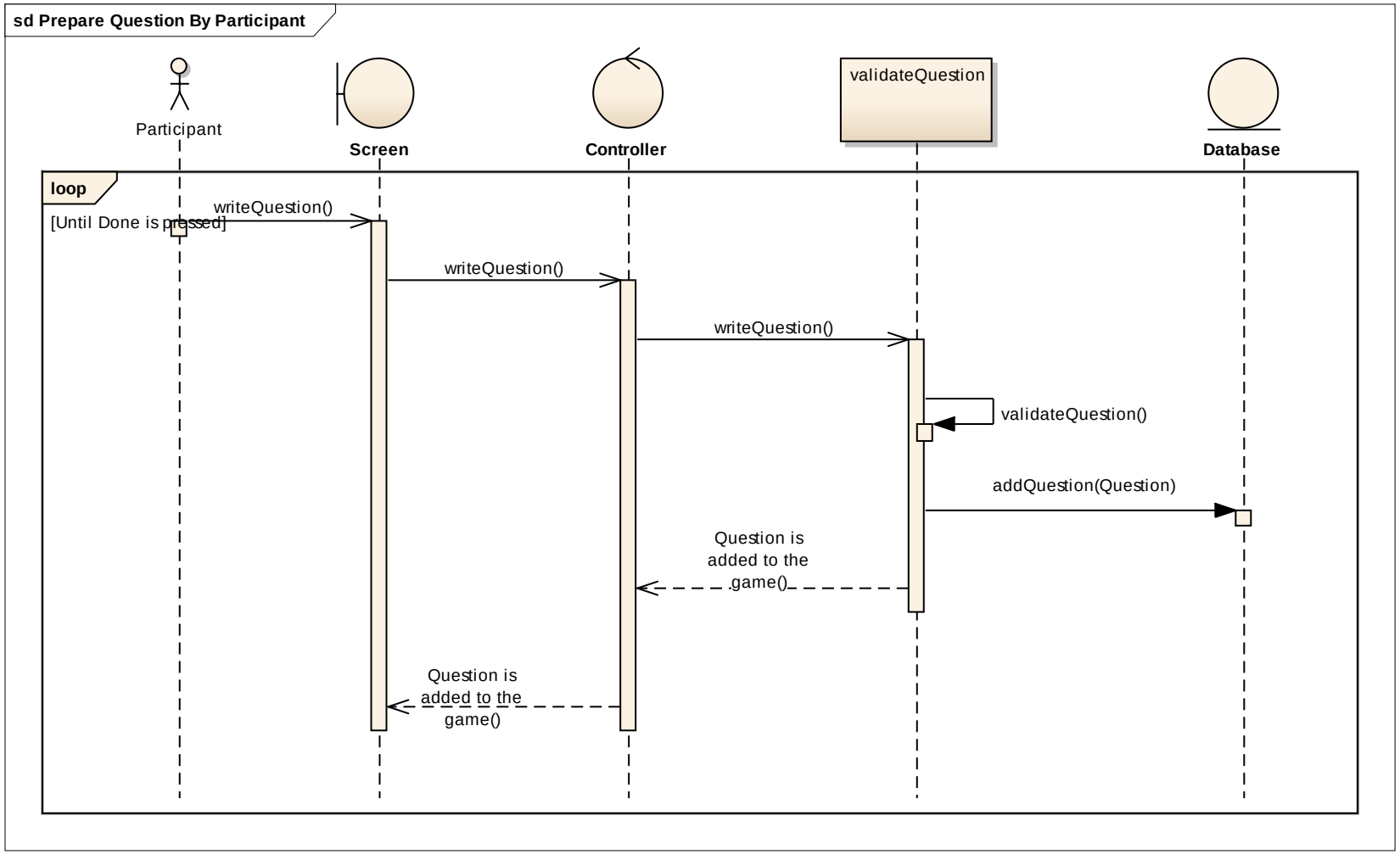

sd Delete Game
Admin
Screen
Games Controller
Game
Database
deleteGame()
deleteGame()
deleteGame()
deleteGame(Game)
Game Deleted()
Game Successfully Deleted()

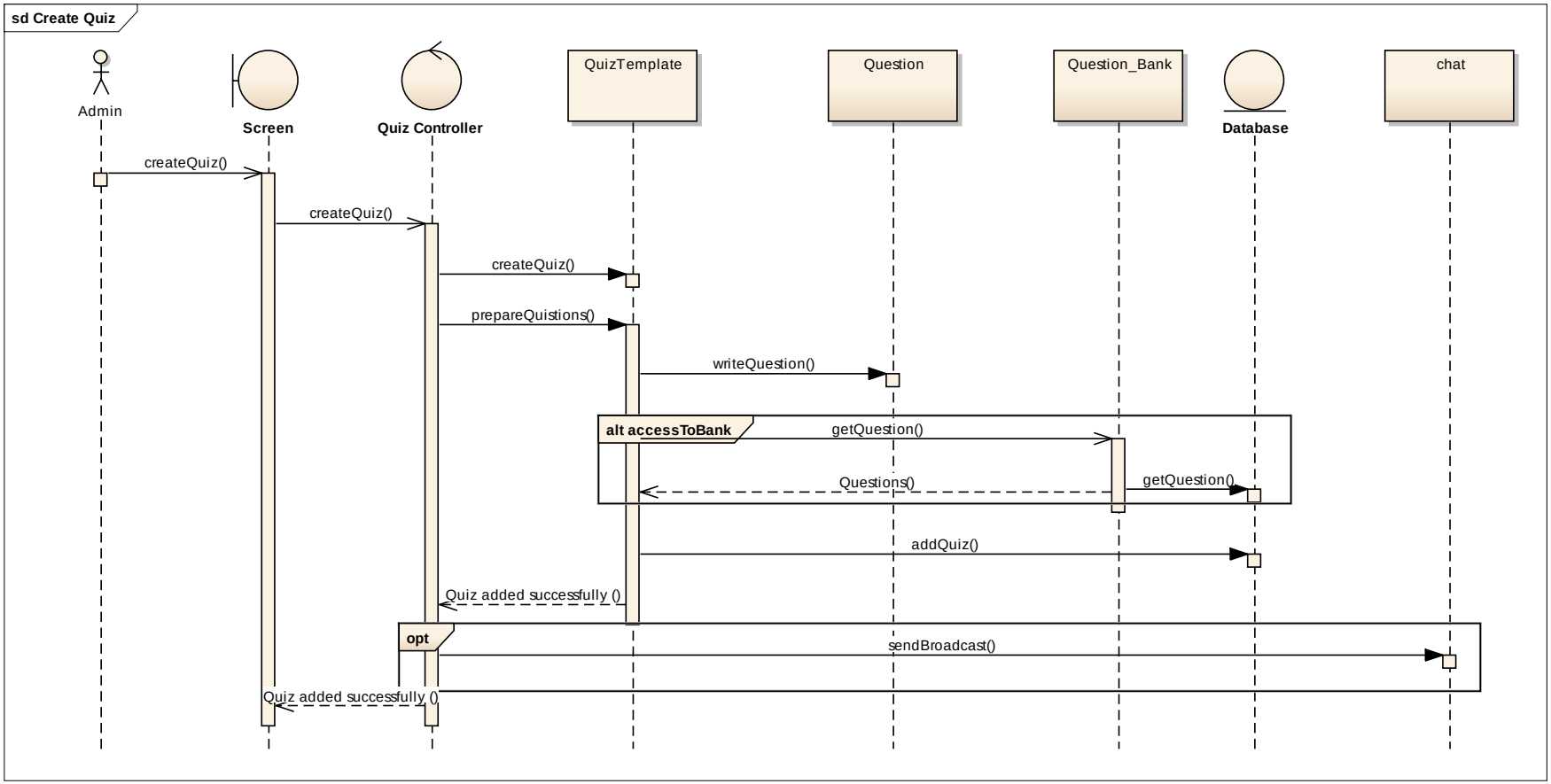

sd Pass the Pin
Participant
Game Screen
Game Controller
Pass the Pin
Question
database
startGame()
startGame()
startGame()
checkStartTime()
alt startTime>currentTime
you cannot play yet()
you cannot play yet()
prepareQuestions()
prepareQuestions()
prepareQuestions()
writeQuestion()
loop
chooseTeam()
[finishTime<currentTime]
chooseTeam()
[rounds<curentReound]
Team()
Team()
Team()
checkTeam()
chooseQuestion()
chooseQuestion()
Question()
Question()
Question()
AnswerQuestion()
AnswerQuestion()
Answer()
Answer()
Answer()
checkAnswer()
getAnswer(question)
alt false answer
decreaseTries()
Worng Answer()
Worng Answer()
checkTries()
alt tries=<0
you lost()
You
Lost()
increaseScore()
correctAnswer()
correctAnswer()

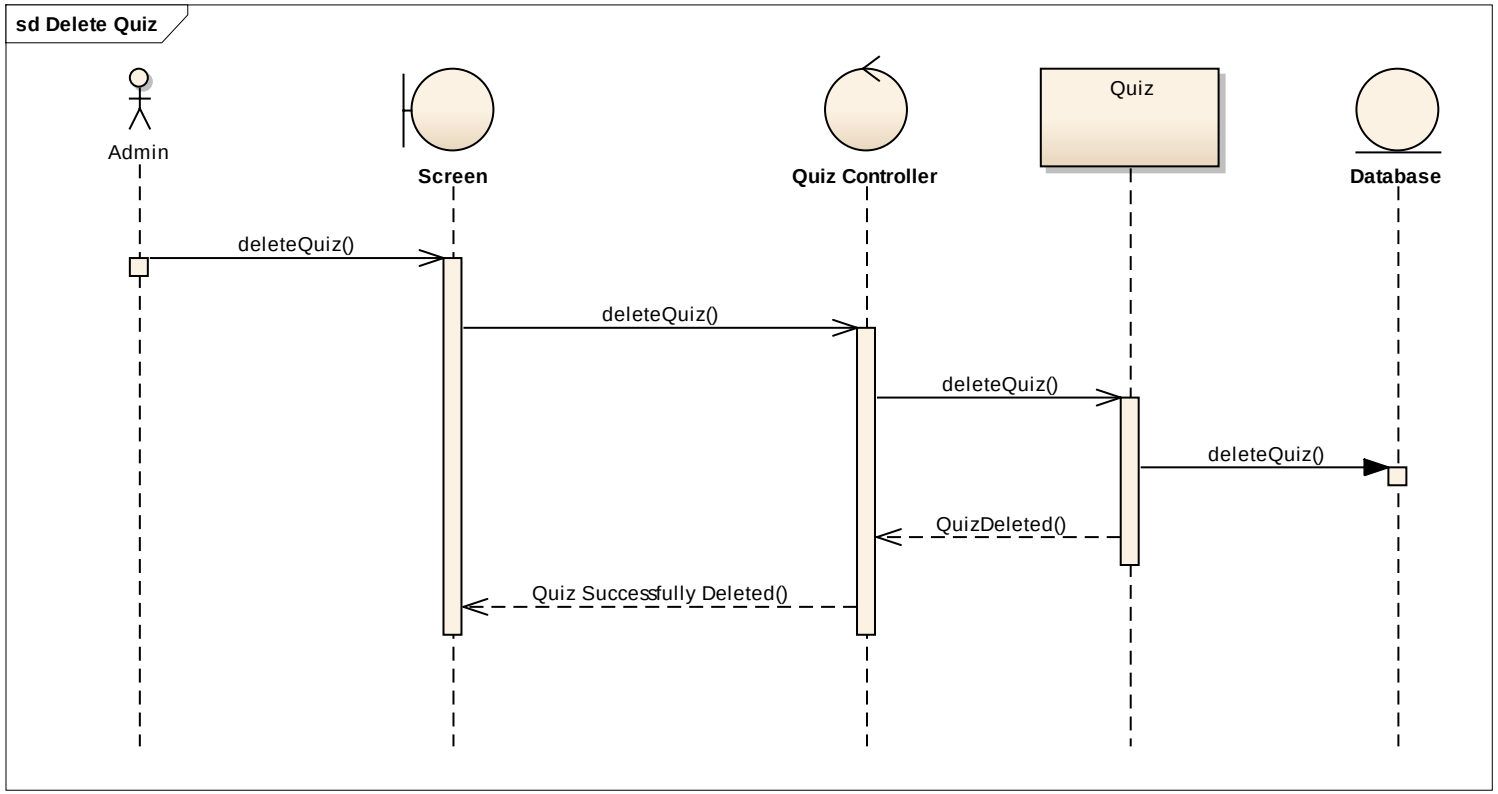

sd Play Fast Answer Race
Participant
Game Screen
Game Controller
Fast Answer Race
database
startGame()
startGame()
startGame()
checkStartTime()
alt startTime>currentTime
you cannot play yet()
you cannot play yet()
loop
[finishTime<currentTime]
Question()
Question()
[rounds<curentReound]
Buzzer()
Buzzer(participant)
Buzzer(participant)
closeBuzzers()
Buzzer Closed()
allowAnswering()
you can answer()
answerQuestion()
answerQuestion()
answerQuestion()
checkAnswer()
getAnswer(question)
alt false answer
decreaseTries()
Worng Answer()
Worng Answer()
checkTries()
alt tries=<0
you lost()
You Lost()
increaseScore()
correctAnswer()
correctAnswer()

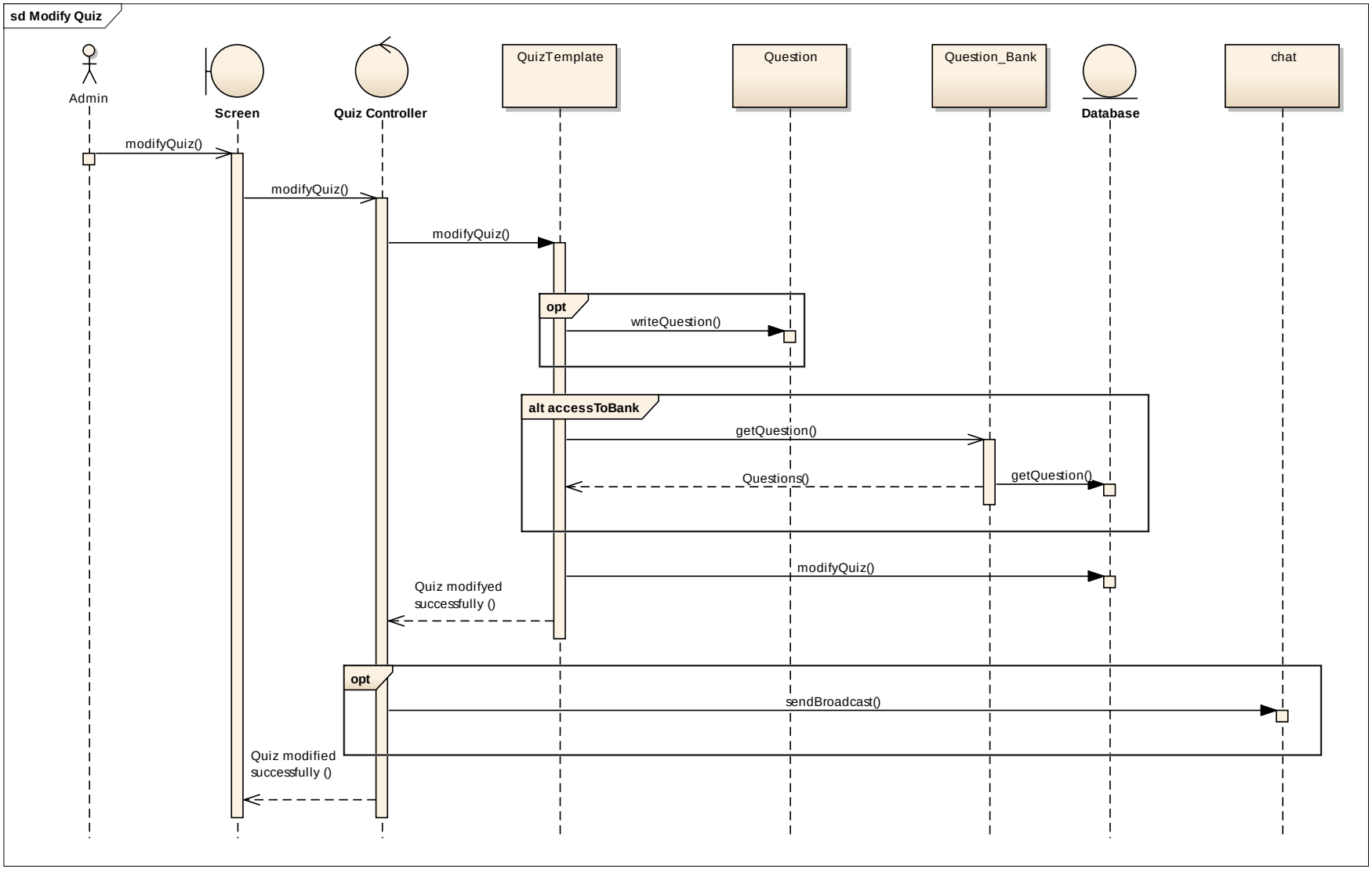

sd Play Weighted Question
Participant
Game Screen
Game Controller
Fast Answer Race
database
startGame()
startGame()
startGame()
checkStartTime()
alt startTime>currentTime
you cannot play yet()
you cannot play yet()
loop
[finishTime<currentTime]
Question list()
Question list()
[rounds<curentReound]
Question Chosen()
Question Chosen(Question)
Question Chosen(Question)
allowAnswering()
you can answer()
answerQuestion()
answerQuestion()
answerQuestion()
checkAnswer()
getAnswer(question)
alt false answer
Worng Answer()
Worng Answer()
increaseScore()
correctAnswer()
correctAnswer()

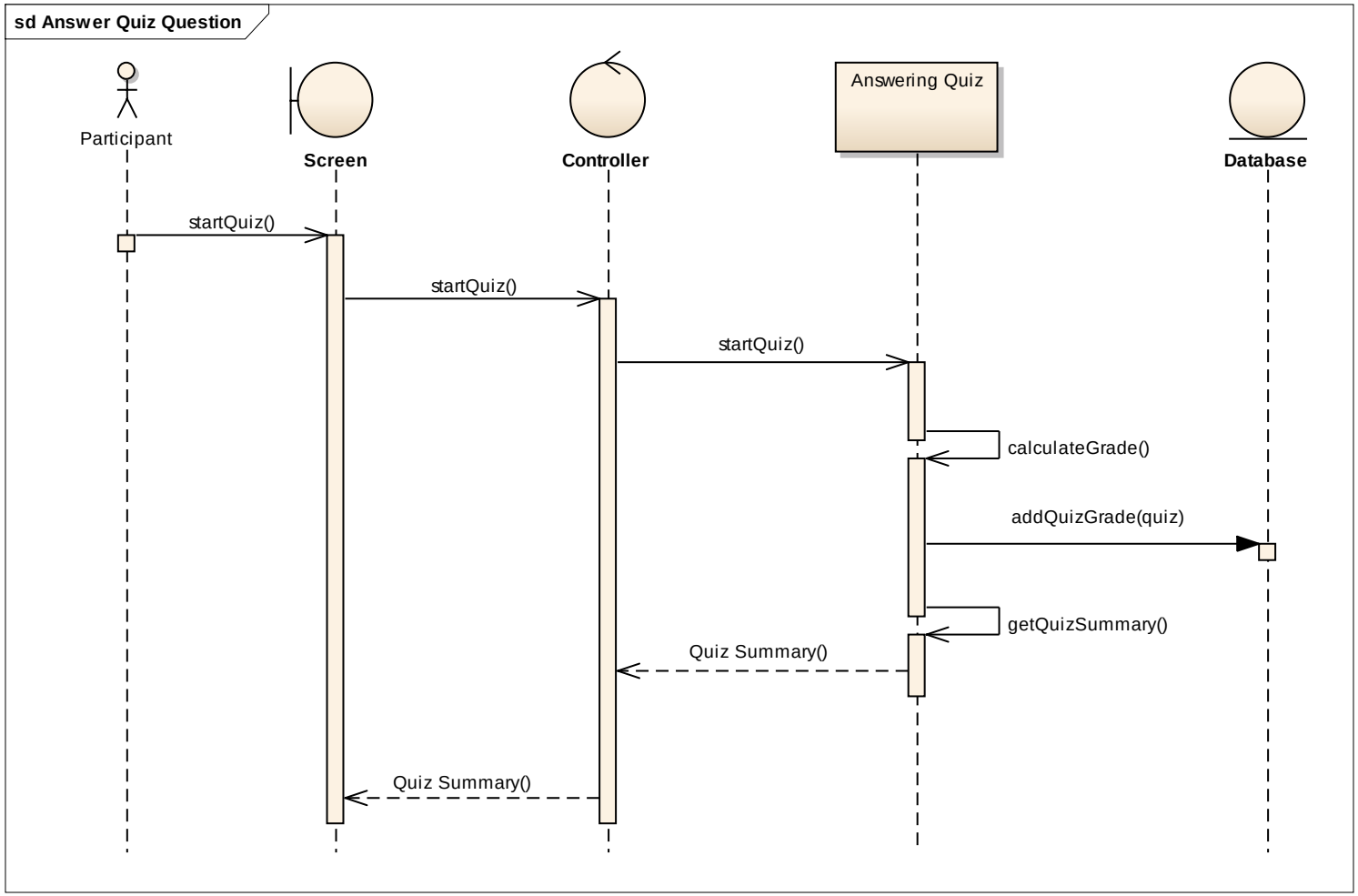

sd Prepare Question By Admin
Admin
Screen
Controller
Question
Question_Bank
Database
loop
[Until Done is pressed]
writeQuestion()
writeQuestion()
writeQuestion()
validateQuestion()
opt
addQuestion(Question)
addQuestion(Question)
Question is added to the game/quiz()
Question is added to the game/quiz()

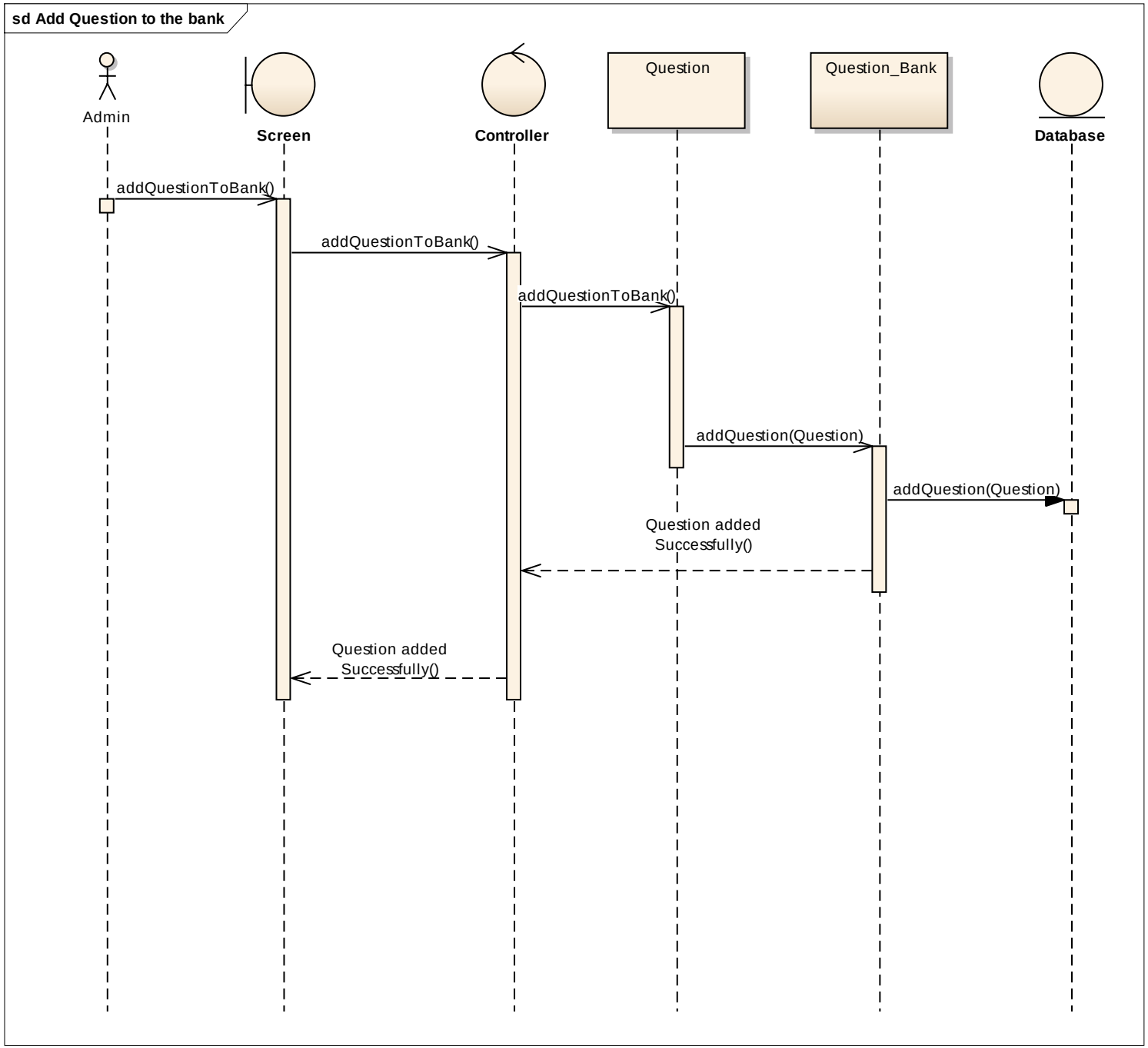

sd Prepare Question By Participant
Participant
Screen
Controller
validateQuestion
Database
loop
[Until Done is pressed]
writeQuestion()
writeQuestion()
writeQuestion()
validateQuestion()
addQuestion(Question)
Question is added to the game()
Question is added to the game()

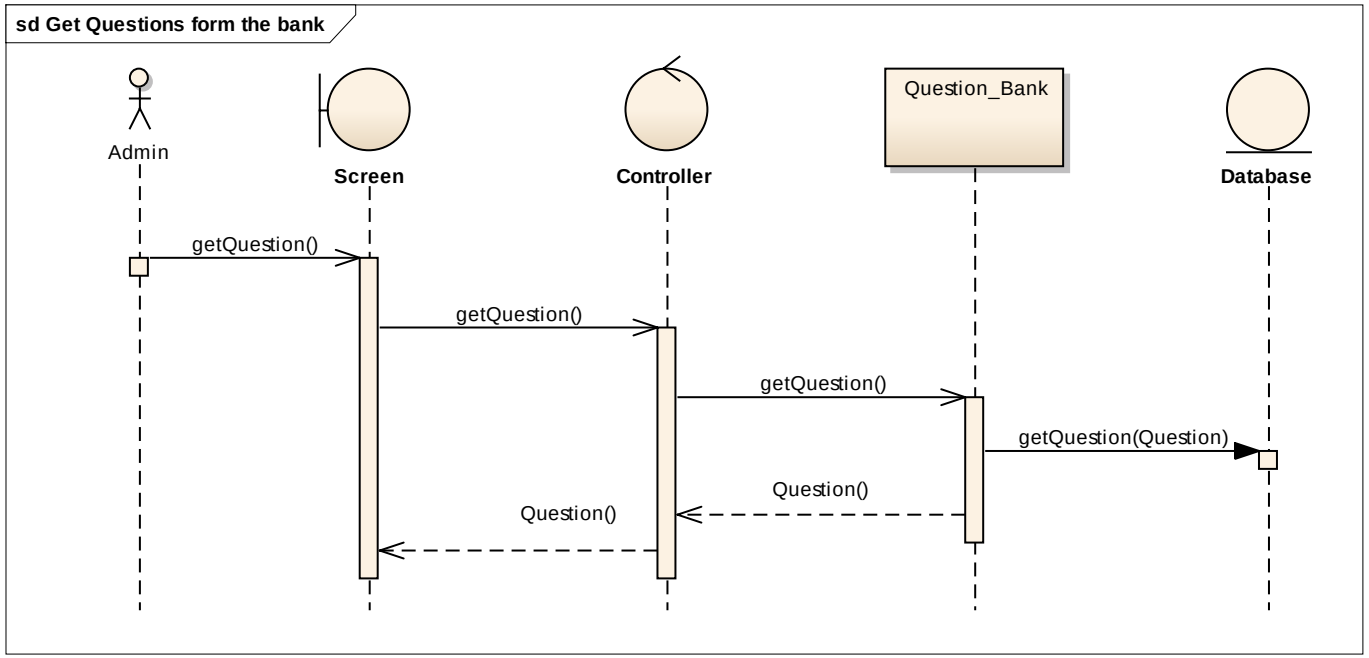

sd Create Quiz
Admin
Screen
Quiz Controller
QuizTemplate
Question
Question_Bank
Database
chat
createQuiz()
createQuiz()
createQuiz()
prepareQuistions()
writeQuestion()
alt accessToBank
getQuestion()
Questions()
getQuestion()
addQuiz()
Quiz added successfully ()
opt
sendBroadcast()
Quiz added successfully ()

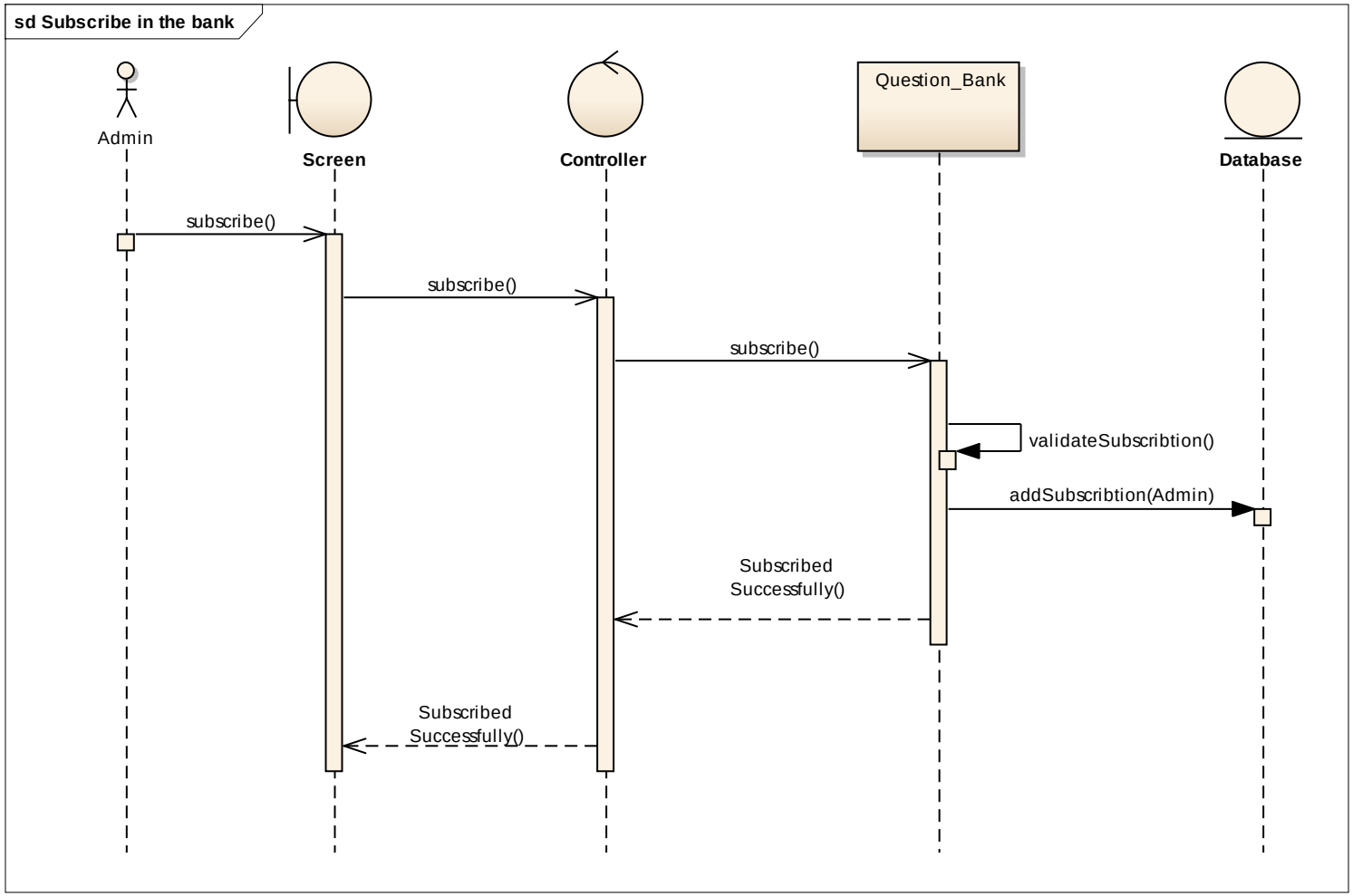

sd Delete Quiz
Admin
Screen
Quiz Controller
Quiz
Database
deleteQuiz()
deleteQuiz()
deleteQuiz()
deleteQuiz()
QuizDeleted()
Quiz Successfully Deleted()

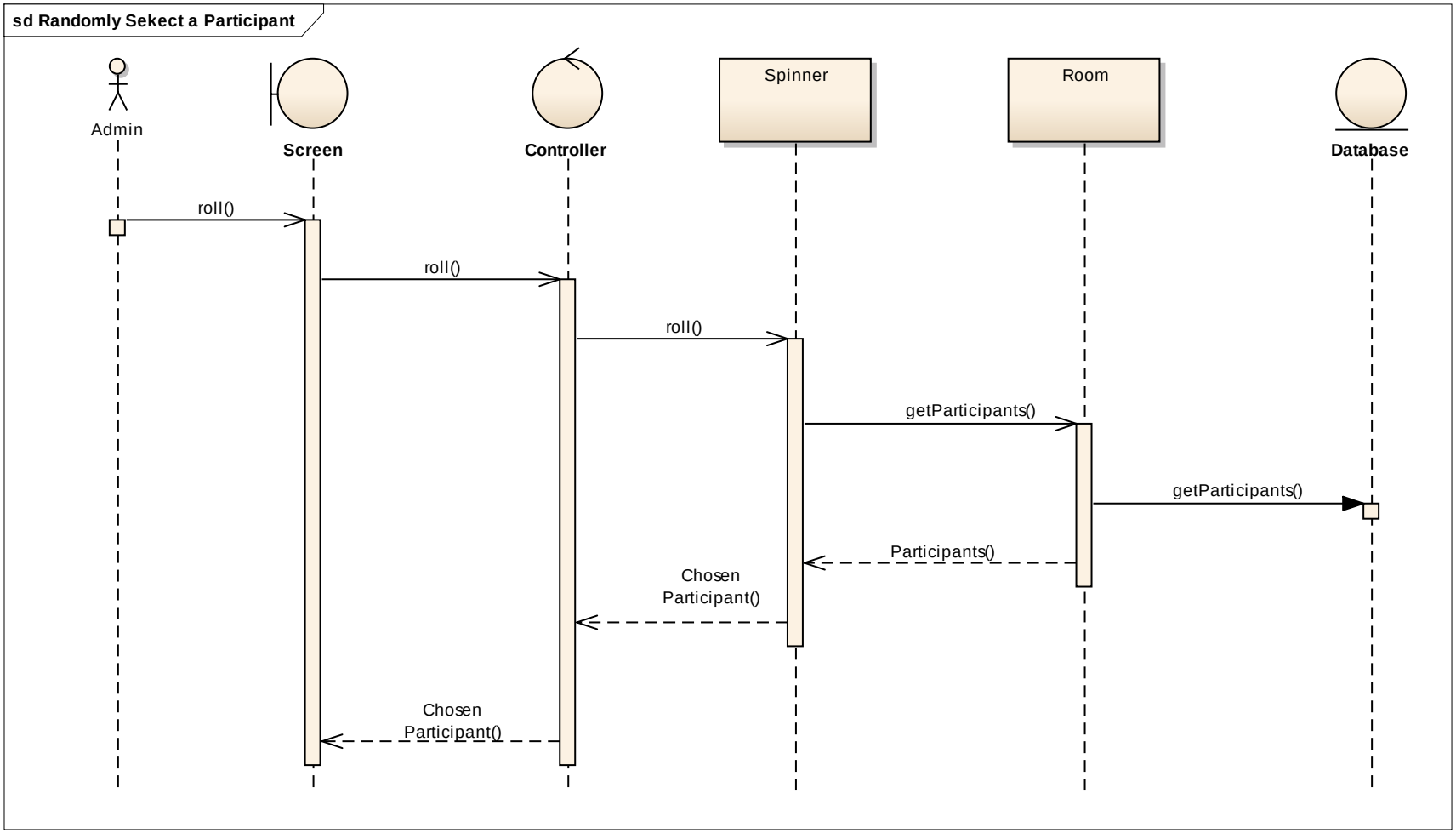

sd Modify Quiz
Admin
Screen
Quiz Controller
QuizTemplate
Question
Question_Bank
Database
chat
modifyQuiz()
modifyQuiz()
modifyQuiz()
opt
writeQuestion()
alt accessToBank
getQuestion()
Questions()
getQuestion()
modifyQuiz()
Quiz modifyed
successfully ()
opt
sendBroadcast()
Quiz modified
successfully ()


sd Answer Quiz Question
Participant
Screen
Controller
Answering Quiz
Database
startQuiz()
startQuiz()
startQuiz()
calculateGrade()
addQuizGrade(quiz)
getQuizSummary()
Quiz Summary()
Quiz Summary()


sd Add Question to the bank
Admin
Screen
Controller
Question
Question_Bank
Database
addQuestionToBank()
addQuestionToBank()
addQuestionToBank()
addQuestion(Question)
addQuestion(Question)
Question added Successfully()
Question added Successfully()

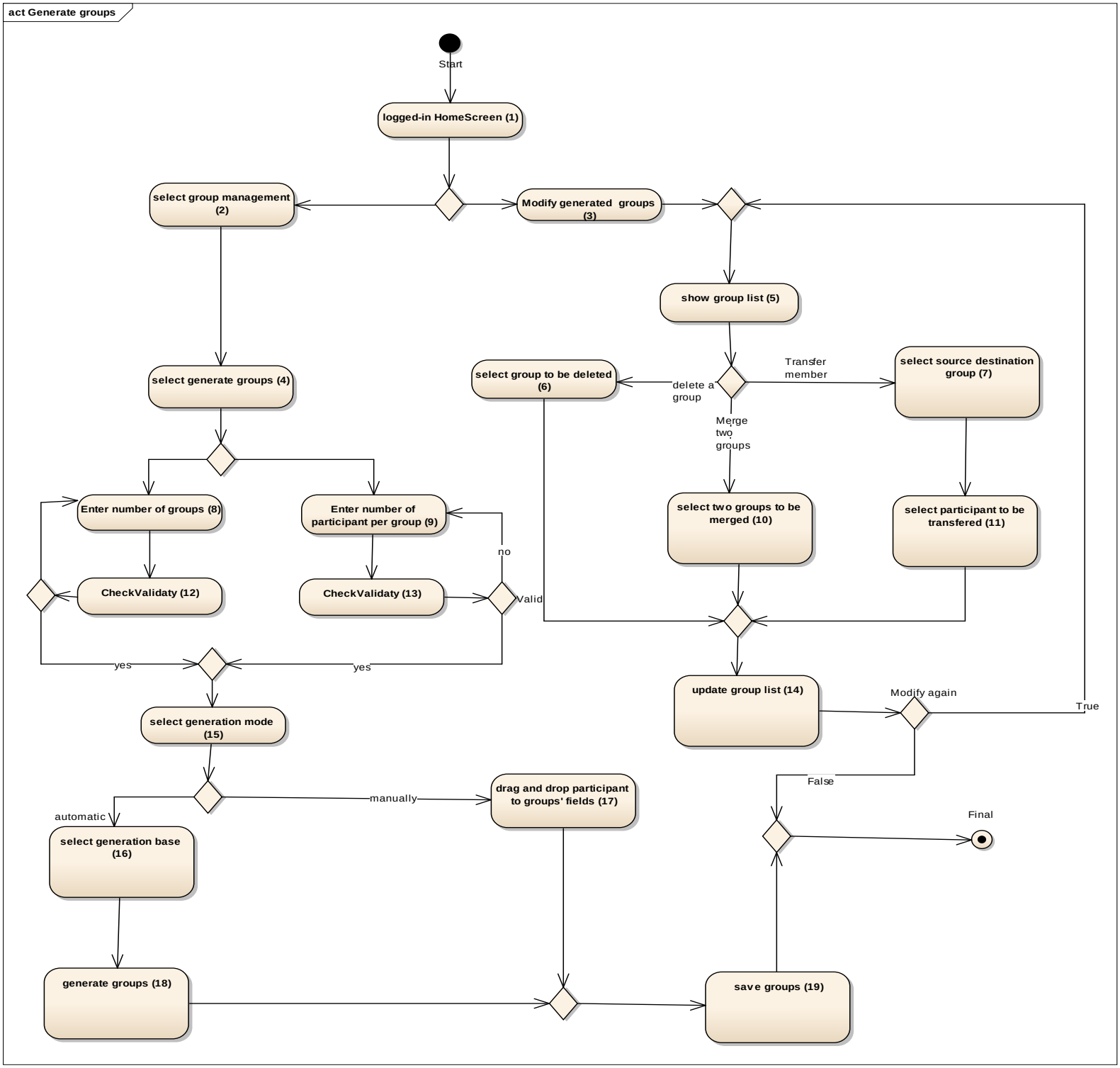

sd Get Questions form the bank
Admin
Screen
Controller
Question_Bank
Database
getQuestion()
getQuestion()
getQuestion()
getQuestion(Question)
Question()
Question()


sd Subscribe in the bank
Admin
Screen
Controller
Question_Bank
Database
subscribe()
subscribe()
subscribe()
validateSubscribtion()
addSubscribtion(Admin)
Subscribed
Successfully()
Subscribed
Successfully()

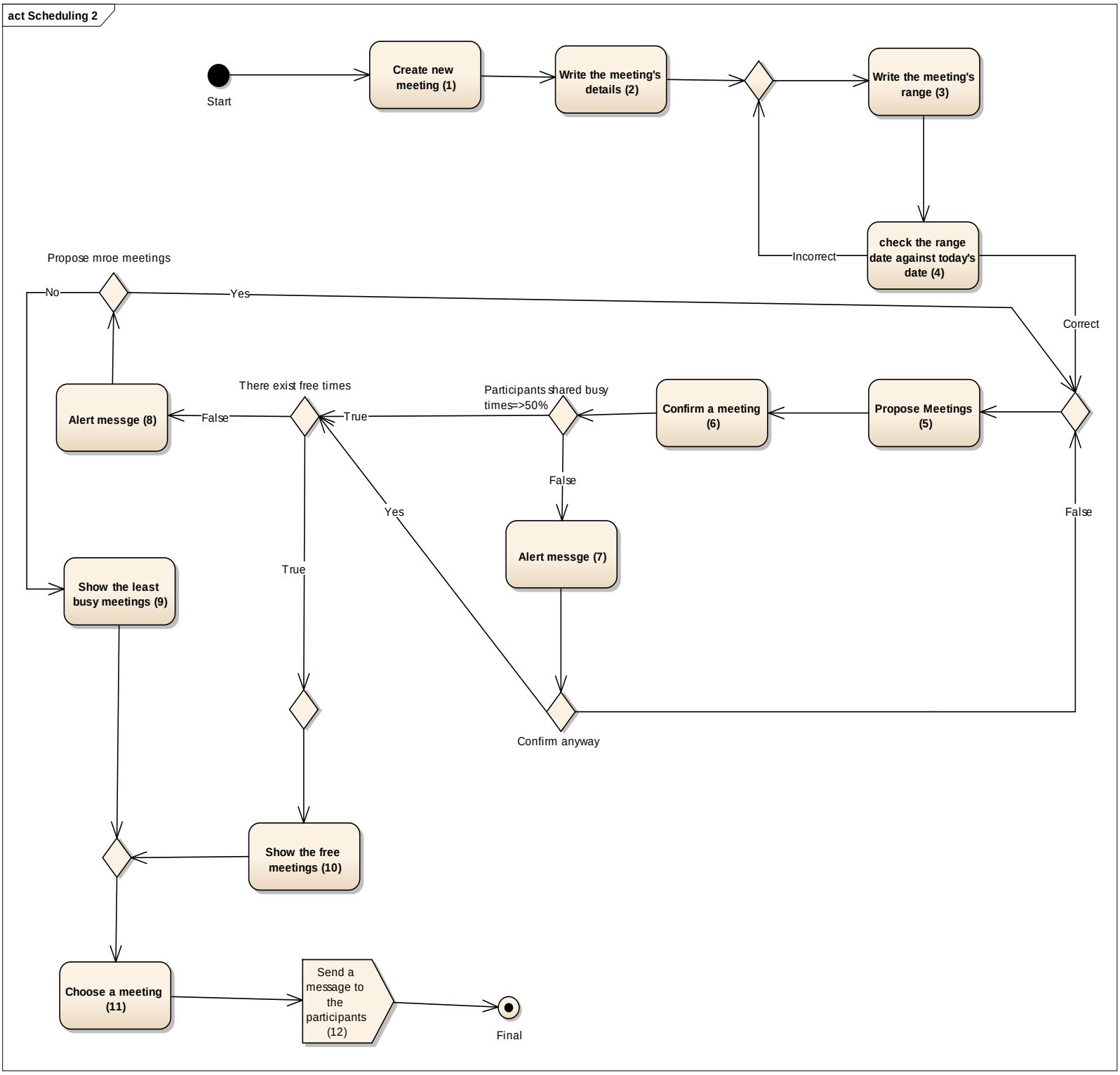

sd Randomly Sekect a Participant
Admin
Screen
Controller
Spinner
Room
Database
roll()
roll()
roll()
getParticipants()
getParticipants()
Participants()
Chosen Participant()
Chosen Participant()

Activity Diagram

Generate Groups



Scheduling


act Scheduling 2
Start
Create new meeting (1)
Write the meeting's details (2)
Write the meeting's range (3)
check the range date against today's date (4)
Incorrect
Correct
Propose mroe meetings
No
Yes
Alert messge (8)
There exist free times
False
True
Participants shared busy times=>50%
Confirm a meeting (6)
Propose Meetings (5)
False
Alert messge (7)
Yes
True
Show the least busy meetings (9)
Confirm anyway
Show the free meetings (10)
Choose a meeting (11)
Send a message to the participants (12)
Final

White-Box Test Cases

Generate Groups

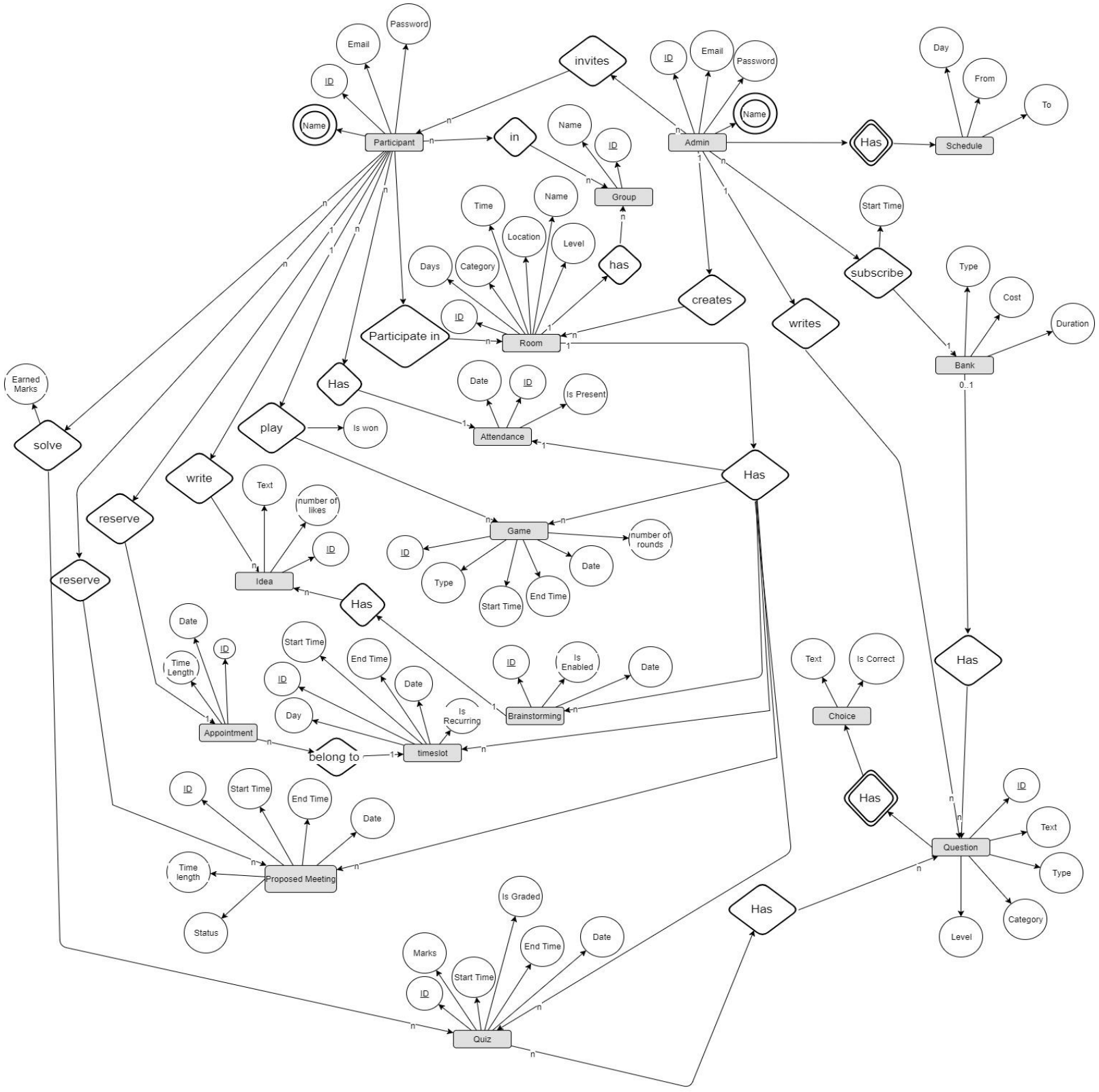
Test Case ID	Initial State	Input	Nodes	Expected Output
TC01	No groups,18 participant	3 groups, automatic	1,2,4,8,12,15,16,18,19	3 groups with 6 participants
TC02	No groups,18 participant	3 Participant, automatic	1,2,4,9,13,15,16,18,19	3 groups with 6 participants
TC03	No groups,16 participant	4 groups, Manual	1,2,4,8,12,15,17,19	4 groups with 4 participants
TC04	No groups,16 participant	4 Participant, Manual	1,2,4,9,13,15,17,19	4 groups with 4 participants
TC05	3 groups, 18 participants	Delete group 1	1,3,5,6,14	Group 1 participants are free, 2 groups
TC06	4 groups, 16 participants	Merge group 2,3	1,3,5,10,14	3 groups
TC07	3 groups, 18 participants	Transfer participant from 2 to 1	1,3,5,7,11,14	3 groups

Scheduling

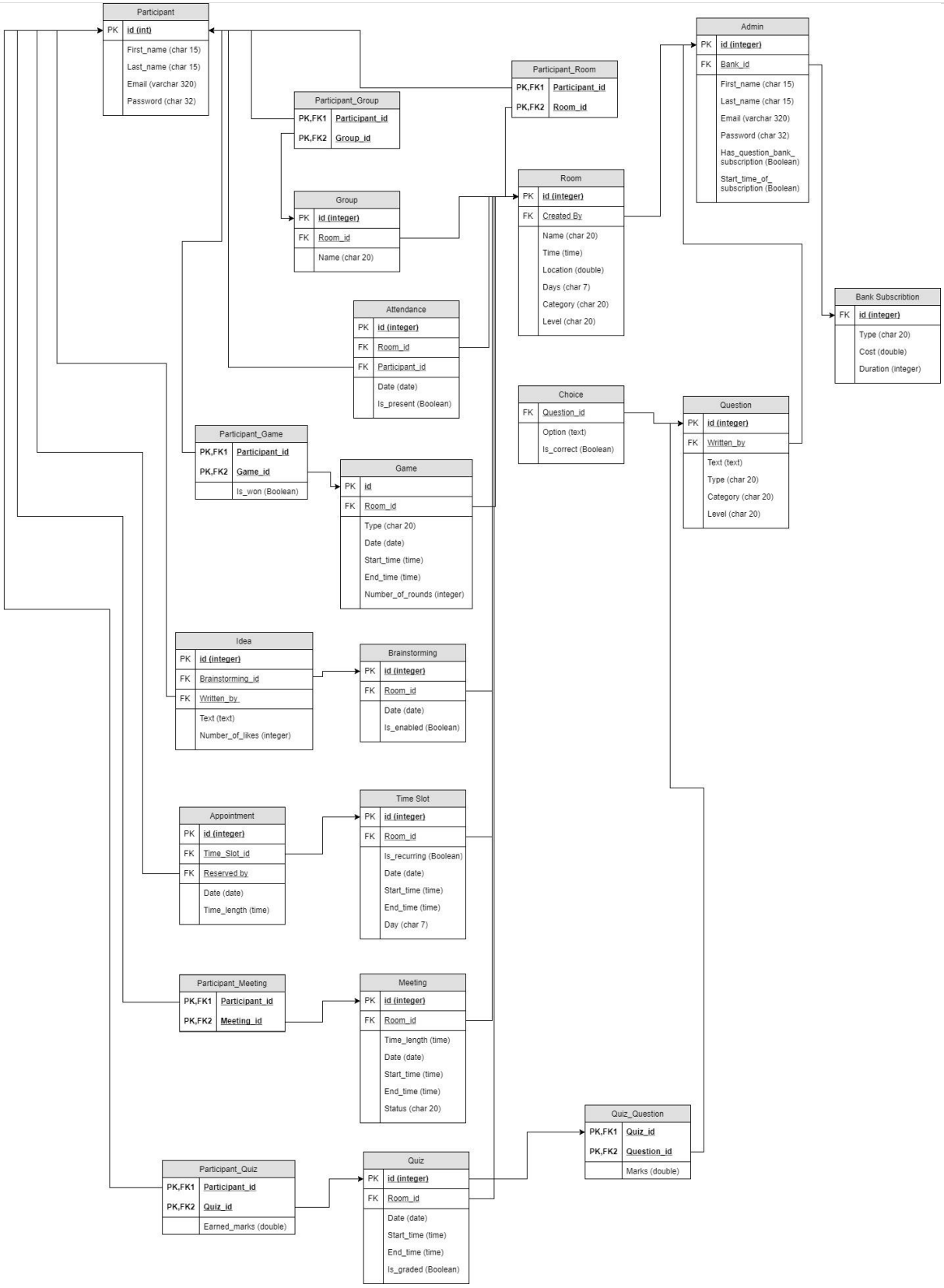
Test Case ID	Initial State	Input	Nodes	Expected Output
TC08	most of participants shared and there exists free times	(1-10/1/2019), 1.5-hour length,	1,2,3,4,5,6,10,11,12	Confirmed Meeting
TC09	most of participants shared and all times are busy	(1-10/1/2019), 1.5-hour length, Confirm anyway	1,2,3,4,5,6,8,9,11,12	Confirmed Meeting
TC10	Not most of participants shared and there exists free times	(1-10/1/2019), 1.5-hour length, Confirm anyway	1,2,3,4,5,6,7,10,11,12	Confirmed Meeting
TC12	Not most of participants shared and all times are busy	(1-10/1/2019), 1.5-hour length, Confirm anyway, Confirm anyway	1,2,3,4,5,6,7,8,9,10,11,12	Confirmed Meeting

Database Design

Entity Relationship Diagram



Relational Schema



Schema Indices

1. **RoomTable**: Index on columns (id, created_by)

2. **GroupTable**: Index on columns (id, room_id)

3. **AdminTable**: Index on columns (id, bank_id)

4. **AttendanceTable**: Index on columns (room_id , participant_id)

5. **QuestionTable:** Index on columns (written_by)

6. **MeetingTable:** Index on columns (room_id , date)

7. **AppointmentTable:** Index on columns (time_slot_id , reserved_by)

8. **IdeaTable:** Index on columns (brainstorming_id , written_by_id)

9. **TimeslotTable:** Index on columns (room_id)

10. **QuizTable:** Index on columns (room_id)