**Senior Project Proposal**

1. **Project Name:**

Jeremy’s Version Control (**jvc**)

1. **Group Size / Contact information**

Jeremy Duong ( 801-604-2224 | [duo13001@byui.edu](mailto:duo13001@byui.edu) )

1. **Abstract/Propose**

As the name suggest, **jvc** is a version control system. We all know the most common of such system is currently **git.** In terms of project structures, data storage, and algorithms, **jvc** will borrow many of the ideas from **git.** However, the purpose of the **jvc** project is to design a version control system that is much simpler for new software developers to use. The vision is to make this system so simple and easy to understand that it gains favor of new students who are taking beginner level classes at BYUI (the CSE 110, CSE 111, and CSE 210 students). Another purpose of the project is so that I myself can gain a rock solid understanding of everything under the hood of **git**, the well known VCS that was created by the genius Linus Torvalds.

1. **Background/Prior Knowledge**

Given that my purpose is to recreate **git,** which deals a lot with file readings, file managements, and hashing, and to really understand git, it would only be appropriate to build the project exclusively in a low level language, C++ to be specific.

C++ is my first programming language. I have been using this language since I first started programming in 2014. Since I have done multiple projects for a few immediate to hard classes in C++, I would consider myself at least a Novice in this language.

As mentioned above, **jvc** is modeled after **git.** However, my goal is to build everything from the ground up. This means that **jvc** will NOT use any thing other than the standard libraries provided by C++. This will be very exciting for me since I have never created anything like this before, even though I have done a good amount of research to at least get started.

In terms of course background that got me interested in this subject, ironically, there is no class that teaches me how git works save a few basic functionalities to get started. The most basic commands of git that all beginners should know are “status”, “add”, “commit”, “log”, “revert”, and “push”. As such, the commands that **jvc** will have are similar to these commands, only simpler.

1. **Description**

Provide the details of your project. In particular, make sure to include:

**Why?** As a TA for CSE 210, I have helped hundreds of students learn how to use a version control system, specifically git. Over the semesters of being a TA, I noticed that most students who take this class have absolutely no background in using **git.** With that said, one very astonishing thing, at least to me, is the fact that we only have ONE week of this class to learn about git before we move onto other OOP topics. How many CSE 210 students really understand what “status”, “add” and “commit” really mean after one week? In my experience, almost ALL of the students I have helped continue to struggle to use these commands correctly even at the end of the semester. As someone who has had some basic experience working in the field of software engineering, we are very familiar to these commands. However, to a completely new student in the field, these terms make absolutely no sense. Commands such as “branch”, “merge”, or “rebase” are even more over their heads. Yet, these commands are all parts of the “git” workflow that these students must learn in the scope of one week. As such, one of the purposes of **jvc** is to create a version control workflow that is so simple, intuitive, and much less time consuming to learn, which would allow students to be completely confident using after just one week.

**What?:** There certainly are other version control systems that have been created before. However, I would say that most of them are no longer used by developers. The one being used most dominantly is git, which has the problem of “complicated” interface that I described above. **Jvc** takes the genius of git internals and marry it with the idea of simple user interface and work flow to create something new that will be of much help to young software engineer students who are freshly new to the idea of version control.

**Who?:** Target Audience – As mentioned above, **jvc** targets a very specific group of audience: **Inexperience CS and SE students** (those who are struggling to pass the beginning level classes such as CSE 110, CSE 111, and CSE 210). If we have to introduce git in CSE 110, we might be hesitant thinking that the students are so inexperience that it might overwhelm them. **Jvc,** on the other hands, aims to be something so simple that make it so that instructors of these courses are confident

**Where?** Platform – Like git, it will be downloaded and installed on the computer of the user.

**How?** Workflow – Similar to git, the program will be used to track changes to a project. The workflow will be as followed:

* Initialize repository with **jvc init.**
* Add/modify files
* Check status using **jvc status.**
* Save changes to the repository using **jvc save** (a combination of add and commit in git)
* **jvc history** to see the history of all saves.
* **jvc revert** to revert back to a specific save.

When? SMART Goal – The project is done when the 5 basic functionalities are implemented and working: **jvc init, jvc status, jvc save, jvc history,** and **jvc revert**

Once these 5 commands are completed, the project can be demo.

1. **Significance**

Reflecting on my own experience as an inexperienced Computer Science major trying to learn git and how confused I was trying to learn the workflow, I think this project will help a lot of new CS and SE students have a much easier time learning how to use version control.

1. **New Computer Science Concepts**

The reason why I chose this project is because I have been using git for a while, but the internal workings of it together with some of its functionalities are still a mystery to me. I would like to learn how files can be read in binary and then hashed using SHA1 (this is how git track changes to a file). I would also want to learn how to most effectively traverse through all the files in a certain directory to check and see if anything is changed compared to the last save. This task seems to require recursion somehow. I’m not sure how it is done yet but I want to learn how to do it.

1. **Interestingness**

Git has always been interesting to me. The idea of having so many different versions of the same program, being able to keep track of all of them, reverting back to a specific version, and then merging different versions when appropriate seem like something really hard to do. On top of that, being able to do this without messing up the whole project is also something that takes a lot of thinking and cleverness. As such, I have always admired Linus Torvalds’ genius in being able to develop this and hoped to be able to reach a level of understanding that will allow me to build my own VCS.

1. **Tasks and Schedule**

* Week 1-4:
  + Goal: implement and test **jvc init** and **jvc status.**
  + Time spent: 3 hours a day, 5 days a week (15hours/week). **Total: 60 hours**
* Week 5-8:
  + Goal: implement and test **jvc save** and **jvc history.**
  + Time spent: 3 hours a day, 5 days a week (15hours/week). **Total: 60 hours**
* Week 9-12:
  + Goals:
    - Implement and test **jvc revert**
    - Consider adding **jvc branch** and **jvc push**
  + Time spent: 3 hours a day, 5 days a week (15 hours/week). **Total: 60 hours**
* Week 13 and 14:
  + Goals:
    - Finish up **jvc branch** and **jvc push** (if implemented)
    - Clean up project structures and code.
  + Time spent: 2 hours a day, 5 days a week (10 hours/week). **Total: 20 hours**

**Total time spent: 200 hours.**

1. **Resources**

I only need the following:

* + The internet.
  + My PC and Laptop
  + Git and Github (ironic haha)
  + Latest C++ compiler

I might also consult a few professors on campus if I get stuck at some point.