Workflow for MoTR project based on Magpie

1. Create and design the Magpie project, follow this step
2. In the file *package.json*, change the "node" version to

```
"node": "^16.x",
```

3. Change the "npm" version to

```
"npm": "^7.x"
```

4. In the file *src/magpie.config.js*, set the correct configurations according to the server you set (here using *Heroku*, where an app has been created with url: *cui-test-3.herokuapp.com*)

```
export default {
  // The following 3 lines can be obatained in *cui-test-3.herokuapp.com* by clicking
  experimentId: '1',
  serverUrl: 'https://cui-test-3.herokuapp.com/',
  socketUrl: 'wss://cui-test-3.herokuapp.com/socket',
  // this will be used in prolific mode
  completionUrl: 'https://...',
  contactEmail: 'cui.ding@uzh.ch',
  // Either 'debug', 'directLink' or 'prolific'
  mode: 'directLink',
  language: 'en'
};
```

4. Upload the project to Github. Make sure the file `deploy-to-gh-pages.yml` exists in the folder `.github/workflows/` so that the Github can automatically build the experiment and commit the output to the `gh-pages` branch. Once you enable GitHub Pages for your repository, GitHub will make the latest version of the `gh-pages` branch available publicly, see here

Deploy server

I guess this part causes problem for you.
The problem is that heroku is not free anymore, so the backend wrote by magpie people cannot work anymore. You have to change “hobby-dev” to “basic” here(I think basic is enough).
If it is not, you can update your plan later

Heroku (online)

1. Clone the repository: https://github.com/magpie-ea/magpie-backend#deployment-with-heroku
2. In the *heroku-deploy.sh* file, change the line 15 *heroku addons:create heroku-postgresql:hobby-dev* to *heroku addons:create heroku-postgresql:mini* (the *mini* can also be *basic* etc. depending on the plan you choose).
3. Then run the script with `sh heroku-deploy.sh cui-test-3 USERNAME PASSWORD` (the *USERNAME* and *PASSWORD* can be different from the ones you log in Heroku). You will be

prompted to the app with link *https://cui-test-3.herokuapp.com

4. In the app you can create new experiment and get the following parameters for the *magpie.config.json* file above by clicking *Edit*:

```
experimentId: '1',
  serverUrl: 'https://cui-test-3.herokuapp.com/',
  socketUrl: 'wss://cui-test-3.herokuapp.com/socket',
```

5. In case the postgresql database has been deleted and created again, run `bash heroku-update.sh` in the magpie-backend folder to connect the database to the app correctly.

Docker (local)

1. In the file *src/magpie.config.js*, set the correct configurations:

```
export default {
  experimentId: '1',
  serverUrl: 'http://localhost:4000/',
  socketUrl: 'wss://cui-test-3.herokuapp.com/socket',
  // this will be used in prolific mode
  completionUrl: 'https://...',
  contactEmail: 'cui.ding@uzh.ch',
  // Either 'debug', 'directLink' or 'prolific'
  mode: 'directLink',
  language: 'en'
};
```

2. Follow the steps in this link. Be aware that when you run `docker-compose up` you don't have to wait till it finishes before you visit `localhost:4000` in your browser.

Host on Github pages

Host only one experiment

Make sure the project folders and files are under the root directory before pushing to Github. In the repo, go to *Settings-Pages-Brach-choose gh-pages* and then *save*. Refresh the page after a few minutes and you will get the link to your experiment, which will be the link sent to the experiment participants. Optionally, you can custom the domain name in the *Custom Domain* on the same page.

Host multiple experiments in the same repository

1. In the root directory, create `experiments/` folder, and put each experiment subfolder (e.g. `pilot-01/` , `pilot-02/`) in it.

2. Replace the content of the file `deploy-to-gh-pages.yml` in `.github/workflows/` with the code in this link, and change the following part in the code according to your directory structure:

   ```
   strategy: matrix: folder:
   - experiments/pilot-01/
   - experiments/pilot-02/
   # ...
   ```

3. Change the following part in `vue.config.js` in each subfolder, e.g. for `pilot-01/`:

```
publicPath:
    process.env.NODE_ENV === 'production'
      ? '/YOUR_REPOSITORY-NAME/experiments/pilot-01/'
      : '/'
```

4. On the Github website of your repository, go to `Settings-Actions-general`, and choose the following option to allow for read and write permission: ![[2023-03-02 14.52.42.png]]
5. Go to *Settings-Pages-Brach-choose gh-pages* and then *save*. The experiments will then be hosted at `https://your-organisation.github.io/your-repository/experiments/pilot-01/` and `https://your-organisation.github.io/your-repository/experiments/pilot-02/`

normally, the github branch “gh-pages” will be created automatically. If you get errors related to “Create branch uses: peterjgrainger/action-create-branch@v2.0.1”, the most straightforward solution would be just create a branch called “gh-pages” manually and then deploy motr on github again.

Manage the experimental results

1. Go to https://cui-test-3.herokuapp.com and log in.
2. Click *RETRIEVE CSV* and the csv file will be downloaded. Note that all the data from the paticipants will be all included and appended in this csv file.

Create customized Magpie base

If you used the recent codes from MoTR repo and followed the readme, you probably have done this part. The original codes by magpie store too many unnecessary data. This part changed the way of storing data.

To economize the data size produced by Magpie implementation. We have to modify the Magpie base.

1. I cloned the `magpie_base` from https://github.com/magpie-ea/magpie-base
2. In `Magpie.js`, which is in the `src` directory, I change the function `flattenData` starting from line 530. Instead of the original way of mapping in line 548 to line 558, I used this codes snippet:

   ```
   var flattenedTrials = map(trials, function (t, index) {
     for (const key in t) {
       if (Array.isArray(t[key])) {
         t[key] = t[key].join('|'); // Convert arrays to strings
       }
     }
   ```

```
      // Merge with experiment data only for the first trial
      if (index === 0) {
        return merge(t, data);
      } else {
        // Create an object with null values for experiment data keys
        const nullExpData = Object.keys(data).reduce((obj, key) => {
          obj[key] = null;
          return obj;
        }, {});

        return merge(t, nullExpData);
      }
    });

    return flattenedTrials;
```

3. I renamed the `magpie-base` directory to `magpie-base-MoTR` and uploaded it to my github repository.

Use customized Magpie base

For all the experiments which you want to use the customized Magpie base, which is now stored in Cui's github, you need to

1. In `package.json` file, change the "magpie-base" in line 22 to `"magpie-base": "git+https://github.com/cuierd/magpie-base-MoTR.git#main",`
2. If you have downloaded/cloned the MoTR repository and have tried to run the MoTR webapplication for your experiment, you will have a `package-lock.json` in your experiment directory. In this case, we recommend you to run the `update_modules.sh` in the experiment folder by open a terminal at this folder and run

```
sh update_modules.sh
```

In this bash file, the first line `rm -rf package-lock.json` will remove the `package-lock.json` file which contains the old project-specific configurations and dependencies.

The second line `rm -rf node_modules/magpie-base` will remove the old `magpie-base` .

The third line `npm install` will install all kinds of small libraries that magpie needs to run, note that in this case it will install `magpie-base-MoTR` instead of `magpie-base` .

The fourth line `npm run build` creates a final build of your project.

The fifth line `npm run serve` runs a development instance of your project for testing.

Note, the `npm run build` and `npm run serve` is switchable because testing your experiment and finalize it can be a circular.

The sixth line `# Also clear the history in the browser!` is a reminder to clear your cookies if sometimes your expected effects do not show up.