Organization: Python Software Foundation

sub-organization: EOS Design System

Project Title: EOS icons API

Name:	**Fawzi Abdulfattah**
Mail:	**iifawzie@gmail.com**
Phone number:	**(+20)1090243795**
Timezone:	**UTC +2**
Github:	**https://github.com/iifawzi**
GitLab:	**https://GitLab.com/iifawzi**

• Motivation:

I've been coding since I was a kid, driven by passion and enthusiasm, through this journey, I've learnt and worked on a lot of open-source projects, and I was always amazed by the way the contributors work together only for one reason, helping the community!

Being able to contribute to a great open source project such as EOS Design system and working with the team to deliver a highly made icons solution to the community will be a lifetime experience. Hopefully, a lot of things will be learnt from the team, beside making a significant contribution to the project.

• Synopsis:

I will be working on the EOS Icons API to make it more reliable, faster, operational and easy to maintain, which will not be done without the guidance of the mentors. This will be made by introducing Mongodb with redis to store the icons with all related metadata, and using docker image to have solid deployment solution, beside refactoring the current codebase, adding more features like generating SVG code-snippets, base64 encoded images, tags recommendation, filtering/searching, and adding unit tests to ensure that the API is working as expected.

I've already dug into the code, fixed some bugs and introduced new ideas that might help making a positive change in the project, having the enthusiasm and the experience as I've been comfortable writing backend applications for a while, beside the community's help, hopefully, would make the process of working on this project a way much easier.

• Project background::

Currently, the EOS Icons API, is in charge of generating the new icons sets, customizing SVGs and PNGs icons, and all other stuff is made on other micro-services or in the frontend (searching and filtering). The goal is to unify all the micro-services to provide a highly and fully made solution in a single backend application as well as adding more features and improving the current codebase as mentioned below.

• What do I plan to do:

The project will be divided into 4 milestones:

A. First Milestone) Improving the current codebase:

the current codebase is well made and fit the needs very well, but wouldn't help us on a scale, and that's because it doesn't enforce any code style, and it depends on a single file (index.js) which's for sure by the time will be so messy, that said, I will be working on refactoring the codebase and the way files are structured, and for this to be achieved, the work will be divided into these points:

1- Adding eslint rules: an eslint configurations will be added to the project, to enforce coding style, which will help making the new contributors able to contribute easily, increase the readability, and to avoid any coding styles conflicts.

2- Splitting and organize the code to a well structured files/directories:
instead of depending on one entry point, the code will be divided to the following files and directories:

I) **Server File:** its responsibility is to initialize the server.

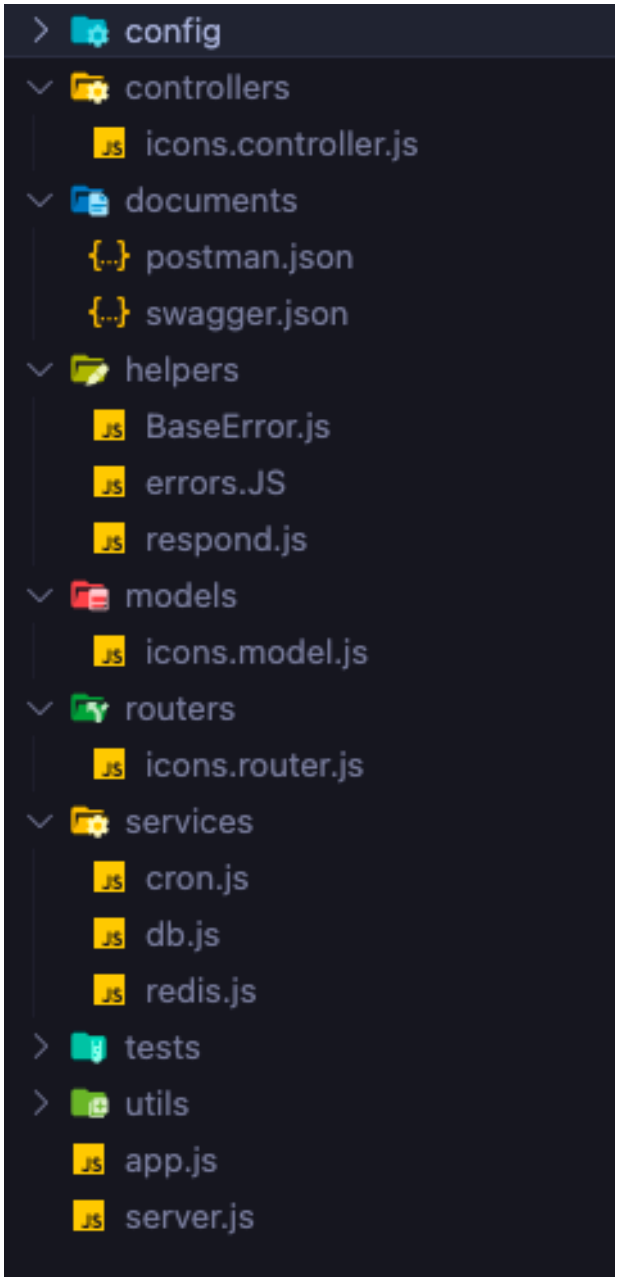
II) **App File:** its responsibility is to initialize all of the other services.

III) **Services Folder:** will contain a file for each functionality e.g (db.js, redis.js, routerSettings.js, mainErrorsHandler, etc…).

III) **Documents Folder:** will contain any documents related to the project, such as swagger documentation, postman documentation, etc…

III) Helpers Folder: a set of helper functions will be added to make the coding process a way much easier. This folder mainly contains a respond function, to be used anytime a response should be sent to the user, to unify that the responses schema, extended Error class and a fixed JSON list of errors, to unify the errors that might be sent to the user.

The image shows an abstract view of the directories and files.



- I was thinking of making a single folder for each entity that would contain the controller, model, router, and any related files in a single folder, but since there're no many entities (Collections on the database) I think it would be better to make it as the image shows, using a separate folder for each purpose (Controllers, Routers, Models).
- A middleware folder might be added if we needed to use any middleware.
- A unit tests will be added using Mocha and Chai, to ensure that the APIs respond as expected. Tests will be added in the end of the fourth milestone since most of the current APIs will be modified.
- Config folder will contain a configuration for each environment; production, development, and test.
- Winston logger will be used to log the errors.

B. Second Milestone) Using MongoDB as a source of truth for the icons.

A MongoDB database will be used to retrieve, edit, add, or update the icons itself, tags, or description.

Mainly, two collections will be created:

1- info collection:

This collection will contain the main info about the icons collection, such as, **current_version**, which would be used in the cronjob to check if there's a newer release with a version higher than the current one.

2- icons collection:

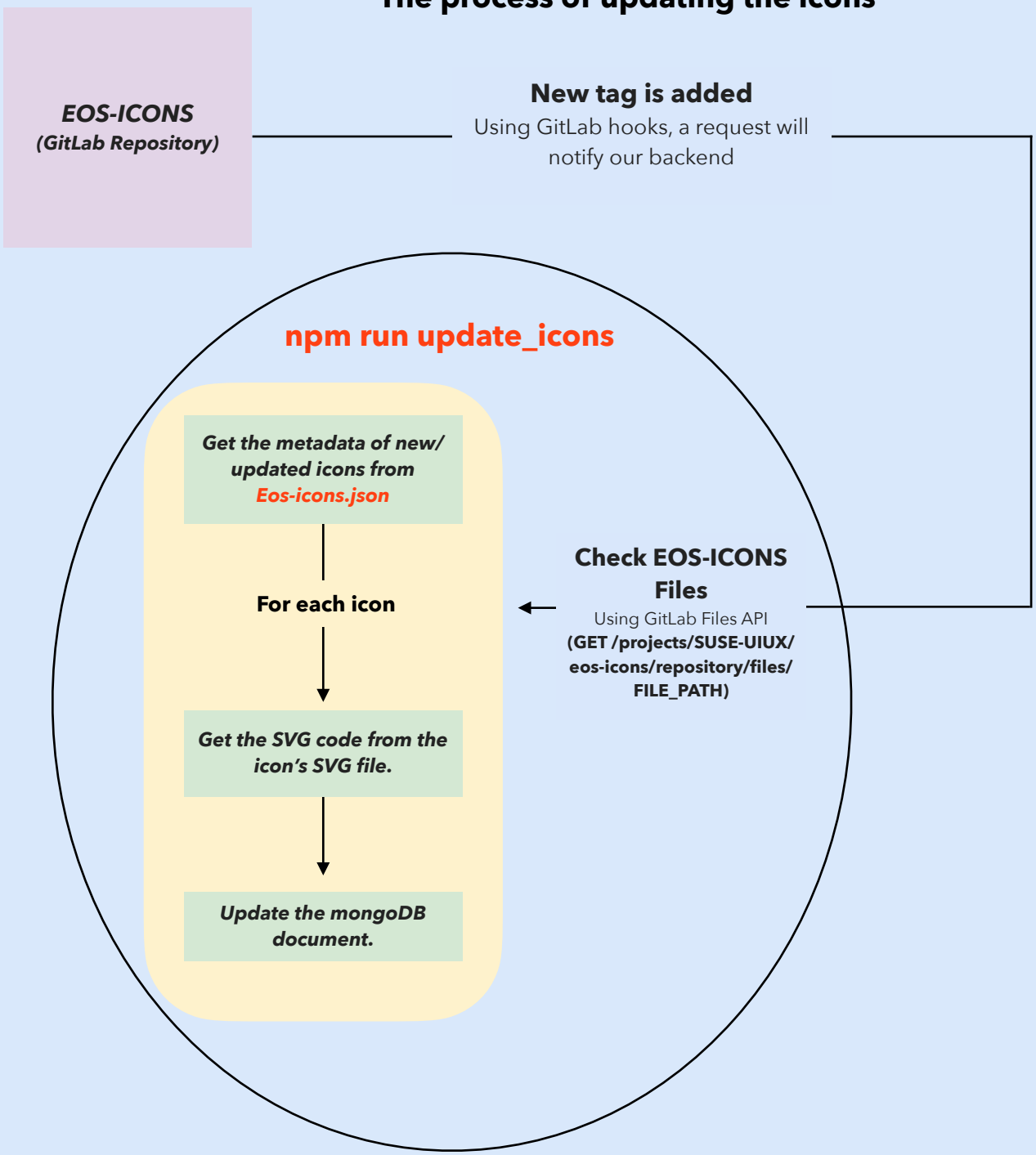
```
{
        icon_id: <ObjectId1>
        icon_name: String
        do: String
        dont: String
        icon_categories: [Strings]
        icon_type: ENUM ("Animated", "static")
        icon_svg: String
        icon_tags: [Strings]
        icon_suggested_tags: [{
                sug_tag_name: String,
                votes: {
                        upVotes: Int,
                        downVotes:Int
                }
        }]
}
```

An **icons** collection will be created with the shown fields, which will be initially filled using a function that will loop on the Eos-icons.json and add the icons to the db, then when a new tag is added in Gitlab a request will be send to our webhook-reciever using GitLab hooks, if there's any new or updated icons, a function will be called to get the meta info from Eos-icons.json and update the db as shown in the diagram below.

- After any new release, we don't need to loop and update all icons, only the modified or new icons should be changed/added, and this will be determined depending on the date field in the json file.

Next Diagram explains the process.

The process of updating the icons



Notes on the diagram:

- An EOS-BOT GitLab account will be created to be used to access the GitLab APIs:
    - An Access-token of EOS-BOT with only READ API scope and READ REPOSITORY scope will be used to access the releases and files APIs, and this token will be published **publicly**, this will be used to read the releases and the Eos-icons.json file.
- The cache of any updated icon will be invalidated.
- To manually update and check if there're any new/updated icons, npm run update_icons script can be ran, and it will be used in the initial setup with –initial argument to add all icons to the db. Or without –initial to only check the new icons.

We have two other solutions to git rid of depending on the GitLab repo (to get the info/SVG of the updated/new icons)

1- Depending on the npm package, but this might lead to conflicts if the package and GitLab repo aren't always in sync, since the releases will be checked from GitLab and the info/files will be taken from the npm package.

2- Writing a script that will loop on the icons in Eos-icons.json and get the actual SVG string and save the SVG string also it in the icon object in the json file, after that, if any icon get updated/added, a function can be run by the person who made that change, which will get the code from the file and put it in the icon's object in Eos-icons.json, or maybe copy and paste the SVG string manually.

That said, when any version is released, the cronjob will only work with the Eos-icons.json, since the SVG string will be already there, and this's might be better than depending on Gitlab APIs to fetch each icon's SVG string.

> **The disadvantage of this approach:** it will require manually running the function that will copy the SVG string from the icon's file.

Those might be an alternative solutions if we faced any issues with the idea proposed in the diagram.

C. Third Milestone) Improving the APIs and provide new ways to consume the icons:

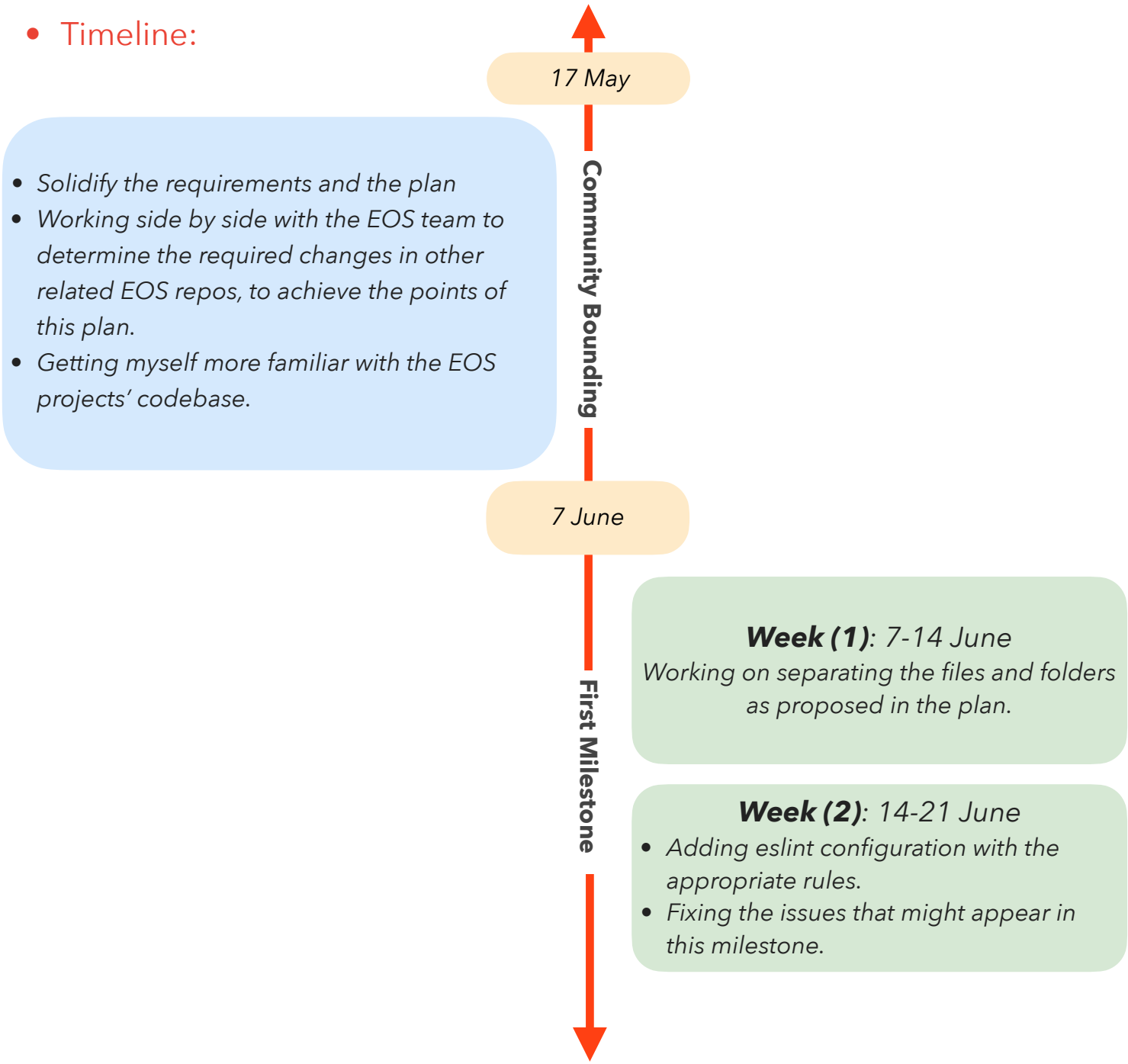
- Current API is only generating SVG icons and PNG images, I will work on adding the ability to get the SVG string, which will be directly fetched from the db, and also adding a Base64 option, which will be generated from the SVG code.
- The system will only store the SVG strings in the db, and all other options will be available by converting the SVG to SVG File, Base64 or PNG.
- There would be a single API that will be able to respond with all needed types of icons either customized (if query parameters are passed) or not customized.
- A Search Api will be added to filter the icons based on tags, categories, name, or type. proper indexes will be created to avoid the full collection scan, beside caching, hopefully all responses will take Milliseconds.
- An API will be added to allow the users to suggest tags for the icons, and maybe we may add a cronjob to check if any suggested tag has more than X upvote, then, will be moved automatically to the permanent tags, or if it has more than X downvote, will be removed from the suggested tags.
- If time allows, finding an alternative of **fontforge,** since it produces unexpected bugs sometimes.

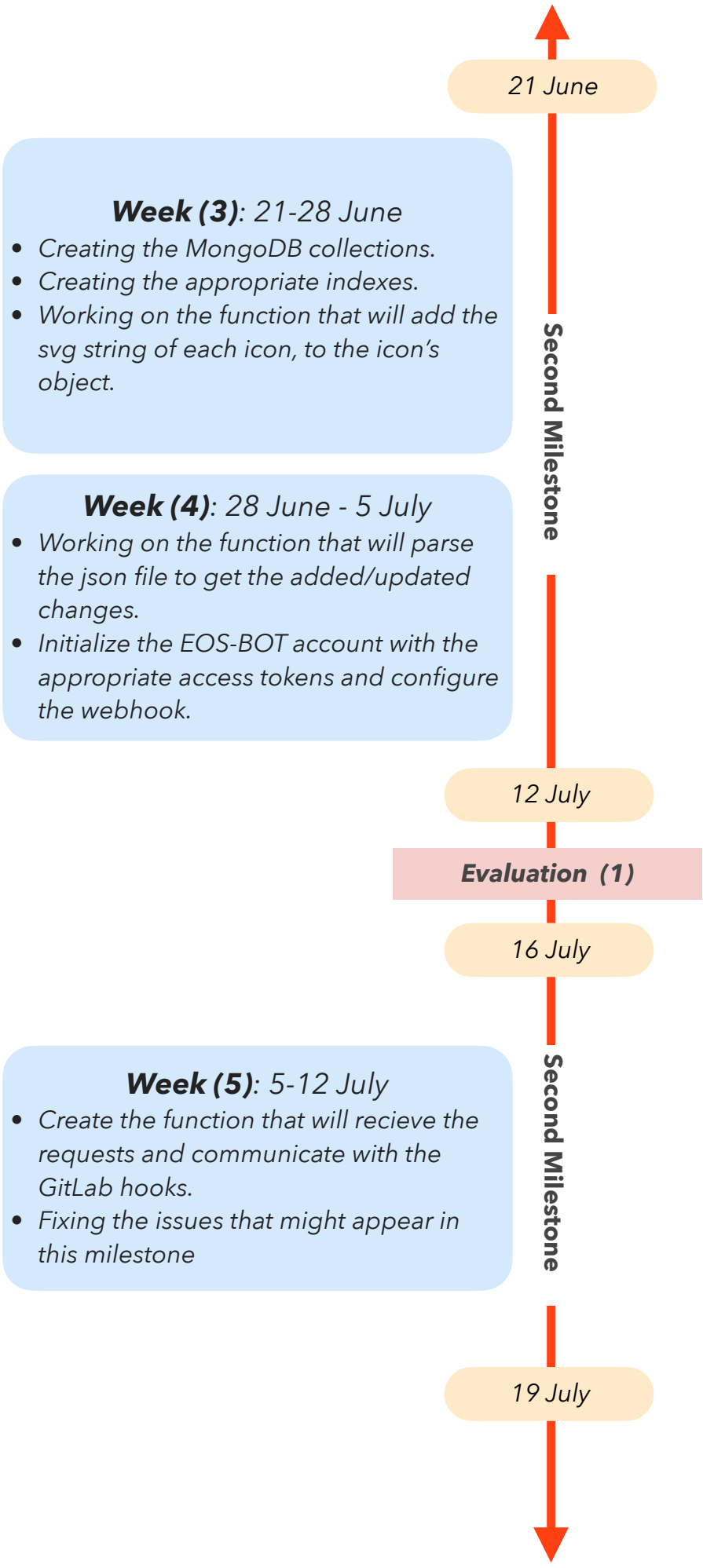
D. Fourth Milestone) Dockerizing, translating, unit testing and documenting.

- Creating a docker image that will start a Node, MongoDB, and redis servers for an easy, fast and solid deployment process.
- Adding unit tests using **Mocha** and **Chai**, to test that all APIs work as expected.
- Adding a translation middleware, to allow multi-language search and requests. Deepl translation service were proposed in the project, but I've found a good open-source alternative which currently supports 12 languages, if it's not enough and we wanted to support more languages we would need to use any other paid services, e.g Deepl or google cloud.
- Adding Swagger documentation for the APIs.

- What I've done so far with the idea:
    - Getting myself familiar with the codebase of the EOS projects.
    - I've created two pull requests that should fix some issues.
    - Getting myself familiar with the dockerizing concepts, as I'vent used to dockerize my applications, i've created a tiny docker image with redis, mysql and node servers where they interact with each other, to practice the concepts.
    - Getting myself familiar with the GitLab Workflow and APIs to propose the idea of depending on EOS-BOT

- Timeline:




21 June
Week (3): 21-28 June
Creating the MongoDB collections.
Creating the appropriate indexes.
Working on the function that will add the svg string of each icon, to the icon's object.
Second Milestone
Week (4): 28 June - 5 July
Working on the function that will parse the json file to get the added/updated changes.
Initialize the EOS-BOT account with the appropriate access tokens and configure the webhook.
12 July
Evaluation (1)
16 July
Week (5): 5-12 July
Create the function that will recieve the requests and communicate with the GitLab hooks.
Fixing the issues that might appear in this milestone
Second Milestone
19 July

Third Milestone

***Week (6)**: 12-19 July*

- *Create the function that will be used to convert SVG string to Base64.*
- *Create the function that will create SVG files from the SVG strings.*
- *Create /icon API that will be used to respond to all requests that need any type of icons (SVG string, file, PNG, or font)*

***Week (7-8)**: 19 July - 2 August*

- *Create the search API that will be used to filter the icons.*
- *Create the APIs that will deal with the suggested icons (add, upvote/downvote)*
- *Cache the non-customized icons.*
- *Fixing the issues that might appear in this milestone.*

2 August

Fourth Milestone

***Week (9)**: 2-9 August*

- *Working on dockerizing the whole backend system.*
- *Documenting the APIs using swagger and update the postman docs.*

***Week (10)**: 9-16 August*

- *Unit testing the APIs and ensure that All of the system functionalities are working as expected.*
- *Fixing the issues that might appear in this milestone*

16 August

Final Evaluation

23 August

- Previous Experience:
    - I've been coding since I was a kid, have written some application in native php, mysql, css, and html. Lately I focused on the backend especially with node.js and express.js.
    - Worked as a freelance for a while and built two full-stack applications using Node.js, Express, Nuxt.js, JS, and sass. Also, I've built some small frontend applications using react. I was responsible for every single detail, starting from the design to the documenting and testing, which's helped me learning and working with a lot of technologies.
    - I've started contributing to open source projects earlier this year and I've made a bunch of hopefully, great contributions:

      PhpMyAdmin
      - Fixing a bug that occurs when default current_timestamp is used with on update current_timestamp **#16653**
      - Fixing a query generation bug that allows specifying a length for JSON fields **#16643**

      Nuxt-i18n
      - Adding direction property and defaultDirection option, so users will be able to choose a default direction that will be applied on the html elements and a locale-specific direction **#1023**
      - Exposing localeProperties property to access the properties of current locale **#1016**
      - Fixing localeProperties became undefined when <i18n> component is used **#1043**

      Joi Validation
      - Fixing an unhandled feature that fills the default values when used in an ordered array schema **#2548**

      GraphQL
      - Fixing a tiny design bug in the graphiql website **#1776**

      Google Chrome
      - Fixing a tiny design bug in their developers' website **#308**

- **I've also created two merge requests to the EOS-ICONS API's project:**
    - Isolating the process of establishing the server from the application logic **#46**
    - Change the process of coping icons to deal with different operating systems **#45**

- About me:
    - An energetic and optimistic undergraduate Egyptian student who's studying computer engineering in Egypt, they're saying that I'm a Geek!, but I'm not, I'm just passionate about computer engineering and coding, especially the backend. When I'm not coding, I'm eating.
    - I spend my free time playing football, reading books, building side projects to enhance my skills, or lately, contributing to open source projects.
    - I love volunteering and helping people.
- Commitments:
    - I will be having 12 days of final exams in the period between July to august, it's not determined yet, anyway, I will manage to work hard until the exams begin and re-arrange the timeline when the schedule is announced, to maybe leave the easy parts to be made during exams.