

Bachelor of Computer and Information Sciences
Contemporary Issues in Software Engineering
Semester 2, 2024

ASSIGNMENT 1A: Set up Development Environment

Worksheet 1 (20% of Ass1A)

VS Code, Git, GitHub, Nest

Deliverables and Due dates:

You are required to complete the Worksheet and keep evidence as you do it by **selectively** taking screenshots of your work, as well as explanations.

Each worksheet should ideally be checked off by the TA by the end of the week's tutorial

This worksheet should be Checked off and Uploaded on CANVAS ideally by end of Tutorial Week 1 – all four worksheets for Assignment 1a are due by week 6, and the knowledge develops cumulatively so don't leave it to the end – that will also make it hard for the TA's to mark and give you feedback.

EXPERIMENT – BE CURIOUS – TEACH OTHERS – TAKE SELECTED SCREENSHOTS FOR KEY ASPECTS

The worksheets will have some theory, a practical exercise, and a worksheet for answers to questions and at least three selectively captured screenshots as evidence. The aim is to be able to learn from the exercise, and evidence that. For each of your three selected screenshots (or sequence of shots) in a brief paragraph or two reflect on why you have selected it. What have you learnt in this part of the worksheet? What was new or surprising? What useful external resource(s) did you consult and why? Provide a link(s) to the resource.

By the end of this Worksheet you should be able to....

1. Create branches and work with them appropriately in GitHub and Git
 2. Keep a local repo synchronised with a GitHub repo using pull and push commands
 3. Write useful commit messages
 4. Use the pull request feature before merging code to Master branch in GitHub
 5. Merge pull requests
 6. Work with Git and GitHub from VS Code locally (make sure you know what is happening in Git!)
- (Use the “GitHub Pull Requests and Issues extension)

<https://code.visualstudio.com/docs/editor/github>

7. Understand what NEST is and how to use the Nest CLI to create a project.

Introduction

This worksheet introduces you to:

The toolset you will need for the team project to enable you to: 1) develop and share code within the team, 2) apply practices to manage local versions and migrate changes to the team repository, 3) ensure the quality of the code developed and released through the workflow to be established, 4) become exposed to the tech stack for the project.

Git, a version control system (VCS) to manage code changes locally for each developer,

GitHub, a cloud-based version control system (VCS) to manage Git code repositories and share code

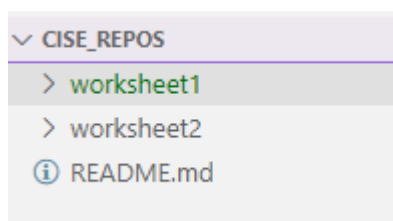
Visual Studio Code (VS Code), an IDE and text editor with some intelligent features and plug-ins for coding and integrating with other tools.

These tools are used to write, test, integrate and share code so developers can collaborate.

Exercise using VS Code, Git, GitHub & Nest

Do the following exercise:

You will need to create folders to store the repository content for each week's worksheet. Previous examples of the typical folder structure that you will have for the worksheets is given below, with the top-level folder being called CISE_Repos:



1. Create a Github account if you have not already (note your token somewhere safe)
2. Create a new GitHub repository called CISE_Repos to hold the worksheets, [and note that a later team repository may be named as a [Main Repository]--- CISE + Team Number + ProjectName]
3. Install Git on your local machine (<https://github.com/git-guides/install-git>)
4. Create a folder for your projects called “CISE_Repos” on your local machine
5. Install VS Code on your local machine <https://code.visualstudio.com/>
6. Install the “GitHub pull requests and Issues” extension for VS Code and authorise if necessary
7. Clone a copy of this empty repo to your local machine in the “CISE_Repos” folder using the following:

From a terminal window in VS Code, at the CLI type:

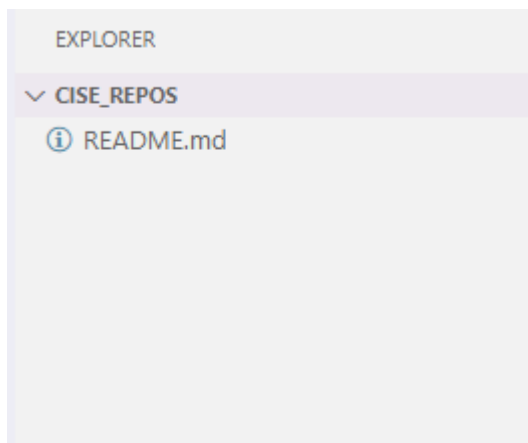
```
>git clone https://github.com/<Your GitHub username>/CISE_Repos
```

```
>cd CISE_Repos to change the working folder  
>git status to check git is working.
```

8. Create a README.md file and push it to the main branch in GitHub

```
>echo "# CISE_Repos" >> README.md creates a README.md file with "CISE_Repos" as the title.  
>git add README.md stages the README.md  
>git commit -m "first commit" saves a snapshot of the changes to README.md with commit message  
>git push -u origin main pushes the latest changes to GitHub remote main branch from the local main  
branch (You should see your README.md in GitHub now).
```

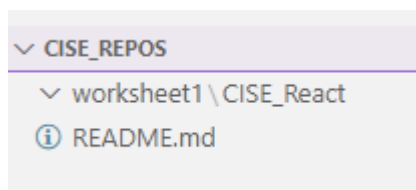
Your folder structure should now look like this:



9. Create locally a “Development” branch off the Master/Main branch and change to working in that branch

```
>git checkout -b Development This creates a new branch and moves you to the branch  
>git branch -a to check what branches you have in your local repo type – green is the current one
```

10. Use VS Code to create a new folder called “worksheet1” (or any name you like) in the CISE_Repos folder for worksheet1’s content. Then create a folder “CISE_React” in folder worksheet1. Stage and commit it, with a useful commit message.



Create a new file called Delete_me.html in the CISE_Repos/worksheet1/CISE_React folder and add the following code to the file Delete_me.html:

```
<!DOCTYPE html>
```

```
<html>

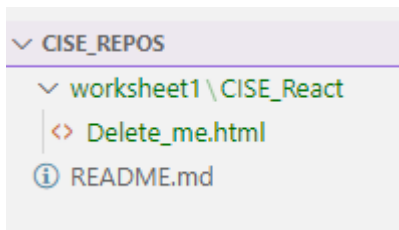
<body>

<h1>My First Heading</h1>

<p>My first paragraph.</p>

</body>

</html>
```



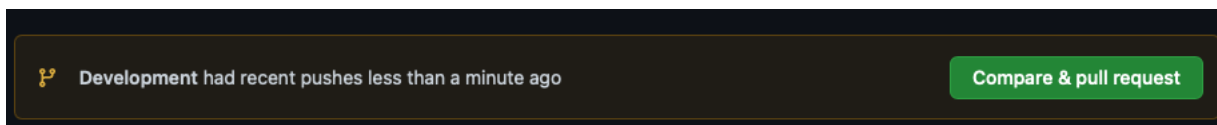
Stage and commit this file, with a useful commit message.

Note: Make sure to write useful commit messages (THIS IS VERY IMPORTANT – it documents the reasons for changing and what was changed). Read the following articles to understand:

<https://betterprogramming.pub/why-every-git-commit-message-must-include-its-commit-context-1171c0b2f710>

<https://dev.to/helderburato/patterns-for-writing-better-git-commit-messages-4ba0>

11. `>git push -u origin Development` to push your changes to GitHub
12. Check `Delete_me.html` has been sent to GitHub in the Development branch in GitHub. You should see a message in GitHub like the following:



13. Press the green button and continue accepting until you have merged this change with the main branch on GitHub. Usually the main branch is the production branch to be deployed to users.
It is common to create a branch structure with feature branches coming off the Development branch like this (Fig 1)



Fig 1

14. We are now going to create a simple React app using a script called “create-react-app”. In order to use the script and some other features we need to install node.js, which is a javascript run-time that allows us to run javascript outside the browser (thank you Google!)
15. Follow the instructions to install node.js on your Operating System
<https://nodejs.org/en/download/>
16. To check it has installed and the version you have, at the command prompt you can type

```
>node -v
```

17. To install all the dependencies for a React application we can install the “create-react-app”. This will create a new folder called cise-react-learn for this app. First navigate to worksheet1 folder in your terminal, then use the commands:

```
>npx create-react-app cise-react-learn
```

```
>cd cise-react-learn to change to the new project directory
```

```
>npm start
```

will start the development web server and open a browser to show you the simple app it created with a spinning logo.

README.md is a markdown file that includes a lot of helpful tips and links that can help you while learning to use Create React App.

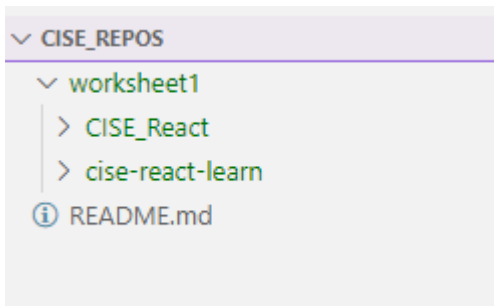
node_modules is a folder that includes all of the dependency-related code that Create React App has installed. You will never need to go into this folder.

package.json that manages our app dependencies and what is included in our node_modules folder for our project, plus the scripts we need to run our app.

.gitignore is a file that is used to exclude files and folders from being tracked by Git. We don't want to include large folders such as the node_modules folder

public is a folder that we can use to store our static assets, such as images, svgs, and fonts for our React app.

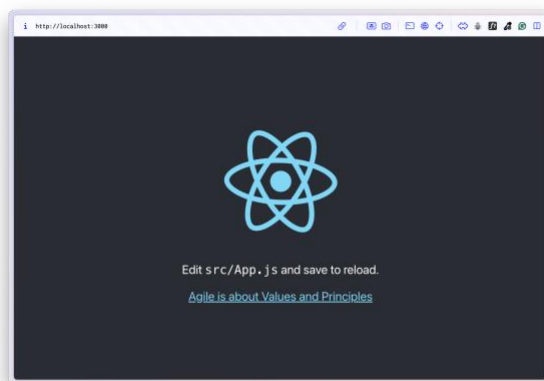
src is a folder that contains our source code. It is where all of our React-related code will live and is what we will primarily work in to build our app.



18.

- >git add . (note the dot) to get Git to stage all files in the cise-react-learn folder.
- >git commit -m "first commit" to commit all the files created
- >git push origin Development to push all the files created to the Development branch of GitHub

19. Make a change to App.js file in the src folder – change “Learn React” to “Agile is about Values and Principles” and save the file. Notice the filename is yellow in VS Code – indicating it is not committed.



Note that your app should have updated in the web browser automatically.


20. Commit this change and push it to GitHub. (Remember the good quality commit message discussed in step 10!)

```

cise-react-learn git:(development) ✗ git add .
cise-react-learn git:(development) ✗ git commit -m 'Update `App.js` link text and include babel dependency in `package.json`'
[development 84cb0fe] Update `App.js` link text and include babel dependency in `package.json`
 2 files changed, 10 insertions(+), 12 deletions(-)
cise-react-learn git:(development) git push origin development
Enumerating objects: 13, done.
Counting objects: 100% (13/13), done.
Delta compression using up to 12 threads
Compressing objects: 100% (7/7), done.
Writing objects: 100% (7/7), 962 bytes | 962.00 KiB/s, done.
Total 7 (delta 3), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (3/3), completed with 3 local objects.
To https://github.com/jackdar/cise_repos.git
 51ce61b..84cb0fe  development -> development
cise-react-learn git:(development)

```

21. (Check that the files were updated in GitHub)

 **development** had recent pushes 42 seconds ago

[Compare & pull request](#)

Name	Last commit message	Last commit date
..		
public	first commit	7 minutes ago
src	Update App.js link text and inclu...	1 minute ago
.gitignore	first commit	7 minutes ago
README.md	first commit	7 minutes ago
package-lock.json	first commit	7 minutes ago
package.json	Update App.js link text and inclu...	1 minute ago

(Ctrl-c in terminal windows will stop the program running)

Collaborate on a Repository with a team

22. In Git create another branch “LogoLink” from the Development branch and move to it
23. Change the App.js file in VSCode so clicking the link takes you to aut.ac.nz instead of reactjs.org
24. Commit and push to GitHub creating the new “LogoLink” branch
25. Invite a classmate or friend who has a GitHub account (or will make one) to be a collaborator with your repository. (or make another GitHub account using a different machine).

“Settings>Manage Access” in GitHub. you will need to know your collaborator’s GitHub account name

26. **Create a pull request** for this merge of the “LogoLink” branch with the Development branch, so the changes are in this branch. Other developers working on the same repository would clone this branch to get the latest code to work on several times per day. You can create the pull request with Git command or use the extension you installed into VS Code (see Fig 3).
27. Go to GitHub on your browser and you should see one open pull request. The Reviewer assigned to the pull request would normally review this code to be merged to check for errors etc and there may be a discussion with the original author of the code to fix something or get clarification. **Get your reviewer to open the pull request and merge this change with the Develop branch**
28. Install the Prettier plug-in to VS Code and see how it works. Read about the purpose of a Linter – we will use ESLint. Try installing the ESLint plug-in for VS Code.

Work with NEST

Step 28: Install NEST CLI on your machine. Open a terminal and use the following command to install the CLI globally: <https://docs.nestjs.com/cli/overview>

```
>npm i -g @nestjs/cli
```

Step 29: Verify the installation of NEST CLI by checking its version. Type the following command:

```
>nest --version
```

(Note you may not have permissions on your system, if you are getting errors run this command for windows OS)

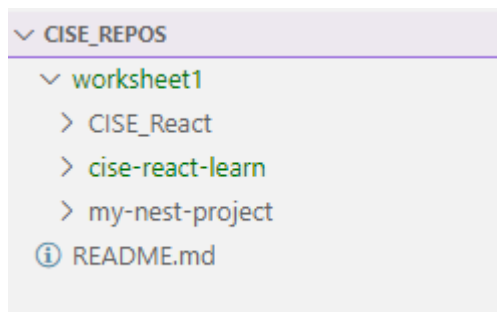
```
> Set-ExecutionPolicy -Scope CurrentUser -ExecutionPolicy RemoteSigned
```

Step 30: Create a new project using NEST CLI. In your terminal, navigate to the directory CISE_Repos/worksheet1 and then create the project and then run the following command:

```
>nest new my-nest-project --skip-git
```

Replace "my-nest-project" with the name you want for your project. NEST CLI will create a new directory with this name and set up a new NEST project inside it (Select npm as the package manager).

Your folder structure at this point should look like this:



Step 31: Navigate into your new project directory using the command:

```
>cd my-nest-project
```

Step 32: Run the NEST application. Inside the project directory, type the following command:

```
>npm run start
```

```
my-nest-project git:(development) ✕ npm run start

> my-nest-project@0.0.1 start
> nest start

[Nest] 87919 - 07/17/2024, 11:57:49 AM LOG [NestFactory] Starting Nest application...
[Nest] 87919 - 07/17/2024, 11:57:49 AM LOG [InstanceLoader] AppModule dependencies initialized +6ms
[Nest] 87919 - 07/17/2024, 11:57:49 AM LOG [RoutesResolver] AppController {/}: +2ms
[Nest] 87919 - 07/17/2024, 11:57:49 AM LOG [RouterExplorer] Mapped {/, GET} route +1ms
[Nest] 87919 - 07/17/2024, 11:57:49 AM LOG [NestApplication] Nest application successfully started +0ms
```


Your NEST application is now running at ``http://localhost:3000``. You can open this URL in your browser to see your application.

Hello World!

Step 33: Make some changes to the application. Open the project in your VS Code, and navigate to ``src/app.controller.ts``. Here, you can change the string returned by the ``getHello()`` function. Save your changes and check your application at ``http://localhost:3000`` again to see the updated message.

Welcome to this new NestJS project!

Step 34: Once you have made your changes, stop the NEST application by pressing ``Ctrl + C`` in your terminal.

Now you can commit everything to this week's repository.

Name: Jack Darlington

Date: 17/07/24

Worksheet Evidence:

This worksheet requires some answers to questions and **at least three selectively captured screenshots** as evidence. The aim is to be able to learn from the exercise, and evidence that.

For each of your three selected screenshots (or sequence of shots) in a brief paragraph or two reflect on:

- Why you have selected it?
- What have you learnt in this part of the worksheet?
- What was new or surprising?
- What useful external resource(s) did you consult and why? Provide a link(s) to the resource.

Evidence	Check
<p>1. What is the purpose of Git and GitHub?</p> <p>Git is a version control system used for source code management. It is free and open source, and widely used in the software engineering industry. One of its biggest advantages over other version control systems is its branching capabilities. Branches are easy to merge and provide isolated environment for each feature or change in a codebase.</p>	
<p>2. Explain the difference between a local repository and a remote repository in the context of Git and GitHub.</p> <p>Git on its own can be run anywhere. The differences between local and remote repositories are the location of the repository. The remote repository is a repository that is hosted in a remote location where it is accessible to multiple users. The local repository is hosted on a local machine where it is accessible to a single user. Git includes technologies to be able to push and pull files to and from a remote repository to keep them both in sync and up to date.</p> <p>GitHub is a Git hosting service that allows users and teams to host their remote repositories online, so they are accessible to users that are allowed access. In this context GitHub hosts the remote repo, and the local repo would be the repo cloned from GitHub which is downloaded and is stored locally on a single user's computer where they can make changes and then push those changes to the remote repository hosted on GitHub.</p>	
<p>3. What is the role of README.md file in a GitHub repository?</p> <p>The README.md is a Markdown formatted text document that usually resides in the root of the repository directory which describes the repo and gives some basic instructions or information. In a repository hosted on GitHub the README.md file is usually rendered as the cover page of a repository and serves as the main page when a user lands on your repository on GitHub.</p>	

4. Explain the purpose of creating branches in GitHub. A screenshot(s) to support your answer may be suitable to show your “Development” branch.

The purpose of creating branches in Git and GitHub is to allow working on features and fixes in isolation of the main codebase. It also allows multiple developers to work on different branches simultaneously without interfering in other work. It also means on GitHub, that developers can create pull requests to merge new code into the main branch, it allows for code review, testing, and QA before changes are integrated.

```
* development
  master
  remotes/origin/development
  remotes/origin/master
( END )
```

5. Explain the steps you took to merge your changes to the main branch on GitHub. A screenshot(s) to support your answer may be suitable.

Merging changes into the main branch requires creating file changes on a different branch in the repo (in this case the development branch). When these changes are committed to the development branch and pushed to GitHub you can then compare branches and the changes between the branches.

In this case, we have changes that can be compared between master and development branches. On GitHub you can now open a pull request for the changes to be merged into the compared branch.

Comparing changes

Choose two branches to see what's changed or to start a new pull request. If you need to, you can also [compare across forks](#) or [learn more about diff comparisons](#).

base: master
compare: development
✓ Able to merge. These branches can be automatically merged.

Discuss and review the changes in this comparison with others. [Learn about pull requests](#)
Create pull request

6. Provide an example of one of your commit messages, adhering to a commit message standard. A screenshot here may be suitable.

2 commits
18 files changed
1 contributor

Commits on Jul 17, 2024

first commit
 51ce61b

jackdar committed 5 days ago

Update App.js link text and include babel dependency in package.json
 84cb0fe

jackdar committed 5 days ago

After my first commit in my development branch, my next commit message follows the standard of using a commit adjective in the present tense as the first word, then stating which files and

code was subject to changes. In this case App.js link text has been updated along with the package.json file.

7. What is the "create-react-app" script used for? A screenshot(s) to support your answer may be suitable to evidence your successful creation of the cise-react-learn React application.

The purpose of the create-react-app script is to be an easy way to set up a new React application. It includes all the boilerplate code to set up a new single page React application with a standard project structure and configuration. This means you can start programming right away without having to set up build tools.

A new way of doing this is using Vite, which works in a similar way but allows for more project customisation, can be used with different technologies and allows for starting projects with different stacks and dependencies. It is generally faster too.

You can see when the project is created, the script automatically downloads all 1482 required packages and template dependencies using the node package manager.

```

● → worksheet1 git:(development) ✖ npx create-react-app cise-react-learn-

Creating a new React app in /Users/jackdarlington/Developer/cise_repos/worksheet1/cise-react-learn-
Installing packages. This might take a couple of minutes.
Installing react, react-dom, and react-scripts with cra-template...

added 1482 packages in 49s
261 packages are looking for funding
  run `npm fund` for details
Installing template dependencies using npm...
added 64 packages, and changed 1 package in 3s
261 packages are looking for funding
  run `npm fund` for details
Removing template package using npm...

removed 1 package, and audited 1546 packages in 2s

Success! Created cise-react-learn- at /Users/jackdarlington/Developer/cise_repos/worksheet1/cise-react-learn-
Inside that directory, you can run several commands:

  npm start
    Starts the development server.

  npm run build
    Bundles the app into static files for production.

  npm test
    Starts the test runner.

  npm run eject
    Removes this tool and copies build dependencies, configuration files
    and scripts into the app directory. If you do this, you can't go back!

We suggest that you begin by typing:

  cd cise-react-learn-
  npm start

Happy hacking!

```

<p>8. What are the roles of the package.json and .gitignore files in a React application?</p> <p>The package.json file is a file that is included in Node.js projects and JavaScript applications including React projects. It includes details about the project, such as the name and version. It also includes a list of dependencies and packages that are required for the project. It means that when inside a project directory the command `npm install` can be used to install all required dependencies listed in the package.json file. It also includes scripts to start and build the project, such as start, test, build, etc. This is different for different projects – Vite projects use the scripts run dev, and run build.</p> <p>A .gitignore file is an optional file in a Git repository. It tells Git which files to ignore and not include in the repository. The files on a .gitignore file are untracked, this is useful for excluding files that are not necessarily required to be tracked, such as builds, temporary files, and sensitive data such as database or API keys.</p> <p>For example, in Node.js projects, the provided .gitignore file ignores the `node_modules` directory by default, which contains project dependencies. It is redundant to include the project dependencies in the Git repository as they are all freely hosted on npm (node package manager). So from only the `package.json` file and running the command `npm install` will all required packages be downloaded from npm and installed into the `node_modules` directory. It also ignores all build files in directories such as dist, build, etc.</p>	
<p>9. Explain the purpose of a pull request in GitHub. A screenshot(s) to support your answer may be suitable, to evidence the open pull request for merging the "LogoLink" branch with the Development branch.</p> <p>A “pull request” is a GitHub specific feature for merging branches with changes from one another. The ability to open a pull request is shown when GitHub detects that one or more branches have changes from the master branch. You can compare the changes and decide whether to open a pull request to merge the changes into the compared branch.</p> <p>Once a pull request has been opened, usually other contributors to the repository can review the branch submitted to be merged and decide whether to merge the changes into the compared branch. If there are conflicts, GitHub will not allow the changes to be merged until all merge conflicts are fixed. Once there are no merge conflicts the changes can be merged into the compared branch.</p> <p>Pull requests are a feature that allow collaboration and code review by peers before being merged into other branches.</p>	
<p>10. A screenshot(s) to support your answer may be suitable to evidence the successful creation of a new project using NEST CLI and running of the NEST application.</p>	

```
✓ Installation in progress... 📦

🚀 Successfully created project my-nest-project
👉 Get started with the following commands:

$ cd my-nest-project
$ npm run start

      Thanks for installing Nest 🙏
    Please consider donating to our open collective
    to help us maintain this package.

👉 Donate: https://opencollective.com/nest

→ worksheet1 git:(development) ✖ cd my-nest-project
→ my-nest-project git:(main) ✖ npm run start

> my-nest-project@0.0.1 start
> nest start

[Nest] 10631 - 07/22/2024, 5:32:11 PM LOG [NestFactory] Starting Nest application...
[Nest] 10631 - 07/22/2024, 5:32:11 PM LOG [InstanceLoader] AppModule dependencies initiali
[Nest] 10631 - 07/22/2024, 5:32:11 PM LOG [RoutesResolver] AppController {/}: +1ms
[Nest] 10631 - 07/22/2024, 5:32:11 PM LOG [RouterExplorer] Mapped {/, GET} route +1ms
[Nest] 10631 - 07/22/2024, 5:32:11 PM LOG [NestApplication] Nest application successfully
S
```

11. A screenshot(s) to support your answer may be suitable to evidence changes you made to the `getHello()` function in the `src/app.controller.ts` file of the NEST application and the updated message in the browser.

```
app.controller.ts X
worksheet1 > my-nest-project > src > app.controller.ts > ...
1 import { Controller, Get } from '@nestjs/common';
2 import { AppService } from './app.service';
3
4 @Controller()
5 export class AppController {
6   constructor(private readonly appService: AppService) {}
7
8   @Get()
9   getHello(): string {
10     return 'Hello this is a new Nest application!';
11   }
12 }
13
```

i http://localhost:3000

Hello this is a new Nest application!