**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>

1. 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:

A screenshot of a computer

Description automatically generated

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.

1. 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:

A screenshot of a computer

Description automatically generated

1. 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

1. 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.

A screenshot of a computer

Description automatically generated

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>

A screenshot of a computer

Description automatically generated

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>

1. >git push -u origin Development to push your changes to GitHub
2. 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:



1. 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)

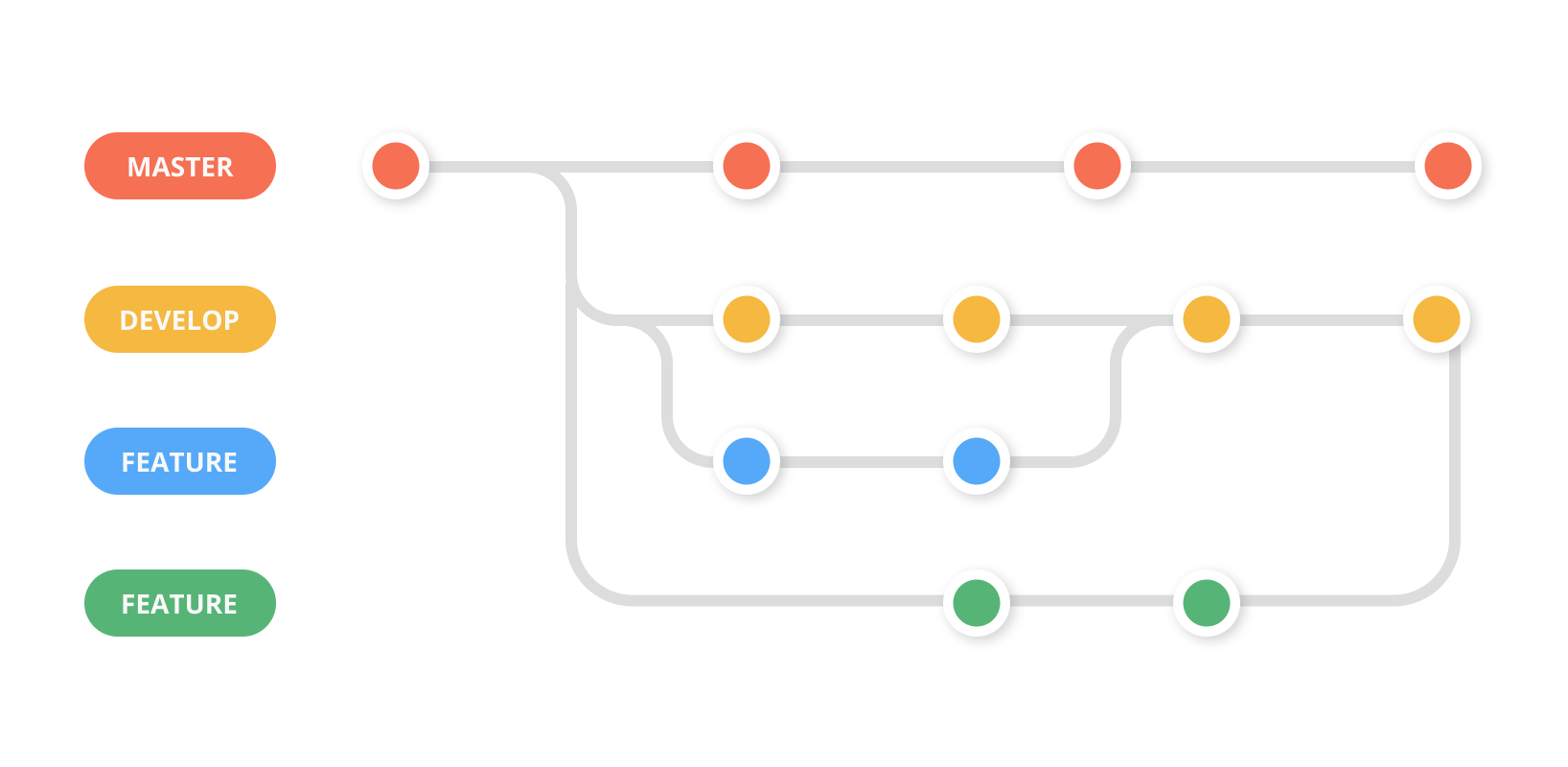


Fig 1

1. 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!)
2. Follow the instructions to install node.js on your Operating System

<https://nodejs.org/en/download/>

1. To check it has installed and the version you have, at the command prompt you can type

>node -v

1. 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.

A screenshot of a computer

Description automatically generated

>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

1. 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.

1. Commit this change and push it to GitHub. (Remember the good quality commit message discussed in step 10!)
2. (Check that the files were updated in GitHub)

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

**Collaborate on a Repository with a team**

1. In Git create another branch “LogoLink” from the Development branch and move to it
2. Change the App.js file in VSCode so clicking the link takes you to aut.ac.nz instead of reactjs.org
3. Commit and push to GitHub creating the new “LogoLink” branch
4. 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

1. **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).
2. 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**
3. 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:**

A screenshot of a computer

Description automatically generated

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

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

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.

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: Date:**

**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** |

|  |  |
| --- | --- |
| 1. What is the purpose of Git and GitHub?   Git is a piece of software where users can download onto their machines which enables Git Versioning. GitHub is an online website where users can upload their local Git Repos to it and create an online repo. Using both these tools changes can be made locally then pushed to the remote repo allowing easier of tracking of changes and collaboration between users. |  |
| 1. Explain the difference between a local repository and a remote repository in the context of Git and GitHub.   Using Git you would clone a remote repo which is hosted on Github. A local repo is then created on your machine where changes are tracked. Once changes have been made you can commit and push the changes back to the remote repo allowing others to see the changes that were made.  Screenshots below showing cloning of the online repo into a local repo, then adding a README.md file to it and pushing it back to github. Initially following the push command it came back with an error turns out it was cause the main branch was using master and not main. After I changed it, the readme had successfully pushed to github as shown in the second screenshot.  A screenshot of a computer  Description automatically generated |  |
| 1. What is the role of README.md file in a GitHub repository?   A README.md file gives people who visit the repository the context of what the project is about like what technology is used to create it, what problems it solves, and what features it offers. The readme may also contain instructions on how to run the application if applicable.  Screenshot below shows the basic readme that was created as a part of this worksheet.A white rectangular object with black lines  Description automatically generated  This second screenshot shows another repo of mine which gives some context to the project and how to run it locally. |  |
| 1. Explain the purpose of creating branches in GitHub. A screenshot(s) to support your answer may be suitable to show your “Development” branch.   Branches in GitHub is an important feature which allows developers to create new branches of pre-existing ones in order to make changes/develop new features without it affecting the made branch. After changes are made a pull request where it would then be merged into the other branch if no conflicts exists.  The screenshot below shows the different branches that were created from this worksheet. Each branch contains their own special features which were created then would be merged into the dev branch before the master branch.  A screenshot of a computer  Description automatically generated |  |
| 1. 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.   After pushing the changes to the new branch, I was able to create a pull request where GitHub would check for any conflicts, after no conflicts were found I merged it from the development branch to the main branch.A screenshot of a computer  Description automatically generated  Instead of accepting the pull request by myself, creating a rule which requires a code review can also be done. In this way if a pull request were to be created to a protected branch then another contributor would have to review the changes made ensuring that nothing is broken and it doesn’t introduce any bugs. After the code review the merge can be processed and changes will be merged into the protected branch.  This second screen shot shows my other account “reviewing” the code after being assigned to it and confirming the merge. This ensures that two developers see the code and review it before merging.  A screenshot of a computer  Description automatically generated |  |
| 1. Provide an example of one of your commit messages, adhering to a commit message standard. A screenshot here may be suitable.   A screenshot of a web page  Description automatically generated |  |
| 1. 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 create-react-app is used to create and install all the dependencies of a basic react app. Running this command allows for easy creation of a react app without the need to set up anything like package.json or install any dependencies individually.  This screenshot shows the command running and installing all the packages/dependencies automatically.    The screenshot below shows the react app running with the changes made to the text, changing it to “Agile is about values and principles”, the link has also been changed to link to AUT instead of the default react website.  A screenshot of a computer  Description automatically generated |  |
| 1. What are the roles of the package.json and .gitignore files in a React application?   The pack.json includes all the of the dependencies of the project, normal ones and dev ones. Normal ones are usually shipped with production to ensure that the Application functions when deployed whereas dev dependencies are only used in a dev environment. Dev dependencies may include stuff like Typescript which is only needed for the developer and is not needed in production. The package.json also includes scripts that can be ran like npm run dev, etc. A .gitignore file is important to not only exclude certain files from tracking like the node\_modules file where it wouldn’t make sense to upload to GitHub, but also not track any files which contain sensitive information like a .env file.A screenshot of a computer  Description automatically generated  This screenshot shows one of the importance of having a gitignore. Whilst this is for the nest project, not having a .gitignore proves to be difficult because the sheer amounts of files in node\_modules would flood the git changes and they are unnecessary. |  |
| 1. 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.   The purpose of a pull request is to allow for code reviews before merging with a stable branch like development or production, etc. Having an open code review allows for developers/testers to test the new features/fixes offered in the PR and provide feedback and recommend changes.  Screenshot below shows the other account as the assigned reviewer for this pull request. In a real world scenario, rules would be set up to protect certain branches and require a minimum of x code reviewers before being able to be merged into the branch. Ensurse that code being merged meets quality standards and doesn’t break anything and allows for better code efficiency from reviewer’s feedback  A screenshot of a computer  Description automatically generated |  |
| 1. 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.   Screenshot below shows that nest has been installed and a project is created via the NEST CLI.  A screen shot of a computer  Description automatically generated  Screenshot shows the Nest application running, one thing that I learn was there is a command “start:dev” where it would watch for changes so you could refresh the page and it would update without needing to start and stop the application each time changes are made. This was found inside the package.json file.  A screenshot of a chat  Description automatically generated  Screenshot shows the commands listed inside package.json  A screen shot of a computer program  Description automatically generated |  |
| 1. 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.   The screenshot below show the changes made to getHello(), by default it showed “Hello World” which was then changed to “Hello I’m a new String”.  A screenshot of a chat  Description automatically generated  A screenshot of a chat  Description automatically generated |  |