Feature Flag Management Platform

About Autofi

AutoFi is the leading commerce platform for end-to-end digital automotive sales and financing, powering billions of online transactions every year. Through its innovative platform and expansive network of trusted lenders, AutoFi empowers sellers, including the nation's top dealers and digital retailers, to sell more efficiently and profitably. Additionally, AutoFi's Lending-as-a-Service API offers customized solutions that power the financing for the largest branded automotive marketplaces in the industry.

Problem statement

This project will involve building the prototype of a tool that would help Autofi to internally manage its Feature Flags. Within Autofi we utilize Feature Flags to perform Soft Rollouts on features and A/B testing. Currently, we use an external tool called LaunchDarkly to manage our Feature Flags across all our repositories and applications.
Although LaunchDarkly provides great benefits like low latency and scalability. It restricts the number of users that have access to turn on/off flags. In Autofi's growing engineering team this is a problem since feature-flagged code needs to be tested throughout all our environments in the release process.

Solution

Autofi needs to create an internal tool that will connect to LaunchDarkly's API and expose some of its features through a web platform for easy access and management.
The two core features of this prototype will be:

- To list all active Feature Flags and their status
- To be able to turn a certain feature flag on or off

Tech Spec

This prototype will require two components, a front end and a back end.

- The back should be in charge of handling the API connection to LaunchDarkly and exposing the necessary through its own API endpoint for it to be consumed by the front end.
- The front end should utilize the back end API to list all the available Feature Flags and provide the ability turn them on and off.

Back-end

- The backend should utilize a JS-based framework, ideally NodeJS and Express
- The backend should be structured in two sections. One for communicating with Launch Darkly and one for communicating with the front end.

For Launch Darkly Communication:

- Get All Feature Flags
  - Endpoint: Get /api/v2/flags/default
  - Parameters:
    - Authorization token: Created via https://docs.launchdarkly.com/home/account-security/api-access-tokens
    - Project key: Should use ‘default’ unless otherwise specified
  - Results: From the results, we only need to grab the following
    - Name: items[x].key
    - Value: items[x].environments.production.on
  - Docs link: https://apidocs.launchdarkly.com/tag/Feature-flags#operation/getFeatureFlags

- Update Feature Flag
  - Endpoint: Patch /api/v2/flags/default/test-1
  - Parameters
    - Authorization token: Created via https://docs.launchdarkly.com/home/account-security/api-access-tokens
    - Project key: Should use ‘default’ unless otherwise specified
    - Feature Flag key: To be grabbed from the list above

```
{
 "patch": [
   {
     "op": "replace",
     "path": "/environments/production/on",
     "value": false
   }
 ]
}
```

  - Docs link: https://apidocs.launchdarkly.com/tag/Feature-flags#operation/patchFeatureFlag

For Front-end Communication:

- Get all Feature Flags
    - Calls the LaunchDarkly Endpoint from above
    - Endpoint GET /GetAllFeatureFlags
    - Returns array of object { name , value }
- Updates Feature Flag
    - Calls the LaunchDarkly Endpoint from above
    - Endpoint POST /UpdateFeatureFlag
    - Parameters: Feature Flag key

Front-end

- The backend should utilize a JS-based framework, ideally NextJS
- There should be two pages;
    - Home Page:
        - On Load it calls /GetAllFeatureFlags and fetches all the feature flags
        - Creates a table a table to display the feature flags and their current value
        - On Click it will redirect to the FeatureFlag Page
    - FeatureFlag Page
        - On Load it displays the current feature flag key and a toggle set to the current value of the Feature Flag
        - It has a save button that will save the Updates

Access

There will be a single Launch Darkly account available. To get access to this reach out to sebastian@autofi.io

Structure

There should be SCRUM standups at least once a week where the progress of the project will be discussed in regular SCRUM and AGILE practices.
If the practices are unclear to the team they will be explained during the first SCRUM.
The time and date of the meeting will be discussed and scheduled according to everyone's schedules.

For Version control, a GitHub account should be created by each one of the team members and a project should be set up for this prototype.

PR reviews are encouraged.