CPE 437 Final Project Specification

Justin Kehl, Jacob Francis, Serbinder Singh

Overview

This document describes the internal operations and REST endpoints of Eblist: an online marketplace similar to Craigslist that connects buyers and sellers. Users of Eblist can post items for sale under one of 14 categories and list details such as price, title, and description of the item. Other users can then look through all the postings by category and find items they wish to buy. Users may contact each other through email to arrange meetings to purchase items.

Error Codes

missingField Field missing from request. Params[0] gives field name

badValue Field has bad value. Params[0] gives field name

notFound Entity not present in DB -- for cases where a Item, Person, etc. is not there.

badLogin Email/password combination invalid, for errors logging.

dupEmail Email duplicates an existing email

noTerms Acceptance of terms is required

noOldPwd Change of password requires an old password

oldPwdMismatch Old password that was provided is incorrect.

dupEnrollment Duplicate enrollment

forbiddenField Field in body not allowed. Params[0] gives field name.

queryFailed Query failed (server problem)

limitExceeded File size limit exceeded for uploading image

User Management and Registration Resources:

<u>Users</u>

All registered users of the application.

GET email=<email or prefix>

Returns array of zero or more users with specified email if provided. No data for other than AU is provided unless AU is an admin. Fields returned are as follows:

email

id

POST

Registers a new user of the application. No AU required. Body must contain all fields as specified below else 400 and a list of missingField errors is reported. Email must be unique else 400 and dupEmail error reported. Role is 0 for all normal users and 1 for admins - any other value for role results in 400 and a badValue error reported. Only an admin may register other admins - otherwise 403. All users must accept the Terms of Use else 400 and badValue error. Zip must be a valid zip code else 400 and badValue error.

email

firstName

lastName

password

role

termsAccepted

zip: A string containing five digits representing the general location of the User

Users/{id}

GET

Returns array of one user with specified id with fields as specified in POST Users less password and termsAccepted. AU must be person in question or admin.

PUT

Updates the specified user with body giving one or more of: *email, firstName, lastName, password, role, zip*. Attempting to change other fields results in 400 and forbiddenField errors. Changing role requires admin AU else 400 and badValue error. An additional field *oldPassword* is required when changing the password else 400 and noOldPwd error. Changing the email to one that already exists causes a dupEmail error. All changes require AU be admin or the person in question.

DELETE

Delete the User in question and all items they have posted. Requires admin AU.

Users/{id}/Items

GET

Returns an array of all items owned by the specified user. AU must be specified user or admin. Each has the same fields as an element returned from a GET on Items

Ssns

Record of all active sessions of users using the application.

GET

Returns a list of all active sessions. Admin-privileged AU required. Returns array of

cookie Unique cookie value for session

prsId ID of Person logged in

loginTime Date and time of login

POST

A successful POST generates a browser-session cookie that will permit continued access for 2 hours. Indicated Person becomes the AU. An unsuccesful POST results in a 400 with a badLogin tag and no further information.

email Email of user requesting login

password Password of user

<u>Ssns/{cookie}</u>

GET

Returns, for the indicated session, a single object with same properties as one element of the array returned from Ssns GET. AU must be admin or owner of session.

DELETE

Log out the specified Session. AU must be owner of Session or admin.

Item Listing Resources:

Unless otherwise specified login is required, but any AU is acceptable for these endpoints.

<u>Categories</u>

The list of available categories is maintained in the database. It includes:

0. Other/None
1. Furniture
2. Appliances
3. Electronics
4. Vehicles
5. Pets/Animals
6. Tools
7. Art
8. Toys/Games
9. Jewelry
10. Clothing
11. Music/Instruments
12. Sporting Equipment
13. Services

GET

Return a list of categories that each have the following fields:

> *id:* Id of the Category
>
> *name*: The name of the Category

Categories/{id}/Items

GET radius=<distance in miles> title=<item title or portion of title>

Returns a list of all items in a given category. Each has the same fields as an element returned from a GET on Items. Results are limited to items within a given *radius* from the user containing *title* in their title if provided.

POST

Adds a new Item to the database that will be listed under the given category. Returns the *id* of the newly posted item. A single image may be associated with this new item by a put to Items/*id*/Image with the returned *id*.

Body contains the following fields:

> *title:* The name of the item user is wishing to sell. Limit to 255 characters
>
> *description:* Body of the item describing the item for sale. Limit to 3000 characters.
>
> *price:* integer representing amount of lowest denomination of currency.

Items

GET radius=<distance in miles> name=<item name or portion of name>

Any AU is acceptable. Return an array of 0 or more elements with one element for each Item in the system. Each element has the following fields.

> *id:* Id of the item
>
> *title:* The title of the listing. Limit to 255 characters.
>
> *price:* integer representing cost of item in cents

ownerId: Owner of the item being sold

email: Email of the owner.

zip: Zipcode of the owner where item is being sold.

distance: Distance in miles from the requester’s zip to the item’s zip.

postTime: Datetime for when the item was posted returned as an integer ms in UTC.

imageUrl: path to the image for an item on the server if there is one.

<u>**Items/{id}**</u>

GET

Return the item with the given id. Return the following for the indicated item:

id: Id of the item

categoryId: id of category the item is posted under.

title: The title of the listing.

description: Body of the item describing the item for sale.

price: integer representing cost of item in cents.

ownerId: Owner of the item being sold.

email: Email of the owner.

zip: Zipcode of the owner where item is being sold.

distance: Distance in miles from the requester’s zip to the item’s zip.

postTime: Datetime for when the item was posted returned as an integer ms in UTC.

imageUrl: path to the image for an item on the server if there is one.

PUT

Update an item listing. Fields that can be updated are title, description, and price. AU must be the owner of the item or admin. Attempting to change any other fields results in 400 and forbiddenField error.

DELETE

Delete an item listing. Only an admin or owner of the item can do this.

<u>Items/{id}/Image</u>

PUT

Upload an image for the specified item. Replaces the existing image if there is one. The entire body is saved as a jpg on the server, accessible through the imageUrl that is now stored with the item. Limit 10mb.

<u>DB</u>

Delete

Clears all content from the Database including all users and items. Resets all autoincrement ids to 1. Adds back one admin user with no items.