ENVISION

This README contains information & set-up instructions for graders under the section Application. The rest of the README is intended for groupmembers to set-up the development workspace.

Table Of Contents

Application

This section of the `README` is specifically intended for graders.

Envision is a visualisation tool for e-mail networks. It is primarily focussed on the Enron dataset, but allows for other similarly formatted datasets to be uploaded.

Using the dataset it can show two different types of visualisations: Disjoint Force-Directed Graphs and Hierarchical Edge Bundling.

The application is available at envision.riswick.net, but can also be manually built using this source code.

To build the application from source, NodeJS (and the included package manager `npm`) need to be installed. The project has been tested to work with the latest LTS version of NodeJS 14.x.x and npm 7.x.x (other versions will probably also work). To then install all project dependencies, navigate to the project directory in a terminal and run `npm install`. This will install all dependencies.

After installing dependencies, all commands listed under Starting the application are available. The primary ones that can be used are `npm run build` to build the application. All files will then be available in the `build` directory. To then start a local web server, `npm run serve` can be used, which will make this directory accessible on localhost:5000.

`npm start` can be used to start the application in development mode, which will make it available at localhost:3000. However, this method will have a performance impact and is therefore not the preferred method of viewing the application.

Very rarely, an issue occurs during installation where npm does not install all dependencies, and on start-up issues occur with missing modules. This can likely be fixed using the steps defined here: Missing Modules during start.

Set-up

This section of the `README` indicates how to set-up a development environment. It is not intended for graders.

Installation

Make sure you have the following installed on your system:

- Git
- NodeJS
- Visual Studio Code

Git Installation Options

If you install Git for Windows using the installer, then you will get a few pages with options. For most, you can just use the default selected. For the page `Choosing the default editor used by Git`, by default Vim is selected. You can change this to Visual Studio Code (not Insiders version). The rest of the install can be done using the regular options.

If you already have Git installed, then you do not have to change any options.

NodeJS Installation Options

When choosing the NodeJS version, please pick the `LTS` version. The rest of the installation can be completed using all default options.

Visual Studio Code Options

After having installed Git, it is useful to change the command line that Visual Studio Code uses to `bash` which was installed along with Git.

To do this, open up Visual Studio Code. Then, by pressing `F1`, you can type in the following command: `Terminal: Select Default Profile`. You will then get a list to choose from, in which you can select `Git Bash`.

There are also some useful extensions listed in this file.

Git Options

Git requires some additional set-up to get started. Open Visual Studio Code and open a terminal (Ctrl+`). Now type the following commands, replacing the e-mail and username with the e-mail and username you use for GitHub.

```
git config --global user.name "John Doe"
git config --global user.email johndoe@example.com
```

Set-up your workspace

To set-up your workspace, you need to `clone` the git repository from GitHub to your local computer. To do this, open up VS Code. In the sidebar, click on the Source Control Tab (Ctrl+Shift+G). If you have installed Git correctly, you should be able to click on the `Clone Repository` button. Then, you will get the option `Clone from GitHub`, after which you will be prompted to login to GitHub in a web-browser.

After you have succesfully logged-in and are back in VS Code, you are asked to select a repository. If the repository is not listed by default, you can type `2IOA0-35/dbl-app` in the search box, after which it should appear.

You will now be prompted to select a folder where you want to store the repository. You can choose any folder that is convenient for you. When it starts cloning, you might be asked to log-in again.

After cloning has been completed, you will get a message in the bottom right to open the folder. If this doesn't appear, you can manually open the folder in `File > Open Folder`.

The repository should now appear in your VS Code installation.

Set-up the application

To set-up the application, you need to open up the terminal (Ctrl+`). If you have configured everything correctly, then you should see a terminal with the following text, in colors `green`, `purple`, `yellow` and `blue`.

```
[YourUserName]@[YourComputerName] MINGW64 [RepositoryFolder]/dbl-app
(main)
$
```

If you do not see a prompt formatted in this way, try restarting Visual Studio Code. If that did not work, go back to Visual Studio Code Options.

You can now type the following command: `npm install`.

If you get a message `bash: npm: command not found`, please try restarting VS Code. If that did not work, make sure you have installed NodeJS correctly.

The command will take a bit of time, but at the end you should get an output that looks something like this:

```
added 2003 packages from 774 contributors and audited 2006 packages in
82.547s

136 packages are looking for funding
  run `npm fund` for details

found 0 vulnerabilities
```

Starting the application

`npm start`

Runs the app in the development mode.
Open http://localhost:3000 to view it in the browser. The first time you run this command, you might get a Windows Firewall message. Please choose `allow`.

The page will reload if you make edits.
Any errors you might have made will appear in the terminal.

To exit and go back to the command line type `Ctrl+C`.

If you press `F5` inside of Visual Studio Code, it will automatically open the app.

If you get issues while compiling that you do not know, please look under Resolving Issues.

`npm run build`

Builds the app for production to the `build` folder.
It correctly bundles React in production mode and optimizes the build for the best performance.

The `build` folder will never be saved to GitHub. It will only be visible on your local system such that you can build as often as you want, without interfering with builds of others.

`npm run serve`

If you have built the app using `npm run build`, all files for the app are available in the build folder.
To open this on your browser you can run `npm run serve`.
Then by opening http://localhost:5000, you can see the app in production mode.

If you make changes to the code after running `npm run build`, these will not automatically show in this mode. You will need to rerun `npm run build`.

To exit and go back to the command line type `Ctrl+C`.

`npm run lint`

ESLint will be executed on all files to check for any warnings or errors. These will be shown in the terminal.

Git & Visual Studio Code

Git Basics

Git works on the basis of `commits`. You commit your changes to the github repository when you are done.

Other people will also have committed their work to the repository which you can then `pull` (download) to your local repository.

When you are done, you can `push` (upload) your changes back to the github repository.

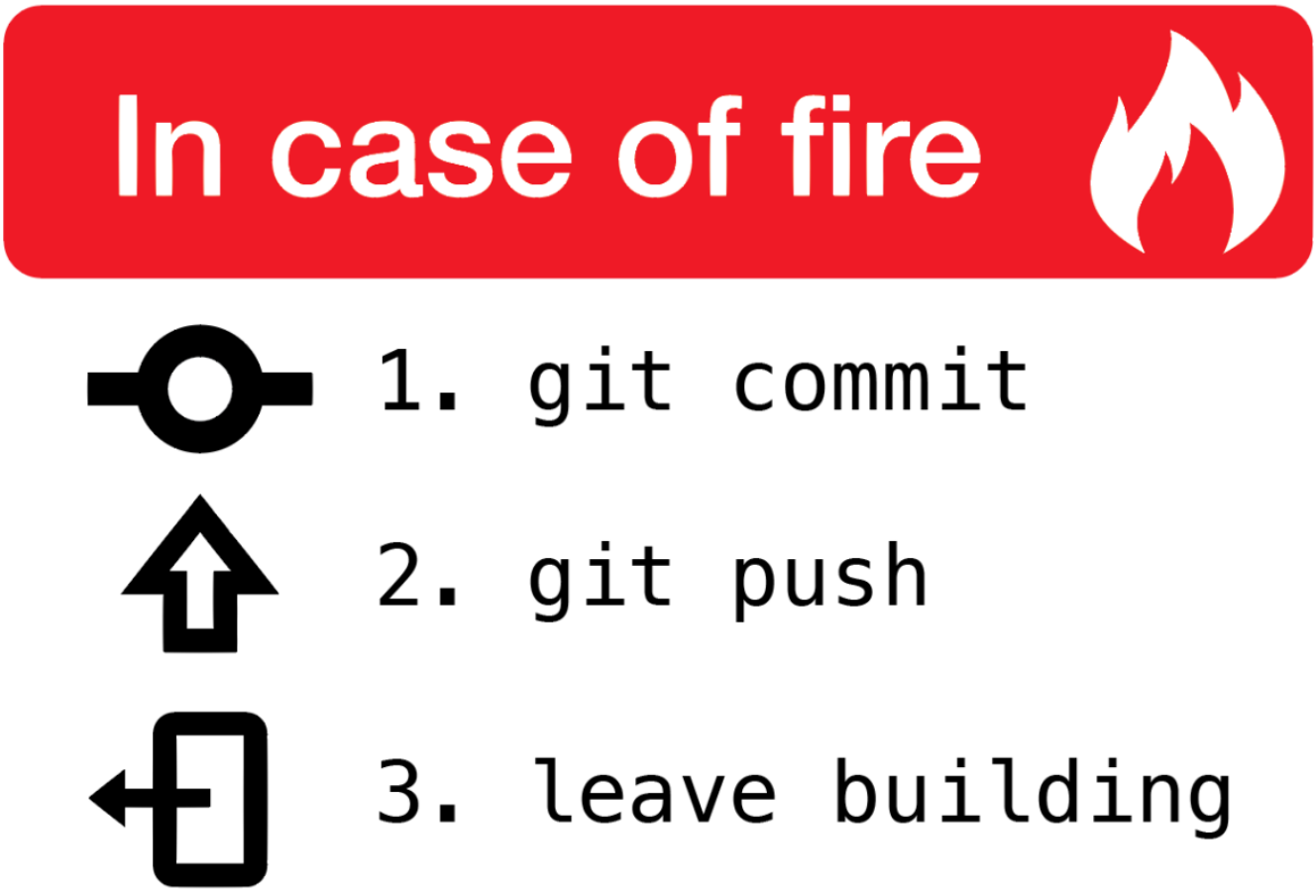
Sometimes, when two people have changed the same file, Git will not know what to do with the conflicts. Visual Studio Code will then ask you to `merge` the files manually.

Workflow

When you are programming in this repository, you should use the following workflow:

- Pulling commits from GitHub before you start.
- Programming! 😃
- Commit your changes.
- Push your changes to GitHub.

Never forget to commit and push after you are done!



Pulling & Pushing Commits

Visual Studio Code has an overview of all commits that need pulling and pushing in the bottom left corner. When there are commits to pull, next to the sync-wheel, a message will appear like: `1↓ 0↑`. This would mean that there is 1 commit to pull and nothing to push.

If you click the sync-wheel it will pull and push what is available.

Even when nothing is showing, it doesn't hurt to press the sync-wheel. Sometimes commits that need to be pulled do not show up, so always click the wheel before you start.

Committing your changes

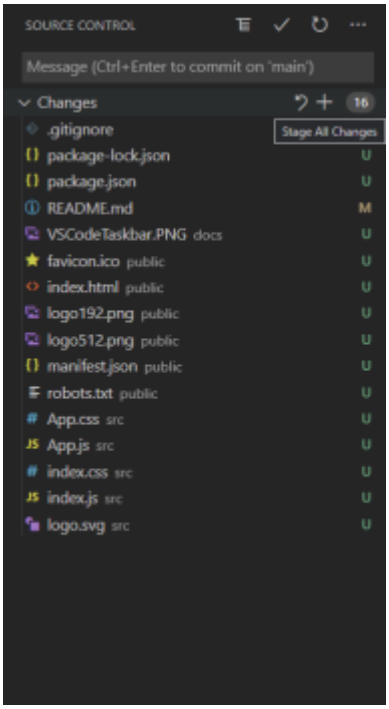
When you want to commit your changes, you can go to the `Source Control` tab in the sidebar (Ctrl+Shift+G).

Here you will see an overview of everything that you have modified. The letters have the following meaning:

- `U` or `A`: Created new file
- `M`: Modified file
- `D`: Deleted file

To commit your changes, you need to tell VS Code which files you want to include in your commit. This way you can decide that you only want to commit a part of your changes, if you are not 100% finished.

To add files to your commit, you can click the `+` next to the file. You can also add all files (which is what you want most often) by clicking the plus on the top (see picture). All of your files should now appear under `Staged Changes`.



The last step of your commit is to provide a `commit message`. This can be typed in the box at the top. The message should be a short description of what you have changed.

Visual Studio Code will complain when you make the first line of your commit message too long. In this case, you need to split it out to the next line.

If you are currently working on a GitHub issue, you can type a `#` + the issue number (so `#14` for issue 14) to automatically link it to the issue.

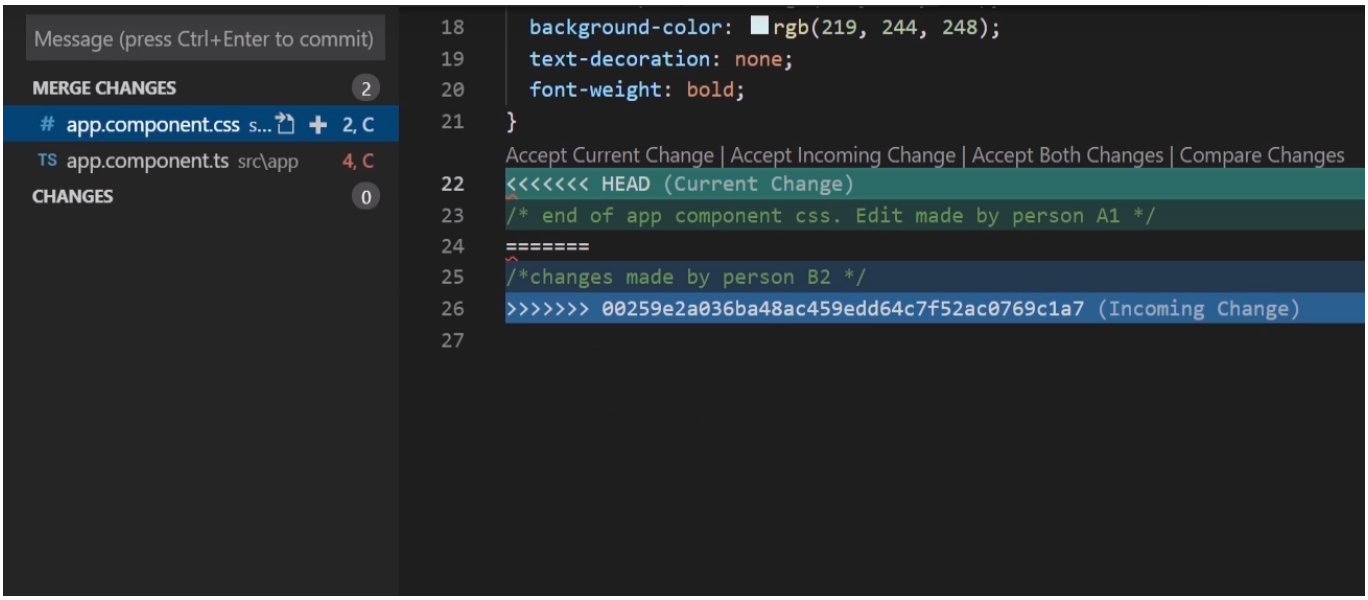
Finally, to save your commit, you click the checkmark, after which the list should clear. Then push your changes.

Resolving Merge Conflicts

When two people have worked on the same file at the same time, it can occur that Git does not know how to merge automatically. In that case, Visual Studio Code will prompt you that it cannot push your commit before you merge.

In the picture below you can see how this would look. You should merge all files in the commit list under the section `Merge Changes`. When you open these, you will see the problematic sections highlighted.

To resolve the conflict, VS Code gives you quick options to accept `current`, `incoming` or `both` changes. Please choose one of these options. After you choose your option, it is allowed to manually modify the file such that it makes sense again.



After you have resolved the conflict, you can press the `+`-sign again and provide a message, just like a commit.

Tools

Browser

It is strongly encouraged to use Google Chrome for developing a React app. There is a useful extension for Chrome called React Developer Tools. It adds 2 new tabs to the Chrome Developer Tools (press F12 on a webpage) where you can see useful information about React and the components.

Visual Studio Code

Here is a list of useful VS Code extensions for this project:

- Debugger for Chrome: Debugging React in the browser
- React Snippets: Useful snippets to quickly add often-used parts to a file.
- ESLint: Checks your code thoroughly for any mistakes.

Debugging

Using the debugger for chrome extension, it is possible to debug apps running in the browser. By pressing F5 within Visual Studio Code, a new chrome browser will launch with a debugger attached. You can then place breakpoints to debug.

Placing breakpoints can be done by clicking next to a line number in any javascript file. Chrome will pause once it hits this breakpoint. Then you can look at all variables that are defined in the debug pane (Ctrl+Shift+D) in VS Code. You also have control to step forward over the code line by line using the controls at the top of the VS Code window.

Linting

ESLint will make sure your code rougly follows a common style. Some common problems, like quotes, can be fixed using the light-bulb that appears in VS Code.

Resolving Issues

Missing Dependencies

When you get a message like: `Module not found: Can't resolve 'x' in ...`, it might be that new modules were added to the repository after you ran `npm install`. In this case, you can fix the problem by re-running `npm install`, which will install all missing modules.

If problems remain, you can force a re-install of all modules by removing the node_modules directory: `rm -r node_modules`. Then, you can once again run `npm install`.

Missing Modules during start

If you ran `npm install` and still have issues during start-up where some modules can't be found, you can try the following debugging step. Removing both the `node_modules` directory and `package-lock.json` file will trigger `npm` to start fresh. Triggering an install using `npm install` will then likely install all dependencies correctly.

If issues persist, a possible cause could be that npm is outdated. The project was tested with npm version 7 (the version can be checked using `npm -v`). To update npm run `npm install -g npm`, after which you can try the previous steps again.