# CSC309H1 Project Phase 2 Models & Endpoints

## Endpoints

**accounts/register/**
Description: Accepts a POST request to create a new user account. Payload contains the user's information such as username, email, password, etc. The register view validates the data and creates a user object, saving it to the database.
Methods: POST
Payloads: username, password, first_name, last_name, email, phone_number (optional), isHost, avatar_file (optional)

**accounts/login/**
Description: Accepts a POST request with the user's credentials (username and password) and checks if they are valid. If valid, it generates an access token and a refresh token that can be used for authentication in subsequent requests.
Methods: POST
Payloads: username, password

**accounts/logout/**
Description: Accepts a POST request to revoke the user's refresh token. This effectively logs the user out of the application since they cannot request new access tokens without a valid refresh token.
Methods: POST
Payloads: refresh (the refresh token to be revoked)

**accounts/token/refresh/**
Description: Accepts a POST request with the user's refresh token. If the refresh token is valid, it generates a new access token that can be used for authentication in subsequent requests.
Methods: POST
Payloads: refresh (the refresh token used to request a new access token)

**accounts/profile/**
Description: This endpoint allows users to fetch and update their profile information. For a GET request, it returns the authenticated user's profile information. For a PATCH request, it updates the user's information with the provided data. The user must be authenticated with a valid access token for both operations.
Methods: GET, PATCH
Payloads (for PATCH): first_name (optional), last_name (optional), email (optional), phone_number (optional), isHost (optional), avatar_file (optional)

**properties/addproperty/**
Description: Accepts only post requests to create a property (subclass of APIView). Must be an authenticated host, the property is assigned to the logged in host. Payload contains all the characteristics of the property except for owner. The only method is a post method that validates the data before creating the object
Methods: POST
Payloads: property_name, address, group_size, number_of_beds, number_of_baths, date_created, price_night, amenities, description

**properties/editproperty/<int:pk>/**
Description: Edits the characteristics of the property with an id = pk. Must be owner of property
Methods: PATCH
Payloads: property_name, address, group_size, number_of_beds, number_of_baths, date_created, price_night, amenities, description

**properties/editingpropertyimages/<int:pk>/**
Description: Edits the images associated with the property with  id=pk. Must be the owner of the property
Methods: PATCH
Payloads: image
Note, can have multiple image keys in payload with different image values to change multiple images

**properties/deleteproperty/<int:pk>/**
Description: Deletes the property with id=pk. Must be the owner of the property
Methods: Delete
Payloads: N/A

**properties/getproperty/<int:pk>/**
Description: Gets the information for a property with id=pk.
Methods: GET
Payloads: N/A

**properties/all/**
Description: Gets the properties that match the filter criteria in sorted order.
Methods: GET
Payloads: All are optional. Groupsize, minprice, maxprice, numbeds, amenities, sort. Amenities must exist, and sort must be either ‘prasc’ for price ascending, ‘prdec’ for price descending, or ‘mgasc’ for groupsize ascending, or ‘mgdec’ for groupsize descending

**properties/createreservation/**
Description: Creates a reservation for the current user and sends a notification to the property owner for a pending reservation request.
Methods: POST
Payloads: property, start_date, end_date

**properties/deletereservation/<int:pk>/**
Description: Deletes a reservation with id=pk if the current user is the property's owner.
Methods: DELETE
Payloads: N/A

**properties/guestreservation/**
Description: Gets the reservations of the user as a guest and can match filter criteria to only show the specified status type.
Methods: GET
Payloads: status (optional)

**properties/hostreservation/**
Description: Gets the properties' reservations of the user as a host and can match filter criteria to only show the specified status type.
Methods: GET
Payloads: status (optional)

**properties/editreservation/<int:pk>/**
Description: Updates the reservation's status depending on the guest's or host's response status to the current reservation status.
Methods: PATCH
Payloads: status

**properties/usernotifications/**
Description: Get all the user's notifications.
Methods: GET
Payloads: N/A

**properties/usernotifications/delete/<int:pk>/**
Description: Delete/clear the current user's notification with id=pk.
Methods: DELETE
Payloads: N/A

**properties/usernotifications/update_read/<int:pk>/**
Description: Updates the current user's notification with id=pk is_read status to True.
Methods: PUT
Payloads: N/A

**properties/createcomment/<int:pk>/**
Description: Creates a root comment (not a reply) from a user on a property where id=pk that the user has completed a reservation at. Can only leave one root comment per reservation at that property.
Methods: POST
Payloads: message

**properties/createresponse/<int:pk>/**
Description: Creates a reply to a comment on a property where id=pk. Only the owner of the property and the guest who created the relevant root comment can participate in this thread, and the same person can't make a direct reply to their own comment (must wait for the other to create a comment first).
Methods: POST
Payloads: parent - the id of the parent comment to the reply being created

**properties/propertycomments/<int:pk>/**
Description: Gets all the comments of a property with id=pk.
Methods: GET
Payloads: N/A

**properties/usercomments/<int:pk>/**
Description: Gets all the comments of a user with id=pk.
Methods: GET
Payloads: N/A

**properties/createusercomment/<int:pk>/**
Description: Creates a comment for a guest with id=pk that has completed or reserved a booking at one of the host's properties. Host must be signed in.
Methods: POST
Payloads: message - the comment that the host wants to leave.

## Models

### Accounts

#### CustomUser

This model extends Django's AbstractUser model and represents a user in the application. It includes all the fields provided by the default User model, with additional fields like phone_number and isHost. It also defines a many-to-many relationship with Django's Group and Permission models for managing user permissions and groups.

#### avatarImage

This model represents an avatar image associated with a user. It has an image field that stores the image file, an image_url field that stores the image URL, and a foreign key relationship with the CustomUser model. Each avatarImage is connected to a single user, and each user can have multiple avatarImage instances associated with them.

### Properties

#### Amenities

Contains an auto-incrementing primary key. Contains the names of the amenities that the website can filter by.

#### PropertyImages

Has an auto-incrementing primary key. It has an associated property (points to Property), the image file, and a path to the image.

#### Notifications

A notification has a recipient (points to a CustomUser), recipient_is_host, reservation, is_read, notification_type, and notification.

#### Reservation

A reservation has a user (points to a CustomUser), property (points to a Property), start_date, end_date, message, and status. There are 8 different status types: PENDING, DENIED, EXPIRED, APPROVED, CANCELED, TERMINATED, COMPLETED, PENDINGCANCELLATION. The status can be updated according to the reservation’s guest or property’s host.

#### Comments

Has an auto-incrementing key. Has an author (points to a CustomUser), a comment_type (user or property), a field to distinguish a reply from a root, a reference to its child comment (is a Comments and can be null), a reference to its parent comment (Comments, can be null), the message of the comment, the property to which the comment refers (points to a Property, can be null), or user about which the comment is written (points to a CustomUser, can be null). For the last two, one must always be null and the other must point to an object for its related field.