Dan’s Cool Beans

# 

# Project Overview

This project aims to build a web app to be used in a frappuccino shop. Includes capability for online ordering by customers, in-person ordering completed by a cashier, queues of drinks to be made and which are already completed for use by the barista(s), submission of hours worked for employees, ability to order and track inventory, change the menu, as well as paying of employees by the manager. Will utilize log-in with encrypted passwords and authentication for various roles and different views for different users.

# Team Organization

Scrum master - Emily Thomas

Software design and developers - Andre Nguyen, Karl Poulson, Maris Hurst

# Software Development Process

The development will be broken up into five 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

| **Policies** | *Emergency, illness, or otherwise* -  If possible, provide at least 1 day’s notice to the team that you are unable to meet your deadline.  Depending on the severity of the situation and availability of everyone else, the remaining team members may divide up your tasks among themselves to attempt to meet the deadline. |
| --- | --- |
| **Procedures** | *Git workflow* -   1. Fork the group repository 2. Clone the forked repository to local machine 3. Create a new branch for the task    1. git checkout -b feature/[task-name-here] 4. Push branch to forked repository on personal account 5. Create PR to the group repository 6. At least 2 people review the PR before it is merged to the master/main branch   *Merge conflicts -*  Team members are responsible for managing their own merge conflicts, not the members merging the PR |
| **Tools** | *Discord* -  Primary communication method. Used to manage conversation and easily reference past conversations for future use  *Text* -  Used for quick, non-task related communications (typically involving scheduling)  *Google Drive* -  Used for collaborating on general documents (primarily for Phase 1) and sharing files between team members  *GitHub* -  Used to implement tasks in code and manage version control  *Jira* -  Used to manage upcoming and current tasks. Also has features for time logging / deadlines |
| **Team meetings** | Team meetings will be held on Tuesday’s or Thursday’s after 2:30pm.  Meeting times will be added/removed each week based on need.  Meeting agenda:   * Update group on status of your in-progress tasks * Discuss any potential issues that may prevent you from meeting your deadline * Plan which tasks are up next for each team member |

# Risk Analysis

| Risk | Details |
| --- | --- |
| Unable to authenticate passwords | Likelihood:   * Low   Severity:   * High   Consequences:   * Unhappy client * All users are unable to access their accounts   Workaround:   * Develop in Django to utilize the password authentication system they have already built |
| Issues storing / retrieving user data and/or inventory from database | Likelihood:   * Low   Severity:   * Medium-High   Consequences:   * Users are unable to view or update account info * System is unable to check if ingredients are available * Manager is unable to track or purchase more ingredients   Workaround:   * Carefully design / construct database schema and verify data flows at each step of the process |
| Broken UI | Likelihood:   * Low   Severity:   * Medium-High   Consequences:   * All user-types are unable to interact with their dashboards / views efficiently * The business cannot run smoothly without an efficient UI   Workaround:   * Two rounds of peer review will occur each time new code is submitted to the repo |
| Money not being removed from / added to accounts correctly | Likelihood:   * Low   Severity:   * High   Consequences:   * Customers could get a drink without having to pay for it * Employees may not get paid * Employees may get paid without manager losing any money * etc   Workaround:   * Implement several validation checks to ensure money is being transferred properly * Only accept money if we can prove money has left an account   + (Reduce potential points of failure) |
| Orders not moving to correct queues after status updates | Likelihood:   * Low   Severity:   * Very High   Consequences:   * Business will be unable to function properly without orders moving through the correct channels   Workaround:   * Increased rounds of testing * Reduce as many potential points of failure as possible |

# Configuration Management

See /docs/README.md for configuration management plan.