How To Develop a New Module In UPYOG

Developing in UPYOG needs a Tech Stack which is - HTML-CSS, Javascript, React.js, JSON, and GIT as well.

1. The first step that is mandatory to develop any module in UPYOG-Micro-UI is enabling the module in MDMS. So Open this Link - https://github.com/upyog/upyog-mdms-data. then go to this directory - data/pg/tenant/citymodule.json and write this piece of code -

```
{
    "module": "PTR",                    // Name of your module in Capital Letters
    "code": "PTR",                      // Give the Same Name as a Code
    "bannerImage": "https://egov-uat-assets.s3.amazonaws.com/PT.png",
    "active": true,                     // set to True is mandatory
    "order":1,
    "tenants": [                        // Tenants denotes the number of cities you want
      {
       "code": "pg.citya"
      },
      {
       "code": "pg.cityb"
      },
      {
       "code": "pg.cityc"
      },
      {
       "code": "pg.cityd"
      },
      {
       "code": "pg.citye"
      }
    ]
  }
```

2. After this Step, remember to restart the MDMS Service. Now Fork and Clone this repository from Github - https://github.com/upyog/UPYOG, open it in VSCode or any other code editor, choose the master branch, and go to this directory - frontend/micro-ui.

   Note - To Run This Locally you can Refer to the Document already made.

3. Now Go to this Directory - frontend\micro-ui\web\micro-ui-internals\packages\modules and make a New Folder and give the name which is the name of your module. After this inside that folder create a file which is **package.json** which is a mandatory file and required for every Module. You can take a reference of this or you can directly copy and paste this file with few changes.

```
{
 "name": "@nudmcdgnpm/upyog-ui-module-asset",  // here change the name from asset
                                                                  to your module name
 "author": "Shivank Shukla - NIUA",              // Change it or you can remove
 "version": "1.0.0",
 "license": "MIT",
 "main": "dist/index.js",
 "module": "dist/index.modern.js",
 "source": "src/Module.js",                                  //this can be same and mandatory
 "files": [
  "dist"
 ],
 "scripts": {
  "start": "microbundle-crl watch --no-compress --format modern,cjs",
  "build": "microbundle-crl --compress --no-sourcemap --format cjs",
  "prepublish": "yarn build"
 },
 "peerDependencies": {
  "react": "17.0.2",
  "react-router-dom": "5.3.0"
 },
 "dependencies": {                              // any external dependency will be added here
  "@nudmcdgnpm/digit-ui-libraries": "1.2.0",                  //mandatory
  "@nudmcdgnpm/digit-ui-react-components": "1.2.0",     // mandatory
  "exif-js": "^2.3.0",
  "jspdf": "^2.5.1",
  "jspdf-autotable": "^3.8.2",
  "lodash.merge": "^4.6.2",
  "microbundle-crl": "^0.13.11",
  "qrcode": "^1.5.3",
  "react": "17.0.2",
  "react-dom": "17.0.2",
  "react-hook-form": "6.15.8",
  "react-i18next": "11.16.2",
  "react-query": "3.6.1",
  "react-router-dom": "5.3.0",
  "xlsx": "^0.18.5"
 }
}
```

4. Now go to this directory - frontend\micro-ui\web\package.json and add the path of your module in the workspace, take a reference of this -

```
"workspaces": [
  "micro-ui-internals/packages/libraries",
  "micro-ui-internals/packages/react-components",
  "micro-ui-internals/packages/modules/common",
  "micro-ui-internals/packages/modules/core",
  "micro-ui-internals/packages/modules/obps"     // add same
 ],
```

   and add the package name under dependencies which are the same as the package name of your module, take a reference - "@upyog-niua/upyog-ui-module-name-of-your-module": "1.0.0".

5. You must repeat The same step in this directory - frontend\micro-ui\web\micro-ui-internals\example\package.json and add the dependency.

6. Now go to this directory - frontend\micro-ui\web\micro-ui-internals\package.json
   Again add the path of your module in the workspace and the **dev** and **build** under the scripts, you can take a reference of **PTR.**

7. Now go under your module folder, and make a new folder named **src**, and under the src folder create a few more folders with names - **components**, **pageComponents**, **pages, config,** and **utils** and one individual file named Module.js, Module.js file is used to import all the components we create or developed all over the modules and register as well.
8. Create one more file Module.js under SRC folder, take a reference of PTR module.js and change the Module code inside that which is similar to your code in Citymodule.json
   In this component, i have imported all the components