2nd City Putt Putt Golf

# Project Overview

This project aims to build a system for running and managing a game of putt putt golf.

The putt putt golf system will be used to manage a sponsored event at a putt putt golf event place. The software will allow participants (players) to track their game from start to finish. Participants will be able to also order drinks. The system will serve to replace paper and pencil for tracking score in a game of putt putt golf. The system can also be accessed by drink servers who will fulfill the orders of participants. Sponsors may access the system to pay for and reserve a day to hold an event at the event place. Managers access the system to perform administrative functions. The entire system will be accessible through a web browser on mobile or desktop.

# Team Organization

Team philosophies include but are not limited to:

* Egoless programming: opinions from any member of the team are just as valid as opinions from another member of the team. Team members will hold each other accountable without placing blame when things go awry
* DRY (Don’t Repeat Yourself): The team will attempt to not duplicate any work, document, or other source of information so as to maintain ‘one source of truth’ per item.
* Iterative is better than waterfall: Our team will focus on quickly iterating and increasing the number of successes we have. Rather than building slow and failing slow, we will prefer to iterate on our wins and build each part bit by bit.

Team members and their roles (for now):

* Austin Galura: Scrum Master – Full Stack Developer
* Jacob Clawson Database Administrator– Full Stack Developer
* Nathan Huntzinger: Integrations/Deployment – Full Stack Developer
* Jackson Miles: Project Management – Full Stack Developer

# 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

Our team will communicate primarily through Discord. Our procedure requires that when we cannot come to a consensus on how to answer a certain question the team has; we will try to go with what the majority desires.

Our tools for communication include: Discord, ClickUp, and GitHub.

We will manage tasks and work objectives for each sprint using ClickUp.

Our pull requests on GitHub must be approved by at least on other person on the team before we can merge to the master branch.

# Risk Analysis

At the beginning of each sprint the team will attempt to define what tasks should be completed during that sprint in a kickoff meeting. In this meeting the team will attempt to identify potential risks that may come up. We, as a team, will analyze the likelihood of each potential risk and how damaging said risk will be to the overall plan of the sprint.

Also, at the beginning of each sprint the team will attempt to determine which tasks depend on other tasks within that sprint. This will allow the team to prioritize those tasks which will/could block other tasks from being completed if delays arise in the task being depended on.

During sprint 2 we took note that the prototypes which were low-fidelity would block us from completing the higher-fidelity prototypes; thus, we prioritized the low-fidelity prototypes over other tickets which could be completed on their own. We saw the risk that these low-fidelity prototypes could go bad as quite low. Later on in the sprint we had to make adjustments to class diagrams to match with use-case diagrams. That was a risk we did not see before which may be something for us to consider during out next Sprint planning session.

# Configuration Management

See the README.md in the Git repository in the same directory as this file.