**Color Code: Completed In-ProgressOmitted**

**MVP: No login required!**

We will need routes for:

1. GET/ : Base page -> redirect to /tasks or replace /tasks.
2. GET /tasks: Home page that shows to do lists (1st view of MVP wireframe) - AHK
3. GET /tasks/categories: Shows items by category (2nd view of MVP wireframe) - AHK
4. ~~GET /tasks/:task\_id : Get a task~~
5. GET /tasks/status/:task\_id : Get the status of task (incomplete, complete)
6. ~~GET /tasks/categories/:task\_id : Get the categories of task~~
7. POST /tasks/add: Adding a task (title + task description for tasks table) - TJ
8. POST /tasks/delete/:task\_id : Deleting a task (Sends a true or false to deleted column in tasks table) - TJ
9. POST /tasks/status/:task\_id : Posts to status\_id in tasks to change based on checkmark status (check is completed, empty is incomplete) - TJ
10. POST /tasks/categories/add/:task\_id/:cat\_name: Adding a category for an existing item (posts to task\_category table)
11. POST /tasks/categories/delete/:task\_id/:cat\_name : Removes a category from an existing item (posts to task\_category table)
12. GET /categories: Get categories list
13. POST /categories/add: Adds a new category to the categories table (Need to set the relationship with the task in task\_category)
14. POST /categories/delete/:cat\_name : Delete a category from the categories table, not just the item relationship. (Is this a necessary feature to have?)

Using routers the paths from above will look like below:

Router for /tasks

1. GET / <- This is GET /tasks.
2. GET /categories
3. GET /:task\_id
4. GET /status/:task\_id
5. GET /categories/:task\_id
6. POST /add
7. POST /delete/:task\_id
8. POST /status/:task\_id
9. POST /categories/add/:task\_id/:cat\_name
10. POST /categories/delete/:task\_id/:cat\_name

Router for /categories

1. GET / <= This is GET /categories.
2. POST /add
3. POST /delete/:cat\_name

**After MVP: Login required!**

1. GET /tasks: Home page that shows to do lists only for the user, if not logged in then redirects to login.
2. GET /users/login: log in page
3. POST /users/logout: logs out user, deletes cookie
4. GET /users/register: registration page
5. POST /users/login: Sends email and password to database for authentication, sets cookie.
6. POST /users/register: Route to accept registration data (email and password) & automatically sets cookies, thereby log in.
7. PUT or POST /tasks/taskdesc/:task\_id : Edit a tasks text (POSTS or PUTS change to tasks table)
8. PUT or POST /users/:user\_id : Change password

Router for /users:

1. GET /login
2. GET /register
3. POST /login
4. POST /logout
5. POST /register
6. PUT or POST /:user\_id

Addition to /tasks router:

1. PUT or POST /taskdesc/:task\_id : Edit a tasks text (POSTS or PUTS change to tasks table)

Using a PUT method directly required the method-override technique:

[**https://web.compass.lighthouselabs.ca/days/w03d4/activities/168**](https://web.compass.lighthouselabs.ca/days/w03d4/activities/168)

[**https://dev.to/moz5691/method-override-for-put-and-delete-in-html-3fp2**](https://dev.to/moz5691/method-override-for-put-and-delete-in-html-3fp2)

**API Routes:**

1. GET /api/users - SELECT \* FROM users;
2. GET /api/tasks - Shows relevant data from all tasks.

**Stretch: Let’s go crazy with features!**

1. GET /tasks/something : Shows time to-do item was created.
2. GET /statuses: Shows statuses
3. POST /statuses/add: add new status
4. POST /statuses/delete/:status\_name Delete a status (need to ask user to change tasks has this status change to which status)

**APIs**

| **Category** | **API Documentation** | **Comment/Example** | **API Keys** | **NPM Package** |
| --- | --- | --- | --- | --- |
| Read (Books) | <https://developers.google.com/books/docs/v1/getting_started> | <https://www.googleapis.com/books/v1/volumes?q=harry%20potter> | Afsan’s Key: AIzaSyC2ZyaS7ndgLzchnCFVbk47Pp8dyCh7FLo | <https://www.npmjs.com/package/google-books-search> |
| Eat (Restaurant) | https://www.yelp.ca/developers/documentation/v3/get\_started | Yelp-Fusion | Afsan’s Key:  no78WOoWmYcy-vTJBTE9C5TCbrcwc3Bz0G-NXqW9J3hHUR2XIfwq5o6Mir5nkRQKNtRy04ZKD2OQWmheoBhRGz1mr0PMjDXFVhUL2-TEqYNJwZ-GDvCNC8QXGVEBYXYx | <https://www.npmjs.com/package/yelp-fusion> |
| Watch (TV/Film) | <https://www.omdbapi.com/> | Limit: 1000/day | Afsan’s Key: 17b20cf  Example Request: https://www.omdbapi.com/?t=Game%20of%20thrones&apikey=17b20cf | <https://www.npmjs.com/package/omdb-client> |
| Play  (Game/Sports) | https://www.uclassify.com/docs/urlrestapi | Games or Sports => Play category.  Limit: 500 / day | TJ’s Key: XtzL8saSlTH4 | <https://www.npmjs.com/package/uclassify> |
| Buy(?)  (Home/Recreation) | https://www.uclassify.com/docs/urlrestapi | Home or Recreation => Buy category.  Limit: 500 / day | TJ’s Key: XtzL8saSlTH4 |  |
| Google Knowledge | [Google Knowledge Graph Search API | Google Developers](https://developers.google.com/knowledge-graph) |  |  |  |
| SerpAPI | [SerpApi: Google Search API](https://serpapi.com/) |  |  |  |
| knowledge-graph-js | <https://www.npmjs.com/package/knowledge-graph-js> |  |  |  |

**uClassify API**

1. **Getting category name (Arts, Business, Computer, Games, Health, Home, Recreation, Science, Society, Sports) :** [**GET https://api.uclassify.com/v1/uClassify/topics/classify/?readKey=YOUR\_READ\_API\_KEY\_HERE&text=**](https://api.uclassify.com/v1/uClassify/Sentiment/classify/?readKey=YOUR_READ_API_KEY_HERE&text=I+am+so+happy+today)**YOUR\_TEXT\_HERE**
2. **Getting specified category name :** [**GET https://api.uclassify.com/v1/uClassify/CLASSIFIER\_NAME\_HERE/classify/?readKey=YOUR\_READ\_API\_KEY\_HERE&text=**](https://api.uclassify.com/v1/uClassify/Sentiment/classify/?readKey=YOUR_READ_API_KEY_HERE&text=I+am+so+happy+today)**YOUR\_TEXT\_HERE Classifier name = `${category\_name}-topics` ex) art-topics, business-topics, home-topics, etc.**

**Categorization Logic:**

Initial thinking for auto categorization logic might look like:

* 1. Check for keywords in the task (watch, eat, buy, read and their synonyms).
  2. Check uClassify/wolfram alpha results for category keywords. If found categorize.
  3. Check google books API, IMDP api, Yelp API (if found, then categorize accordingly).
  4. If none of these work, ‘category = other’ or something.
* I think to keep it simple we should keep the categories like in the project example:
* Core: Watch/Read/Eat/Buy/Other
* Stretch: More categories- Visit/Music(to listen)/Games

**‘Watch’/Read/Eat/Buy & Synonyms**

**What scripts will we need?**

|  | **Script Purpose** | **Comments** |
| --- | --- | --- |
| 1 | Load all tasks on screen load. | Partially made, linked to /api/tasks. |
| 2 | Load all tasks on category view screen load. | Partially made, needs to link to DB. |
| 3 | Delete -> Pop up confirmation -> If yes, delete task and reload. If no, then no change. | Made.  On delete click.  Will call the first script at some point within the ‘Yes’ code. |
| 4 | Add new task -> Reload all tasks with new task on top. | Made.  On add-task click.  Call first script after adding data to DB. |
| 5 | Edit button -> should trigger some form to change the current task description and title. | On click for edit button.  Will reload the tasks after making edit (first script again.) |
| 6 | Checkbox -> on click should remove the current item. | Made. |
| 7 | View toggle (between all tasks and category view). It doesn’t need to be a script, can just be a GET/tasks/categories. | On click, change between all tasks list and categorized lists. |
|  | Add more if needed... |  |

**Category icons (Font Awesome)**

| **Category** | **Icon** |
| --- | --- |
| **Read** | [**https://fontawesome.com/v5.15/icons/book?style=duotone**](https://fontawesome.com/v5.15/icons/book?style=duotone)  **<i class="fad fa-book"></i>** |
| **Eat** | [**https://fontawesome.com/v5.15/icons/utensils?style=solid**](https://fontawesome.com/v5.15/icons/utensils?style=solid)  **<i class="fas fa-utensils"></i>** |
| **Watch** | [**https://fontawesome.com/v5.15/icons/film?style=solid**](https://fontawesome.com/v5.15/icons/film?style=solid)  **<i class="fas fa-film"></i>** |
| **Play** | [**https://fontawesome.com/v5.15/icons/icons?style=solid**](https://fontawesome.com/v5.15/icons/icons?style=solid)  **<i class="fas fa-icons"></i>** |
| **Buy** | [**https://fontawesome.com/v5.15/icons/shopping-cart?style=solid**](https://fontawesome.com/v5.15/icons/shopping-cart?style=solid)  **<i class="fas fa-shopping-cart"></i>** |

**Milestones and roles**

1. **Project target: July 29, 2021 Afternoon**
2. **Dry run demo: July 29, 2021 Evening**
3. **MVP target: July 27(?), 2021 - will need to agree when Mohammad is back**
4. **MVP roles**
   1. **\* As a user, I want to view my to-do list. (Landing page iterate to-do list with <% for %>)**
   2. **\* As a user, I want to add items to a to-do list. (Form, button, route. Do we want to auto-load script in MVP like tweeter? Maybe 1 - 2 examples) - TJ wants to do this part**
   3. **\* As a user, I want to remove items from my to-do list (Button that triggers an alert with two buttons (yes/no). Clicking yes triggers a post route.)**
   4. **\* As a user, I want to check-off or mark my completed Items on my to-do list (Checkbox with two states. Clicking triggers a post route).**
   5. **\* As a user, I want to manually add categories. Question from Afsan: For MVP - add categories in general, or only to an item?**
   6. **\* As a user, I want to manually remove categories from an item. (Button that triggers a post route)**