**City University of Seattle**

**CS 504 - Software Engineering Summer 2021**

**Independent Project 4**

**Vaccine Scheduler**

**Anil Erturk**

**Introduction**

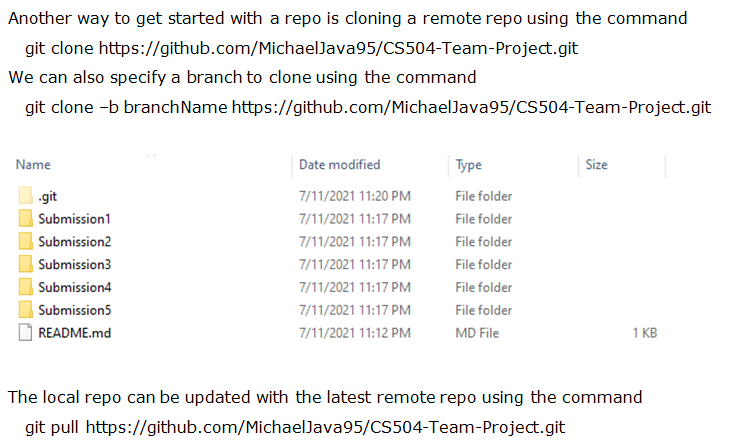
The independent projects covered the sections of software engineering that are used for software projects. The internet affected software development a lot. The internet allowed the agile process to exist because the internet changed deployment by making it possible to deploy the new version to the end user every two weeks or so. The Internet also changed the development through Git and GitHub. The requirements are taken from the customer but the communication with the customer does not end after the initial requirement engineering. Especially in the agile process. Then the design is divided into small tasks and given to developers. The developers push their commits to a repo where the team lead can see and accept them. The team lead also gives developers feedback on their code.

Having a process for the entire project is going to allow the team to have plans and institutions for everything needed for a project. Not having a process is going to leave the team without plans and tools needed to overcome challenges. Working without a process is going to cause projects to fail more often.

In software development, design patterns exist because there are common problems that use common solutions. Software engineering is the design pattern for the entire project. All projects have common problems and solutions. Instead of reinventing the wheel, it is better to come up with a process to guide the team along during the project.

**Configuration Management System**

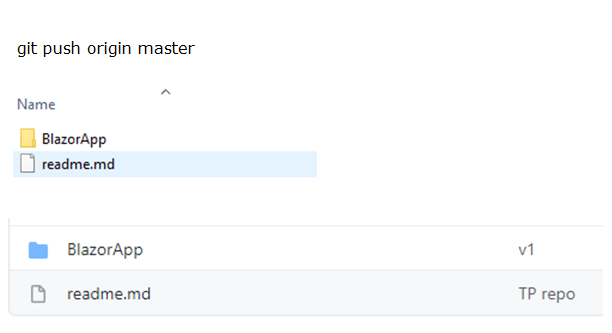
Git and GitHub are the Configuration Management System (CMS) of today. It is going to help us with getting the code from a remote repo for the first time or getting the latest version of the repo.



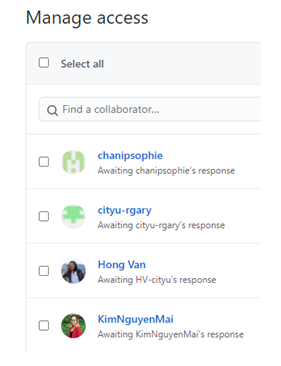
After changes are made to the code on the local machine, the changes can selectively be added to a commit. This commit can be given a message and then pushed to the remote repo.



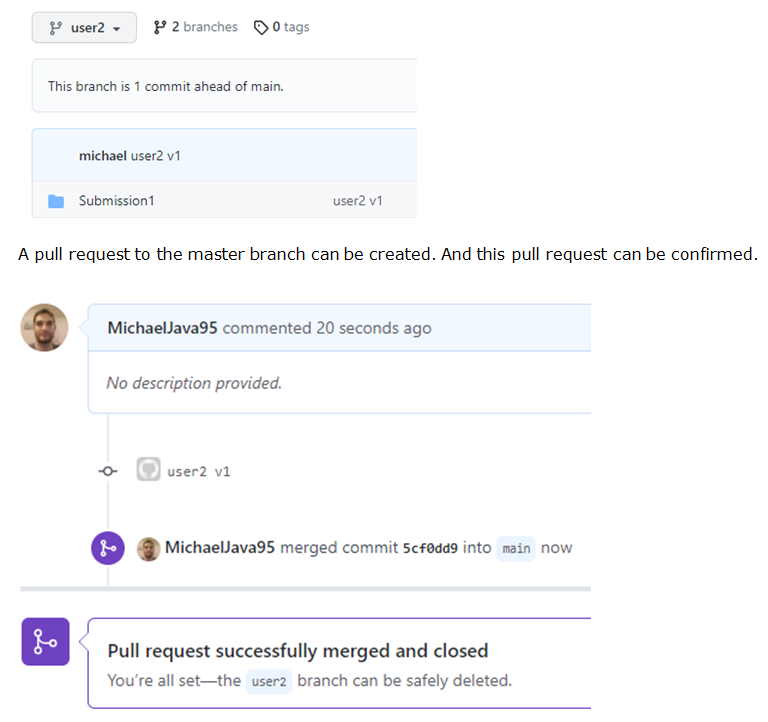




This system is how people pull and push code in today’s software development world. For this to be possible, multiple people must be able to work on the same project. Which is possible with GitHub. Push access to a project can be given to people by managing the access of people to a project.



Different people generally push their code to different branches, which can then be merged to the main branch.



With these tools, we generally want people to work on different modules. We wouldn’t want them to overwrite each other’s work. If collisions do exist during merges, GitHub requests to handle each collision.

**Requirements and Architecture**

Like established earlier, it is important to come up with a process and plan for the project that will guide the team. The requirement engineering is part of this processing and planning. The functional and non-functional requirements will guide the team and help create the design. The design will be divided into tasks for developers to implement. Architecture, dependencies, languages to be used, tools to be used, versions to be used, compatibility, etc are also parts of software engineering. These high-level decisions must be made at first to come up with a plan to follow. Having a plan will also mean that the team has an end game that can at least work in theory. If the correct architecture and compatibility of tools and versions are not decided from the beginning, there might be costly refactors later on.

Functional requirements for the vaccine scheduler were the user stories. These cover vaccine selection, inputting zip code to get the list of closest destinations, choosing a specific type of the vaccine, selecting the dose, selecting the date and time, inputting other details such as name.

Nonfunctional requirements for the vaccine scheduler are accessibility, privacy, security, and safety. Accessibility is required for vaccines to be available to everyone. Accessibility is gained through the internet because a web application can be accessed from many different software and hardware platforms. Privacy is gained through security. Web applications have common security challenges. They have to be robust against common cyber attacks such as SQL injection and cross-site scripting. Safety is gained through robust programming and the handling of exceptions.

The architecture appropriate for this project is MVC (Model View Controller). It will help create a dynamic web application that checks all the requirements of the user inputting data and the server responding dynamically.

**Test Plan and Cases**

The testing part of the project will tell the team whether the project is ready for deployment or not. The team will determine what level of quality they want to ensure and deploy the product after the test results reach that level. The testing starts at the unit level, then the team moves on to integration testing, system testing, and lastly the acceptance testing.

The objectives of the tests are generally covering functional and nonfunctional requirements.

Test methodology will take advantage of the MVC architecture. MVC divides the software into three, and each part is further divided into multiple files. This gives the team a starting point with the unit tests. Also, integration and system tests are straightforward in this architecture. Lastly, regression testing will allow the team to ensure quality as they make changes to the software.

IDEs and other tools such as Selenium give the team automated testing for all kinds of applications including web applications. Selenium tests can act like a human and use a website which is a very valuable tool for web application development. The test cases will cover the functional and non functional requirements.

**Software Metrics**

A software metric is a measure of software characteristics that are quantifiable or countable.

In order to understand why these metrics are necessary, one needs to understand the purpose of software testing.

The purpose of software testing is to ensure that the software product is above a predetermined quantifiable level of quality. This is because the software team agrees to spend a certain amount of resources on testing to ensure the number of tests, coverage, rate of bugs and failed tests. Metrics allows the development team to determine whether the software product is above the predetermined level of quality which makes it ready for deployment. This is why metrics are necessary for acceptance testing of a software product.

**Lessons Learned**

Software engineering requires a process to increase the success rate. Software engineering processes act like programming design patterns. There are common problems and there are common solutions to these problems. The same thing applies to software engineering. For every project, there are going to be the same common problems. The team needs to have a system, a process in order to professionalize their work. The process helps the team monitor and handle the project.

Instead of diving right into the project, doing requirements and design engineering first gives the team a better chance at success. Because at least the team now has a plan that can work if implemented correctly. Which means the team is one step closer to success. Not having a plan is a good way of running into costly refactors or complete project cancellations.