Requirements Definition

Tool Co-op

1. Introduction and Context

This document describes the requirements for the Tool Co-op website that is being developed for Joe's company. This website will be a way for the company to modernize its approach to doing business, as well as decrease the risk of human error. The website seeks to autotomize various tasks that Joe must complete daily including, but not limited to: Creating and deleting tools in the database, lending tools, checking in tools that have been returned, viewing a list of customer profiles, buying a new tool, and viewing the current status of a certain tool.

In general, this website is designed as an interface for company employees only, but has potential to be expanded so that customers of the Tool Co-op may also use the website as a way to check out and request tools. With that in mind, the main purpose of this document is to describe the website pertaining to company employees. Further adjustments will be made as the project expands.

The requirements presented in this document are both functional and non-functional. The functional requirements seek to explain how the website will work while not necessarily describing how it will do so. The non-functional requirements describe an overall design and architecture of the product. We will build the website using the Django web framework and utilize Git and the Scrum method to work as a team to achieve our goal.

2. Users and their Goals

See use case diagrams in Docs folder.

3. Functional Requirements

- A tab to display all tools
    - All tools will come from the database and this tab will load them into view.
- Database for the tools (In our case we will utilize the Django models feature to accomplish this)
    - We will provide an init view to display and test database results'.
    - Database will contain the current status of any given tool.
- Shopping cart to check out the tools
    - Maximum of five tools per user per checkout.
    - When checking out tools enter the name of the person checking it out, date, and due date.
- Display whether tool is available or not
    - We will ensure that tools that are checked out are unable to be added to the cart.
- Delete a tool and remove from the database.
    - Provide a confirmation message when deleting a tool.
- Reorder a tool.
    - One-click reordering of any tool in the database.
    - Provide a confirmation message when reordering a tool.

    - Order gets sent via fax/email/etc to tool providing company.
    - Order status is available to view from an orders tab.

4. Non-functional Requirements

- We will follow an agile method and will have weekly meetings either in person or via Slack.
- We will implement unit tests for each of the functions in our project.
- Website must be able to support both customer and employee user privileges.
- Website must be able to operate at all times, unless Joe requests otherwise.

5. Future Features

- Expand the website to include a customer interface alongside the employee interface.
    - This portion of the website would allow users to view the tools in the database and check them in and out. It would not however have the administrative functionality of an employee interface (i.e. a customer cannot delete or create tools from the database.)
    - Extra precautions would need to be taken with cybersecurity as we would store users login and personal information on hand.
- Add customer records to the website.
    - Customer records would include a picture of the customer, their full name, a history of tools they've checked out, and any tools they currently have checked out.
    - Customers can be added manually to the system.
    - If a tool is checked out and the system sees the person checking it out is not in the system, it manually creates a record for them.
    - If a tool is checked out and the system sees the person is already in the system, it asks the employee to confirm the person is the correct customer.
- Add tool kits to the website.
    - Add a set of tools as a 'tool kit' that can simultaneously be checked out at once.
    - Tool kits would consist of tools that might all be needed for the same job a customer is working on.
- Add accessible tool usage data.
    - An employee-accessible tab would allow employees to see which tools are in the highest demand (searched on the website, requests made to check out, etc), which tools are checked-out the most often, and other useful snippets of data.

6. Glossary

Database: A structured set of data held in a computer, especially one that is accessible in various ways.

Django: A Python-based web framework, which follows the model-template-view (MTV) architectural pattern in order to ease the creation of complex, database-driven websites.

Django Models: A customizable storage structure that contains the essential fields and behaviors of the data being stored in the database.

Init View: Path that will initialize the database with dummy entries for testing.

Slack: A work-centric online messaging service.

Unit Test: A software testing method by which individual units of source code, sets of one or more computer program modules together with associated control data, usage procedures, and operating procedures, are tested to determine whether they are fit for use.

User Privileges: A privilege given to a user to perform a particular action or set of actions within a computer system. Privileges can vary between users based on permissions given by the system.