**Software Project Management - OSS Project Proposal**

**By Team#1 - Git Mavericks**

## OSS: **Git** <<https://git-scm.com>/>

## Team Profile:

Name: **Git Mavericks**

Members/Roles:

* Amelia Mazer - SCRUM Master
* Rahul Sridhar Bangalore - Product Owner
* Jingyu Xu - Q/A Tester
* Tina Zou - Tester
* Jonas Hernando - Tester
* Vinit Jahagirdar - Software Engineer
* Yunus Kocabey - Software Engineer
* Thao T Thu Nguyen - Support

Vision: Help our developers be more efficient in their collaboration so that our customers can be more proficient in their optimal pricing.

## Company Profile:

Name: **IntelliPrice**

Founded: 2017

Employees: 50

Headquarters: San Jose, CA

### About us:

IntelliPrice is on a journey to make e-commerce product pricing simple. We provide cloud-based price optimization solutions, which assist small-to-medium-sized businesses with automating their pricing either fully or partly by leveraging machine learning and advanced algorithms.

Our solutions are designed to be flexible and customizable, allowing businesses to tailor their pricing strategies to their unique needs and objectives. Whether it is to optimize prices for a specific product or customer segment or adjust prices in response to supply and demand, we provide the tools and insights to make informed pricing decisions. Some key features include:

- Competitor Pricing Analysis

- Revenue Tracking, Process Automation

- Product Management

- User-friendly Portal

Our company has a total of 50 employees and it’s growing consistently, and our technical team makes up half of the total staff. The number of developers is 50% of the total technical staff. The rest of the team includes data scientists, project managers, product owners, test leads, and release managers, among others.

**Mission Statement:** We make your price right for you so you can achieve pricing success with confidence and ease.

**Vision Statement:** To create and innovate the best software that gives optimal pricing for all.

**OSS Problem statement: Internal to the company**

* A fast-growing number of software developers in the company increases the difficulty in collaboration among project teams. This leads to errors and duplication of efforts, ultimately decreasing productivity and prolonging development time.
* IntelliPrice’s developers currently use **Apache Subversion (SVN)** as its version control system. While this solution worked for us when we were starting due to its easier learning curve and trunk-branch-tag workflow, we have found that our work has been inhibited by the below limitations:
  + As our workload has increased, relying on updating one central repository has slowed down our work due to an increase in network bandwidth.
  + During unforeseen work interruptions, our developers have not been able to continue with their work due to lost connections or if the central repository goes down, leading to our developers not being able to commit their code until the central repository is restored.
  + As our company moves towards embracing remote work, we need to be able to prevent potential code merge conflicts. Subversion’s more central approach to contributions has resulted in a breakdown of cohesion if our team members cannot communicate with each other.

**Business Objectives:**

* Facilitate collaboration among software developers
  + Enabling effective domestic remote work
  + Version control

**Proposed Solution:**

* Migration from a Centralized Version Control System (Apache Subversion - SVN) to Distributed Version Control System (Git).
* Adapting to Git will help us tackle most of the challenges related to:
  + Increase collaboration and streamline development
  + Allow distributed teams to work effectively and improve productivity
  + Performance bottleneck
  + Merge issues

[https://www.geeksforgeeks.org/centralized-vs-distributed-version-control-which-one-should-we-choose](https://www.geeksforgeeks.org/centralized-vs-distributed-version-control-which-one-should-we-choose/)/

**Key Sponsors & Stakeholders:**

* Change Management Sponsors:
  + CIO (Executive Sponsor)
  + Director of IT Security
  + Manager of Operations
  + Volunteer testers

**Hypothesis**: How a source code management software will be able to facilitate collaboration among developers:

We believe that using a Distributed Version Control System (Git) is better than a Centralized Version Control System (Apache Subversion) in terms of collaboration among developers due to the following reasons:

* *Version control*: SCM software allows for versioning of the source codes, enabling IT teams to keep track of changes made to the codebase over time. This helps in ensuring that only authorized changes are made and provides a clear audit trail of who made the changes and when. (1, 6)
  + Using SVN worked well when we were starting off as a company where we only had a small team of developers. SVN’s branch-trunk approach had an easier learning curve, so implementing it across our organization was done smoothly and in a timely manner.
  + As our organization has grown in staff headcount and continues to embrace remote and hybrid work, we believe that Git, even with a steeper learning curve since the use on local repositories requires staged changes and merges, will work better for our organization’s work in delivering a valuable product while also maintaining organization health.
* *Collaboration:* SCM software lets any number of developers work on features and fixes in parallel, without the worry of accidentally affecting another team member's changes. As a result, it can encourage team members to collaborate and create a plan of how best to approach the project. (5)
  + The top-down approach of SVN worked well for us when we started off as a company as it was relatively easy to manage access control for our small developer team. As we have grown and found ourselves in various work locations, working off of one central repository has led to merge conflicts amongst our employees if communications between developers break down.
  + In contrast, with Git, we hope to see real benefits in its ability for developers to work offline, and allowing for each developer to work on their own piece of code to be staged for merge will ideally enhance productivity and reduce conflict.
* *Branch protection:* SCM software provides branch policy features that help teams protect their important branches of development. Policies enforce code quality and change management standards. These can include requiring code review approvals, preventing direct pushes, enforcing specific commit message formats, and ensuring that changes to critical branches follow established policies (2, 3, 4)
  + With SVN we have again been able to manage access to our team members when our headcount was smaller, but we have also run into problems if the central repository unexpectedly goes down. This single point of failure has had ramifications for our company’s ability to continually develop and improve our services.
  + With Git, we hope that the distributed version control model will help to eliminate the single point of failure, and will allow our work to continue unimpeded in the case of downtime.

Source for hypothesis:

1. <https://www.atlassian.com/git/tutorials/source-code-management>

2.<https://www.ibm.com/docs/en/cognos-analytics/10.2.2?topic=SSEP7J_10.2.2/com.ibm.swg.ba.cognos.ug_fm.10.2.2.doc/c_settingupmultiuserenvironment.htm>

3. <https://www.varonis.com/blog/git-branching>

4. <https://learn.microsoft.com/en-us/azure/devops/repos/git/branch-policies?view=azure-devops&tabs=browser>

5.<https://ca.indeed.com/career-advice/career-development/what-is-source-code-management#:~:text=Encourages%20better%20collaboration,best%20to%20approach%20the%20project>.

6. <https://git-scm.com/video/what-is-version-control>

**Why Git? / Other Reasons for Git:**

* Git is compatible & in-built into most IDEs,
* Easy to Integrate with tools like JIRA, Cloud storage services, ERP, etc.,
* Good open community support.

<https://www.linkedin.com/pulse/migrating-git-version-control-just-philip-armour/>

**Return on Value:**

* Consistent delivery and improvements in our products
* Employee Productivity and contribution, how much work employees have done compared to other months and quarters
* Employee satisfaction
* Customer Satisfaction and Renewal

<https://www.batimes.com/articles/roi-vs-rov-an-agile-insight/#:~:text=ROV%20(Return%20on%20value)%20is,delivery%20and%20platform%20technology%20respectively>.

<https://www.projecttimes.com/articles/return-on-value-versus-return-on-investment-an-agile-insight/>

**Advantages of Git:**

* The increased productivityand reduced development time and costs associated with merging and resolving conflicts by the collaborating teams,
* Better risk management by a version control system that provides a complete history of code changes, enabling organizations to track and revert changes, reducing the risk of data loss and errors without any additional subscription/payment for the storage.

<https://www.atlassian.com/git/tutorials/what-is-version-control>

<https://www.perforce.com/blog/vcs/8-version-control-best-practices>

**Disadvantages of Integrating Git:**

* The learning curve/Time, not everyone may know how to use Git
* Cost of migrating and configuring other software & hardware that are git compatible