Bunk'O'Bed

-Make your stay away from home feel like home

Sayf Hussain Z

2018103059

Suryaa V S

2018103610

We define ourselves as " A social website that connects people who have space to spare with those who are looking for a place to stay".

Niche in the Market:

Lots of homeowners find it difficult to rent out their property for extended stay. Bunk'O'Bed connects these homeowners to people who are looking for affordable housing for a short period of time.

Scope:

Hotels all over the world offer a very limited set of facilities for a very steep price, Bunk'O'Bed provides users the opportunity to search for accommodations that suit their needs and budgets. Tourist whether it be local or international would prefer to stay in the warmth of a house with amenities that just aren't present in hotels. Guests can now choose to live in an entire apartment by themselves at an affordable price. Tourists now get an opportunity to stay in a local nieghbourhood and get to know the place better which couldn't be done with a regular hotel. Bunk'O'Bed doesn't just cater to tourist, but to anyone looking for short term accommodations at an affordable price.

People who own multiple homes in a city get the perfect opportunity to supplement their income by renting out properties for a short term and with very little maintenance of the property. Elderly people with spare rooms who do not want to rent out their rooms on a

permanent basis can now rent out their rooms on short term basis to prospective guests.

Anyone moving to a new city in search of work need not commit to a long term rental plan right away and can instead ease into the city by choosing a short term accommodation suiting their needs.

Implementation:

Creating an account:

Users are required to enter details regarding themselves like their phone number, ssn/aadhar card details and if they are the host some paperwork for the property to prove ownership.

Adding a listing:

Homeowners who are willing to rent out their homes or spare room register onto the site and make a listing with details on the type of property, amenities offered, the location with pictures of the property. Amenities can include loads of features like availability of home cooked food, the electrical appliances that the guests can use, the furnishing available, access to recreational facilities. The host also decides the price of stay for each day.

Categorising properties:

Guests can log in into the system and search for listings based on various factors like the city, the cost, the type of property whether it’s a single room, an apartment, or even a bungalow. They can also search based on the amenities offered like access to a swimming pool, the furnishing available.

Making a reservation:

Users can choose a listing and read through its description, pictures posted and even check the profile of the host. Users can also request to talk to the host even before they have made the reservation for the property. Contact details are kept private until a reservation is

made and whenever a reservation is made the host and the guest are notified with each other's contact details.

Rating system:

All users (including the hosts and the guests) have a rating which is visible to everyone. The rating for a host is generated based on reviews of other guests whom the host has previously hosted and for the guests its based on the hosts at whose properties they have stayed.

Grievance support:

24/7 user support is available that takes up queries from the users and gets back to them at the earliest with solutions for their problems.

Contacting freelance photographers:

Homeowners can contact freelance photographers through the site and request them to click pictures of the property.

Blacklisting listings:

The admin goes through the rating system and if he finds listings violating user policy he can blacklist the listing and remove the listing and also temporarily ban the user.

Agency bookings

Travel agencies can connect with the hosts and offer stay at apartments instead of traditional hotels in their travel itinerary.

Actors:

Host: A special type of user with permissions to add listings for multiple properties.

Guest: A user who primarily uses the system to browse through the different listings in various categories and makes a reservation for the desired property.

Admin: Maintains control over the entire system, integrity of the system and acts upon the directions of the grievance system to ban users either temporarily or permanently.

Customer support: Responds to the queries or complaints made by any kind of user to assist them. Customer support also passes on information about each user to the admin.

Travel Agencies: Another special user with the permission to include multiple hosts in the travel plans promoted by them.

Photographers: Photographers can contact different hosts offering their services at a nominal cost. They also play a role of validating users to eliminate fraudsters uploading fake pictures for their listings.

USECASE DIAGRAM

The purpose of use case diagram is to capture the dynamic aspect of a system. Use case diagrams are used to gather the requirements of a system including internal and external influences.

These requirements are mostly design requirements. Hence, when a system is analysed to gather its functionalities, use cases are prepared and actors are identified.

Use case diagrams are considered for high level requirement analysis of a system. When the requirements of a system are analysed, the functionalities are captured in use cases.

Use cases:

I. Create an account.
II. Logging in.
III. Rating system.
IV. Grievance system.
V. Add a listing.
VI. Contact freelance photographers.
VII. Make payment to photographer.
VIII. Request to contact host.
IX. Categorising properties.
X. Making a reservation.
XI. Make payment.
XII. Disclosure of details.
XIII. Blacklisting users.
XIV. Temporary Ban.
XV. Contact agencies and advertisers.
XVI. Details verification.
XVII. Register agency.
XVIII. Upload travel itinerary.
XIX. Bulk bookings

Use case: Creating account

Primary actor: Host, Guest.

Goal of the use case: Allow people to register onto the system by entering their details. The entered details are then validated by the admin after which a verification email is sent to the user.

Use cases involved at a lower level of abstraction:

I. Enter details.
II. Validate details.
III. Send verification mail.
IV. Validate mail.
V. Pay registration fee.

Use case: Rating System

Primary actor: Host, Guest.

Goal of the use case: Allow any user to provide ratings to other users they have interacted with (i.e rented their property to or had rented their property).The users also have the option of adding a detailed review of the host, guest and the property. The new rating for the host, guest and the property is also calculated here.

Use cases involved at a lower level of abstraction:

I. Add rating for host
II. Add rating for guest
III. Add rating for property
IV. Add a review
V. Update rating

Use case: Logging in

Primary actors: Host, Guest

Goal of the use case: Registered users are required to enter their credentials before accessing their account. The entered credentials are cross checked with existing records before the user is allowed access to any features reserved for registered users.

Use case: Adding a listing

Primary actors: Host.

Goal of the use case: Users can list their property on the website. Users are required to enter information about the property such as the location, size of the property and the features available.

Use case: Contact freelance photographers

Primary actors: Host.

Goal of the use case: The host can contact photographers to take pictures of the property to increase the trust factor of the property as the pictures of the property come from a third party.

Use case: Make payment for photographer

Primary actors: Host

Goal of the use case: The photographers is paid for his services.

Use case: Categorising properties

Primary actors: Guest.

Goal of the use case: Users can categorise properties while searching for them based on various factors like location, type of property, amenities available.

Use case: Making a reservation

Primary actors: Guest

Goal of the use case: Users can Finalize details with the host and make a reservation for the property for the specified dates.

Use case: Disclosure of details

Primary actor: Guest

Goal of the use case: Contact details are exchanged between the host and the user which were kept private before this stage.

Use case: Make payment

Primary actor: Guest.

Goal of the use case: The user can now make payment for the rented property. This payment is first held in the system, the admin then confirms with the host if the guest has checked in and then transfers money to the host.

Use cases involved at a lower level of abstraction:

I. Make payment
II. Withhold payment
III. Confirm check-in with the host.
IV. Transfer money to the host.

Use case: Blacklisting user

Primary actor: Admin.

Goal of the use case: The admin blacklists users with lower ratings and users who are reported repeatedly. If a user gets blacklisted more than 5 times the user is banned.

Use case: Temporary ban

Primary actor: Admin.

Goal of the use case: User who are blacklisted more than 5 times in a period of 6 months are banned from accessing the system.

Use case: Contact agencies and advertisers

Primary actor: Admin.

Goal of the use case: The admin can invite various travel agencies to use the platform and can also contact advertisers to market the platform efficiently.

Use case: Details verification

Primary actor: Admin

Goal of the use case: The Admin has to verify the information entered by the user while listing properties like check ownership of the property.

Use case: Register Agency

Primary actor: Travel Agency

Goal of the use case: Travel agencies can register with the system to integrate the features offered by the platform into their travel plans.

Use case: Upload Travel Itinerary

Primary actor: Travel Agency.

Goal of the use case: Travel agencies can create travel plan for tourists with the housing provided by various hosts on the system and can upload the itinerary on the platform.

Use case: Bulk bookings

Primary actor: Travel Agency.

Goal of the use case: The travel agency can rent out multiple properties offered by multiple hosts at the same time.

Use case: Help desk

Primary actors: Host, Guest, Photographers, Customer support.

Goal of the use case: Users can send in their complains and queries which are then replied to by the customer support.

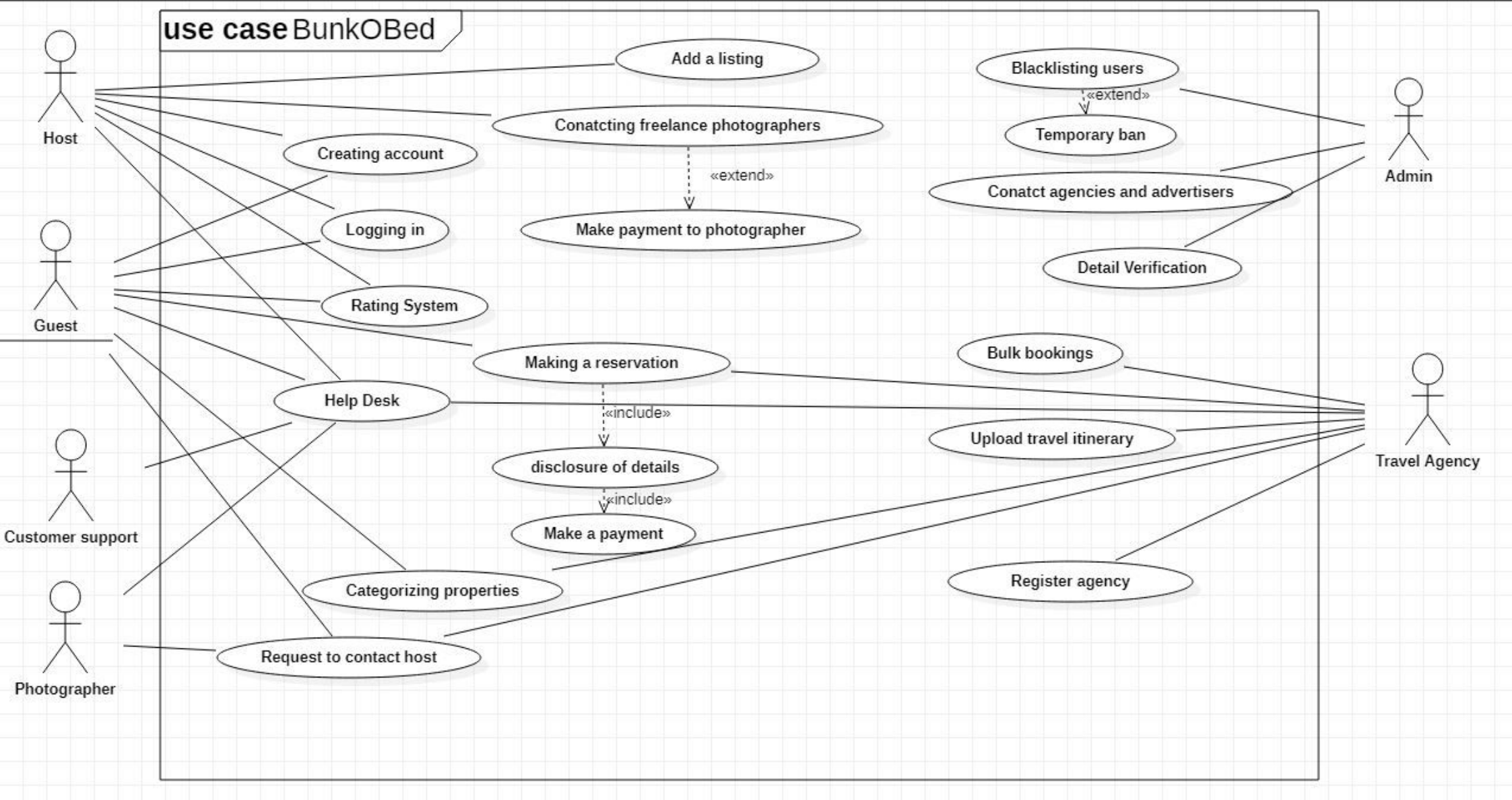
Use cases involved at a lower level of abstraction:

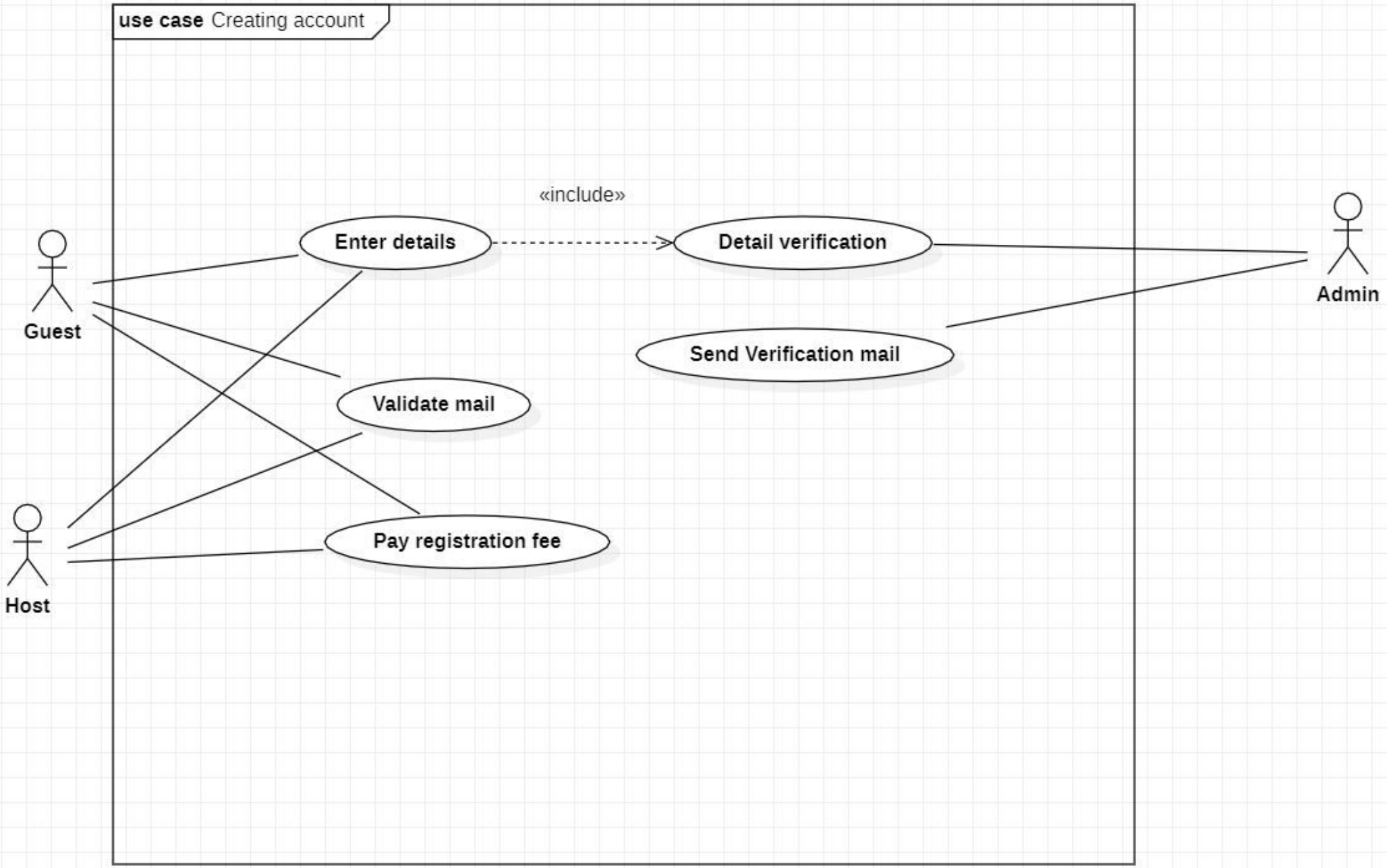
I. Report abuse
II. Avail refund
III. Schedule meeting with admin
IV. Contact company

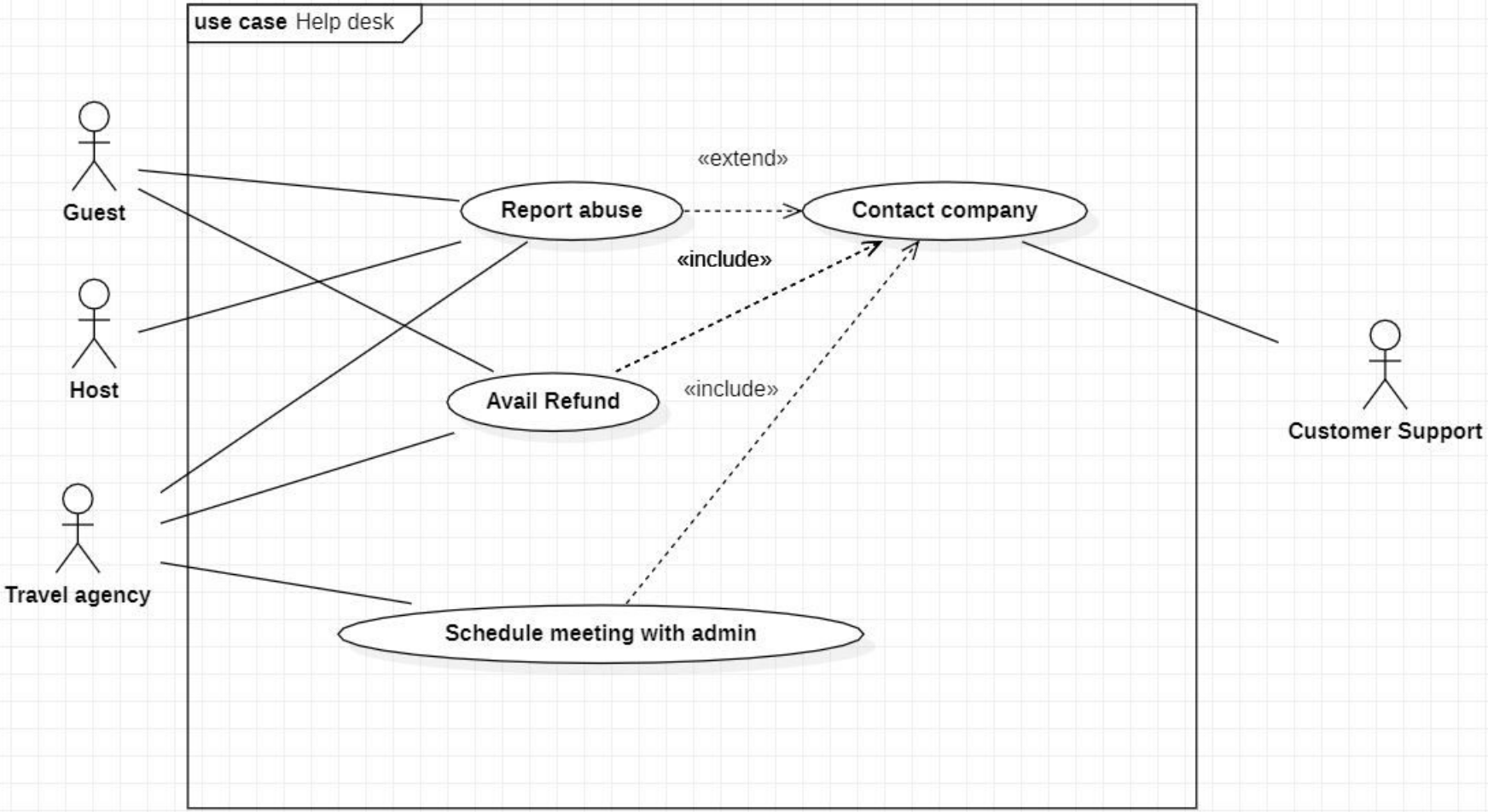
Use case: Request to contact host

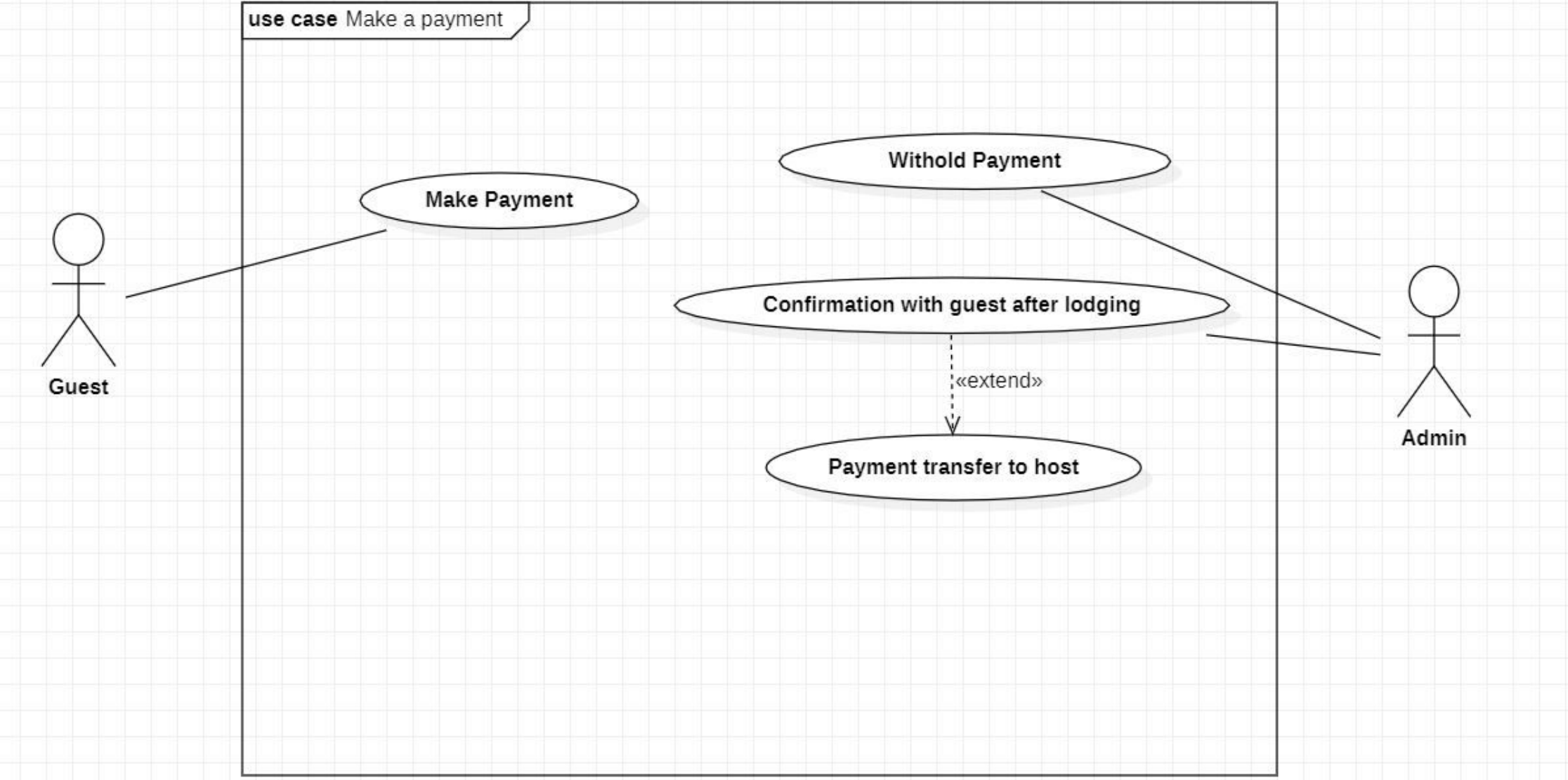
Primary actor: Guest, Photographers, Travel Agency.

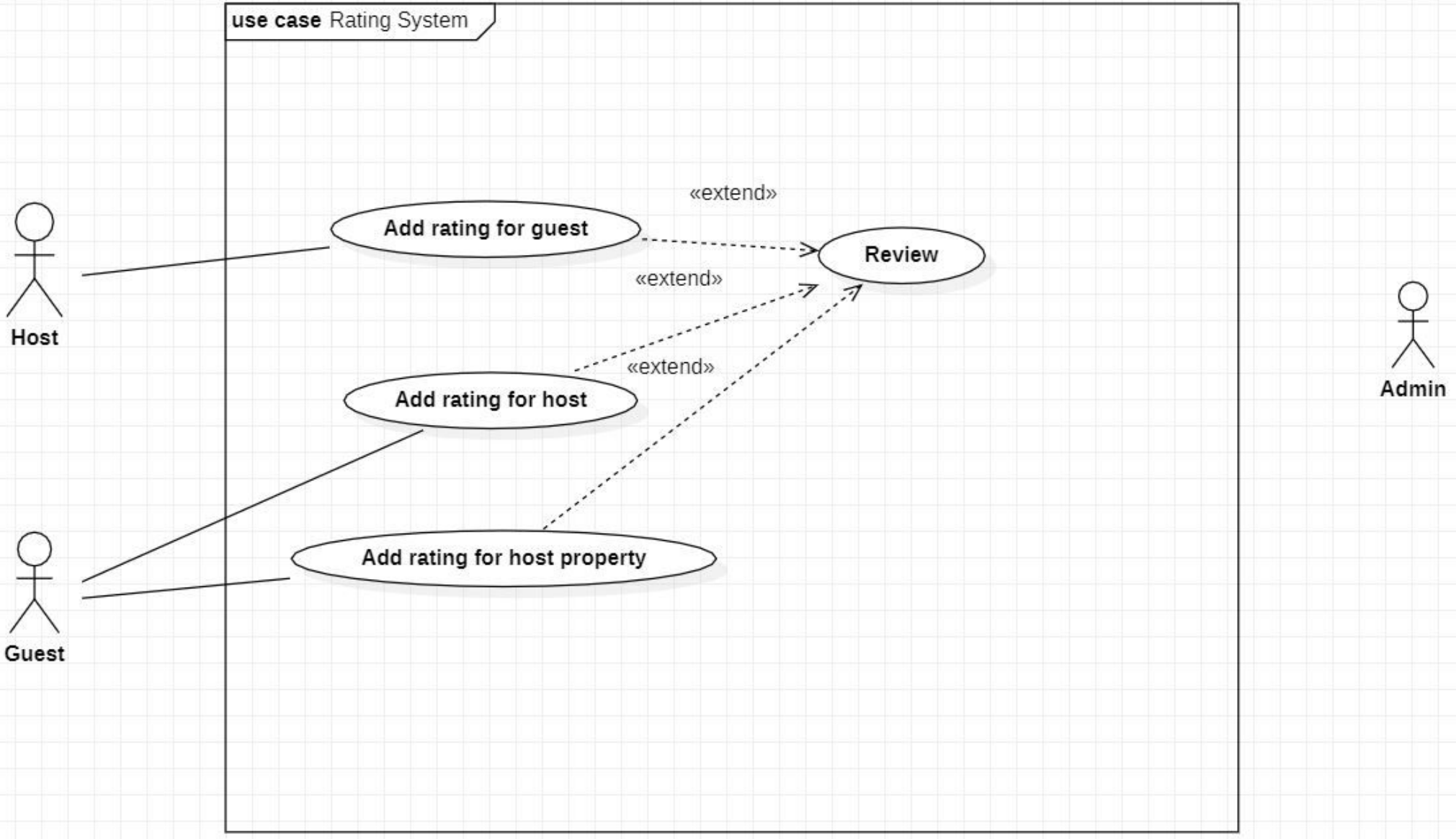
Goal of the use case: Guest, Photographers and Travel agencies can request to contact host to either discuss details or sign them up for their travel plan.


use case BunkOBed
Host
Guest
Customer support
Photographer
Admin
Travel Agency
Add a listing
Conatcting freelance photographers
Creating account
Logging in
Rating System
Help Desk
Make payment to photographer
«extend»
Blacklisting users
«extend»
Temporary ban
Conatct agencies and advertisers
Detail Verification
Making a reservation
«include»
disclosure of details
«include»
Make a payment
Bulk bookings
Upload travel itinerary
Register agency
Categorizing properties
Request to contact host


use case Creating account
«include»
Enter details
Detail verification
Send Verification mail
Validate mail
Pay registration fee
Guest
Host
Admin


use case Help desk
Guest
Host
Travel agency
Report abuse
«extend»
Contact company
«include»
Avail Refund
«include»
Schedule meeting with admin
Customer Support


use case Make a payment
Make Payment
Withold Payment
Confirmation with guest after lodging
«extend»
Payment transfer to host
Guest
Admin


use case Rating System
Host
Add rating for guest
«extend»
Review
«extend»
«extend»
Add rating for host
Admin
Guest
Add rating for host property

DOMAIN MODEL

A *domain model* is a *visual* representation of conceptual classes or real - situation objects in a domain. Domain models have also been called *conceptual models*, *domain object models*, and *analysis object models*.

Domain classes:

I. User
II. Host
III. Guest
IV. Property listings
V. Rating
VI. Transaction
VII. Admin
VIII. Photographers
IX. Queries
X. Costumer support
XI. Travel agencies
XII. Meeting
XIII. Refund
XIV. Rented properties
XV. Bulk bookings

Host ,Guest and Travel agencies are subclasses of User

Associations and Multiplicity:

Admin ***Manages*** User (1 to 1..*)

User ***Has*** User details (1 to 1)

Host ***Adds a*** property listings (1 to 1..*)

Guest ***Books a*** Property listing (1 to *)

Property listing ***Added to*** Rented Properties (1 to *)

Guest ***Initiates*** transaction (1 to *)

Travel agency ***Initiates*** Transaction (1 to *)

Transaction ***Forwards to*** Host (1 to 1)

User ***Makes a*** Queries (1 to *)

Queries ***Resolved by*** Customer support (1 to 1..*)

Customer support ***initiates*** Refund (1 to *)

Refund ***Requires*** Transaction (1 to 1)

User ***Has a*** Rating (1 to 1)

Property ***Has a*** Rating (1 to 1)

Host ***Hires a*** photographer (1 to *)

Meeting ***Scheduled with*** admin (1 to 1)

Travel agency ***calls for a*** meeting (1 to 1)

Attributes:

1. User
    - User ID
    - Password
    - Name
    - Number of blacklists
2. User details
    - Bank account details
    - Aadhar number
    - PAN card ID
    - Email id
    - Phone number
3. Guest
    - Number of bookings made
4. Host
    - Number of properties registered
5. Travel Agency

- Government Registration ID
- Number of bookings made

6. Rating
    - Number of stars
    - Comments
7. Customer support
    - Staff ID
    - Location
    - Rating
    - Languages spoken

8. Refund
    - Issue
    - Guest ID
9. Queries
    - Queryid
    - Type of Query
    - Grievance
    - User contact details
    - Status
10. Photographers
    - Photographer ID
    - Name
    - Location
    - Bank details
11. Admin
    - Admin ID
    - Position of responsibility
    - Password
12. Meeting
    - Meeting ID
    - Travel agency ID

- Date
- Time

13. Bulk bookings
    - Travel agency
    - Number of people
    - Duration of travel
    - Accommodation type
14. Property listing
    - Property ID
    - City
    - Address
    - Type of property
    - Host ID
15. Rented Property
    - Property ID
    - Guest ID
    - Booking dates
    - Cost of booking

16. Transaction
    - Transaction ID
    - Amount
    - Mode

Functions:

1. User:
    - Get_userID()
    - Cancel_booking()
    - Make_payment()
    - Notify_user()
2. Guest:
    - Calculate_cost()

3. Transaction:
    - Get_bank_details()
    - With_hold_payment()
    - Make_payment(mode,money)
    - Complete_transaction(bankdetails)
4. Queries:
    - Categorise()
    - Check_status()
    - Close_query()
5. Travel agency:
    - Calculate_cost()
6. Host:
    - Notify_host()
    - Hire_photographers()
7. Customer support:
    - Change_status()
    - Send_query()
    - Notify(queryid)
8. Refund:
    - Set_refund(amount)
9. User details
    - Get_Bank_details()
10. Property listings
    - Get_cost_per_day()
    - Checks_availability(time)
    - Book_property(userid)
    - Payment_complete(status)
    - Add_propertyphoto()
    - Notify(status,propertyid)
11. Ratings
    - Addtosuslist()
12. Rented Property
    - Get_booking_dates()

- Checks_availability(id,time)
- Add_property()

13. Meeting
    - Block_date_time(userid,adminid)
14. Admin
    - Modify_blacklist(user)
    - Ban(user)
    - Notify(transactionid)
    - Notify(userid,propertyid)

Class Responsibility Collaborations (CRC) :

Class	Attributes	Functions	Collaborations
User	• User ID • Password • Name • Number of blacklists	•Get_user_info() •Raise_query() •Notify_user	•User details •Admin •Queries •Ratings
User details	•Bank account details •Aadhar number • PAN card ID • Email id • Phone number	•get_Bank_details()	•User
Guest	• Number of bookings made	•Cancel_booking() •Make_payment()	•Property listings •Transaction
Host	• Number of properties registered	•Hire_photographers()	•Property listings •Photographers •Transaction
Travel agency	• Government Registration ID • Number of bookings made	• Calculate_cost()	•Transaction •Meetings •Bulk bookings
Rating	• Number of stars • Comments	• Addtosuslist()	•User •Property listings
Customer support	• Staff ID • Location • Rating • Languages spoken	• Change_status() • send_query() •notify(queryid)	•Refund •Queries
Refund	• Issue • Guest ID	• Set_refund(amount)	•Customer support •Transaction
Queries	• Type of Query • Grievance • User contact details • Status •Queryid	• Categorise() • Check_status() •Close_query()	•Customer support •User
Photographers	• Photographer ID • Name • Location	•Notifyphotographers()	•Host
Admin	• Admin ID • Position of responsibility	•Modify_blacklist(user) • Ban(user)	•User •Meeting

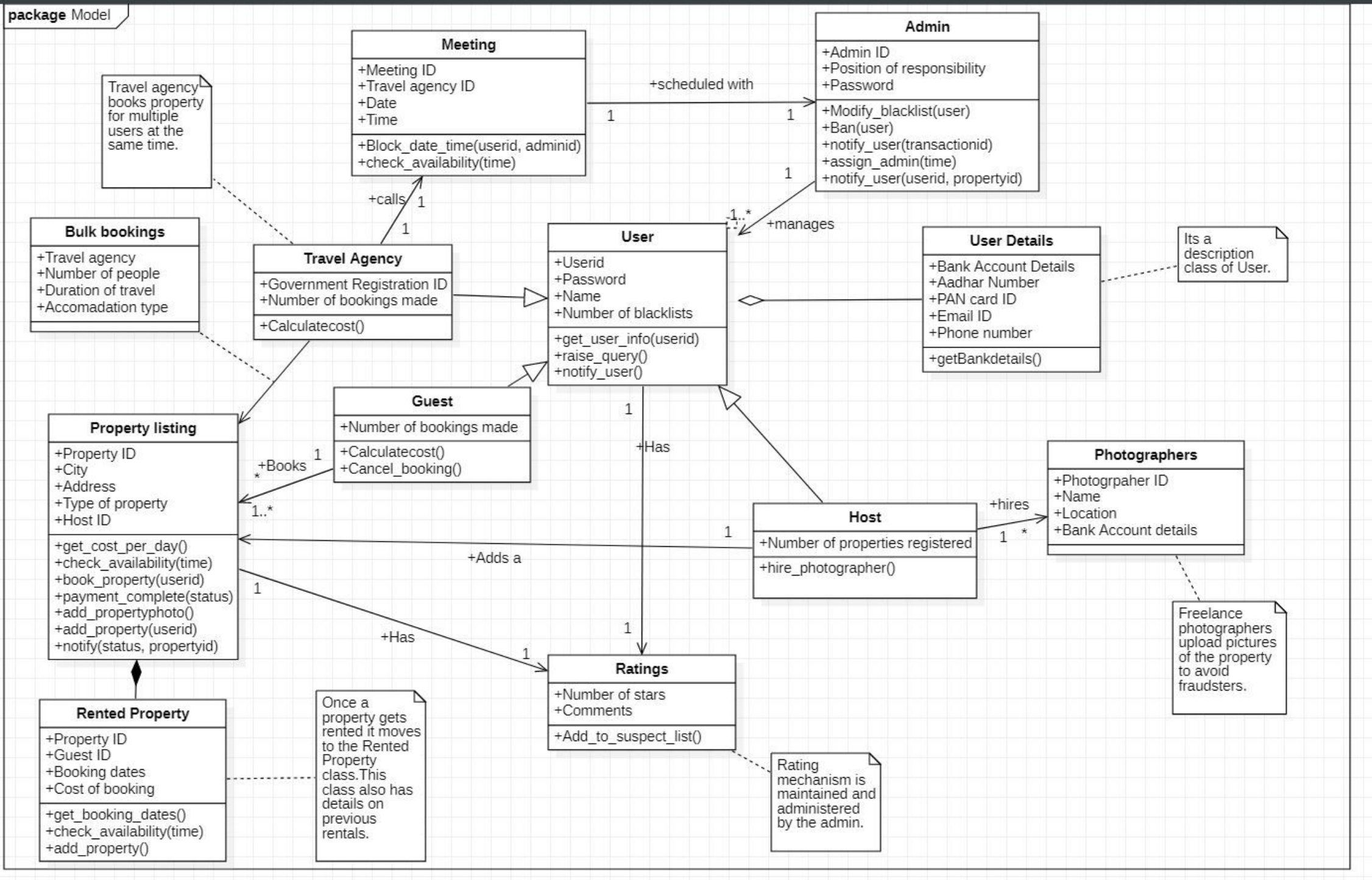
	• Password	•Notify(transactionid) •Notify(userid,propertyid)	
Meeting	• Meeting ID • Travel agency ID • Date • Time	• Block_date_time(userid,adminid)	•Travel agency •Admin
Bulk bookings	• Travel agency • Number of people • Duration of travel • Accommodation type		•Travel agency
Property listing	• Property ID • City • Address • Type of property • Host ID	• Get_cost_per_day() • Checks_availability(time) • Book_property(time) • Payment_complete(status) •Add_propertyphoto() •add_property() •notify(status,userid)	•Host •Rented property
Rented Property	• Property ID • Guest ID • Booking dates • Cost of booking	• Get_booking_dates() •Checks_availability(time) •Add_property()	•Property listings
Transaction	•Transaction ID •Amount •Mode	•with_hold_payment() •make_payment(mode,amount) •complete_transaction(bankdet)	•Travel Agency •Guest •Host •Refund

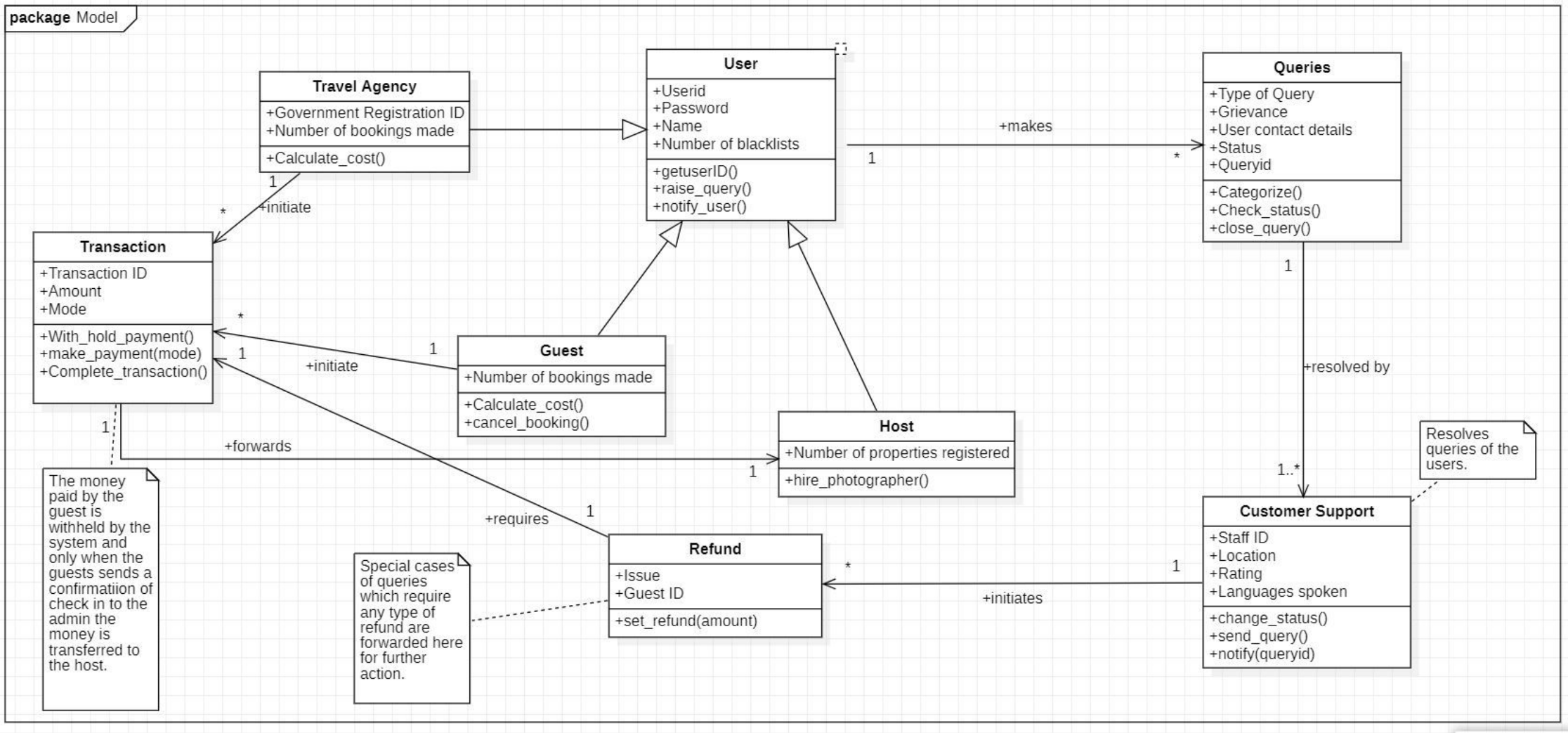
CLASS DIAGRAM

Class diagram is a static diagram. It represents the static view of an application. The purpose of class diagram is to model the static view of an application.

Class diagrams are the only diagrams which can be directly mapped with object-oriented languages and thus widely used at the time of construction.

Class diagram is basically a graphical representation of the static view of the system and represents different aspects of the application.


package Model
Meeting
+Meeting ID
+Travel agency ID
+Date
+Time
+Block_date_time(userid, adminid)
+check_availability(time)
+scheduled with
1
1
Admin
+Admin ID
+Position of responsibility
+Password
+Modify_blacklist(user)
+Ban(user)
+notify_user(transactionid)
+assign_admin(time)
+notify_user(userid, propertyid)
Travel agency books property for multiple users at the same time.
+calls
1
1
1
1..*
+manages
Bulk bookings
+Travel agency
+Number of people
+Duration of travel
+Accomadation type
Travel Agency
+Government Registration ID
+Number of bookings made
+Calculatecost()
User
+Userid
+Password
+Name
+Number of blacklists
+get_user_info(userid)
+raise_query()
+notify_user()
User Details
+Bank Account Details
+Aadhar Number
+PAN card ID
+Email ID
+Phone number
+getBankdetails()
Its a description class of User.
Guest
+Number of bookings made
+Calculatecost()
+Cancel_booking()
Property listing
+Property ID
+City
+Address
+Type of property
+Host ID
+get_cost_per_day()
+check_availability(time)
+book_property(userid)
+payment_complete(status)
+add_propertyphoto()
+add_property(userid)
+notify(status, propertyid)
+Books
*
1
1..*
1
+Has
Host
+Number of properties registered
+hire_photographer()
+hires
1
*
Photographers
+Photogrpaher ID
+Name
+Location
+Bank Account details
Freelance photographers upload pictures of the property to avoid fraudsters.
+Adds a
1
+Has
1
Ratings
+Number of stars
+Comments
+Add_to_suspect_list()
Rented Property
+Property ID
+Guest ID
+Booking dates
+Cost of booking
+get_booking_dates()
+check_availability(time)
+add_property()
Once a property gets rented it moves to the Rented Property class.This class also has details on previous rentals.
Rating mechanism is maintained and administered by the admin.


package Model
Travel Agency
+Government Registration ID
+Number of bookings made
+Calculate_cost()
User
+Userid
+Password
+Name
+Number of blacklists
+getuserID()
+raise_query()
+notify_user()
Queries
+Type of Query
+Grievance
+User contact details
+Status
+Queryid
+Categorize()
+Check_status()
+close_query()
+makes
1
*
1
+initiate
*
Transaction
+Transaction ID
+Amount
+Mode
+With_hold_payment()
+make_payment(mode)
+Complete_transaction()
1
*
1
+initiate
1
Guest
+Number of bookings made
+Calculate_cost()
+cancel_booking()
1
+resolved by
1..*
1
+forwards
1
Host
+Number of properties registered
+hire_photographer()
Resolves queries of the users.
The money paid by the guest is withheld by the system and only when the guests sends a confirmatiion of check in to the admin the money is transferred to the host.
+requires
1
Refund
+Issue
+Guest ID
+set_refund(amount)
Special cases of queries which require any type of refund are forwarded here for further action.
*
1
+initiates
Customer Support
+Staff ID
+Location
+Rating
+Languages spoken
+change_status()
+send_query()
+notify(queryid)

SEQUENCE DIAGRAM

Sequence diagrams can be useful references for businesses and other organizations. Try drawing a sequence diagram to:

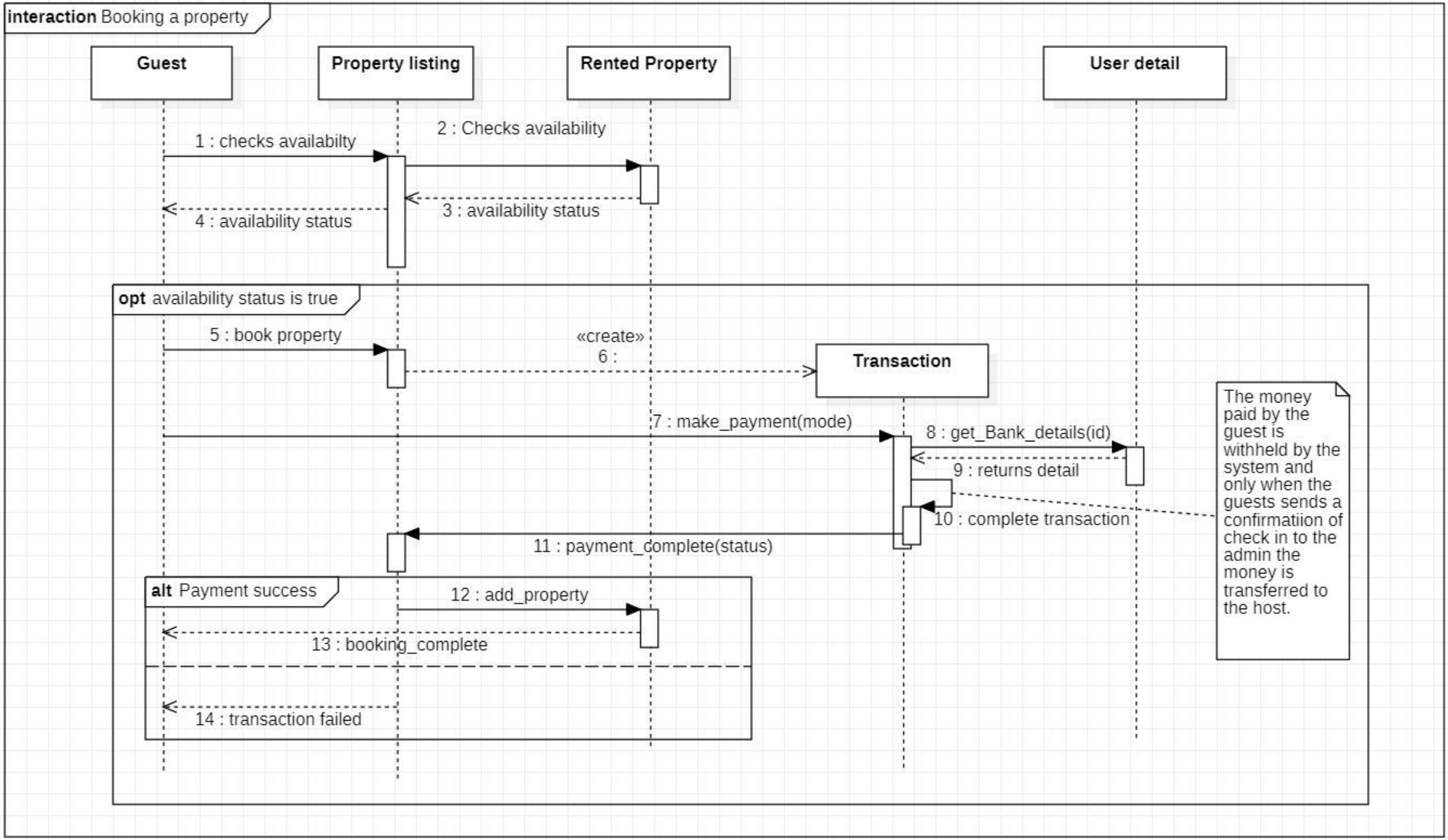
- Represent the details of a UML use case.
- Model the logic of a sophisticated procedure, function, or operation.
- See how objects and components interact with each other to complete a process.
- Plan and understand the detailed functionality of an existing or future scenario.

Sequence Diagram 1: Booking a property

Objects involved: Guest, Property listing, Rented Property and User Details.

Description:

- This sequence diagrams shows the steps involved in a Guest booking a property.
- The Guest first has to check for the availability of the property for the concerned dates, Only if the property is available can the guest continue to book the property.
- While booking a property a new instance of the transaction class is created.
- This instance gets the bank details of the user and completes the transaction and sends the transaction status to the property listing.
- Based on the status received by the property listing object the property is moved to the Rented property class and the Guest is notified.

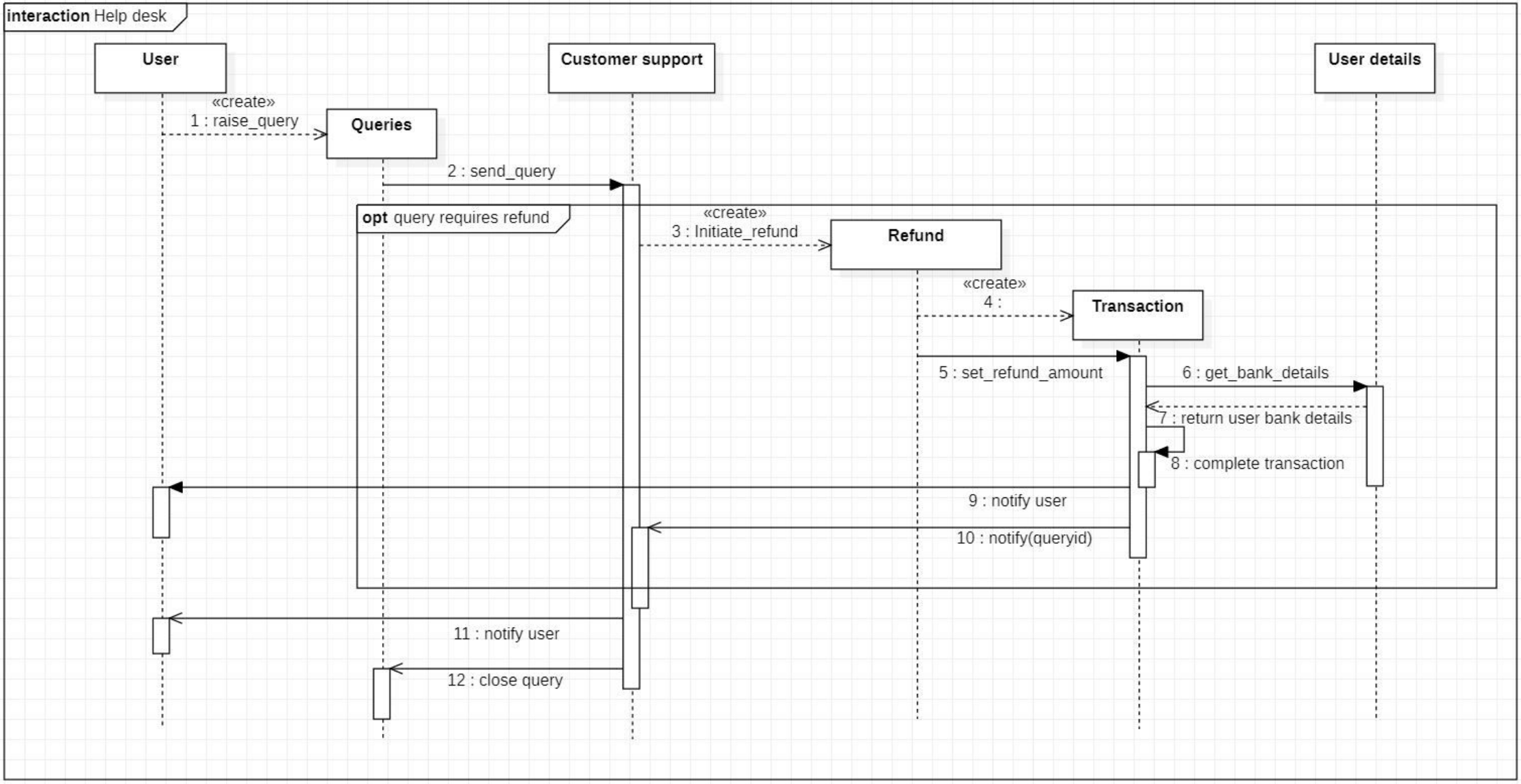

interaction Booking a property
Guest
Property listing
Rented Property
User detail
1 : checks availabilty
2 : Checks availability
3 : availability status
4 : availability status
opt availability status is true
5 : book property
«create»
6 :
Transaction
7 : make_payment(mode)
8 : get_Bank_details(id)
9 : returns detail
10 : complete transaction
11 : payment_complete(status)
The money paid by the guest is withheld by the system and only when the guests sends a confirmatiion of check in to the admin the money is transferred to the host.
alt Payment success
12 : add_property
13 : booking_complete
14 : transaction failed

Sequence Diagram 2: Help Desk

Objects involved: User, Queries, Customer support, Refund, Transaction and User details.

Description:

- This sequence diagram shows the steps involved in resolving User Queries.
- Any query raised by the User will create a new instance of the Queries class.
- The Customer support receives these queries and if refund is required, a new instance of Refund class is created.
- This instance will create a new instance of the Transaction class which gets the User details and completes the Transaction
- The instance of the Transaction class notifies both the User and the Customer support.
- The Customer support finally changes the Query status.

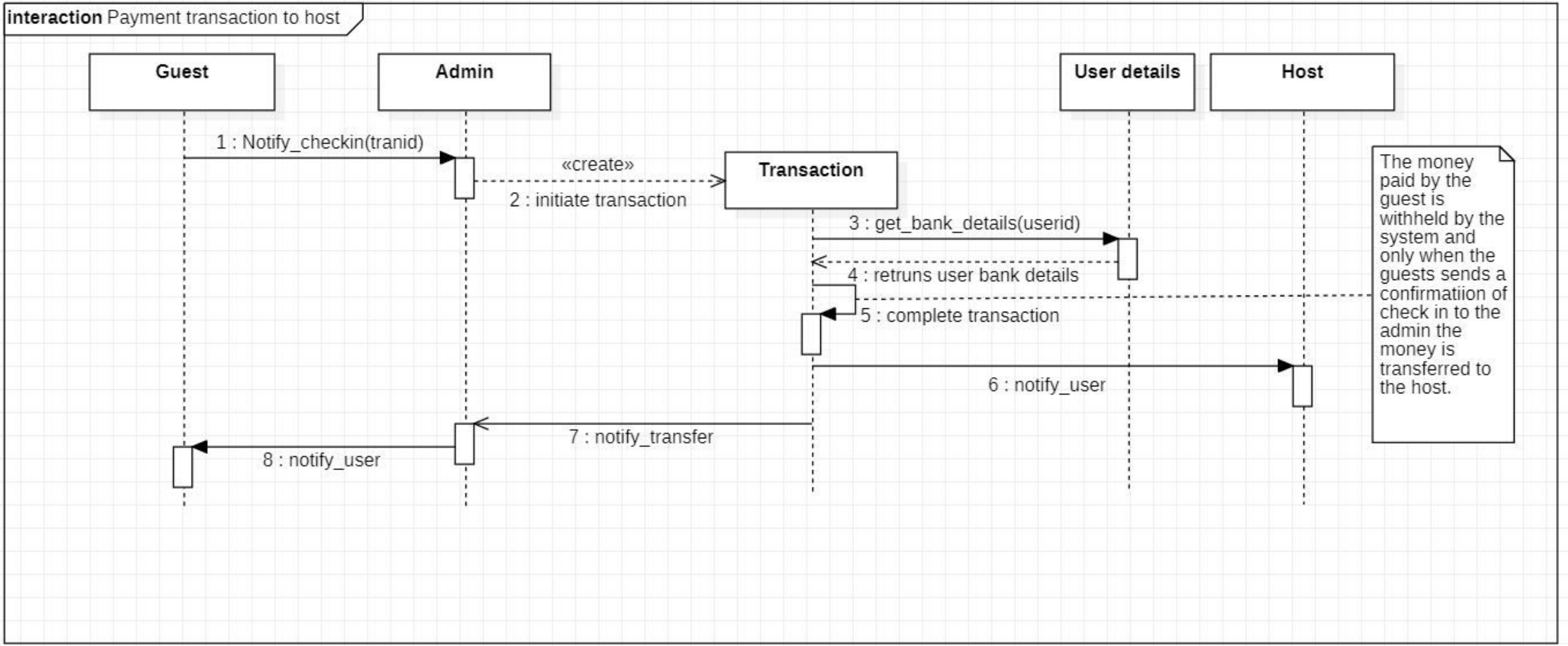

interaction Help desk
User
Customer support
User details
«create»
1 : raise_query
Queries
2 : send_query
opt query requires refund
«create»
3 : Initiate_refund
Refund
«create»
4 :
Transaction
5 : set_refund_amount
6 : get_bank_details
7 : return user bank details
8 : complete transaction
9 : notify user
10 : notify(queryid)
11 : notify user
12 : close query

Sequence Diagram 3: Payment Transaction to Host

Objects involved: Guest, Admin, Transaction, User details and Host.

Description:

- This sequence diagram shows the steps involved in forwarding the money to the Host
- The Guest notifies the admin of check-in.
- The Admin now creates a new instance of the Transaction class.
- This instance gets the Host's bank details and completes the transaction.
- This instance notifies the Host and the admin of the transaction status.
- The Admin then notifies the Guest.

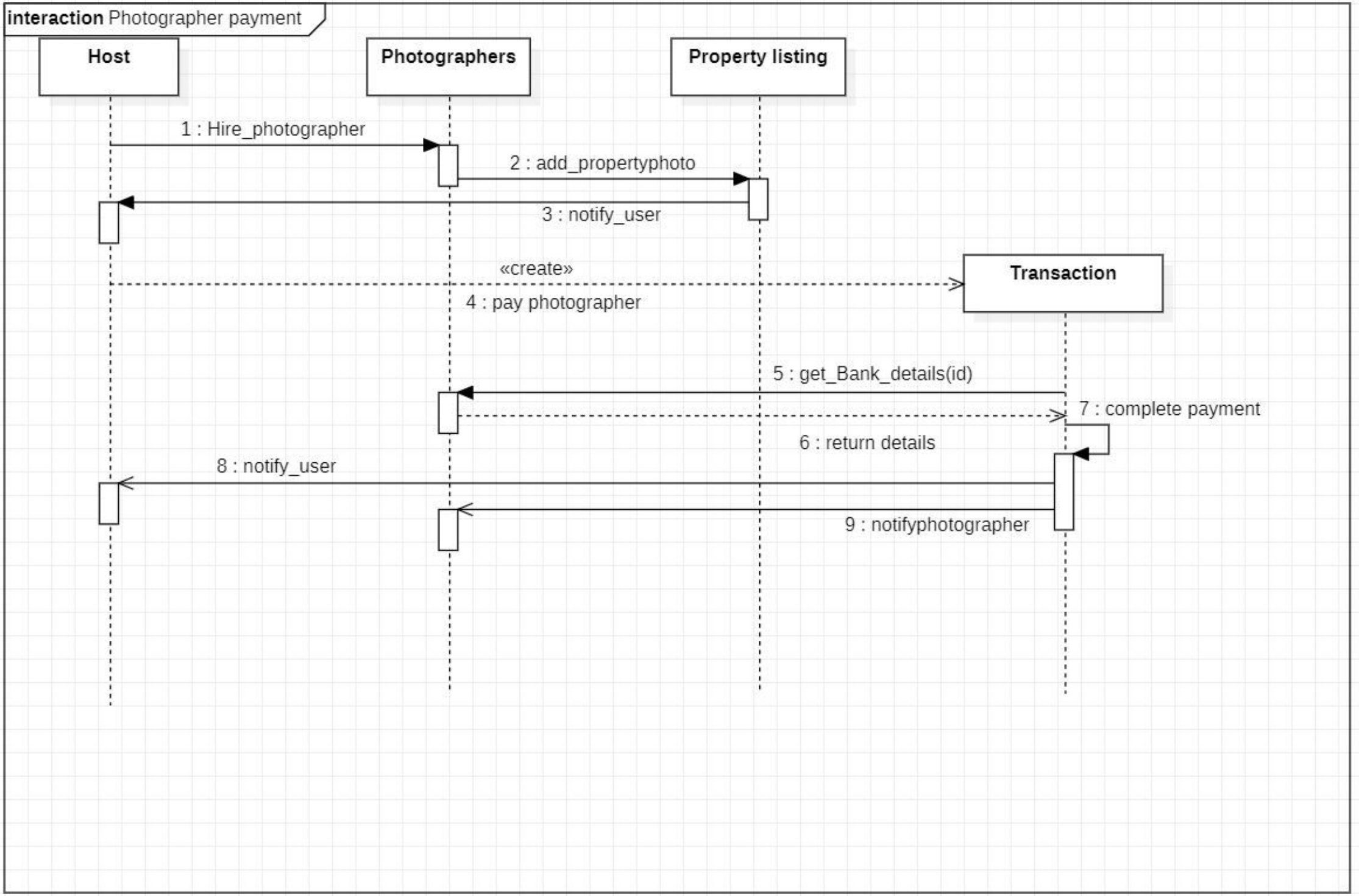

interaction Payment transaction to host
Guest
Admin
User details
Host
1 : Notify_checkin(tranid)
«create»
Transaction
2 : initiate transaction
3 : get_bank_details(userid)
4 : retruns user bank details
5 : complete transaction
6 : notify_user
7 : notify_transfer
8 : notify_user
The money paid by the guest is withheld by the system and only when the guests sends a confirmatiion of check in to the admin the money is transferred to the host.

Sequence Diagram 4: Photographer Payment

Objects involved: Host, Photographers, Property listings and Transaction.

Description:

- This sequence diagram shows the steps involved in Hiring and paying a Photographer.
- The Host hires a photographer who can then add photos of a property to its description in the Property listings.
- The Host now creates a new instance of the Transaction class.
- This instance gets the bank details of the photographers and completes the transaction.
- This instance notifies both, the Host and the Photographer of the payment status.

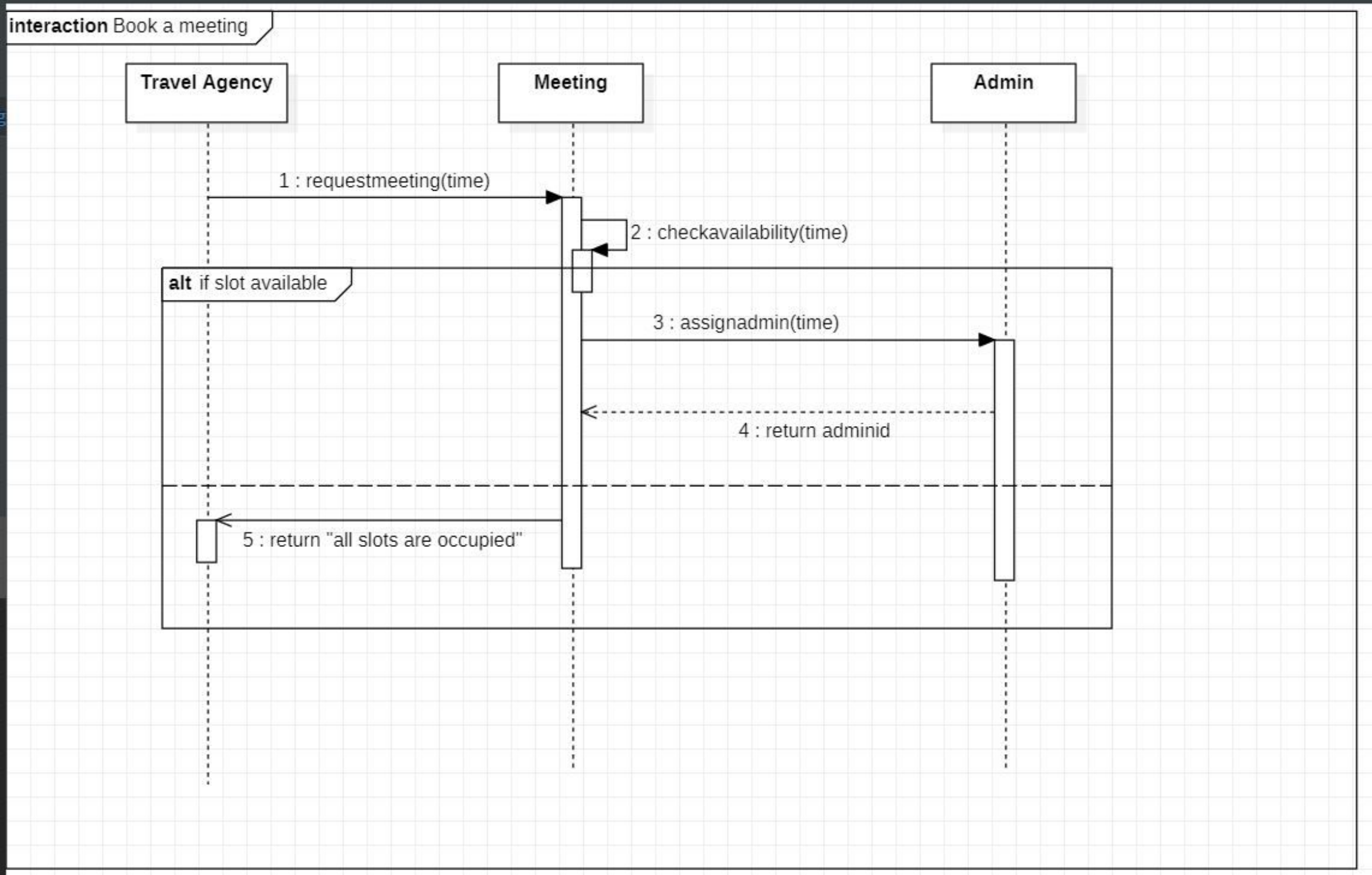

interaction Photographer payment
Host
Photographers
Property listing
1 : Hire_photographer
2 : add_propertyphoto
3 : notify_user
«create»
Transaction
4 : pay photographer
5 : get_Bank_details(id)
7 : complete payment
6 : return details
8 : notify_user
9 : notifyphotographer

Sequence Diagram 5: Book a Meeting

Objects involved: Travel agency, Meeting, Admin

Description:

- This sequence diagram shows the steps involved in a travel agency booking a Meeting with the Admin.
- The Travels agency first requests for a meeting at the given time.
- The Meeting class object checks for availability at the given time.
- If a slot is available at the requested time, an Admin is assigned to the Meeting.
- If no slot is available at the requested time then the Travel agency is notified that all slots are booked.


interaction Book a meeting
Travel Agency
Meeting
Admin
1 : requestmeeting(time)
2 : checkavailability(time)
alt if slot available
3 : assignadmin(time)
4 : return adminid
5 : return "all slots are occupied"

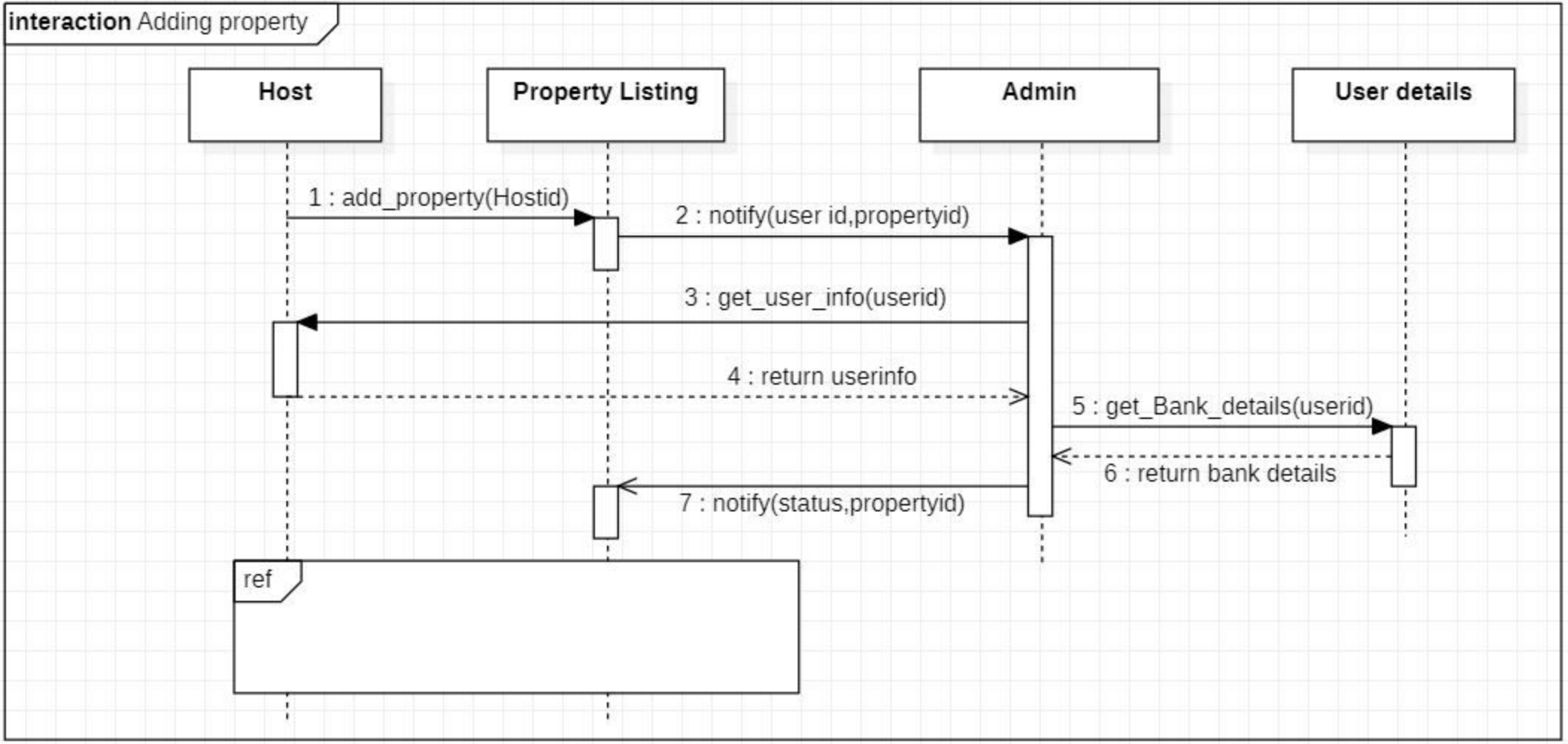
Sequence Diagram 6:Adding Property

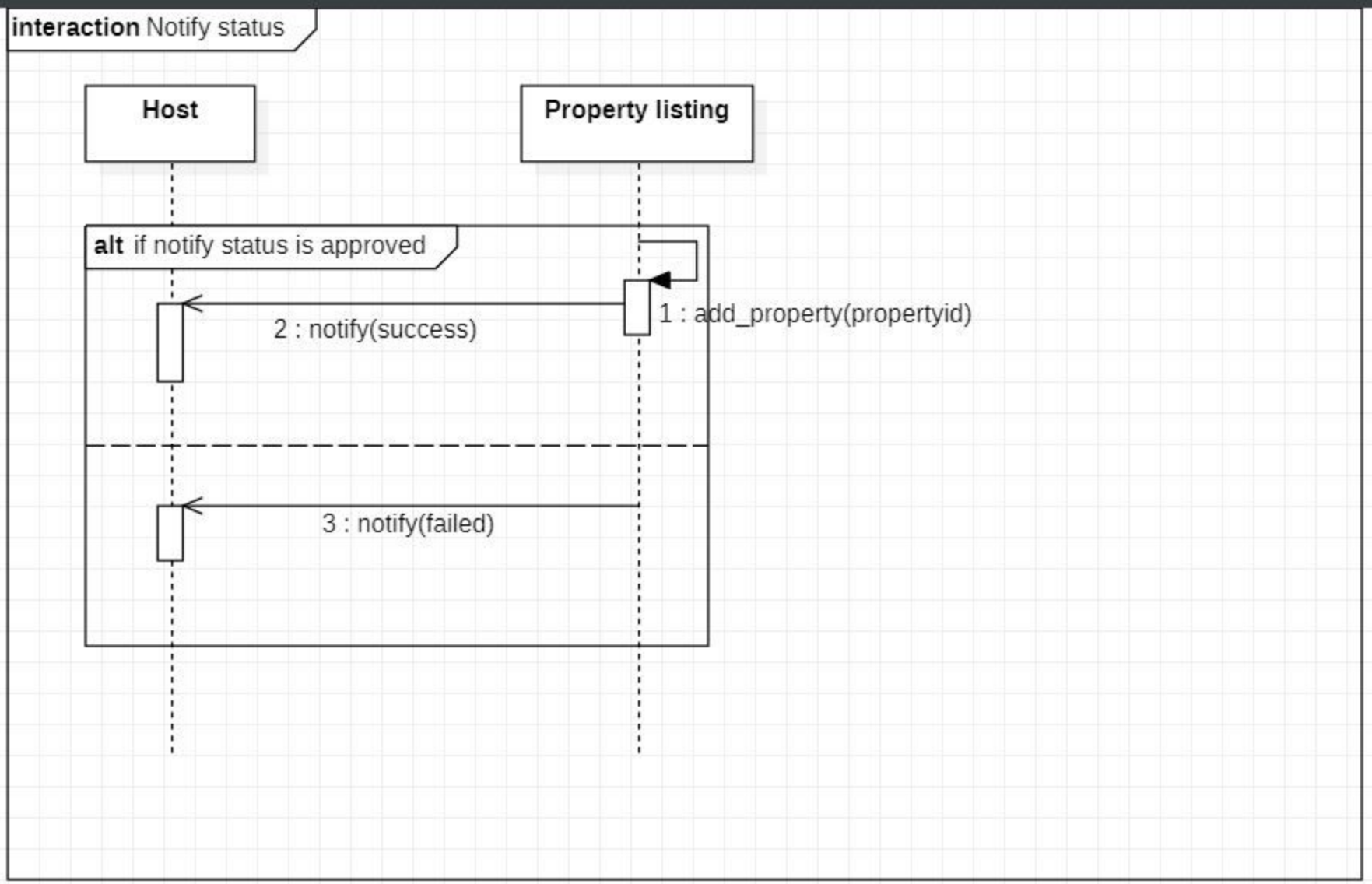
Objects involved: Host, Property Listing, Admin and User details

Description:

1. Host wants to add property listing which triggers Property Listing.
2. Property Listing notifies Admin who needs to validate user information and bank details of the host.
3. User information and bank details are verified and is notified to the Property listing.
4. If the status is success then the property list is added and notified to host else the host is notified failure.

Note: Ref is used in the case of notifying status to host and adding property listing to make the diagram clutter-free.


interaction Adding property
Host
Property Listing
Admin
User details
1 : add_property(Hostid)
2 : notify(user id,propertyid)
3 : get_user_info(userid)
4 : return userinfo
5 : get_Bank_details(userid)
6 : return bank details
7 : notify(status,propertyid)
ref


interaction Notify status
Host
Property listing
alt if notify status is approved
1 : add_property(propertyid)
2 : notify(success)
3 : notify(failed)

STATE MODEL DIAGRAM

A state machine models the behaviour of an object as it passes through a number of states in its lifetime due to some events as well as the actions occurring due to the events. A state machine is graphically represented through a state transition diagram.

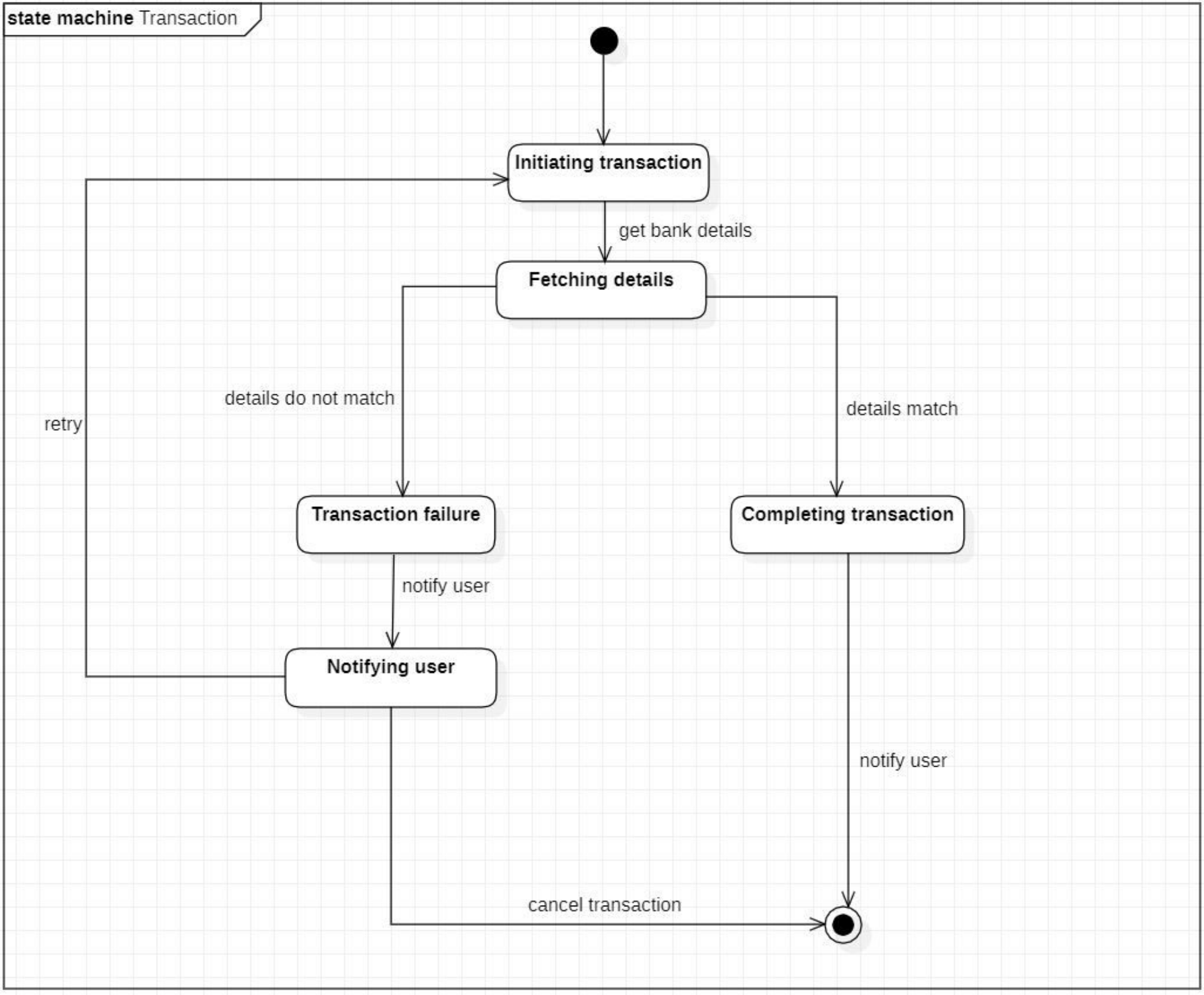
State Diagram 1: Transaction

Classes involved: Transaction

Description: Transaction class facilitates completion of payment for booking property and for refund if query is raised.

Flow of diagram:

1. Event: The guest initiates payment or customer support initiates refund payment
   Response: move to Initiating Transaction
2. Event: Bank details are being accessed
   Response: move to Fetching details
3. Event: If details fetched match with entered details
   Response: move to Completing transaction
4. Event: If details fetched do not match
   Response: move to Transaction failure
5. Event: In case of failure, user is notified
   Response: move to Initiating transaction (if retry)/ final state(cancel transaction)
6. Event: In case of success, user is notified
   Response: move to final state


state machine Transaction
Initiating transaction
get bank details
Fetching details
details do not match
retry
details match
Transaction failure
Completing transaction
notify user
Notifying user
notify user
cancel transaction

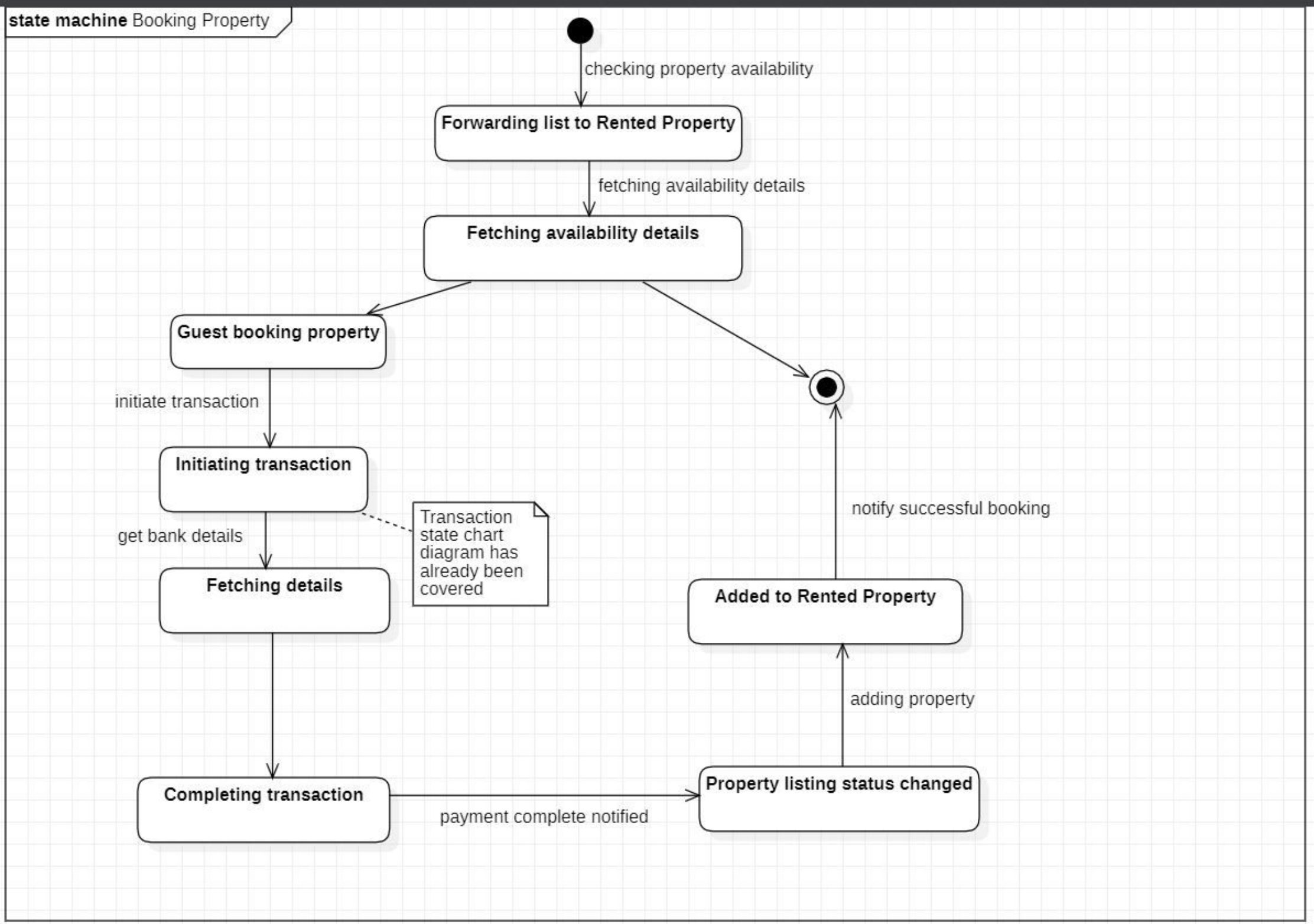
State Diagram 2: Booking Property

Classes: Property listing

Description: Property listing checks for availability and initiates transaction in case of availability, else moves to final state.

Flow of diagram:

1. Event: Checking property availability
   Response: moves to Forwarding list to Rented Property (Rented property is another object consisting of all the details about previous and current bookings)
2. Event: Trying to fetch all the availability details of the property
   Response: moves to Fetching availability details
3. Event: In case of availability
   Response: moves to Guest booking property -> initiates transaction (Previous state diagram) -> moves to the final state
4. Event: In case of non-availability
   Response: moves to final state


state machine Booking Property
checking property availability
Forwarding list to Rented Property
fetching availability details
Fetching availability details
Guest booking property
initiate transaction
Initiating transaction
get bank details
Fetching details
Transaction state chart diagram has already been covered
Completing transaction
payment complete notified
Property listing status changed
adding property
Added to Rented Property
notify successful booking

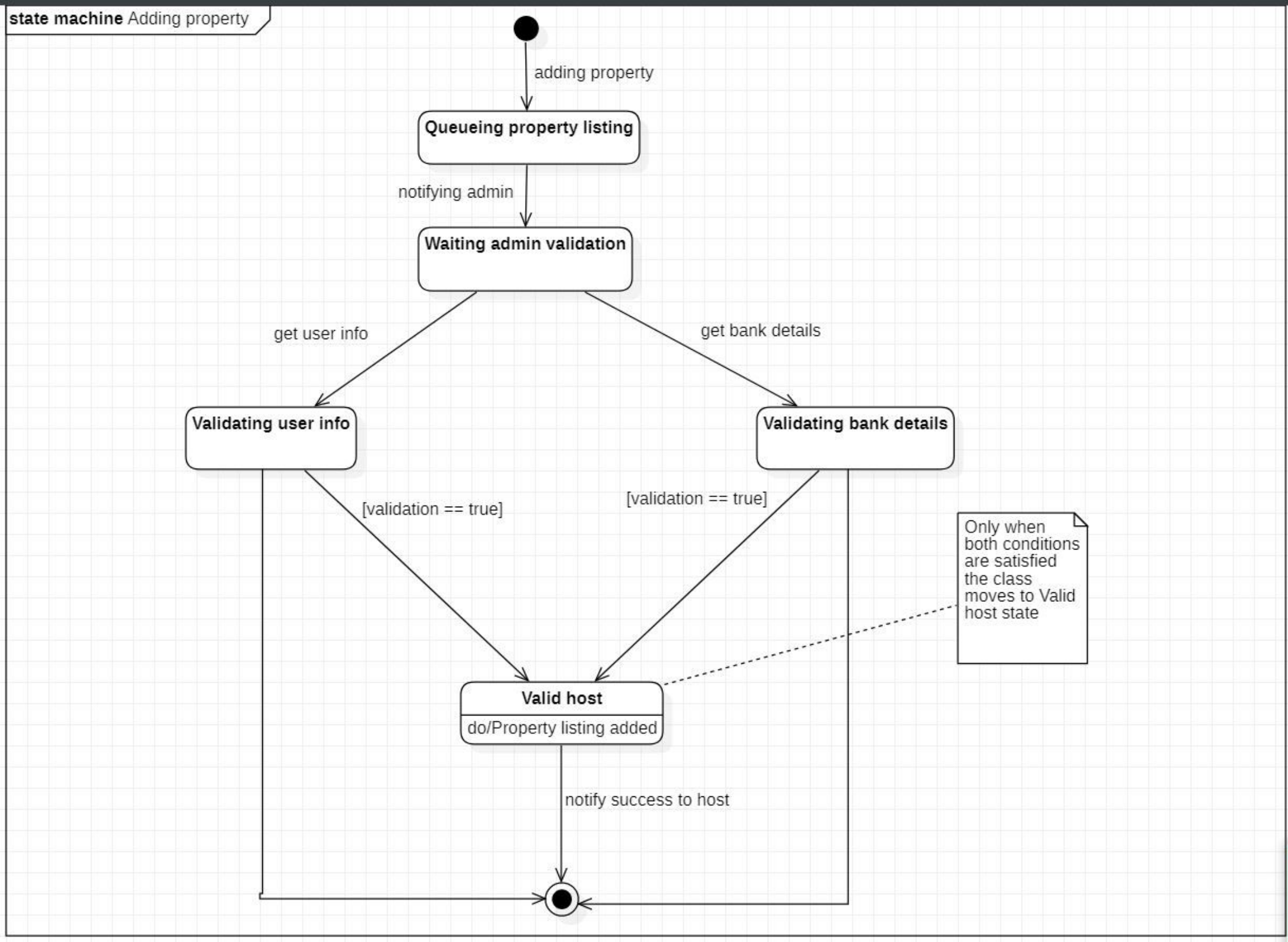
State Diagram 3: Adding property

Classes: Property listing

Description: In case of adding a new property to the property listing, admin needs to validate user and the bank details, after which the property would be added to the listings.

Flow of diagram:

1. Event: Adding to property listing
   Response: move to Queueing property listing
2. Event: Notifying admin
   Response: moves to Waiting admin validation
3. Event: Gets user information and bank details
   Response: move to Validating user info -> Valid Host (if true) / Validating bank details -> Valid Host (if true)
4. Event: If each information is valid, then notify success
   Response: move to final state


state machine Adding property
adding property
Queueing property listing
notifying admin
Waiting admin validation
get user info
get bank details
Validating user info
Validating bank details
[validation == true]
[validation == true]
Only when both conditions are satisfied the class moves to Valid host state
Valid host
do/Property listing added
notify success to host

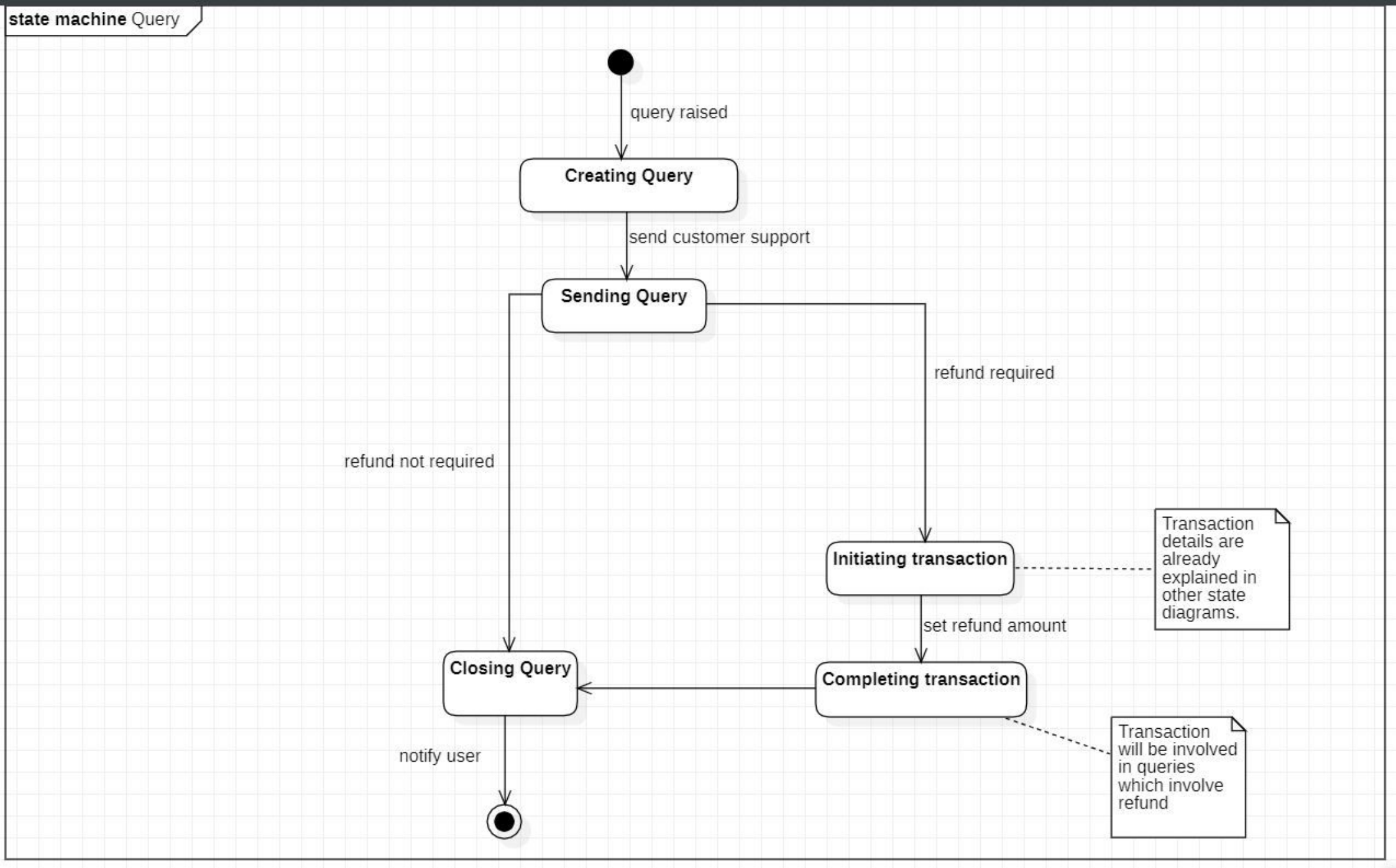
State Diagram 4: Query

Classes: Queries

Description: The query raised by user is taken up by customer support which is then checked for refund and if required then the payment is processed and then the query closed.

Flow of diagram:

1. Event: if query is raised by the user
   Response: moves to Creating Query
2. Event: the query is forwarded is sent to customer support
   Response: moves to Sending Query (waits for response)
3. Event: If the refund is required
   Response: moves to initiate transaction (Transaction state diagram has already been covered)-> Completing transaction-> Closing Query
4. Event: If the refund not required
   Response: moves to Closing Query
5. Event: Once the Closing Query state is reached
   Response: moves to final state


state machine Query
query raised
Creating Query
send customer support
Sending Query
refund required
refund not required
Initiating transaction
Transaction details are already explained in other state diagrams.
set refund amount
Closing Query
Completing transaction
Transaction will be involved in queries which involve refund
notify user

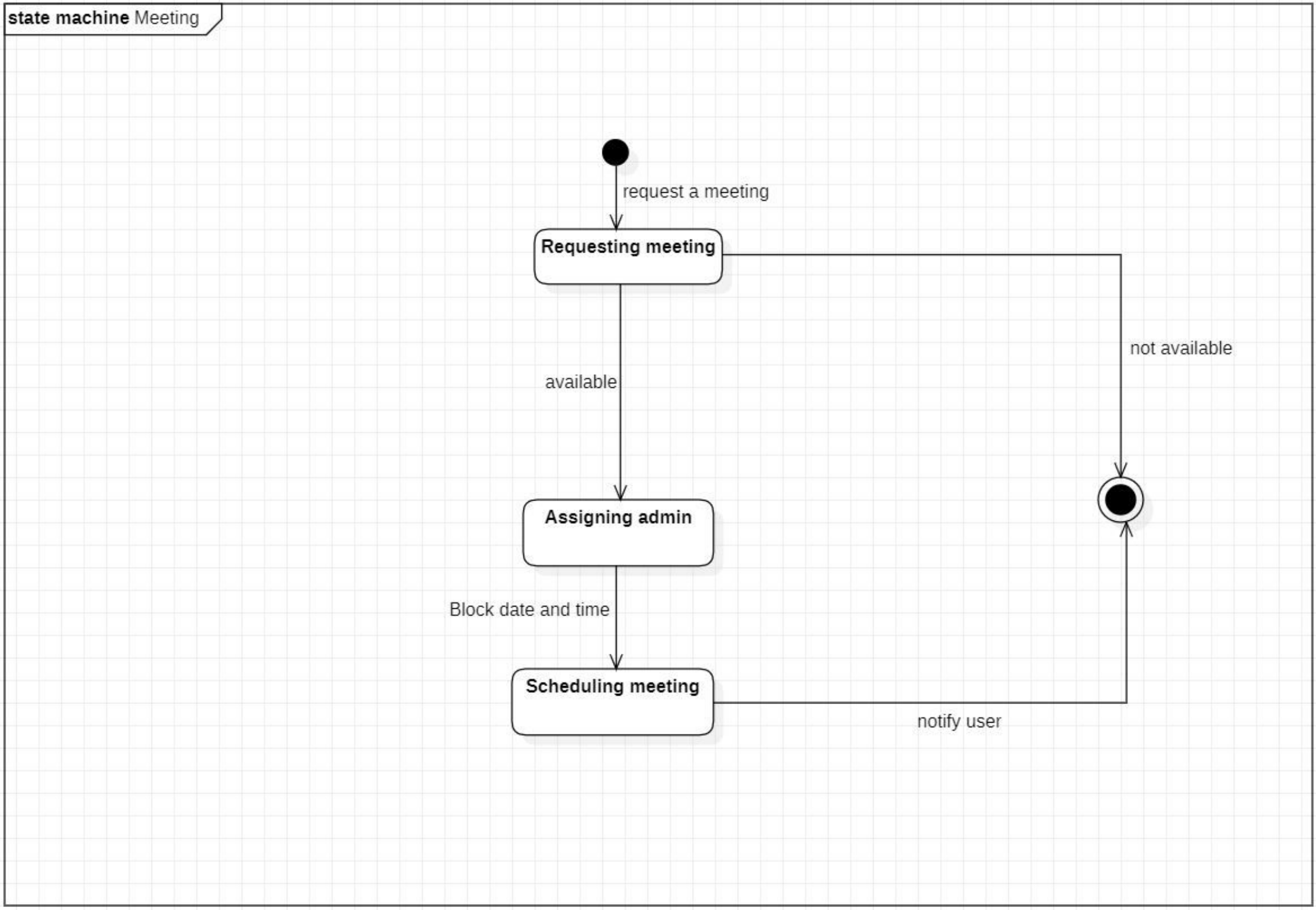
State Diagram 5: Meeting

Classes: Meeting

Description: If the travel agency wants to schedule a meeting with the admin then the meeting object is called to schedule the meet.

Flow of diagram:

1. Event: If the meeting is requested by Travel agency
   Response: moves to Request Meeting
2. Event: If meeting slot if available/ not available
   Response: moves to Assigning admin / moves to final state (notifies user)
3. Event: The meeting date and time is being blocked
   Response: moves to Scheduling meeting
4. Event: Scheduled meeting is notified to the user
   Response: moves to the final state


state machine Meeting
request a meeting
Requesting meeting
not available
available
Assigning admin
Block date and time
Scheduling meeting
notify user

ACTIVITY DIAGRAM

Activity diagram is another important behavioural diagram in UML diagram to describe dynamic aspects of the system. Activity diagram is essentially an advanced version of flow chart that modeling the flow from one activity to another activity.

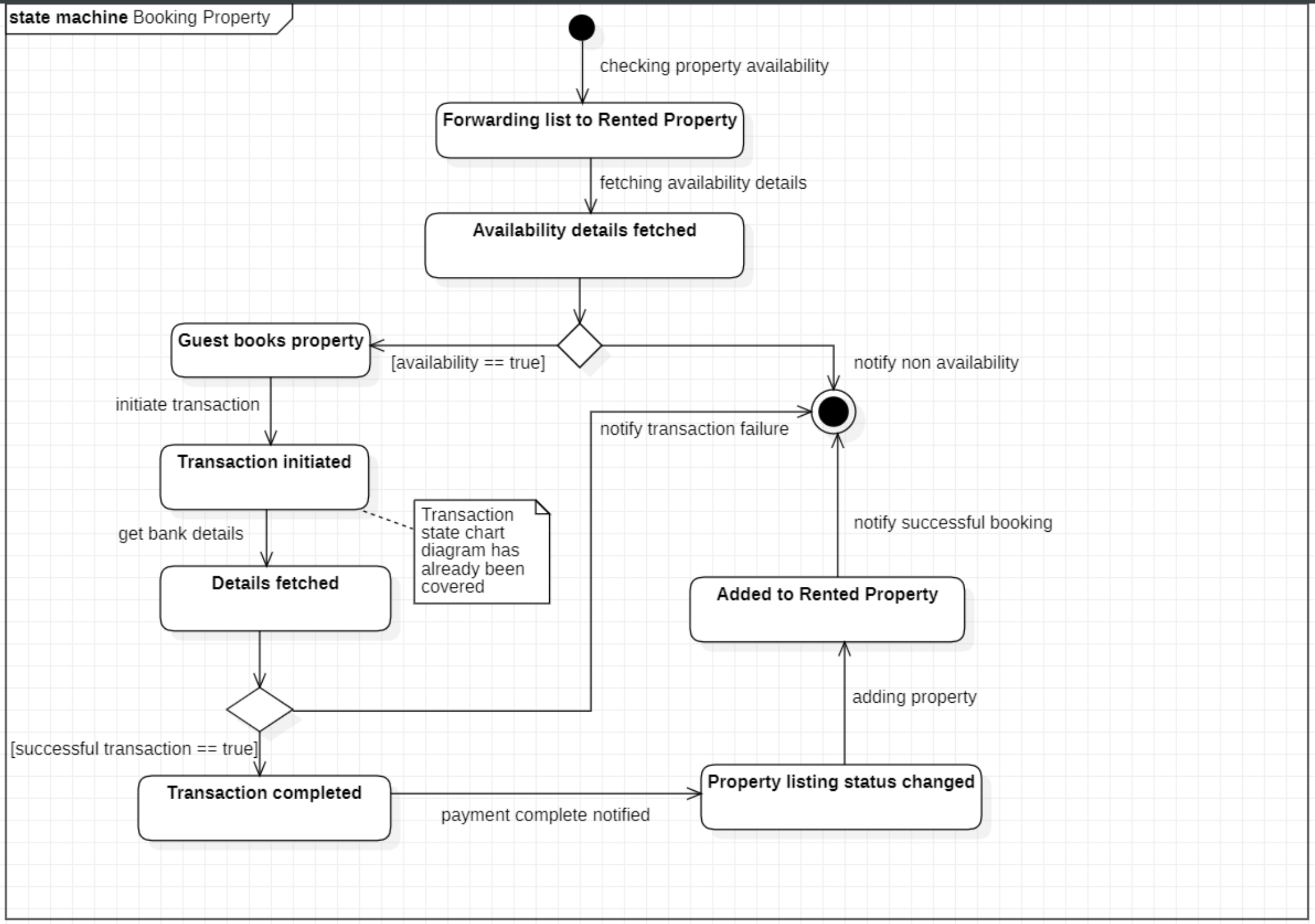
Activity Diagrams describe how activities are coordinated to provide a service which can be at different levels of abstraction.

Activity diagram 1: Booking a property

Objects Involved: Guest, Property Listings, Rented Property and Transaction.

Scenario:

- The Guest checks if the property is available for the given dates.
- If the property is not available, the flow of activities stop and final stage is reached.
- If the property is available for the given dates, the Guest continues to Book the property.
- The guest initiates a Transaction.
- User details are fetched by the Transaction class.
- If the Transaction fails the flow of activities stop and final stage is reached.
- If the Transaction is successful, the property listing status is changed and its moved to Rented Property.

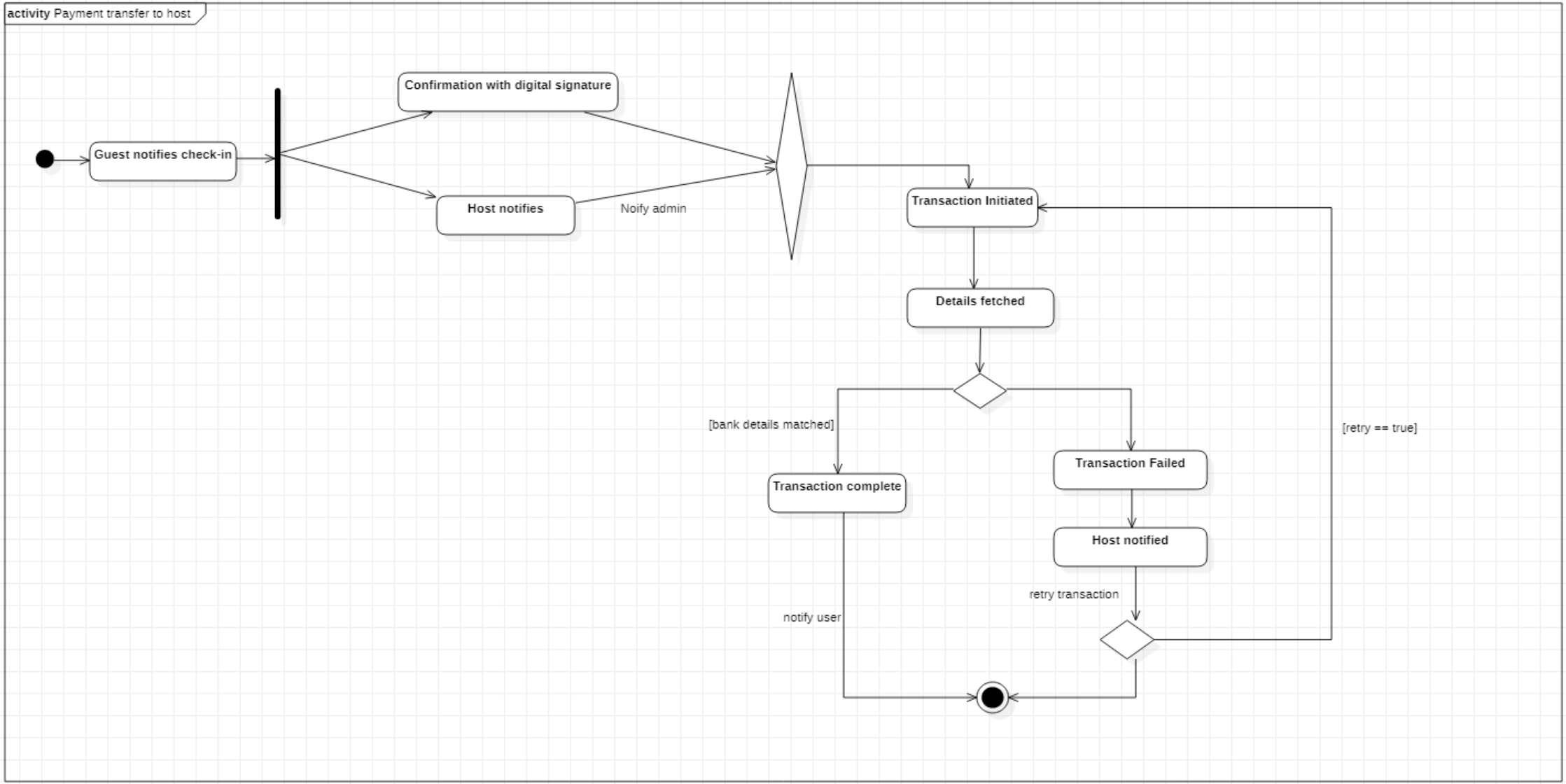

state machine Booking Property
checking property availability
Forwarding list to Rented Property
fetching availability details
Availability details fetched
Guest books property
[availability == true]
notify non availability
initiate transaction
notify transaction failure
Transaction initiated
Transaction state chart diagram has already been covered
get bank details
notify successful booking
Details fetched
Added to Rented Property
adding property
[successful transaction == true]
Transaction completed
payment complete notified
Property listing status changed

Activity Diagram 2: Payment transfer to Host

Objects involved: Guest, Host, Admin, Transaction, User details.

Scenario:

- The guest notifies the Host and the Admin concurrently on check-in.
- The Host notifies the Admin of the Guest check in after confirmation with digital signature.
- The Admin initiates transaction.
- User Bank details are then fetched from the User details class.
    - If the bank details match the transaction is completed and final stage is reached.
    - If the bank details do not match, the Host is notified and given an option to retry the transaction
        - If the Host choses to retry then the transaction is initiated again.
        - If the Host doesn’t retry then the transaction reaches the final stage.

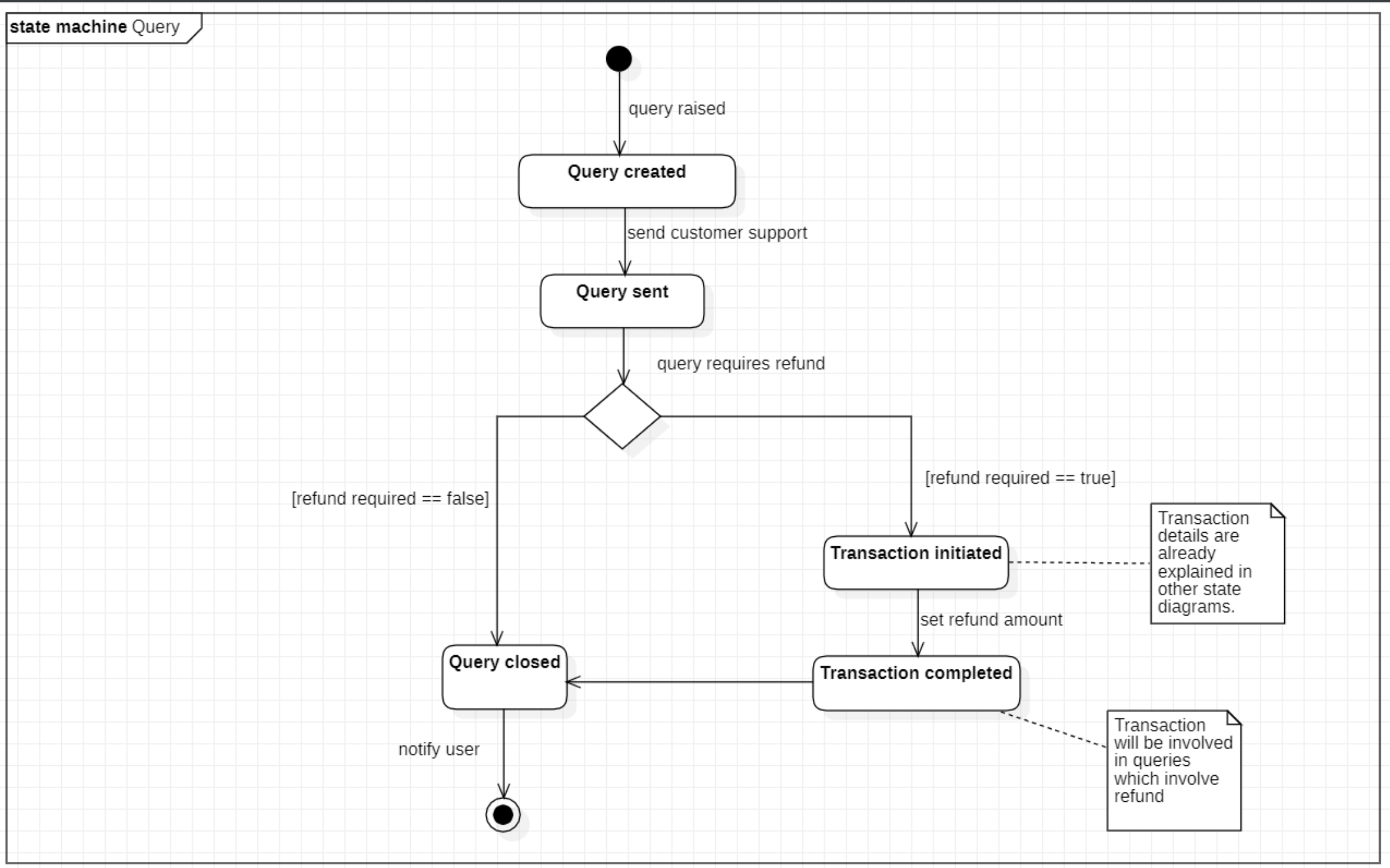

activity Payment transfer to host
Guest notifies check-in
Confirmation with digital signature
Host notifies
Noify admin
Transaction Initiated
Details fetched
[bank details matched]
Transaction complete
notify user
Transaction Failed
Host notified
retry transaction
[retry == true]

Activity Diagram 3: Query

Objects involved: Queries, Customer Support and Transaction

Scenario:

- The user raises query then query is created in Queries object and query is sent to Customer Support.
- The Customer Support returns notifying whether a refund is required or not
- If the refund is not required measures are taken and the Query is closed and the Queries object ends
- If the refund is required then the Customer Support initiates Transaction object which completes after successful transfer of money to the user. The completed transaction triggers Query to Query closing.
- From Query closing, the Queries moves to end state terminating Customer Support and Transaction objects.

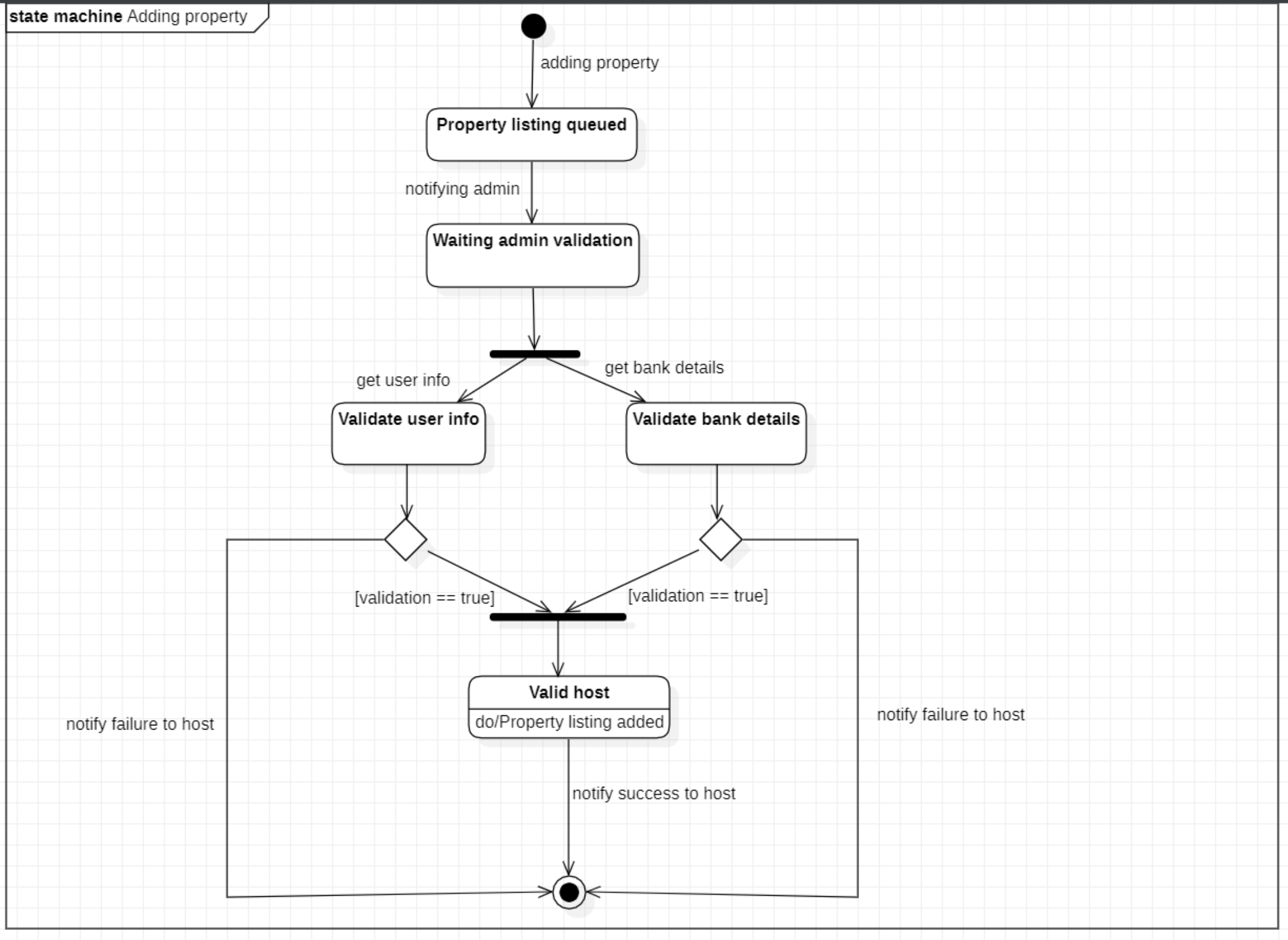

state machine Query
query raised
Query created
send customer support
Query sent
query requires refund
[refund required == false]
[refund required == true]
Transaction initiated
Transaction details are already explained in other state diagrams.
set refund amount
Query closed
Transaction completed
Transaction will be involved in queries which involve refund
notify user

Activity diagram 4: Adding Property

Objects Involved: Host, Property Listing and Admin

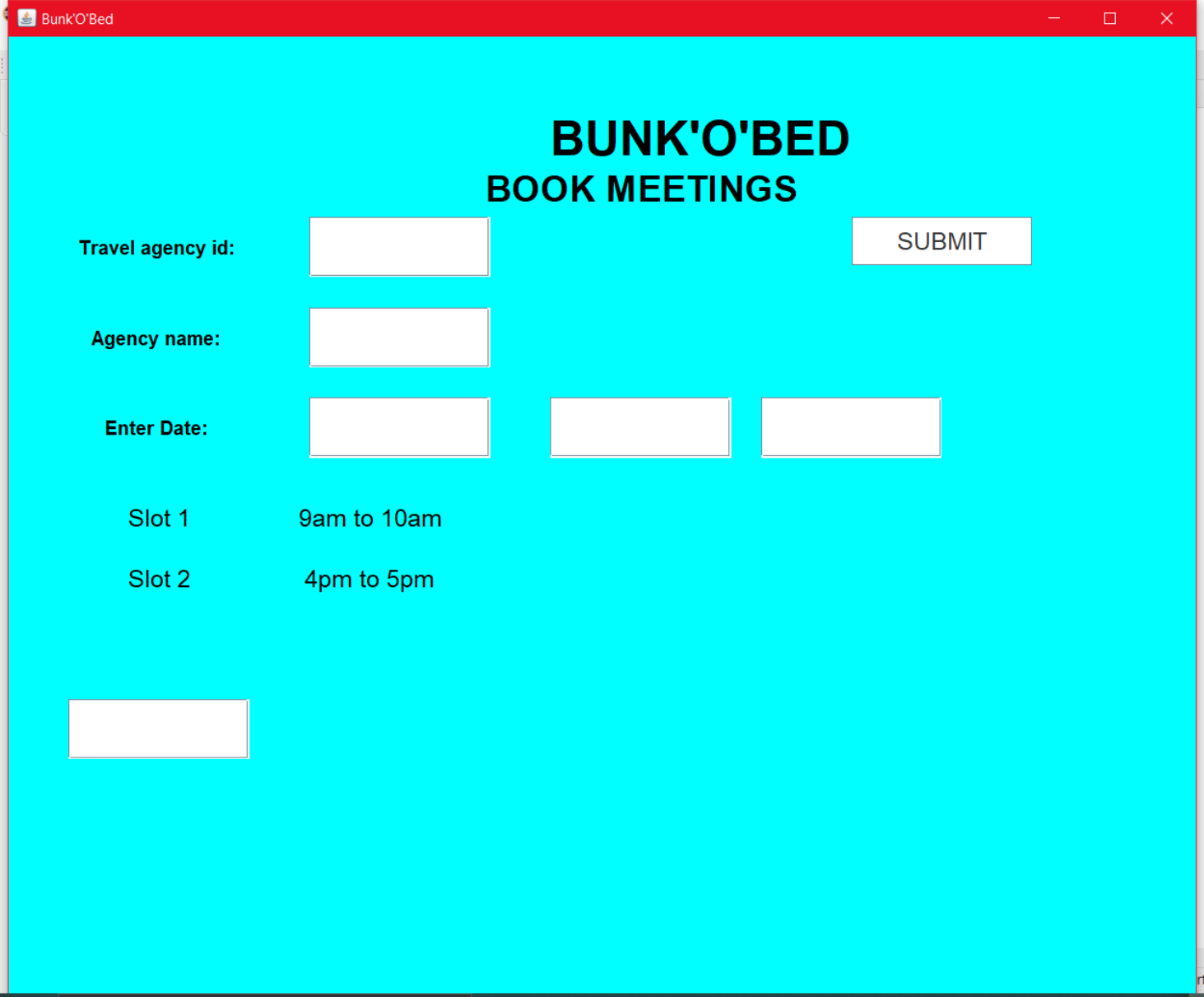
Scenario:

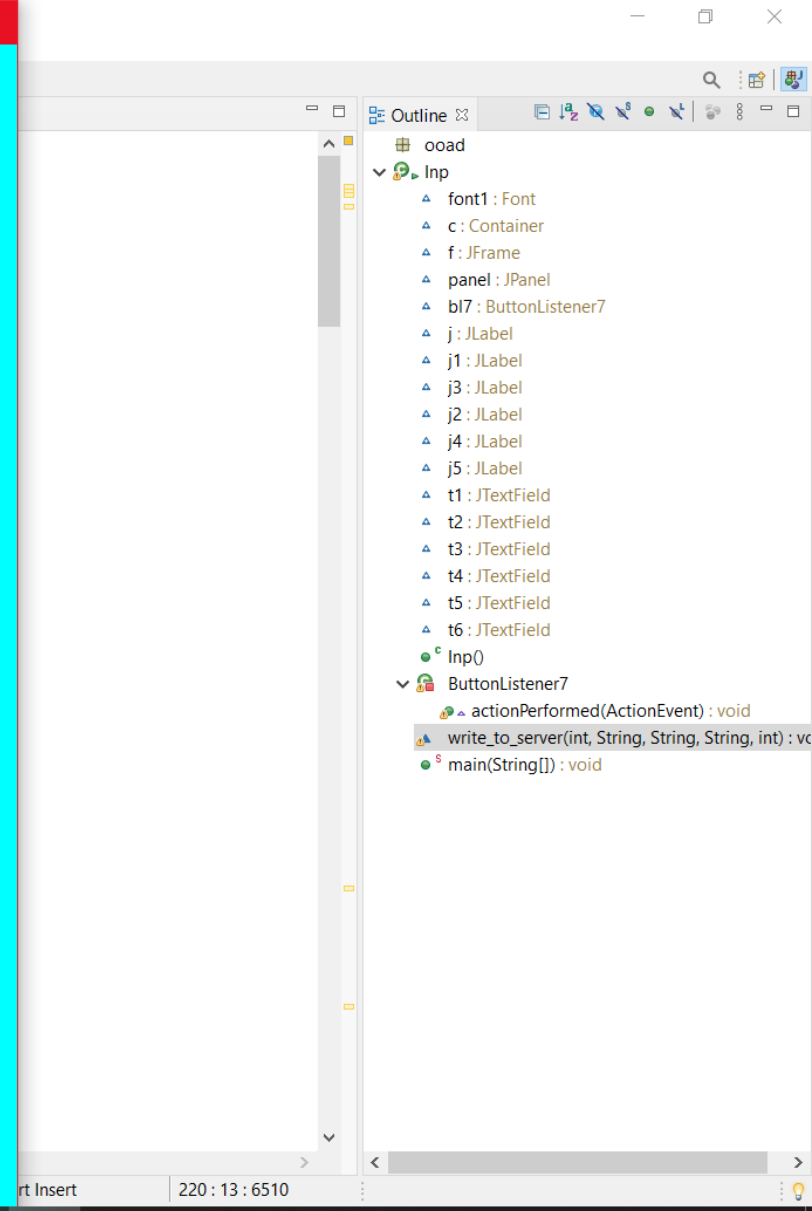
- If Host wants to add property, then Property Listing object gets triggered and moves to Property Listing queued
- In turn triggers Admin object and waits for admin validation
- Admin validates both user information and bank details of the user
- If both the information is valid then the Host is valid and the Property listing object is notified which adds the queued property listing.
- If either of the information is not valid, the Host is notified the failure
- In both cases, the objects terminate and move to the final/end state.


state machine Adding property
adding property
Property listing queued
notifying admin
Waiting admin validation
get user info
get bank details
Validate user info
Validate bank details
[validation == true]
[validation == true]
Valid host
do/Property listing added
notify failure to host
notify failure to host
notify success to host

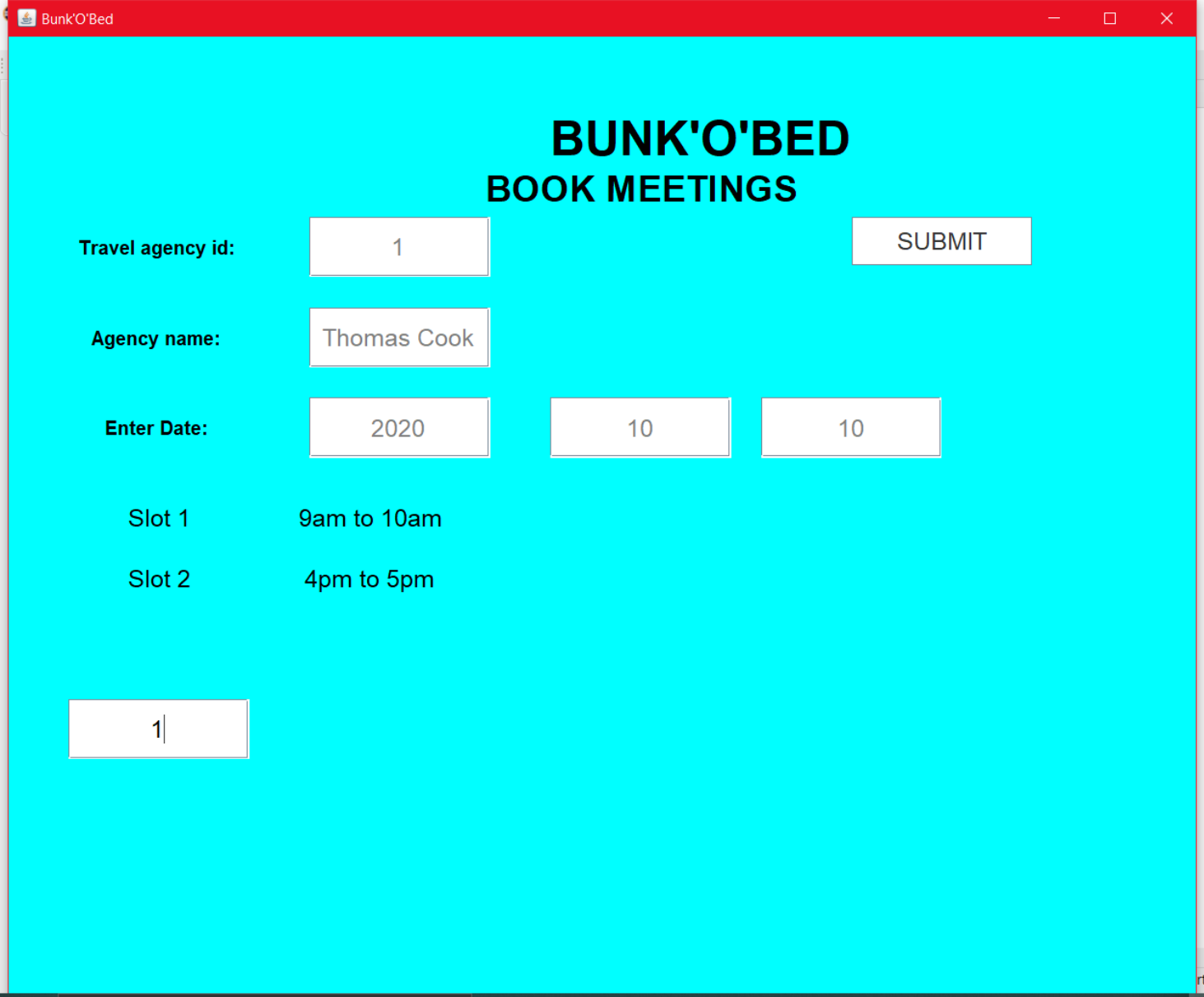
IMPLEMENTATION

A single use case has been identified with the use of MySQL and various Java libraries.


Bunk'O'Bed
BUNK'O'BED
BOOK MEETINGS
Travel agency id:
SUBMIT
Agency name:
Enter Date:
Slot 1
9am to 10am
Slot 2
4pm to 5pm


Outline
ooad
Inp
font1 : Font
c : Container
f : JFrame
panel : JPanel
bl7 : ButtonListener7
j : JLabel
j1 : JLabel
j3 : JLabel
j2 : JLabel
j4 : JLabel
j5 : JLabel
t1 : JTextField
t2 : JTextField
t3 : JTextField
t4 : JTextField
t5 : JTextField
t6 : JTextField
Inp()
ButtonListener7
actionPerformed(ActionEvent) : void
write_to_server(int, String, String, String, int) : vo
main(String[]) : void
rt Insert
220 : 13 : 6510


Type here to search
ENG
3:54 AM
07-Nov-20



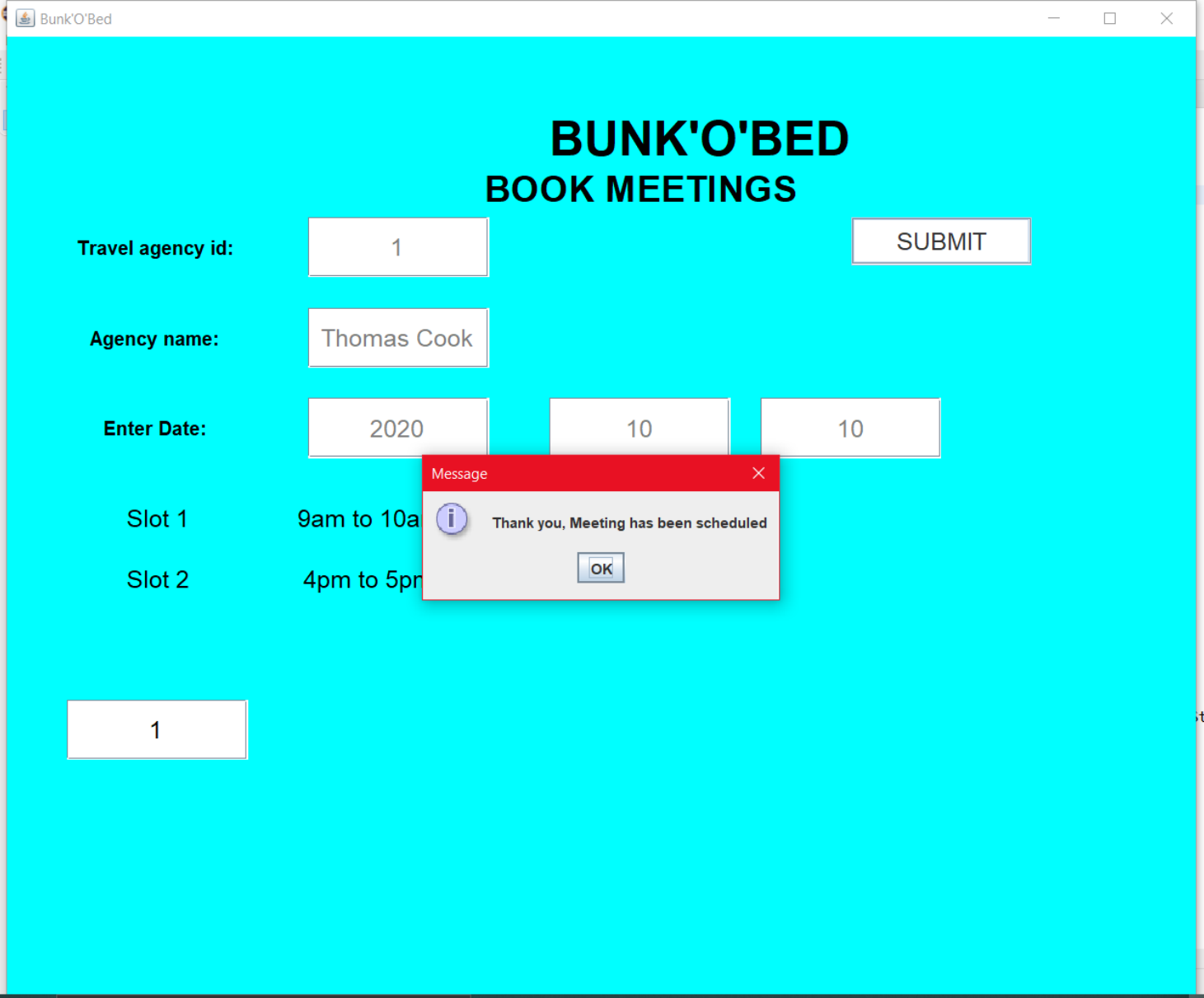
Outline

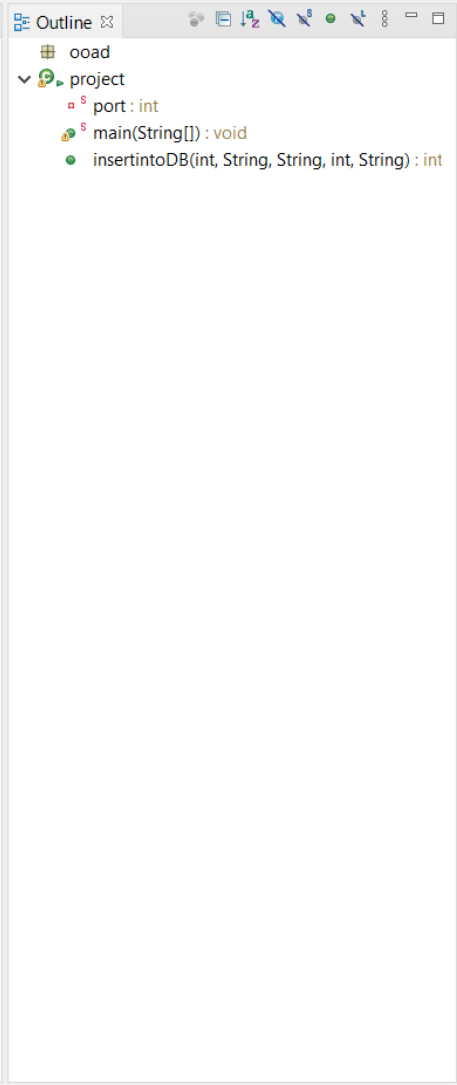
- ooad
- Inp
  - font1 : Font
  - c : Container
  - f : JFrame
  - panel : JPanel
  - bl7 : ButtonListener7
  - j : JLabel
  - j1 : JLabel
  - j3 : JLabel
  - j2 : JLabel
  - j4 : JLabel
  - j5 : JLabel
  - t1 : JTextField
  - t2 : JTextField
  - t3 : JTextField
  - t4 : JTextField
  - t5 : JTextField
  - t6 : JTextField
  - Inp()
  - ButtonListener7
    - actionPerformed(ActionEvent) : void
    - write_to_server(int, String, String, String, int) : vo
  - main(String[]) : void

rt Insert | 220 : 13 : 6510

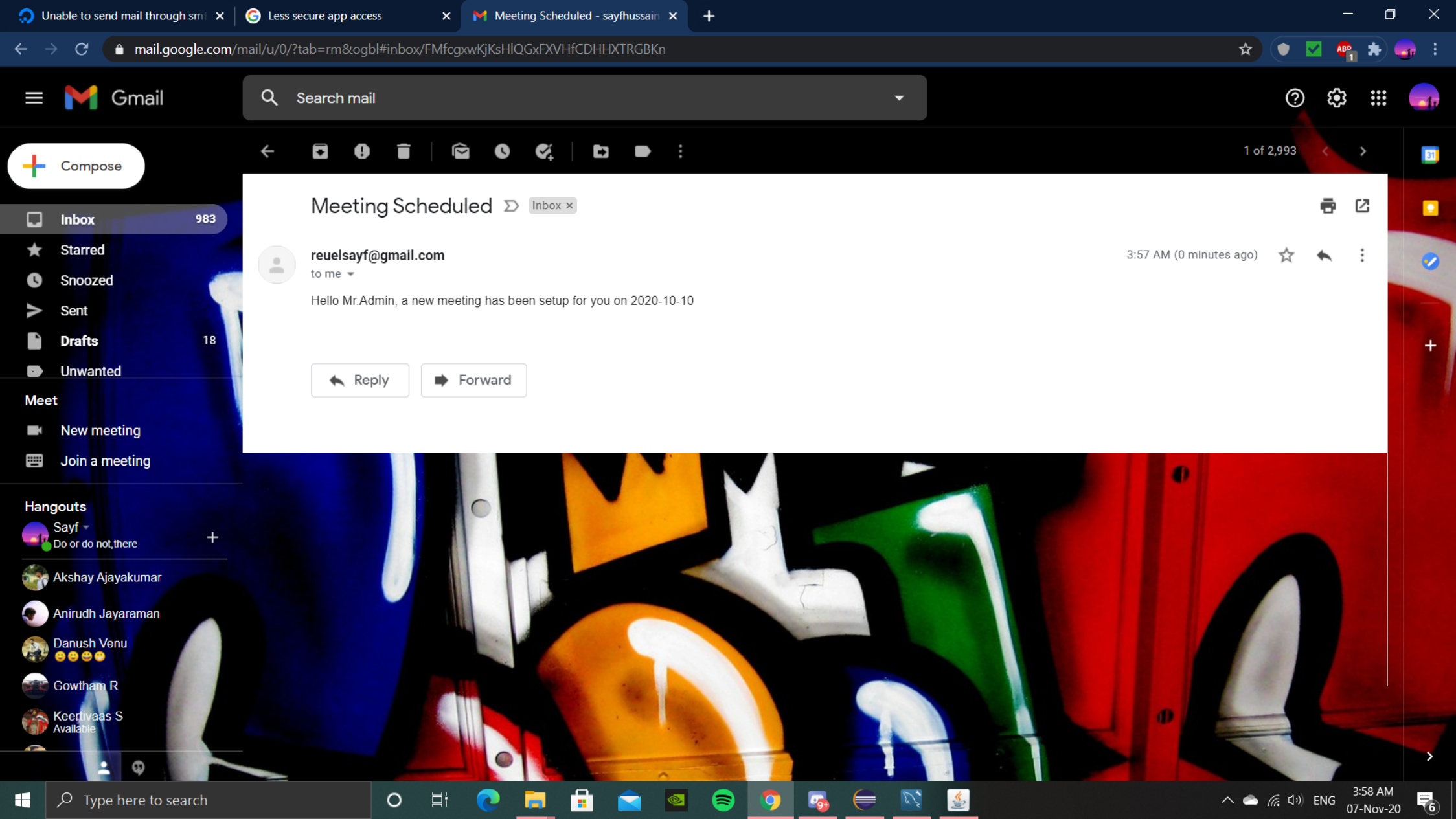
Type here to search

ENG 3:55 AM 07-Nov-20


Bunk'O'Bed
BUNK'O'BED
BOOK MEETINGS
Travel agency id:
1
SUBMIT
Agency name:
Thomas Cook
Enter Date:
2020
10
10
Message
Thank you, Meeting has been scheduled
OK
Slot 1
9am to 10a
Slot 2
4pm to 5pr
1


Outline
ooad
project
port : int
main(String[]) : void
insertintoDB(int, String, String, int, String) : int


Type here to search
ENG
3:56 AM
07-Nov-20


Unable to send mail through smt
Less secure app access
Meeting Scheduled - sayfhussain
mail.google.com/mail/u/0/?tab=rm&ogbl#inbox/FMfcgxwKjKsHlQGxFXVHfCDHHXTRGBKn
Gmail
Search mail
Compose
Inbox 983
Starred
Snoozed
Sent
Drafts 18
Unwanted
Meet
New meeting
Join a meeting
Hangouts
Sayf
Do or do not,there
Akshay Ajayakumar
Anirudh Jayaraman
Danush Venu
Gowtham R
Keerthivaas S
Available
1 of 2,993
Meeting Scheduled
Inbox ×
reuelsayf@gmail.com
to me
3:57 AM (0 minutes ago)
Hello Mr.Admin, a new meeting has been setup for you on 2020-10-10
Reply
Forward
Type here to search
ENG
3:58 AM
07-Nov-20