LAB 4: ELECTRON FEATURES

In this exercise, you will:

- Add native menu
- Add native notifications
- Add new browser window
- Perform File Read operation and IPC communication from main process to renderer process
- Use axios library for http requests

1. Check out the `Lab-Electron-Features` branch from the remote repository.

```
git checkout origin/Lab-Electron-Features
```

Note: The HEAD of this branch provides the completed code for the entire exercise.

---

2. Run `npm install` to ensure all the dependencies have been installed.
3. Create a new local branch for your work based on the "starting point..." commit.

```
git log --oneline
git checkout <hash for starting point commit>
git checkout -b Lab-Electron-Features-mine
```

4. Start the app.

```
npm run start
```

Add native menu

5. Create a reference to the Menu class from the electron package. Use below line of code to replace the old first line of code in main.js file (its present at the root level of project structure).

```
const { app, BrowserWindow, Menu } = require('electron')
```

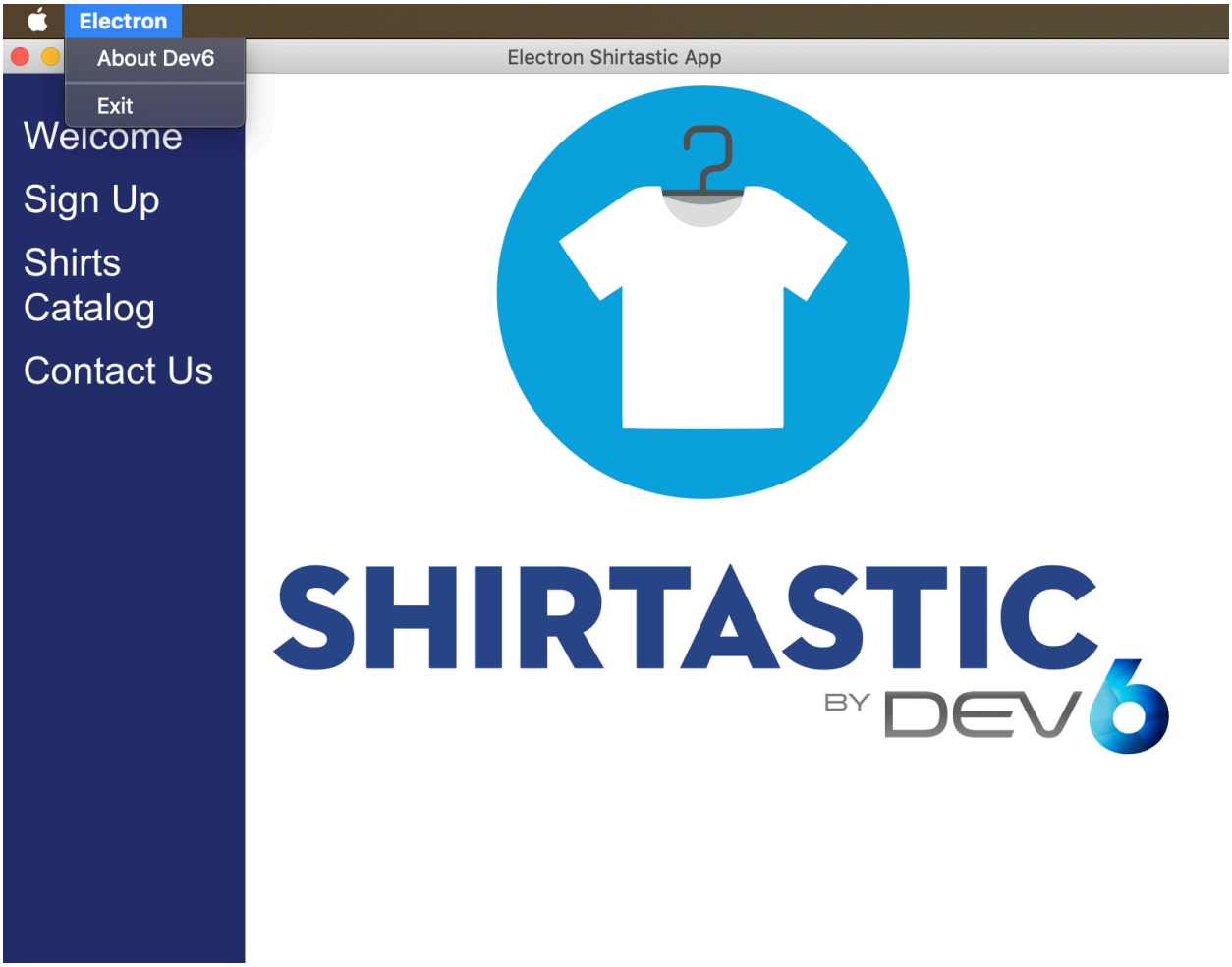
6. Next step is to declare Menu and add it to existing code of createWindow function inside main.js file. Place below code just below win.on('closed') event handler.

```
var menu = Menu.buildFromTemplate([
    {
        label: 'Menu',
        submenu: [
            {
                label: 'About Dev6',
                click() {
                    console.log('clicked');
                }
            },
            {type: 'separator'},
            {
                label: 'Exit',
                click() {
                    app.quit()
                }
            }
        ]
    }
])
```

```
Menu.setApplicationMenu(menu);
```

7. Restart the app.

```
npm run start
```



Note: Now you can check your app has native menus on screen.

8. Now let's add some functionality to About Dev6 menu item. We will use Shell module, to open external links. Place below line of code in main.js file just under previous declarations done.

```
const shell = require('electron').shell
```

9. Next step is to use openExternal function from shell module. Observe in below screenshot how we used openExternal function to open an external link.

```
var menu = Menu.buildFromTemplate([
  {
    label: 'Menu',
    submenu: [
      {
        label: 'About Dev6',
        click() {
          shell.openExternal('https://www.dev6.com/')
        }
      },
      {type: 'separator'},
      {
        label: 'Exit',
        click() {
          app.quit()
        }
      }
    ]
  }
])
  Menu.setApplicationMenu(menu);
}
```

10. Restart the app.

```
npm run start
```

Note: Now you will notice when you click on About Dev6, it opens the URL in user's default browser. Try clicking on Exit menu link and observe how it kills the app.

Add native notifications

11. Let's add native notification for successful signup. Open file signup.js under assets/js folder. Add reference to path module as top line.

```
const path = require('path');
```

12. Then place below line of code in init function.

```
const notification = {

        title: 'Signup Alert',

        body: 'Welcome to Dev6',

        icon: path.join(__dirname, '../images/TShirt.png')
```

```
        }
```

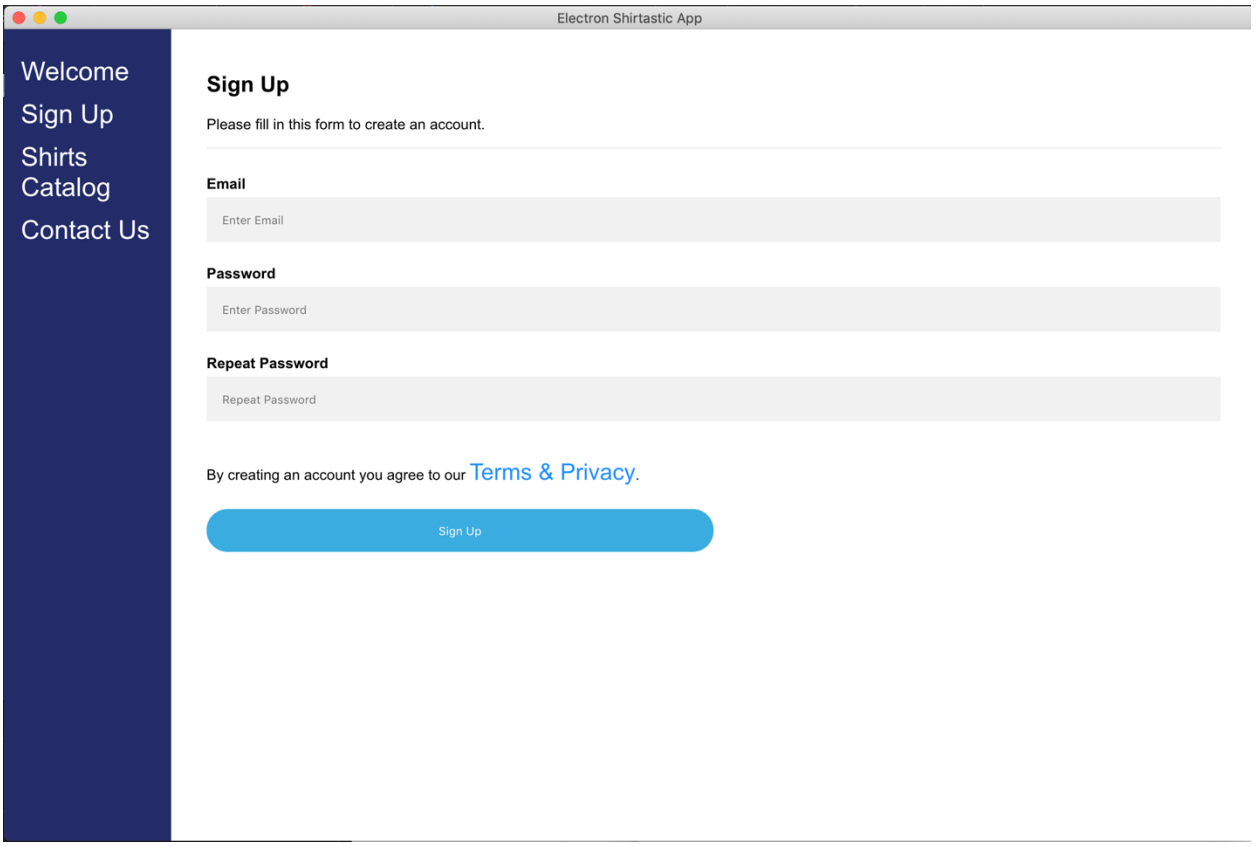
13. Last step is to display notification. This is done by placing below line of code inside success handler of axios call. Please refer to screenshot below to know the exact location for placement of this line of code.

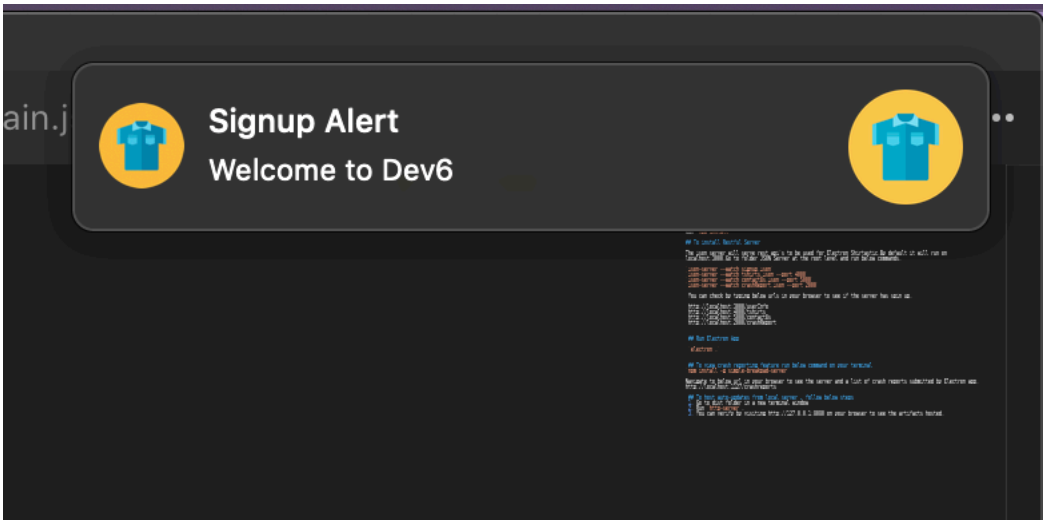
```
const myNotification = new window.Notification(notification.title,
notification);
```

14. Restart the app.

```
npm run start
```

Note: Now as soon you perform the signup in Signup screen, you will see a native alert saying Signup Alert. Please refer below screenshots to know which screen to test and the look of native notification on macOS.





Add new browser window

15. So, in next series of steps, we will add functionality of opening a new browser window for Contact Us link. In order to do so, lets create a new js file under assets/js folder named as new-browser-window.js. Place below line of code in this file. The main hero in this entire set of code is electron.remote.BrowserWindow which is providing the capacity

to open a new browser window for us. You can read more about this by visiting this URL https://electronjs.org/docs/api/browser-window.

```
const path = require('path');

const electron = require('electron');

const BrowserWindow = electron.remote.BrowserWindow;

const customerFeedbackBtn = document.getElementById('customer-feedback');

customerFeedbackBtn.addEventListener('click', function (event) {

    const modalPath = path.join('file://', __dirname, '../screens/customer-feedback.html');

    let win = new BrowserWindow({ width: 800, height: 700, frame: true });

    // // Open the DevTools.

    win.webContents.openDevTools()

    win.loadURL(modalPath);

    win.show();

});
```

Note: Here in above line of code, we created a new browser window on the click on ContactUs link and displayed the content of customer-feedback.html.

Perform File Read Operation and IPC communication from main process to renderer process

16. In order to perform file operations, we will require the help of ipcMain and ipcRenderer modules for communications between main process and renderer process. Let's start with the set up. Open file customer-feedback.js under assets/js folder. Paste below lines of code.

```
selectDirBtn.addEventListener('click', (event) => {

    event.preventDefault();

  ipcRenderer.send('open-file-dialog')

})



ipcRenderer.on('selected-directory', (event, path) => {

  document.getElementById('selected-file').innerHTML = `You selected:
${path}`

})
```

Note: The renderer process is all set to communicate and send messages to main process.

---

17. Now, open main.js (which is present at the root level). And paste below lines of code to send and receive messages from customer-feedback.js (renderer process). You can paste these lines anywhere in main.js file.

```
const {ipcMain, dialog} = require('electron')

ipcMain.on('open-file-dialog', (event) => {

  dialog.showOpenDialog({

    properties: ['openFile', 'openDirectory']

  }, (files) => {

    if (files) {

      event.sender.send('selected-directory', files)

    }
```

```
  })

})
```

Note: Below are the screenshots for entire code of customer-feedback.js and main.js after performing above steps. Please go through these screens and placement of code, in case any of the instructions is not clear.

Customer-feedback.js

```
const axios =require('axios');

const {ipcRenderer} = require('electron');

const selectDirBtn = document.getElementById('attach-file');

selectDirBtn.addEventListener('click', (event) => {
    event.preventDefault();
  ipcRenderer.send('open-file-dialog')
})

ipcRenderer.on('selected-directory', (event, path) => {
  document.getElementById('selected-file').innerHTML = `You selected: ${path}`
})

submitBtn.addEventListener('click', (event) => {
  event.preventDefault();
  let formData = new FormData(document.querySelector('form'));
  let userInfo = {
    firstname: formData.get('firstname'),
    lastname: formData.get('lastname'),
    attachedFile: formData.get('selected-file'),
    subject: formData.get('subject')
  };
  axios({
    method: 'post',
    url: 'http://localhost:5000/contactUs',
    data: userInfo,
    })
    .then(function (response) {
        //handle success
        console.log(response);
        alert('Thanks for your feedback.We will get in touch soon...');

    })
    .catch(function (response) {
        //handle error
        console.log(response);
    });
})
```

Main.js

```
const { app, BrowserWindow, Menu } = require('electron')
const log = require('electron-log');
const isDev = require('electron-is-dev');
const path = require('path')
const shell = require('electron').shell
const {ipcMain, dialog} = require('electron')

ipcMain.on('open-file-dialog', (event) => {
  dialog.showOpenDialog({
    properties: ['openFile', 'openDirectory']
  }, (files) => {
    if (files) {
      event.sender.send('selected-directory', files)
    }
  })
})
```

Note: You can read more about IPC by visiting this URL https://electronjs.org/docs/api/ipc-main and https://electronjs.org/docs/api/ipc-renderer

---

HTTP using axios library

18. For this part of lab, we will just observe how axios is being used in making http get and http post in our electron app.
19. For HTTP Post, please refer to the code used in signup.js under assets/js folder.
20. For HTTP Get, please refer to the code used in catalog.js under assets/js folder.