Tools Co-op

# Project Overview

This project aims to build a system that facilitates a library-like loaning system for tools.

The tool system will allow users to reserve tools through our website, while also providing them with specific instructions and information related to the transaction. The underlying database will keep track of customer accounts, lists of available tools and toolsets, and a superuser account that will allow employees to audit both tools and customers.

# Team Organization

Our team is composed of four software engineers with varying backgrounds and skill levels. As equal partners on the project, we will plan out tasks according to project needs, delegating them out in a way that gives each team member an equal workload, while also matching tasks to individual skillsets. As there are no team leaders or management positions, each member will help audit the work of their peers.

# Software Development Process

The development will be broken up into four phases. Each phase will be a little like a Sprint in an Agile method and a little like an iteration in a Spiral process. Specifically, each phase will be like a Sprint, in that work to be done will be organized into small tasks, placed into a “backlog”, and prioritized. Then, using on time-box scheduling, the team will decide which tasks the phase (Sprint) will address. The team will use a Scrum Board to keep track of tasks in the backlog, those that will be part of the current Sprint, those in progress, and those that are done.

Each phase will also be a little like an iteration in a Spiral process, in that each phase will include some risk analysis and that any development activity (requirements capture, analysis, design, implementation, etc.) can be done during any phase. Early phases will focus on understanding (requirements capture and analysis) and subsequent phases will focus on design and implementation. Each phase will include a retrospective.

|  |  |
| --- | --- |
| **Phase** | **Iteration** |
| 1. | Phase 1 - Requirements Capture |
| 2. | Phase 2 - Analysis, Architectural, UI, and DB Design |
| 3 | Phase 3 - Implementation, and Unit Testing |
| 4 | Phase 4 - More Implementation and Testing |

We will use Unified Modeling Language (UML) to document user goals, structural concepts, component interactions, and behaviors.

# Communication policies, procedures, and tools

For group communication, we have decided to use GroupMe for general team communication. We are also using Git (and GitHub) as our configuration management system, with ClickUp as our team/task management system.

We use ClickUp to create, organize, and assign tasks to group members. Whenever a member completes a task, it is given the “review” tag, where it is audited by all other group members and revised before finally marked as “done”. At the end of each sprint, all group members are expected to thoroughly inspect and test each software component to ensure every submission meets our standards of quality.

# Risk Analysis

Currently, there are several risks that can be foreseen before we begin development on this project. Although not exhaustive, this list represents several of the major tasks that must be done, and the risks associated with them:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Task | Description | Likelihood | Severity | Consequences | Work-arounds |
| (1) Build a database | A database must be built. It must be secure, yet accessible for users and employees | Medium | Critical | Data could be lost, insecure, or otherwise compromised. | Rather than use a database, use a big list. Not effective. |
| (2) Populate database with initial data | Data must initially be inputted by hand from business ledger into database (in actuality, there is a .csv file to be read in). | Medium | Low | Tool numbers could be misrepresented, leading to small errors in the system and great frustration to customers. | Give all current tools to charity, and then buy exact numbers of new tools. |
| (3) Build a payment system | Customers must be able to pay for the service and should not get the service when they don’t pay. However, this is very difficult to build. | High | Medium | Customers may either not have access or have free access to the system. The work around may be the easiest option. | Have customers pay in person, with employees updating their account status on their behalf. |
| (4) Build a reservation system | Build a system that correctly identifies what tools are available, tracks number of requests, and allows users to directly reserve tools. | Low | Medium | Same as in task (2). Additionally, system may incorrectly indicate which tools should be purchased. | None. This system really needs to be functioning. |
| (5) Manage users/superusers | Employee accounts must have special access. | Low | Medium | Customers can access or modify restricted data. | Alternate system without superuser accounts. |

# Configuration Management

See the README.md in the Git repository.