**Project Plan**

**Money LAWNdering**

**Project Overview**

This project aims to build a system that allows customers to post yard work-related jobs and for workers to accept those jobs.

The system will allow customers to post a job with its listing details. The system will allow workers to accept jobs that they want. The system will allow customers to pick who they want to work on the job. When a worker notifies the system that they have completed the job the system will transfer the price listed by the customer from the customers account to the workers account while taking a 10% cut. If the customer and worker continue to work with each other and decide to stay in contact over the phone rather than through posting job listings on the web app, then the customer can continue to pay the worker directly through the web app where the system will take a 5% cut on said scenarios.

**Team Organization**

Project Manager: Samuel Johnson

Designers and Developers: Tom Waite, Andrew Peterson, Xander Matheson

**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** |
| Phase 1 | Requirements Capture |
| Phase 2 | Analysis, Architectural, UI, and DB Design |
| Phase 3 | Implementation, and Unit Testing |
| 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**

Group Message - Main channel for communication. Used to schedule meetings and coordinate efforts.

Google Drive - Storage for files needing collaborative effort and review for Milestone 1.

GitHub - Formal repository user for submissions, version control, data tracking, backlog tracking, and communication with Professor Dan Watson and Rob Johnson.

**Risk Analysis**

* Database Structure
  + Likelihood - Low
  + Severity - Very high
  + Consequences - Ineffective data tracking can lead to confusion on job listings and responses, account balances, transactions, reviews, and account information, possibility of financial fraud.
  + Work-Around - There are none. System would lose all its value and basic functionality without a proper database implementation
* Login
  + Likelihood - Low
  + Severity - Med-High
  + Consequences - Dissatisfactory customer experience concerning transaction information
  + Work-Around - There exist some third party libraries for Django that implement better security features such as password strength checking.
* UI
  + Likelihood - Low
  + Severity - Very high
  + Consequences - Inability to interact with users in an efficient way
  + Work-Around - None, obviously we must have a UI or users won't be able to use the app and thus the whole app becomes useless
* Hosting
  + Likelihood - Low
  + Severity - Med
  + Consequences - Inability for system to host or serve information essential for system functionality
  + Work-Around - Host system through a hosting service
* Money Transactions
  + Likelihood - Low
  + Severity - Very High
  + Consequences - Customers are extremely mad, app loses its trustability, possible legal consequences
  + Work-Around - There are workarounds that would work if we had more time in this class and if it were real money, but for this class the workaround is that we only deal with fake money

**Configuration Management**

See the README.md in the Git repository