IPL ELECTRON APP DEVELOPMENT DOCS

Please note that this application was developed on a Kali Linux Distribution and the instructions provided in this doc may vary across systems. Also, this doc assumes that you already have emscripten installed and configured on your machine. If not, please check instructions here. Other prerequisite dependencies include **nodejs** and **python3.**

Create Electron Project using Electron Forge:

```
npm init electron-app@latest ipl
```

You may get a prompt:

```
Need to install the following packages:
  create-electron-app@6.0.4
Ok to proceed? (y)
```

Type Y and click enter. Let the installation process run its course.

```
✓ Locating custom template: "base"
✓ Initializing directory
✓ Preparing template
✓ Initializing template
✓ Installing template dependencies
```

After installation is completed, you will get a directory structure

```
forge.config.js  node_modules  package.json  package-lock.json  src
```

The src directory will contains the files needed for app development.

```
index.css  index.html  index.js  preload.js
```

More info about Electron Forge can be found here.

IPL APP DEVELOPMENT:

The application was made from transpiled C++ code from Lab 3 with very minimal changes. The main of the application was defined in wc.cpp. It defines all the methods exposed to the client as well as other methods used to manipulate the dom in the browser. EM_JS was used to embed JavaScript code within the C++ code. The embedded JavaScript was then used to perform specific operations within the browser.

BUILDING IPL APPLICATION:

C++ files were placed in Code directory while JavaScript files generated by Electron Forge were places in the Code/ipl directory. The C++ code was then transpiled into ipl.js and ipl.wasm and placed in Code/ipl/src/js directory.

The target distribution for this APP is Linux. To build the distribution, navigate to ipl directory and run

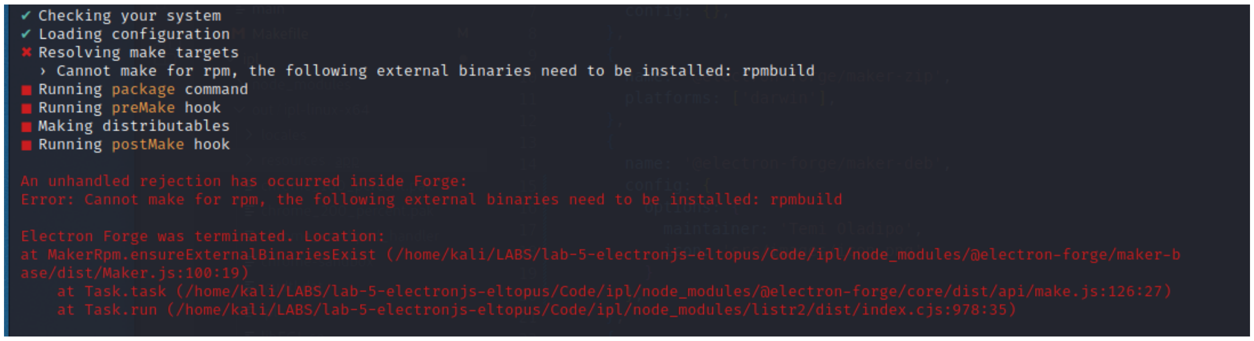
```
npm run make
```

```
> ipl@1.0.0 make
> electron-forge make

✓ Checking your system
✓ Loading configuration
✓ Resolving make targets
  › Making for the following targets: deb, rpm
✓ Running package command
  ✓ Preparing to package application
  ✓ Running packaging hooks
    ✓ Running generateAssets hook
    ✓ Running prePackage hook
  ✓ Packaging application
    ✓ Packaging for x64 on linux [13s]
  ✓ Running postPackage hook
✓ Running preMake hook
✓ Making distributables
  ✓ Making a deb distributable for linux/x64 [55s]
  ✓ Making a rpm distributable for linux/x64 [1m19s]
✓ Running postMake hook
  › Artifacts available at: /home/kali/LABS/lab-5-electronjs-
eltopus/Code/ipl/out/make
```

You can install the .deb on local Linux machine by navigating to Code/ipl/out/make/deb folder and running

```
sudo dpkg -i ipl_1.0.0_amd64.deb
```

Application will be installed on local machine. If you encounter an issue such as:



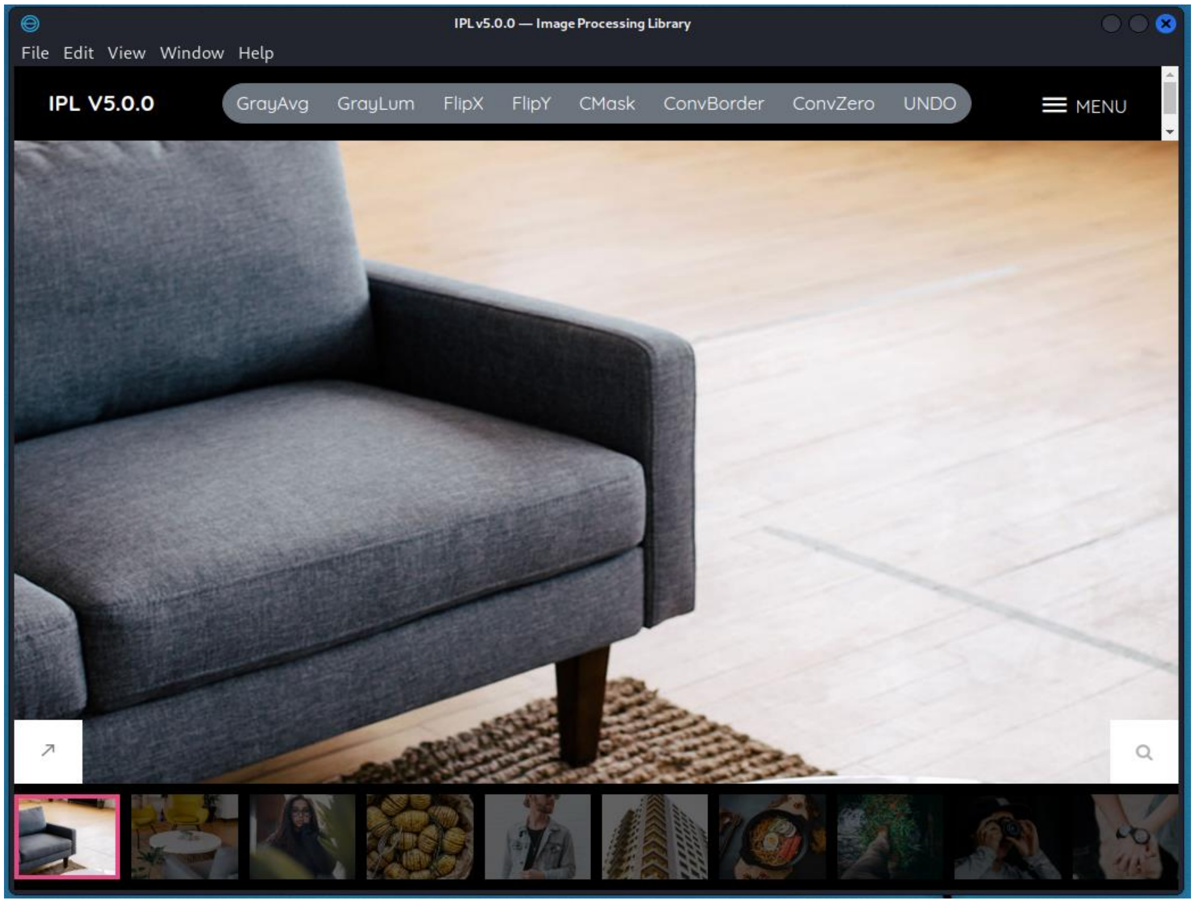
```
✔ Checking your system
✔ Loading configuration
✖ Resolving make targets
  › Cannot make for rpm, the following external binaries need to be installed: rpmbuild
■ Running package command
■ Running preMake hook
■ Making distributables
■ Running postMake hook

An unhandled rejection has occurred inside Forge:
Error: Cannot make for rpm, the following external binaries need to be installed: rpmbuild

Electron Forge was terminated. Location:
at MakerRpm.ensureExternalBinariesExist (/home/kali/LABS/lab-5-electronjs-eltopus/Code/ipl/node_modules/@electron-forge/maker-b
ase/dist/Maker.js:100:19)
    at Task.task (/home/kali/LABS/lab-5-electronjs-eltopus/Code/ipl/node_modules/@electron-forge/core/dist/api/make.js:126:27)
    at Task.run (/home/kali/LABS/lab-5-electronjs-eltopus/Code/ipl/node_modules/listr2/dist/index.cjs:978:35)
```

Run the following command to fix the issue.

```
sudo apt-get install rpm
```

IPL APP DESCRIPTION

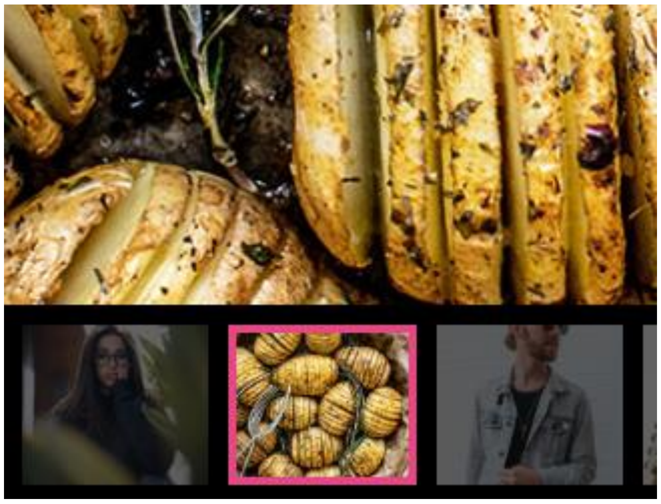
IPL is an Electron Native App that can process images using specific pre-defined techniques. After application on local machine, you can look for the icon on applications list and double click to startup the app. On app startup, you are presented with:



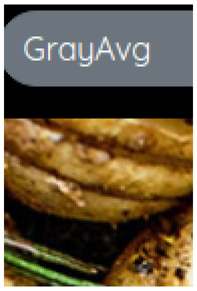
The app starts up with a list of predefined images. Subsequent versions will allow user to upload and edit custom images, but this version does not.

To test a feature, select an image listed in at the bottom of the page and click the button corresponding to the feature. For example, to perform a Gray Scale Average operation on a selected image:

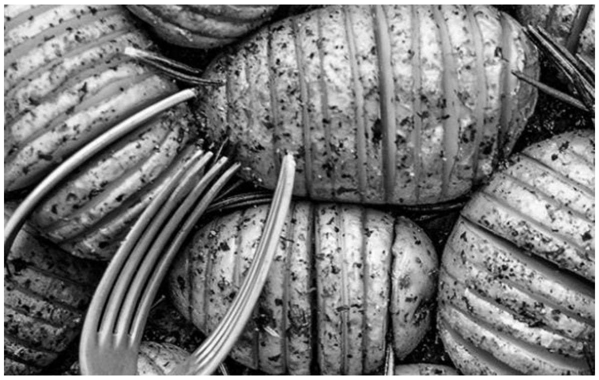
- Selected the image



- Click GrayAvg button.



- Grayscale is applied to image



- You can undo any operation by clicking the undo button

Other operations than can be performed on images include

- Gray Scale Luma: applies grayscale with human visible luma.
- FlipX flips image on X axis.
- FlipY flips image on Y axis.
- Color Mask Applies predefined color mask to image.
- Convolution Border applies predefined convolution to image.
- Convolution Zero applies predefined convolution to image.
- Undo reverts applied operation.

NATIVE CLIENT NOTIFICATION

Native client notification was implemented to notify user after operation has been completed. To see the feature in action,

- Launch application.
- Select any image.
- Click on any operation button.
- Notification pops up on the right-top corner saying what operation was completed.