# Shelby El-rassi, T3A1, Flextrack, MEL

## Q1 |Provide an overview and description of a standard source control process for a large project.

#### **What is source control?**

Source control (also known as version control) is the tracking and management of files/code and any changes made. The practice of source control is the tracking of any modifications and the ability to revert back to any previous versions in the event of mistakes or backtracking, a key feature is the ability to know which version is the current iteration. An excellent example of this is Git.

**Git**

Git is a free and open source version control system, it is designed for source code management, however it does have the ability to management other sets of files. Git is currently one of the most popular version control system for software development, as it is a distributed version control system, most Git operations are local, with every user/client having their owner server and full back up of the repository locally. It has the advantages of speed, feature branching, useful staging areas and the ability to run multiple workflows. Below is an overview of the git-flow branching model, which has become a popular version control model for software teams.

**Git-flow Workflow**

Git-flow workflow first published by Vincent Driessen at nvie. It describes a firm branching model based around a projects release. The hefty framework is ideal for management of larger projects.

**Overview of this workflow:**

* Getting started with this workflow starts with an actual toolset available for installation, after installation (which is available on multiple operating systems), utilising git-flow for a project involves executing *git flow init*. It essentially is an extension of the standard *git init* command, that creates branches.
* A central repository houses the ‘Master’ branch, which is for the core of the project reflecting a production-ready base, along with another branch named ‘Develop’, this branch runs parallel with the master and reflects a state of latest features for next release (also known as an integration branch). An important note is the ‘Develop’ branch will house the entire history of the project, whereas the ‘Master’ will only contain the latest version.
* When *git flow init* is utilised, this command will create the ‘Develop’ branch.
* When a new feature is started, a new branch is created for this feature. This can be done via git-flow extension by the command *git flow feature start feature\_branch.*
* Each developer pulls and pushes to the origin ‘Develop’ as the parent branch.
* Developers may also pull changes from other peer’s branches, which can be useful for working together in sub-teams.
* Each branch should have a clear descriptive name with a clear purpose.
* Each feature branch should also be pushed to the central repository branch ‘Develop’, this means the code can be shared for collaboration.
* When complete these feature branches get merged back into ‘Develop’, never interacting with the ‘Master’ branch directly.
* Once a feature branch is finalised, the command *git flow feature finish feature\_branch* can be executed.
* When the ‘Develop’ branch has accumulated the required features for release or a scheduled date is imminent, a ‘Release’ branch gets forked off ‘Develop’. This ensures the next release cycle beings, so new features cannot be added after this point, only fixes, documentation and other release centred tasks.
* Once this ‘Release’ branch is ready to go, it gets merged into ‘Master’ and marked with a release number.
* The usage of a ‘Release’ branch for initial preparation of the release ensures that the rest of the team is able to continue working on features for the succeeding release date.
* It is important that the ‘Release’ branch also gets merged back into ‘Develop’ as well as ‘Master’ for any updates possibly added which new features might need access too. Once this process is complete the current ‘Release’ branch can be deleted.
* Another type of branch operated in this flow are ‘Hotfix’ branches, which are in place to swiftly fix up production releases. They operate just like feature branches, however, are parented off the ‘Master’ branch. A ‘Hotfix’ branch is the only branch forked directly off the ‘Master’. Once the fix is fully complete, these branches get merged immediately into ‘Master’ and ‘Develop’, with the ‘Master’ tagged with an updated version number. Operating a dedicated branch just for branch is a key advantage to this workflow as it ensures issues can be addresses by the team, with total interruption of the rest of the workflow.

In summary, this workflow is great for the large projects operating on a release-based schedule, offering the specific branches enable this workflow management, minimising interruption.

#### [Sources]---

1. <https://www.wrike.com/project-management-guide/faq/why-should-i-use-version-control-in-project-management-software/>
2. <https://www.automationworld.com/home/blog/13316896/configuration-management-and-version-control>
3. <https://hackernoon.com/top-10-version-control-systems-4d314cf7adea>
4. <https://nvie.com/posts/a-successful-git-branching-model/>
5. https://www.atlassian.com/git/tutorials/comparing-workflows/gitflow-workflow

## Q2 |What are the most important aspects of quality software?

1. In ffgfgfg

#### [Sources]---

1. [label](https://github.com/Shelby219/T3A1_workbook/blob/master/link)

## Q3 |Outline a standard high level structure for a MERN stack application and explain the components.

#### [Sources]---

1. [label](https://github.com/Shelby219/T3A1_workbook/blob/master/link)

## Q4 |A team is about to engage in a project, developing a website for a small business. What knowledge and skills would they need in order to develop the project?

#### [Sources]---

1. [label](https://github.com/Shelby219/T3A1_workbook/blob/master/link)

## Q5 |With reference to one of your own projects, discuss what knowledge or skills were required to complete your project, and to overcome challenges.

#### [Sources]---

1. [label](https://github.com/Shelby219/T3A1_workbook/blob/master/link)

## Q6 |With reference to one of your own projects, evaluate how effective your knowledge and skills were for this project, and suggest changes or improvements for future projects of a similar nature.

#### [Sources]---

1. [label](https://github.com/Shelby219/T3A1_workbook/blob/master/link)

## Q7 |Explain control flow, using an example from the JavaScript programming language.

#### [Sources]---

1. [label](https://github.com/Shelby219/T3A1_workbook/blob/master/link)

## Q8 |Explain type coercion, using examples from the JavaScript programming language.

#### [Sources]---

1. [label](https://github.com/Shelby219/T3A1_workbook/blob/master/link)

## Q9 |Explain data types, using examples from the JavaScript programming language.

#### [Sources]---

1. [label](https://github.com/Shelby219/T3A1_workbook/blob/master/link)

## Q10 |Explain how arrays can be manipulated in JavaScript, using examples from the JavaScript programming language.

#### [Sources]---

1. [label](https://github.com/Shelby219/T3A1_workbook/blob/master/link)

## Q11 |Explain how objects can be manipulated in JavaScript, using examples from the JavaScript programming language.

#### [Sources]---

1. [label](https://github.com/Shelby219/T3A1_workbook/blob/master/link)

## Q12 |Explain how JSON can be manipulated in JavaScript, using examples from the JavaScript programming language.

#### [Sources]---

1. [label](https://github.com/Shelby219/T3A1_workbook/blob/master/link)

## Q13 |For the code snippet provided below, write comments for each line of code to explain its functionality. In your comments you must demonstrates your ability to recognise and identify functions, ranges and classes.

class Car {

constructor(brand) {

this.carname = brand;

}

present() {

return 'I have a ' + this.carname;

}

}

class Model extends Car {

constructor(brand, mod) {

super(brand);

this.model = mod;

}

show() {

return this.present() + ', it was made in ' + this.model;

}

}

let makes = ["Ford", "Holden", "Toyota"]

let models = Array.from(new Array(40), (x,i) => i + 1980)

function randomIntFromInterval(min,max) { // min and max included

return Math.floor(Math.random()\*(max-min+1)+min);

}

for (model of models) {

make = makes[randomIntFromInterval(0,makes.length-1)]

model = models[randomIntFromInterval(0,makes.length-1)]

mycar = new Model(make, model);

console.log(mycar.show())

}

#### [Sources]---

1. [label](https://github.com/Shelby219/T3A1_workbook/blob/master/link)