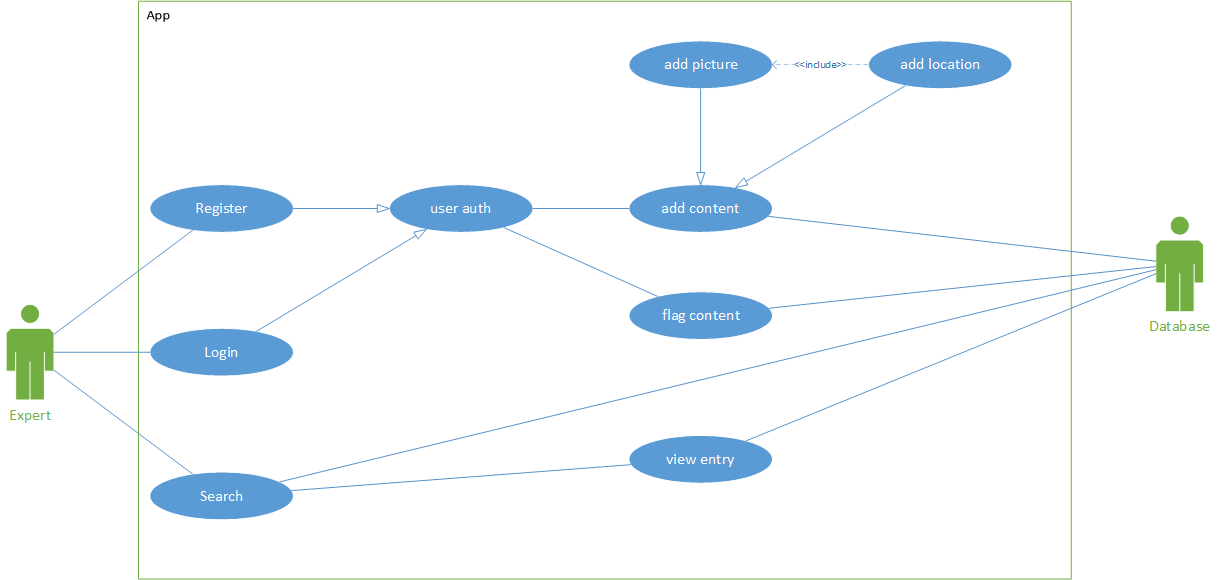
WORK IN PROGRESS

**Application:** Bugipedia (Placeholder name) **🤔**

*Website Functionality | App Functionality*

* Target Audience : EXPERTS
* Point of the app: Reference on the go for field work
  + Users will be able to look up bugs based off of their features
    - Entries for each bug includes:
      * Common name & scientific name
      * Pictures
      * List of characteristics
      * Paragraph/mini wiki entry about bug
      * Map of sightings
      * Additional information/advice (ex. Look for florida roaches under dried palm fronds)
  + View photos of bugs taken by other users
    - Flag inappropriate pictures
  + View locations on a map where the bugs have been seen in their area
  + Submit images / spot locations
  + a form where users can submit new species to be added to the DB (need mod approval)
  + Edit entries about bugs (need mod approval)
  + Mods can approve edits/new species



**Database**

|  |
| --- |
| -- users CREATE TABLE `app`.`users` (  `id` INT(10) UNSIGNED NOT NULL AUTO\_INCREMENT ,  `username` VARCHAR(70) NOT NULL ,  `password` VARCHAR(512) NOT NULL ,  `admin` BOOLEAN NOT NULL ,  PRIMARY KEY (`id`),  UNIQUE (`username`) ) ENGINE = InnoDB; |
| -- session CREATE TABLE `app`.`session` (  `id` INT(10) UNSIGNED NOT NULL AUTO\_INCREMENT ,  `user\_id` INT(10) UNSIGNED NOT NULL ,  `session\_id` VARCHAR(512) NOT NULL ,  `api\_key` VARCHAR(512) NOT NULL ,  `session\_time` DATETIME NOT NULL ,  `is\_api` BOOLEAN NOT NULL ,  PRIMARY KEY (`id`),  INDEX (`user\_id`) ) ENGINE = InnoDB; |
| -- sightings CREATE TABLE `app`.`sightings` (  `id` INT(10) UNSIGNED NOT NULL AUTO\_INCREMENT ,  `bugid` INT(10) UNSIGNED NOT NULL ,  `latitude` FLOAT(10, 6) NOT NULL ,  `longitude` FLOAT(10, 6) NOT NULL ,  PRIMARY KEY (`id`),  INDEX (`bugid`) ) ENGINE = InnoDB; |
| -- pictures CREATE TABLE `app`.`pictures` (  `id` INT(10) UNSIGNED NOT NULL AUTO\_INCREMENT ,  `bugid` INT(10) UNSIGNED NOT NULL ,  `num\_flags` INT(10) UNSIGNED NOT NULL ,  `picture\_link` VARCHAR(1024) NOT NULL ,  PRIMARY KEY (`id`),  INDEX (`bugid`) ) ENGINE = InnoDB; |
| -- bugs CREATE TABLE `app`.`bugs` (  `id` INT(10) UNSIGNED NOT NULL AUTO\_INCREMENT ,  `common\_name` VARCHAR(70) NOT NULL ,  `scientific\_name` VARCHAR(70) NOT NULL ,  `class\_id` INT(10) UNSIGNED NOT NULL ,  `order\_id` INT(10) UNSIGNED NOT NULL ,  `family\_id` INT(10) UNSIGNED NOT NULL ,  `genus\_id` INT(10) UNSIGNED NOT NULL ,  `color\_id\_1` INT(10) UNSIGNED NOT NULL ,  `color\_id\_2` INT(10) UNSIGNED NOT NULL ,  `general\_type\_id` INT(10) UNSIGNED NOT NULL ,  `mouth\_parts\_id` INT(10) UNSIGNED NOT NULL ,  `wings` BOOLEAN NOT NULL ,  `antenna` BOOLEAN NOT NULL ,  `hind\_legs\_jump` BOOLEAN NOT NULL ,  `hairy\_furry` BOOLEAN NOT NULL ,  `thin\_body` BOOLEAN NOT NULL ,  `description` TEXT NOT NULL ,  `additional\_advice` TEXT NOT NULL ,  `approved` BOOLEAN NOT NULL ,  PRIMARY KEY (`id`),  INDEX (`class\_id`, `order\_id`, `family\_id`, `genus\_id`, `color\_id\_1`, `color\_id\_2`, `general\_type\_id`, `mouth\_parts\_id`) ) ENGINE = InnoDB; |
| |  |  | | --- | --- | | -- color CREATE TABLE `app`.`color` (  `id` INT(10) UNSIGNED NOT NULL AUTO\_INCREMENT ,  `color` VARCHAR(20) NOT NULL ,  PRIMARY KEY (`id`) ) ENGINE = InnoDB; | -- class CREATE TABLE `app`.`class` (  `id` INT(10) UNSIGNED NOT NULL AUTO\_INCREMENT ,  `name` VARCHAR(70) NOT NULL ,  PRIMARY KEY (`id`) ) ENGINE = InnoDB; | | -- order CREATE TABLE `app`.`order` (  `id` INT(10) UNSIGNED NOT NULL AUTO\_INCREMENT ,  `name` VARCHAR(70) NOT NULL ,  PRIMARY KEY (`id`) ) ENGINE = InnoDB; | -- family CREATE TABLE `app`.`family` (  `id` INT(10) UNSIGNED NOT NULL AUTO\_INCREMENT ,  `name` VARCHAR(70) NOT NULL ,  PRIMARY KEY (`id`) ) ENGINE = InnoDB; | | -- genus CREATE TABLE `app`.`genus` (  `id` INT(10) UNSIGNED NOT NULL AUTO\_INCREMENT ,  `name` VARCHAR(70) NOT NULL ,  PRIMARY KEY (`id`) ) ENGINE = InnoDB; | -- mouth\_parts CREATE TABLE `app`.`mouth\_parts` (  `id` INT(10) UNSIGNED NOT NULL AUTO\_INCREMENT ,  `name` VARCHAR(70) NOT NULL ,  PRIMARY KEY (`id`) ) ENGINE = InnoDB; | |
| -- general\_type CREATE TABLE `app`.`general\_type` (  `id` INT(10) UNSIGNED NOT NULL AUTO\_INCREMENT ,  `name` VARCHAR(70) NOT NULL ,  PRIMARY KEY (`id`) ) ENGINE = InnoDB; |

**API**

* Switched from PHP to Flask RESTful API using Python
* Using SQLAlchemy to interact with DB

Endpoints List

* Users
  + Register
  + Login
  + ~~Add~~
  + ~~Delete~~
  + ~~Edit~~
* Bugs
  + Add
  + Delete
  + Edit
  + Search
* Pictures
  + Add
  + Delete (when flagged enough times)

**API Docs**

Register API

Endpoint: /api/register

Request: POST

|  |  |  |
| --- | --- | --- |
| Parameter | Mandatory | Description |
| username | yes | Username to be registered |
| password | yes | Plaintext password for registration |
| isAdmin | yes | If the user is an admin (This probably shouldn’t be a thing) |

Response:

|  |  |
| --- | --- |
| Parameter | Description |
| success | 1 if successful, -1 if not |
| error | If failed, then this will contain a message describing the error |

Example:  
Request body:{"username": "wew", "password": "test", "isAdmin": 1}

Response body: {"error":"User already registered","success":-1}

Alternate Response body: {"success":1}

Login API

Endpoint: /api/login

Request: POST

|  |  |  |
| --- | --- | --- |
| Parameter | Mandatory | Description |
| username | yes | Username to be registered |
| password | yes | Plaintext password for registration |

Response:

|  |  |
| --- | --- |
| Parameter | Description |
| success | 1 if successful, -1 if not |
| error | If failed, then this will contain a message describing the error |
| sessionID | If successful, this field contains a 32 character random string that is your sessionID. Store this because it is what you will use to auth to other endpoints |

Request body:{"username": "wew", "password": "test"}

Response body: {"sessionID":"rGIxucrETI20XgQgpqgcdJElXZaMUedO","success":1}

Alternate response body: {"error":"Incorrect password provided","success":-1}

Search API

Endpoint: /api/search

Request: GET

Response:

|  |  |
| --- | --- |
| Parameter | Description |
| success | 1 if successful, -1 if not |
| error | If failed, then this will contain a message describing the error |
| Fields | An array of json objects. Each object will contain a “label”, a “type” and maybe an “options” field,.  Types:  TEXT: text box, send a string back  DROP: Drop down menu, send a string back, contains options.  CHECK: Checkbox, send array of strings back. Contains options. |
|  |  |

Response body: {"fields":[{“label”: “Common Name”, “type”: “TEXT”},

{“label”: “Colors”, “type”: “CHECK”, “options”: [“red”, “green”]],

"success":1}

Request: POST

All fields will be anded together, if multiple check boxes are chosen the check boxes will be ored together.

This only returns names and bug id for the bugs found, then the user will click on the bug they want more info for and you will use a different endpoint to request full bug info.

Request:

|  |  |  |
| --- | --- | --- |
| Parameter | Mandatory | Description |
| <Label> (TEXT) | no | String that was entered into text box matching <label> |
| <Label> (DROP) | no | String that was selected from drop down menu matching <label> |
| <Label> (CHECK) | no | Array of strings from boxes that were checked in the field matching <label> |

Response:

|  |  |
| --- | --- |
| Parameter | Description |
| success | 1 if successful, -1 if not |
| error | If failed, then this will contain a message describing the error |
| Results | Array containing json objects for each bug found by the search. |
| results[n].common\_name | The common name for that bug |
| results[n].id | The bug ID for the bug |
| results[n].scientific\_name | The scientific name for the bug |
| results[n].thumbnail | URL to first image stored in db for the bug. If no image exists, a url for a thumbnail of a generic bug is returned. |

Request body:{"Colors": ["yellow", "red", "blue"], "Has wings": "Yes", "Has hind legs for jumping": "No", "Common Name": "spider"}

{"results":[{"common\_name":"spider","id":1,"scientific\_name":"spider", "thumbnail": "http://imgur.com/lmao"}],"success":1}

Sightings API

Endpoint: /api/getSightings

Request: POST

|  |  |  |
| --- | --- | --- |
| Parameter | Mandatory | Description |
| id | yes | BugID for the bug you want sightings for. |

Response:

|  |  |
| --- | --- |
| Parameter | Description |
| success | 1 if successful, -1 if not |
| error | If failed, then this will contain a message describing the error |
| sightings | Array of sightings for the bug |
| sightings[n].latitude | Latitude of sighting. Type: Float |
| sightings[n].longitude | Longitude of sighting. Type: Float |

Example:  
Request body:{"id": 1}

Response body: {"success":1, "sightings": [{"longitude": 5.5, "latitude": 4.4}] }

Endpoint: /api/addSighting (User authentication required for this endpoint)

Request: POST

|  |  |  |
| --- | --- | --- |
| Parameter | Mandatory | Description |
| id | yes | BugID for the bug you want to add a sighting for. |
| longitude | yes | Longitude of sighting, a float |
| latitude | yes | Latitude of sighting, a float |

Response:

|  |  |
| --- | --- |
| Parameter | Description |
| success | 1 if successful, -1 if not |
| error | If failed, then this will contain a message describing the error |

Example:  
Request body:{"id": 1, “latitude”: 5.5, “longitude”: 5.5} (remember to add authentication header)

Response body: {"success":1}

Pictures API

Endpoint: /api/getImages

Request: POST

|  |  |  |
| --- | --- | --- |
| Parameter | Mandatory | Description |
| id | yes | BugID for the bug you want image links for. |

Response:

|  |  |
| --- | --- |
| Parameter | Description |
| success | 1 if successful, -1 if not |
| error | If failed, then this will contain a message describing the error |
| images | Array of bug image info |
| images[n].imageId | Image ID, to be used for flagging |
| images[n].url | Image url |

Example:  
Request body:{"id": 1}

Response body: {"images":[{“url”: "https://upload.wikimedia.org/wikipedia/commons/1/1e/Ctenidae 2C\_Ctenus 2C\_PE\_Carlos\_Botelho 2C\_S C3 A3o\_Paulo.jpg”, “imageId”: 5}”],"success":1}

Endpoint: /api/addImage (User authentication required for this endpoint)

Request: POST

|  |  |  |
| --- | --- | --- |
| Parameter | Mandatory | Description |
| id | yes | BugID for the bug you want to add a sighting for. |
| image | yes | Base64 encoded image data |

Response:

|  |  |
| --- | --- |
| Parameter | Description |
| success | 1 if successful, -1 if not |
| error | If failed, then this will contain a message describing the error |

Example:  
Request body:{"id": 1, "image": "LONG BASE64 STRING"} (remember to add authentication header)

Response body: {"success":1}

Endpoint: /api/flagImage (User authentication required for this endpoint)

Request: POST

|  |  |  |
| --- | --- | --- |
| Parameter | Mandatory | Description |
| imageId | yes | Image id of image you want to flag |

Response:

|  |  |
| --- | --- |
| Parameter | Description |
| success | 1 if successful, -1 if not |
| error | If failed, then this will contain a message describing the error |

Example:  
Request body:{"imageId": 1} (remember to add authentication header)

Response body: {"success":1}

Register Bug API

Endpoint: /api/registerBug

Request: POST

|  |  |  |
| --- | --- | --- |
| Parameter | Mandatory | Description |
| common\_name | yes | Common name of bug. This is taken from POST request to check if a duplicate exists. |
| scientific\_name | yes | Scientific name of bug |
| class | yes | Class characteristic of bug |
| order | yes | Order characteristic of bug |
| family | yes | Family characteristic of bug |
| genus | yes | Genus characteristic of bug |
| color\_1 | yes | Main color characteristic of bug |
| color\_2 | yes | Off color characteristic of bug |
| general\_type | yes | General type characteristic of bug |
| mouth\_parts | yes | Mouth part characteristic of bug |
| wings | yes | Does bug have wings? (boolean) |
| antenna | yes | Does bug have antennae? (boolean) |
| hind\_legs\_jump | yes | Does bug use hind legs for jumping? (boolean) |
| hairy\_furry | yes | Is the bug hairy/furry? (boolean) |
| thin\_body | yes | Does the bug have a thin body or not? (boolean) |
| description | yes | Brief description of what the bug is |
| additional\_advice | yes | Additional advice regarding the bug |
| approved | no | Bugs registered are done by admins already, so this is default to True when it’s added to the DB |

Response:

|  |  |
| --- | --- |
| Parameter | Description |
| success | 1 if successful, -1 if not |
| error | If failed, then this will contain a message describing the error. |

Example:

Request Body:  
{"common\_name":"Florida woods cockroach", "scientific\_name":"Eurycotis floridana", "class":"Insecta", "order":"Blattodea", "family":"Blattidae", "genus":"Eurycotis", "color\_1":"brown", "color\_2":"red", "general\_type":"Cockroaches and Termites", "mouth\_parts":"Biting/Chewing", "wings": 1, "antenna": 1, "hind\_legs\_jump": 0, "hairy\_furry": 0, "thin\_body": 0, "description":"The Florida woods cockroach or palmetto bug is a large cockroach species. When alarmed, adults can eject an extremely foul-smelling directional spray up to 1 m. Two other naming variations include Florida cockroach and Florida woods roach. The Florida woods cockroach is slower moving than many other cockroach species. It prefers damp locations with abundant moisture, and does well in warm, damp climates. It is found in its native habitats, such as Florida and the West Indies. The species wanders indoors at times, especially into damp locations, such as bathrooms; however, it prefers the outdoors and is not considered a major pest in the home.", "additional\_advice":"They can be found in palmetto trees, sheltered outdoor locations, such as under leaf litter, in tree holes, and under lumber and boards, and other crevices, as well as in bushes and wooded areas"}

Response Body:

{“success”: 1}

Get Bug API

Endpoint: /api/getBug

Request: POST

|  |  |  |
| --- | --- | --- |
| Parameter | Mandatory | Description |
| id | yes | Bug ID that corresponds to a bug entry in the database |

Response:

|  |  |  |
| --- | --- | --- |
| Parameter | Level (1-n, 1 being highest level) | Description |
| success | 1 | 1 if successful, -1 if not |
| error | 1 | If failed this will contain a message describing the error. |
| sightings | 1 | An array of latitude/longitude pairs representing locations where bug was seen |
| pictures | 1 | Array of urls that link to pictures of the bug |
| description | 1 | Description of bug from database |
| common\_name | 1 | Common name of bug from database |
| additional\_advice | 1 | Additional advice from database |
| characteristics | 1 | Characteristics group in the JSON response that encapsulates some bug characteristics |
| characteristics.antenna | 2 | True/False whether bug has antenna |
| characteristics.class | 2 | Class of bug from database |
| characteristics.color1 | 2 | Main color of bug from database |
| characteristics.color2 | 2 | Off color of bug from database |
| characteristics.family | 2 | Family of bug from database |
| characteristics.general\_type | 2 | General type of bug description from database |
| characteristics.genus | 2 | Genus of bug from database |
| characteristics.hairy\_furry | 2 | True/False whether bug is hairy/furry |
| characteristics.hind\_legs\_jump | 2 | True/False whether bug uses hind legs to jump |
| characteristics.mouth\_parts | 2 | Mouth parts description from database |
| characteristics.order | 2 | Order of bug from database |
| characteristics.scientific\_name | 2 | Scientific name of bug from database |
| characteristics.thin\_body | 2 | True/False whether bug has thin body |
| characteristics.wings | 2 | True/False whether bug has wings |

Example: lamolmaolmaolamolamolmaolamolmaolamolmaolamolmaolamolmaolamo

Request Body: {"id": 1}

Response Body:

Note: The fields are reorganized alphabetically since they are put into a python dictionary before sent out as a JSON response.

{"additional\_advice":"lmao","characteristics":{"antenna":true,"class":"test1","color1":"red","color2":"blue","family":"test","general\_type":"test","genus":"test","hairy\_furry":true,"hind\_legs\_jump":true,"mouth\_parts":"test","order":"test","scientific\_name":"spider",”thin\_body”:false,"wings":true},"common\_name":"spider","description":"lmao","pictures":[{"url":"http://fancybear.group/wew.png"}],"sightings":[{"latitude":5.5,"longitude":7.7},{"latitude":6.6,"longitude":7.7}],"success":1}

Edit Bug API

Endpoint: /api/editBug (admin only)

Request: POST

|  |  |  |
| --- | --- | --- |
| Parameter | Mandatory | Description |
| bugid | yes | Bugid of the bug being edited |
| common\_name | yes | Common name of bug. This is taken from POST request to check if a duplicate exists. |
| scientific\_name | yes | Scientific name of bug |
| class | yes | Class characteristic of bug |
| order | yes | Order characteristic of bug |
| family | yes | Family characteristic of bug |
| genus | yes | Genus characteristic of bug |
| color\_1 | yes | Main color characteristic of bug |
| color\_2 | yes | Off color characteristic of bug |
| general\_type | yes | General type characteristic of bug |
| mouth\_parts | yes | Mouth part characteristic of bug |
| wings | yes | Does bug have wings? (boolean) |
| antenna | yes | Does bug have antennae? (boolean) |
| hind\_legs\_jump | yes | Does bug use hind legs for jumping? (boolean) |
| hairy\_furry | yes | Is the bug hairy/furry? (boolean) |
| thin\_body | yes | Does the bug have a thin body or not? (boolean) |
| description | yes | Brief description of what the bug is |
| additional\_advice | yes | Additional advice regarding the bug |

Response:

|  |  |
| --- | --- |
| Parameter | Description |
| success | 1 if successful, -1 if not |
| error | If failed, then this will contain a message describing the error. |

Example:

Request Body:  
{"common\_name":"Florida woods cockroach", "scientific\_name":"Eurycotis floridana", "class":"Insecta", "order":"Blattodea", "family":"Blattidae", "genus":"Eurycotis", "color\_1":"brown", "color\_2":"red", "general\_type":"Cockroaches and Termites", "mouth\_parts":"Biting/Chewing", "wings": 1, "antenna": 1, "hind\_legs\_jump": 0, "hairy\_furry": 0, "thin\_body": 0, "description":"The Florida woods cockroach or palmetto bug is a large cockroach species. When alarmed, adults can eject an extremely foul-smelling directional spray up to 1 m. Two other naming variations include Florida cockroach and Florida woods roach. The Florida woods cockroach is slower moving than many other cockroach species. It prefers damp locations with abundant moisture, and does well in warm, damp climates. It is found in its native habitats, such as Florida and the West Indies. The species wanders indoors at times, especially into damp locations, such as bathrooms; however, it prefers the outdoors and is not considered a major pest in the home.", "additional\_advice":"They can be found in palmetto trees, sheltered outdoor locations, such as under leaf litter, in tree holes, and under lumber and boards, and other crevices, as well as in bushes and wooded areas", “bugid”: 1}

Response Body:

{“success”: 1}

Endpoint: /api/getEdits (no auth)

Request: GET

|  |  |  |
| --- | --- | --- |
| Parameter | Mandatory | Description |

Response:

|  |  |
| --- | --- |
| Parameter | Description |
| success | 1 if successful, -1 if not |
| error | If failed, then this will contain a message describing the error |
| submissions | Array of edit submissions |
| sightings[n].bug\_id\_new | Bug id of new bug |
| sightings[n].bug\_id\_old | Bug id old |

Example:  
Request body:{}

Response body: {"submissions":[{"bug\_id\_new":9,"bug\_id\_old":2,"submission\_id":2}],"success":1}

Endpoint: /api/approveEdit (admin only)

Request: POST

|  |  |  |
| --- | --- | --- |
| Parameter | Mandatory | Description |
| id | yes | Submission id you want to approve or deny |
| approve | yes | True or false value of if you want to approve the edit or not |

Response:

|  |  |
| --- | --- |
| Parameter | Description |
| success | 1 if successful, -1 if not |
| error | If failed, then this will contain a message describing the error |

Example:  
Request body:{"id": 2, "approve": true}

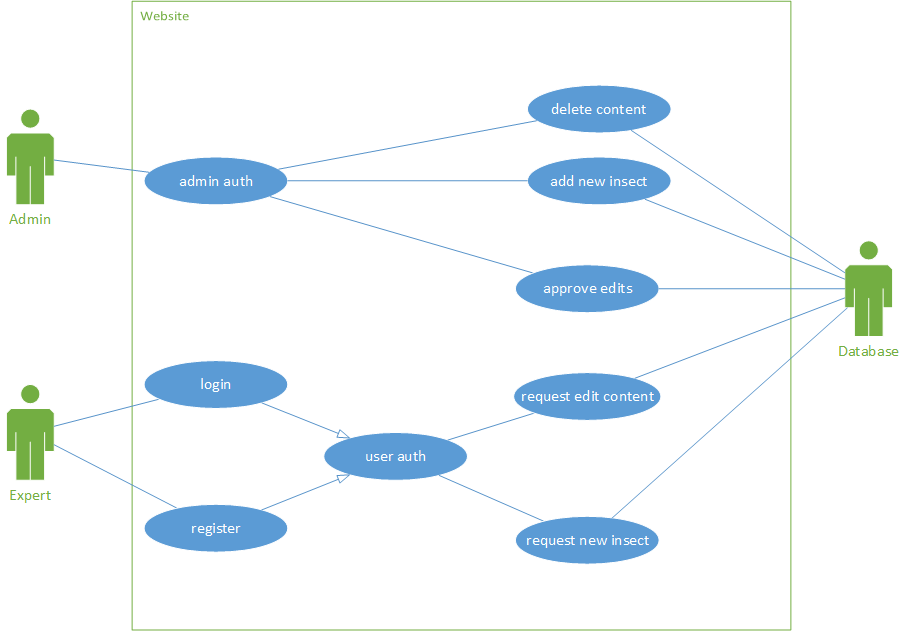
Response body: {"success":1}

**Application**

* **Main Functions**
  + **Search**
    - Enterable fields
      * Dropdowns
      * Checkboxes
      * Above two obtained from DB
    - Submit Search request
    - Retrieve Search entries
      * In form of picture, name, sci name
    - Viewing a specific entry
      * All the fields for that entry from DB
  + **Upload Picture**
    - Take picture from camera/choose from pictures
    - Upload
  + **Adding Sighting**
    - Active Google Maps
    - Choose a location on map
    - (Optional) Manual Sighting Location
* **Framework**
  + - Ask forum to identify bug based on picture
      * Comments for identifications
  + Permissions/External Intents needed
    - Camera
    - Location
    - Internet Access (API Calls)
* Networking
  + **Logged in/ Sessions**
    - When logged in, we receive a permanent **SessionID (API Key**)
    - Needs to be put into persistent storage
    - Only removed when the user logs out.
  + **Searching / Main activity**
    - Get a list of all characteristics
    - Search for entries
      * **Picture link**
        + Retrieve from URL that will be passed
      * **Common Name**
      * **Scientific name**
      * **Reference ID to the Database Entry**
  + View DB Entry
    - Entry Object Special Handling
      * Pictures - retrieve from URLs that will be passed to
    - Everything else is a string or list of strings that can be directly displayed
    - Displaying the map (w/ location data obtained from DB)
  + Gallery View
    - Reporting a inaccurate photo
      * **Send back the URL of the photo**
      * **Send back the SessionID token**
  + Uploading Photos
    - **Base64 of the image to be uploaded**
    - **SessionID token**
  + Uploading Sighting
    - **Lat/Long Values - obtain from the Maps activity directly**
    - **SessionID token**
  + API handling classes for main networking activity (Retrofit)
    - Requires support from each activity for implementation

**Website**

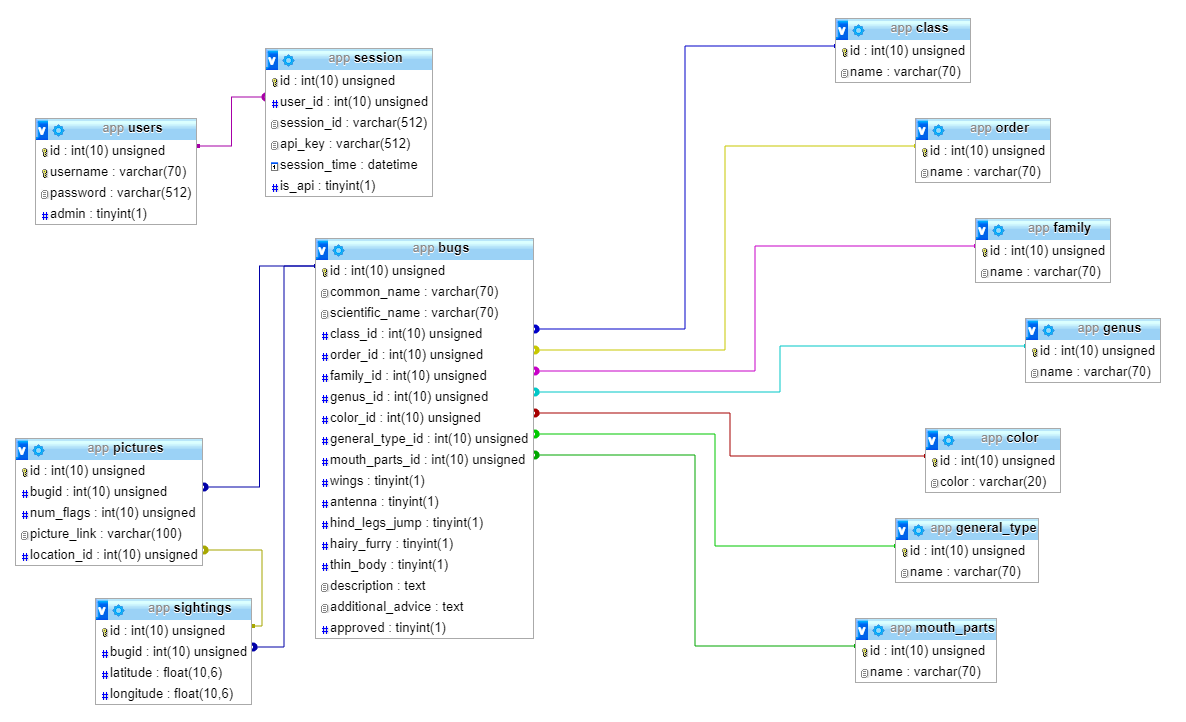
* Any searching will be done by name



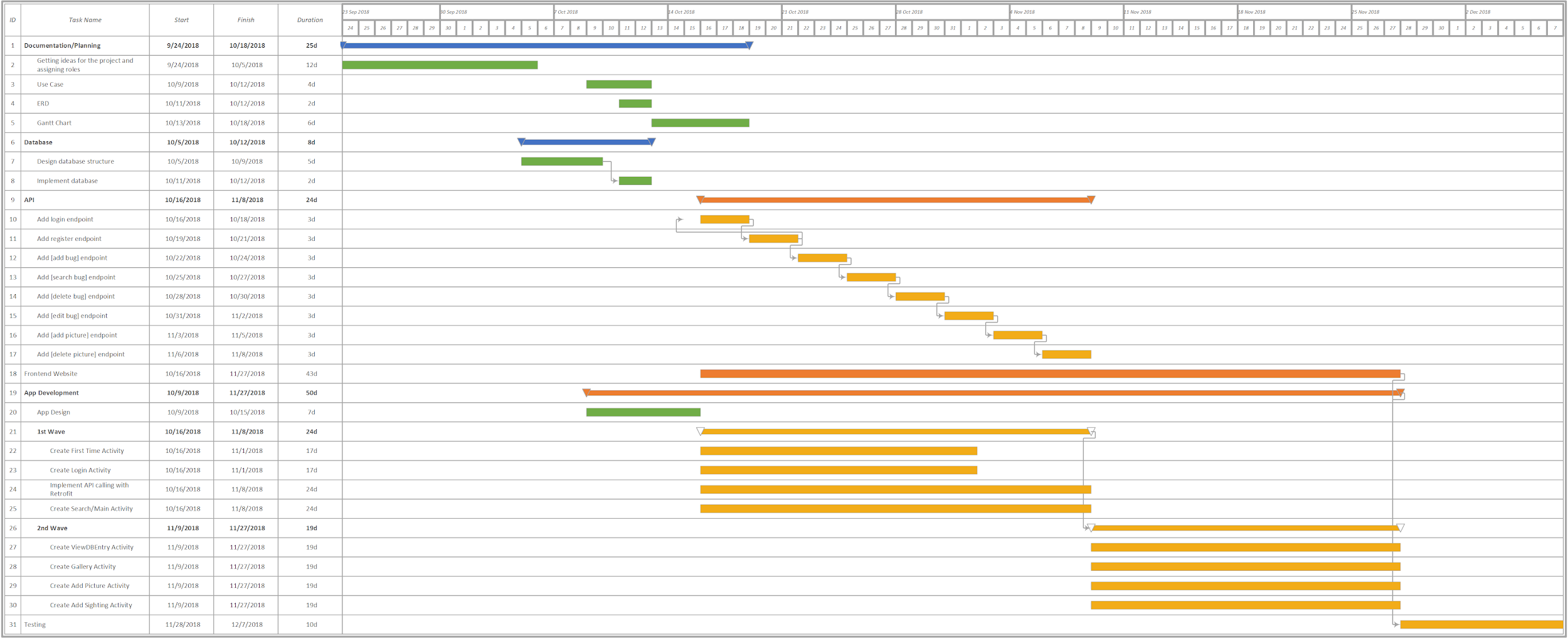
**Money**

* Channel in Discord for discussion
* Subscription based service
* Donations
* Ads on website/app



**ERD**

**In progress:**



****