**Project Management Plan**

**<Your Name Here>**

**<Your Project Number and Name Here>**

**<Course, Semester, Year>**

Contents of this Document

Project Overview

Applicable Standards

Deliverables

Software Life Cycle Process

Tools and Computing Environment

Configuration Management

Quality Assurance

Risk Management

Table of Work Packages, Time Estimates, and Assignments

GANNT or PERT Chart

Technical Progress Metrics

Plan for tracking, control, and reporting of progress

**Project Overview**: <Include a 1-paragraph description of what your project is about.>

With this property management web app, tenants can make support requests regarding issues in their apartments. Requests can include a description, images, and videos. The landlord can monitor the requests and take appropriate actions to fix the issues. The landlord may then assign the request to a handyman and negotiate a price and time with them. Each user can have a rating based on how reputable they are.

**Applicable Standards**

* Coding Standard <The goal of a coding standard is to make maintenance easier. What is the minimal acceptable standard for code on your project? Include such things as required documentation, naming conventions, indentation style, etc. Make sure to cite your sources.>
  + Using clear names for functions and variables
  + Documenting the most frequently used portions of the code
  + Separating into modules to avoid large messy files
  + Using an auto-format tool to stay consistent in minor details
* Artifact Size Metric Standard <What are appropriate measures of "size" for your project? Specifically, how is each measured? Note that you will use these measures to monitor your progress, so it is important that you choose useful measures. For instance, predicted number of classes, number of input sources, APIs, data files can be examples of artifact size metrics.
* <Optional: any other relevant standards>

<NOTE: Choose metrics that are clear and easy to monitor. You will be expected to follow any standards you list in this section.>

**Deliverables**

|  |  |
| --- | --- |
| **Artifact** | **Due Dates**  <some will have multiple deliveries> |
| Individual Weekly Progress Reports | Weekly (Fridays) submission throughout the semester through webcourses |
| Concept of Operations |  |
| Software Project Management Plan (SPMP) |  |
| Software Requirements Specification (SRS) |  |
| Test Plan |  |
| High-Level Design |  |
| Detailed Design |  |
| Test Plan |  |
| Test Results |  |
| Source, Executable, Build Instructions |  |

**Software Life Cycle Process:** <What process will you follow? Give few sentences description of the process and the rationale for selecting this process. See chapter 2 of your textbook for background information. You may decide to implement a "hybrid" model which combines multiple processes.>

**Tools and Computing Environment:** <What operating system, programming languages, compilers, libraries, tools, and integrated environment, etc. will you use to design and build your project?>

**Configuration Management:** <How will you handle version control and change control? What procedures will be followed? You are required to use GitHub for configuration management (CM). If you plan to use any other CM tool, you need an approval from the course instructor and you need briefly discuss here the reasoning of your CM tool choice.>

**Quality Assurance:** <What QA activities will you do and when will each activity occur? ... Who is responsible for making sure this occurs (just list yourself)? How will the results be reported?>

**Risk Management:** <Identify potential risks for this project. For each risk, how will you manage the risk? It is expected that this information will be at a high-level at the beginning of the project.>

**Table of Work Packages, Time Estimates, and Assignments:** <Break down your project into a hierarchy of work packages. For each work package, estimate how much work time it will take to complete. For each work package, state who is responsible for its completion (should just be yourself). It is expected that this information will be at a high-level at the beginning of the project.>

**Technical Progress Metrics:** <You must estimate and track your technical progress using appropriate metrics for each phase of your project. What is a useful metric for each phase of your project? For example, for requirements phase, the total number of requirements, the number of requirements changes, the number of TBDs, etc.>

<For OO analysis and design, you might want to count UML diagrams completed. For detailed design and code, you might want to count packages, classes, methods. You will also want to think about other technical metrics such as: memory usage, execution speed, size of various documents, complexity of code (using any of the complexity metrics). These can help in planning and in tracking your project work.>

<Choose your metrics carefully -- select metrics that will be easy to collect, easy to report, and easy to interpret. The goal is to give management insight into the progress and risks of your project.>

**Plan for tracking, control, and reporting of progress**

<Briefly describe what data to collect, when to collect it, how and when to interpret it, how and when to report it. Following is an example that you can base your project plan on.>

You will post your progress report weekly through webcourses: the weekly report should include time spent in each activity, completed task(s), tasks in-progress, tasks planned for the following week, and individual issues and problems.

Each week, you read and analyze the logs; examine the technical content of the work done to date; examine the technical progress metrics; consider the QA results; reassess the potential project risks; and take corrective action if necessary.