Vin.ly API Documentation

Current Active Routes:

Black Text indicates currently operating API endpoints
Blue Text indicates either proposed endpoints OR future changes to the endpoint and related data.
Red Text indicates currently completed but non operational or not usable in the current version. Extended periods of Red Text may indicate future deprecation.
Yellow Text with a Red Highlight Indicates that Endpoint will be deprecated and removed in future versions.

## Wine Routes:

- /api/v1/wine/
  - GET: Returns general listing of all wines available in the system database. Will retrieve all data on the wines unless otherwise specified. Will change and likely be disabled for anyone but administrators as database grows.
  - POST: Allows the creation of wines with the following fields in demo:
    - Varietal: the wine type, e.g, Chardonnay, Sangiovese, Pinot Noir (currently unique)
    - Flavor_profile: the primary / prominent flavors of the wine
    - Notes: the after finish and other flavors that commonly occur
    - Body: Sweet to Bone Dry
    - Description: A description of the wine
    - PROPOSED FUTURE FIELDS:
      - Name: if the wine has a specific name, it goes here
      - Winery: the vintner makes the wine
      - Vintage: the year of corkage / produced
      - Region: The general region the wine is produced in, e.g, Sonoma, Napa, Bordeaux, Westphalia, Martha's Vineyard
- /api/v1/wine/:wine_id
  - GET: Gets a wine by a certain ID, currently not used by App, but is useful for administration through postman.
  - PUT: Proposed update structure for admins and advanced curated users to make changes
  - DELETE: Remove a wine from the system, especially if a duplicate occurs.
- /api/v1/wine/name/:wine_name
  - GET: Get a wine by its name. This will return mutliple results of a named wine.
- /api/v1/wine/winery/:winery

        - GET: Get a wine by its winery. This will return multiple results if the winery has numerous wines in production.
    - /api/v1/wine/region/:wine_region
        - GET: Get a wine by its region. If a region has multiple wineries or wines associated with it, there may be a lot of results.
    - /api/v1/wine/region/:wine_region/:winery
        - GET: Get a wine by its region and associated winery. Will return fewer results than above.
    - /api/v1/wine/vintage/:winery/:vintage/:varietal
        - GET: Get wines from a winery, a vintage year, and a varietal. This can reduce the number of wines returned by a winery.
    - /api/v1/wine/like/:wine_like
        - GET: Get wines containing a certain string pattern in their names. This can be useful for finding like named wines or if the user does not remember the name of the wine exactly.
    - /api/v1/wine/match/:varietal
        - GET: Current working function, matches a wine to various foods that pair well with its varietal.
        - PROPOSED EXPANSION:
            - Match varietal and flavor profile to specific dishes, e.g., a spicy zinfandel works better with firmer steak cuts such as New York, while chardonnay is better with creamy pasta dishes or creamed chicken. A dry Chenin Blanc goes better with hard cheeses and crudite, whereas a sweet might go better with fish or deserts.

## Beer Routes:

- /api/v1/beer
    - GET: Lists all beers and the information pertaining to them.
    - POST: Creates a beer with the following fields
        - Beer_Type: indicates the type of beer, e.g. Bock, Pilsner, or Pale Ale
        - Notes: The primary flavors of the beer. Most beer is fairly bready or bitter in flavor, but there are definite notes in different beers depending on the base they use. Wheat bears tend to be fruitier and have banana flavor, while IPAs are floral and stark.
        - Intensity: How strong the underyling flavors come through, as well as the beers accessibility (may need to be split off.)
        - Color: The base color of this particular type of beer, will be expanded if beer is expanded out like wine.
        - Description: A general note on the beer's appearance, origins, and style.
        - PROPOSED BEERS FIELDS

        - Beer_Name: The beer's name, e.g, Lagunitas Lil' Somethin' Somethin', Kona Fire Rock Ale. etc. (Possibly unique, but two brewers may later create similarly or same named beer)
        - Brewery: The brewery that makes the beer
        - Region: The general region the beer comes from. E.g, Sonoma, California, Golden, Colorado, Newcastle, UK
    - /api/v1/beer/:beer_id
        - GET: Gets a  beer by a certain ID, currently not used by App, but is useful for administration through postman or curl.
        - PUT: Proposed update structure for admins and advanced curated users to make changes
        - DELETE: Remove a beer from the system, especially if a duplicate occurs.
    - /api/v1/beer/name/:beer_name
        - GET: Get a beer by its name. This will return mutliple results of a named beer.
    - /api/v1/beer/brewery/:brewery
        - GET: Get a beer by its brewery. This will return multiple results if the brewery has numerous beers in production.
    - /api/v1/beer/region/:beer_region
        - GET: Get a beer by its region. If a region has multiple breweries or beers associated with it, there may be a lot of results.
    - /api/v1/beer/region/:beer_region/:brewery
        - GET: Get a beer by its region and associated brewery. Will return fewer results than above.
    - /api/v1/beer/:brewery/:beer_type
        - GET: Get beer types rom a brewery. This can reduce the number of wines returned by a brewery, often to one or two. Some breweries, such as Lagunitas or MacLeod Brewery have a rotating palette of similar typed beers that can be very distinct.
    - /api/v1/beer/like/:beer_like
        - GET: Get beers containing a certain string pattern in their names. This can be useful for finding like named beers, or if the user does not remember the name of the beer exactly.
    - /api/v1/beer/match/:beer_type
        - GET: Current working function, matches a beer to various foods that pair well with its type.
        - PROPOSED EXPANSION:
            - Match beer type and note/intensity profile to specific dishes, e.g., a floral IPA, such as Angel City works well with Indian or Cajun food dishes such as Vindaloo or Spicy Shrimp. Cream Ales are

better with creamier dishes. A crisp pilsner or lager might go well with a cheese plate or fish.

## Food Routes:

- /api/v1/food
  - GET: Lists all foods and the information pertaining to them.
  - POST: Creates a food with the following fields
    - Food_Type: Indicates the primary feature of the food, such as pork, steak, hamburger, or pasta.
    - Primary_Flavor: Currently underutilized, indicated as savory or sweet. As dishes become prominent, more detailed flavor profiles will be generated.
    - Food_Group: More specificly which food group the food belongs to: Red Meat, Poultry, Seafood, Starch, Dairy, etc. May be merged into Food_type or expanded.
    - Description: A description of the food or dish.
    - PROPOSED BEERS FIELDS
      - Food_Name: The name of the dish. (Unique)
      - Region: The cuisine the food is identified with, usually region of a nation and the nation itself, e.g, Jaipur, India, Southwestern, United States, Paris, France
- /api/v1/food/:food_id
  - GET: Gets a food by a certain ID, currently not used by App, but is useful for administration through postman or curl.
  - PUT: Proposed update for admins and advanced curated users to make changes
  - DELETE: Remove a food from the system, especially if a duplicate occurs.
- /api/v1/food/math/beer/:food_type
  - GET: Matches a food type to its constituent beer pairings. Currently very little data is in the database. Currently associated by IDs.
- /api/v1/food/math/wine/:food_type
  - GET: Matches a food type to its constituent wine pairings. Currently very little data is in the database. Currently associated by IDs.

USER ROUTES:

- .
  - /api/v1/user
    - GET: Gets all of the user information. This is a potential security risk currently, but no real users exist yet.

        - POST: Register a new user with a hashed password. The input data will be changed from the current schema, schema still in development. May be limited to a username, email, and password moving forward.
            - Username: a custom user-name, unique
            - Password: Encrypted by bcrypt currently
            - Email: the user's email, should be unique.
            - First_Name, Middle_Name, Last_Name: Fields currently in effect to collect users name. - likely to be removed due to privacy issues.
            - Country: Get the user's country to comply with local drinking laws. Due to cultural sensitivity issues, users from countries where the consumption of alcohol is forbidden may be prevented from registration, if the app is available in their app store. Otherwise, post gentle reminder to obey the laws and dictates of their homeland.
            - DOB: Get date of birth to confirm if user is 16 (Western Europe), 18 (Western Hemisphere, Australia, UK, Eastern Europe, most of Asia), 19 (Canada), 20 (Japan), or 21 (US).
    - /api/v1/user/id/:user_id
        - GET: Get a user by their id number.
        - PUT: Edit a user's data accordingly for administrative purposes. Likely to ban or remove users who are disruptive or edit/reset passwords. (Separate route)
        - DELETE: Remove a user.
    - /api/v1/user/id/:user_name -- id will be changed to un for clarification.
        - GET: Get a user by their username, not currently active.
    - /api/v1/user/password_check/:username/:password
      This will be removed prior to the next version of the API, it was simply a utility function to ensure that passwords were being recognized and decrypted properly by the hashing algorithm.

    - /api/v1/user/login/
        - POST: Log in a user with the matching password and username.