Models Diagram:

User:

- email field is the field that contains the user's email and we use this field instead of username for user login so every user have a email and it is unique
- phone field is the user's phone number and this field is optional and must follow the format mentioned in the accounts section of the endpoints
- avatar field the avatar or portrait image of the user and this will be set to the default portrait image if user does not provide a image
- first_name field contains the user's first name and it is optional
- last_name field contains the user's last name and it is optional
- comments field contains the list of comments about the user and it is a list of comments object
- rating field is the overall rating of the user

notifications:

- receiver field is an User object that corresponds to the receiver of the notification
- content field contains the actually content of the notifications
- created_time field contains the time that the notification is created and this field is not changed once the notification is created and the default value is set to the time when this notification is created
- content_object field is a field that is either property, reservation, or User. If the notification is sent because of a comment to a user, then the User that is being commented is passed (the User that we can find this comment on). If the notification is sent because of a reservation request or because a reservation is coming up, then the reservation is passed. If the notification is sent because of a comment to a property, then the property being commented is returned

reservations;

- host field is an User object that corresponds to the host of the reservation
- reserver field is an User object that corresponds to the user that place this reservations
- property field is an property object that corresponds to the property that this reservation reserve
- reserve_time field contains the time where this reservation is placed and this field is not being modified and it is always set to the time that the reservation is placed
- start_day contains the date of the start of this reservation
- end_day contains the date of the end of this reservation
- status contains the current status of the reservation
- notified is a boolean value that indicate whether this reservation has been notified to be upcoming or not

property:

- owner field is the owner of the property

- address field is the address of the property
- comments field contains the list of all comments that is associated with this property
- rating field is the rating of this property and we keep 2 decimal places for the rating and it is the average of all ratings for this property
- available_start field is the start day of the available interval for this property that the owner entered
- available_end field is the end day of the available interval for this property that the owner entered
- amenities field contains the list of amenities for this property and we store it as a string of all amenities for this property each separated by a comma
- image_cover field contains the cover image for the property (The image that will show up when the user search for properties)
- field image1 to image5 are 5 images that the user can upload for their properties so overall, a user can upload at most 6 images for a property
- num_guest field contains the number of guests that can live in this property
- description field contains the description of the property
- price field contains the price of the property

comments:

- sender field is an User object that corresponds to the user who send or post the comment
- content field contains the actual content of the comment
- created_time field is the time where this comment is created and it is not modified
- is_reply is a boolean value that indicate whether the comment is a reply or not true means the comment is a reply and false otherwise
- parent_comment is another comment object that is set only when the current comment is a reply and thus, this field is set to the comment that this reply replies and if the comment is a comment not a reply, this field is set to null
- rating contains the rating that the user give associated with the comment
- receiver is an User object that corresponds to the user who receives this comment or in other words, the receiver is the user that can reply to this comments
- content_object is either User or property where the comment is either a comment on User or on property

Endpoints:

To run our startup.sh to activate our virtual environment, you need to cd into the directory with the file startup.sh and run it with source startup.sh otherwise the virtual environment won't be activated and then, run ./run.sh to start the server

Accounts:

Important Note when entering phone number:

- The phone number must be entered using the international format which is +1XXXXXXXXXX for phone numbers in US where we have +country code then the actual phone number otherwise an error will be returned

Endpoint: /accounts/register/
Methods: POST
Fields/Payloads: email, phone, first_name, last_name, avatar, password1, password2
Authentication Needed: No
Validation Error:

- “The two password fields didn’t match” when password1 not same as password2
- “This password is too short. It must contain at least 8 characters" when password1 or password2 is shorter than 8 characters
- “A user with this email already exists” when a user registered with this email already
- “This field is required” when email, password1 or password2 is missing
- "Enter a valid email address." when the email address is not valid
- “The phone number entered is not valid” when the phone number is not entered in the format mentioned in the important note section of account endpoints

On Success:

- Create the new user and if the avatar field is not provided, it will be set to the default profile avatar

Endpoint: /accounts/obtainToken/
Methods: POST
Fields/Payloads: email, password
Authentication Needed: No
Validation Error:

- “No active account found with the given credentials” when the email and password does not match a record in database

On Success:

- Return an access token and a refresh token where the user can use the access token in the header to run any endpoints that needs authentication

Endpoint: /accounts/update/
Methods: PUT
Fields/Payloads: email, phone, first_name, last_name, avatar
Authentication Needed: Yes (Need access token in header)
Validation Error:

- “A user with this email already exists” when the email is used by someone else
- "Enter a valid email address." when the email address is not valid
- “The phone number entered is not valid” when the phone number is not entered in the format mentioned in the important note section of account endpoints

On Success:

- The account of the currently logged in user’s profile will be changed according to the payload
- The email field can be empty string or not provided and we treat them as not wanting to change the email so the email stays the same and the same rule apply for avatar as well but all other fields are set to empty string if empty string is passed in
- If any of the other fields are not passed in, then we treat that as not wanting to change those field (Note that not passing in the field is not the same as passing in empty string)

Endpoint: /accounts/change_password/
Methods: PUT
Fields/Payloads: old_password, password1, password2
Authentication Needed: Yes (Need access token in header)
Validation Error:

- “The two password fields didn’t match” when password1 not same as password2
- “This password is too short. It must contain at least 8 characters" when password1 or password2 is shorter than 8 characters
- “This field is required” when old_password, password1 or password2 is missing

On Success:

- The password of the currently logged in user is changed to the new password provided
- On success, the information about the user that changed the password will be returned

Endpoint: /accounts/detail/
Methods: GET
Fields/Payloads: None
Authentication Needed: Yes (Need access token in header)
Validation Error:

- None

On Success:

- Returns the information about the currently logged in user

Endpoint: /accounts/refreshToken/

Methods: POST
Fields/Payloads: refresh (The refresh token given when obtain token)
Authentication Needed: Yes
Validation Error:

- Returns Token is invalid or expired when an invalid token is given
- Return this field is required in no refresh token is provided in the body

On Success:

- Returns a new access token

Notifications:

Endpoint: /notifications/list/
Methods: GET
Authentication Needed: Yes (Need access token in header)
Parameter: page (used for pagination where you can set page to the page number that you want to look at)
Validation Error:

- None

On Success:

- Returns a list of notifications that is sent to the user logged in ordered by the time these notifications are created and ordered in descending order meaning most recent notifications is at top

Endpoint: /notifications/read/<int:pk>/
Methods: GET
Authentication Needed: Yes (Need access token in header)
Validation Error:

- If the id that is passed in is not a valid notification id, then not found will be returned

On Success:

- Returns the detail of notification with id pk which is passed in inside the url

Endpoint: /notifications/clear/<int:pk>/
Methods: DELETE
Authentication Needed: Yes (Need access token in header)
Validation Error:

- If the id that is passed in is not a valid notification id, then not found will be returned

On Success:

- Deletes the notification with id pk which is passed in inside the url

Endpoint: /notifications/notupcoming/
Methods: POST
Authentication Needed: Yes (Need access token in header)
Validation Error:

- None

On Success:

- If there is at least one reservation that the logged in user reserved that start at tomorrow, then one notification is created for each reservation and returned in json format where receiver is the current user's id, content is the message and the reservation is the reservation id
- Note that if there are multiple reservations made by the current user that start from tomorrow, then there are multiple notifications created and the endpoint will return an array of notification information
- If there is no reservation made by the current user that starts tomorrow, then return {"message": "No upcoming reservation for the logged in user"}

Property:

Important Note:

- For any of the endpoints that accept amenities, you should structure your input as a string of all amenities separated by comma. For example, wifi,kitchen,washer is a property that has wifi, kitchen and washer and nothing else.

Endpoint: /property/all/
Methods: GET
Authenticated: No
Class: Paginator
Parameters: amenities, num_guest, address, available_start, available_end, order_by, page (used for pagination where you can set page to the page number that you want to look at)
Validation Error:

- amenities in list of valid amenities ('wifi', 'kitchen', 'washer', 'dryer', 'air_cond', 'heating', 'hair_dryer', 'TV', 'pool','hot_tub', 'bbq', 'gym')
- available_start and available_end are valid datetime variables (format: YYYY-MM-DD)
- order_by in list of valid order_by ('price', 'rating')

On Success:

- Return all properties in database filtered by parameters and ordered by order_by

Endpoint: /property/details/<int:pk>/
Authenticated: No
Method: GET
Validation Error: None
On Success: return the detail of the property with id=pk.

Endpoint: /property/update/<int:property_id>/
Method: PUT
Authenticated: Yes (only owner of the property could edit it)
Fields: address, description, image1, image2, image3, image4, image5, image_cover, price, available_start, available_end, num_guest, amenities
Validation Error:

- available_start before available_ends
- num_guest positive
- amenities are valid
- price is positive

On success: update the property, return the updated property

Endpoint: /property/create/
Method: POST
Authenticated: Yes
Fields: address, price, available_start, available_end, num_guest, amenities, image_cover, description, image1, image2, image3, image4, image5
Validation Error:

- address, image_cover, price, available_start, available_ends, num_guest should be none empty
- available_start before available_ends
- num_guest positive
- amenities are valid
- price is positive

On Success: add the property

Endpoint: /property/delete
Method: DELETE
Authenticated: Yes (a user could only delete the properties they owns)
Validation Error: None
On success:

- delete the chosen property
- send notification to users that had reservations.

Reservations:

Note:

All reservations' start day and end day are inclusive meaning that if you want to reserve from start_day to end_day, then we assume that for both start_day and end_day, the guest still stayed in the property

Endpoint: /reservations/list/<str:user_type>/<str:state>/
Method: GET
Authenticated: Yes (a user could list the reservations that they owns)
Parameter: page (used for pagination where you can set page to the page number that you want to look at)
Error Check:

- Return Not Found if the state or the user_type is not a valid input since we do not have a page for a invalid input

On success:

- List all the reservations satisfying the user input's condition

Note:

- user_type can only be either "host" or "guest" or "both" where host means the host of all the reservations are the currently logged in user and guest means the reserver of all the reservations listed are the currently logged in user and both means the reservations listed are all reservations where the logged in user is involved regardless of whether the user is host or guest
- State can only be one of the 9 states which are "pending", "denied", "expired", "approved", "canceled", "cancel_pending", "terminated", "completed" or "all" where all means all states

Endpoint: /reservations/reserve/
Method: POST
Authenticated: Yes
Fields: property_id, start_day, end_day (start_day and end_day must be in format YYYY-MM-DD)
Error Check:

- Return this field is required if any of the three fields are not given or empty
- Return "There is no property with the given id" if the property_id is not in the database
- Return "The format of the day is invalid. Should be YYYY-MM-DD." if the format of start_day or end_day is not following the format of YYYY-MM-DD

- Return “Your start day cannot be later than your end day of reservation.” if the start_day is later than end_day. Note that the key is date in this case since this could be an error with start_day or end_day so we have a new key for this error
- Return “Your start day cannot be earlier than the available start day of the property.” if the start_day is before the available start day of the property
- Return “Your end day cannot be later than the available end day of the property.” if the end_day is after the available end day of the property
- Return “There is time conflicts on this property. Please choose another time interval.” if the time interval entered conflicts with a reservation that is made(we consider only reservations that are in status pending, approved or cancel_pending since all other reservations are no longer active so it will not conflict with any reservations)

On success:

- A new reservation on the property with the property_id is created starting from start_day to end_day and a notification is sent to the host of the property letting them know that there is a reservation request pending

Endpoint: /reservations/cancel/<int:pk>/
Method: PUT
Authenticated: Yes
Error Check:

- Return Not found if the logged in user is not the reserver of the reservation
- Return this reservation cannot be canceled if the status of the reservation is not pending or approved

On success:

- If the original status is pending, then the reservation will be canceled
- If the original status is approved, then the reservation will be cancel_pending and a notification will be sent to the host of the reservation mentioning the reserver want to cancel the reservation

Endpoint: /reservations/pendingAction/<int:pk>/
Method: PUT
Authenticated: Yes
Field: action (either approve or deny)
Error Check:

- Return Not found if the logged in user is not the host of the reservation or the reservation is not in pending status
- Return this field is required if no action is provided
- Return this is invalid action if the action is not approve or deny

On success:

- If action is approve, the status is changed to approved, otherwise status is changed to denied and in either case, a notification will be sent to the reserver saying their reservation request is either approved or denied

Endpoint: /reservations/cancelAction/<int:pk>/
Method: PUT
Authenticated: Yes
Field: action (either approve or deny)
Error Check:
- Return Not found if the logged in user is not the host of the reservation or the reservation is not in cancel_pending status
- Return this field is required if no action is provided
- Return this is invalid action if the action is not approve or deny

On success:
- If action is approve, the status is changed to canceled, otherwise status goes back to approved and in either case, a notification will be sent to the reserver saying their cancelation request is either approved or denied

Endpoint: /reservations/terminate/<int:pk>/
Method: PUT
Authenticated: Yes
Error Check:
- Return Not found if the logged in user is not the host of the reservation
- Return this reservation is no longer active so you cannot terminate it if the status of the reservation is not pending, approved, or cancel_pending

On success:
- The reservation with id pk will be terminated and a notification will be send to the reserver of the reservation saying the reservation is terminated

Endpoint: /reservations/markExpired/
Method: POST
Authenticated: Yes
Error Check:
- None

On Success:
- Mark all reservations that the currently logged in user involves as expired if the start day of the reservation has been passed and the reservation's status remains at pending
- If no such reservation exists, then return No reservation should be marked expired

Endpoint: /reservations/markCompleted/
Method: POST
Authenticated: Yes
Error Check:

- None

On Success:

- Mark all reservations that the currently logged in user involves as completed if the start day of the reservation has been passed and the reservation's status is approved (here we assume that if the status is approved and the start day already passed, then the reserver went to the property) or cancel_pending (we assumed that if the cancelation request is not responded by the host, then the guest actually went to the property)
- If no such reservation exists, then return No reservation should be marked completed

Comments:

Endpoint: /comments/property/<int:id>/comments/
Method: POST
Authenticated: Yes (a user who have a completed or terminated reservation can comments that property)
Field: content, rating
Error Check:

- Return rating error if you don't input rating or rating exceed value
- Return content error if you don't input anything

On success:

- Return a json format with "sender_id, content, is_reply, object_id, receiver_id, rating"

Note:

- After create the comments with the rating. The receiver will receive a notification, and the rate of the property will be adjust according to new rating.
- the ratting can only be integer number between 1 to 5.

- Only the reservation with the completed state or terminated state can comment under the property page. For one reservation, user can only post one comments under the property page.

Endpoint: /comments/user/<int:id>/comments/
Method: POST
Authenticated: Yes (a owner who has a completed reservation can comments that user who apply that resercation)
Field: content, rating
Error Check:
- Return content error if you don't input anything
- Return rating error if you don't input rating or rating exceed value

On success:
- Return a json format with "sender_id, content, is_reply, object_id and receiver_id"

Note:
- After create the comments with the rating. The receiver will receive a notification, and the rate of the user will be update according to new rating.

Endpoint: /comments/property/<int:id>/reply/
Method: POST
Authenticated: Yes (after an user comment a property, the owner can reply this user)
Field: content
Error Check:
- Return content error if you don't input anything
- If you are not the owner or the sender, you cannot reply the post

On success:
- Return a json format with "sender_id, content, is_reply, object_id and receiver_id"

Note:
- After create the reply. The receiver will receive a notification.
- There is no limitation for reply, so if the client start a comments, the owner and client can both reply to the comments and reply

Endpoint: /comments/user/<int:id>/reply/
Method: POST
Authenticated: Yes (after an ownner comment a user, the user can reply this owner)
Field: content
Error Check:
- Return content error if you don't input anything
- If you are not the owner or the sender, you cannot reply the post

On success:

- Return a json format with “sender_id, content, is_reply, object_id and receiver_id”

Note:

- After create the reply. The receiver will receive a notification.
- There is no limitation for reply, so if the client start a comments, the owner and client can both reply to the comments and reply

Endpoint: /comments/property/<int:id>/
Method: GET
Authenticated: No
Parameter: page (used for pagination where you can set page to the page number that you want to look at)
Error Check:

- If the property not exist, raise error

On success:

- Return a json format with all comments and reply under that property with “id, sender_id,receiver_id, content, created_time, is_reply, object_id and rating”

Note:

- Any user can view the property, and all the user can review the comments of a property

Endpoint: /comments/user/<int:id>/
Method: GET
Authenticated: Yes
Parameter: page (used for pagination where you can set page to the page number that you want to look at)
Error Check:

- If the user not exist, raise error
- If the user cannot view the comment, then raise error

On success:

- Return a json format with all comments and reply under that user with “id, sender_id,receiver_id, content, created_time, is_reply, object_id and rating”

Note:

- Only host who has had or currently having a reservation with the user can view the user’s comment and of course, the user can see comments made to them so the currently logged in user can list comments about themselves