SE 4485: Software Engineering Projects

Fall 2025

Project Management Plan

Group Number	3
Project Title	Create VSCode Extensions Update Utilities
Sponsoring Company	Everfox
Sponsor(s)	Dustin Endres
Students	1. Brice Duke 2. Alan Nguyen 3. Maximilian Do 4. Chase Uherek 5. Jose Saucedo 6. Dustin Nguyen

ABSTRACT

- This Software Project Management Plan (SPMP) will define how the team will plan and deliver two Linux utilities that allow VSCode extension updates in an airgapped environment. The first utility will download the latest VSIX packages for a list of extensions, compiles and zips an archive ready to be transferred. The second utility will download the zip file, unpack to the correct VSCode extension directory. Both utilities will log their activity to syslog. The document shows project organization, lifecycle model, risk analysis, resource needs, deliverables, schedule, and governance mechanisms, aligned with selected IEEE/ISO standards.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

INTRODUCTION

This document is intended to provide a clear plan and roadmap for how the team will manage themselves throughout the duration of the project, as well as the scope of the project to be delivered at different milestones. It will also present possible risks to be posed to the project and possible mitigation/prevention strategies to ensure a smooth development process. The product being created is a VSCode extension updater designed for use in airgapped environments. The entire process of updating extensions will not be covered in the scope of this product; instead, it will focus on packing and unpacking the extensions, logging its changes and following a YAML configuration file to determine the extensions to update and the target directories to write to. The document will begin with a description of the project organization, or the way in which the development team is organized. Following that, there will be descriptions on the lifecycle model that the team will use, a risk analysis section, and the software and hardware requirements, both functional and non-functional. The document will then outline the scope of the deliverables and their schedule, the ways in which the project documents will be produced and the monitoring and reporting mechanisms to track them, and lastly the professional standards the team will follow throughout the duration of the project.

PROJECT ORGANIZATION

The team will include six contributors, all of whom are tasked with working on all phases of the project's lifecycle. Their specific responsibilities will be determined later on as the requirements and specifications are fleshed out. The team members are:

1. Brice Duke – specializes in web application development, focusing on web application performance, security, and distributed systems. Brice will also act as group leader for the project.
2. Alan Nguyen – specializes in NodeJS, and is experienced with making bot applications using Discord SDKs.
3. Maximilian Do – specializes in Java and C++, has experience in quality assurance.
4. Chase Uherek – specializes in game development, mainly modifying games, as well as web development and databases.
5. Jose Saucedo – specializes in web development and hardware-level engineering.
6. Dustin Nguyen – specializes in game development.

Specific responsibilities will be determined by matching as closely as possible the area of expertise of the team member with the skills most needed to successfully complete the task. There are no specific tasks yet created, but the six team members should have very similar roles and will all work together at each phase of the lifecycle in order to ensure all major decisions are made with a maximally diverse set of viewpoints.

LIFECYCLE MODEL USED

- For this project, our team will incorporate an incremental and iterative lifecycle model. This allows us to deliver two utilities in manageable stages which ensures that core functionality is available early while adjustments and additional features are added incrementally.
- Rationale:
    - Small, well defined scope - The project contains two related but distinct utilities (the packaging utility and the installation utility). The incremental approach enables us to complete and validate each tool independently while ensuring they integrate smoothly.
    - Flexibility with requirements - The sponsor may refine expectations after seeing initial results, so an iterative lifecycle allows feedback-driven improvements without major rework.
    - Risk management - By producing working software early, risks such as version incompatibility and corrupted updates are discovered sooner and mitigated incrementally rather than at the end of development.

RISK ANALYSIS

- Project's scope may become too broad (Medium Risk, Medium Likelihood)
    - Lack of requirements engineering
    - Gold plating
        - Frequent communication with the sponsor to discuss required functionalities via email or Zoom
        - Proper and detailed requirements identification to act as a framework during development, avoid changes or additions unless necessary

Rationale: Team may get sidetracked on addressing concerns or Quality of Life features instead of developing core functionalities

- Deadlines for deliverables agreed upon may not be met (Medium risk, Low Likelihood)
    - Time loss from code rework
    - Team members' personal circumstances preventing them from completing work
        - Frequent communication and check-ins among members, sharing workload when

needed
        - Code reviews whenever changes are made to ensure quality work

Rationale: During development, addressing all items before the deadline can be challenging, as team members' schedules and life circumstances vary, they cannot be fully devoted to the project. As team members can be flexible to work on the project more, it is also reasonable to be realistic with the deadlines. Reworking code or functionalities are also expected when bugs or changes in logic occur.

- Project's major architectures and designs are changed during development (High Risk, Low Likelihood)
    - Changes in the air-gap environment that may require proper adaptations of the product
    - Changes in VSCode that may require proper adaptations of the product
        - Communication with sponsor should that happen to reassess scope, expectations, and deliverables where applicable

Rationale: There could be a scenario where air-gap environment protocols are changed, and the utilities need to operate differently compared to initial specifications. Similarly, changes in how VSCode operates can also affect its expected behaviors. These drastic changes will affect the final deliverables.

- Project requiring expertise outside the team's capabilities (Medium Risk, Low Likelihood)
    - Specialized knowledge identified during development
    - Edge-case interactions that require a thorough understanding of development environments
        - Prioritize finding affordable alternatives that can satisfy the problem
        - Communicate with sponsor to reassess scope and functionalities

Rationale: It is possible that a particular interaction/behavior is required to satisfy the air-gap environment constraints, which is difficult to implement.

SOFTWARE AND HARDWARE RESOURCE REQUIREMENTS

- Management and Documentation
    - Team Internal Communication
        - Discord will be used for the majority of team internal communications due to its ease of access and familiarity among team members. A team group has been set up to facilitate messaging and sharing of untracked static resources when necessary.
    - Sponsor Communication
        - Email will be the primary form of communication with the project sponsor. All team members, the project sponsor, the course professor, and both TAs will be copied on all email communications per project management requirements and to keep all parties involved and informed.
        - Zoom will be used for live communication when needed during update meetings, demonstrations, and presentations per the sponsor's requirements.
    - Documents
        - Google Docs will be used to synchronize editing of text documentation artifacts, allowing team members to collaborate on documents live.
        - Finalized documents will be pushed to a documentation directory in the team's GitHub for review and publication.
        - Changes to a live document after its initial publication will be annotated with comments in Google Docs and those comments will be consolidated into a commit message and pushed to GitHub for review and publication on a

roughly-weekly schedule.

- Software
    - Visual Studio Code will be used as a development and testing target and likely as an environment for developing the application.
    - GitHub will be used for version management of code and corresponding periodic documentation updates.
- Hardware and Testing Environment
    - Team members may use local virtual machines, Windows Subsystem for Linux, or a dedicated system running an x86_64 Linux-based operating system for development.
    - One team member has provided two remote dedicated servers running Ubuntu Server to use as a testing environment. These servers will be made available at all times via SSH and SFTP to all team members with a working internet connection. These servers are being used because they were already being rented; no additional monetary cost will be incurred for their use in this project.

DELIVERABLES AND SCHEDULE

- Project Management Plan - Due 2025-09-12
    - The Project Management Plan outlines how development of the project will proceed from its outset. This is a living document and may be updated as the project progresses and the team's understanding of the project improves.
    - The team will discuss strategy as a whole then each team member will produce a portion of the document for team review based on their experience and preference. Team members will review each other's work and commentary before committing a new version of the document.
    - All future project activities will be documented in and dependent on the Project Management Plan.
    - Initial expected time: 5-10 person-hours.
- Requirements Documentation - Due 2025-09-26
    - The Requirements documentation will include a use case model describing how the system will be used in both graphic and textual form. Non-functional requirements will be listed in textual form.
    - Team members will work together to define the application's requirements, first focusing on the requirements set forth by the project goals, then individually defining requirements for team review as they relate to specific subsystems.
    - The requirements and use cases determined when this document is finalized will be used to inform the architectural structure of the software.
    - Expected time: 5-15 person-hours.
- Architecture Documentation - Due 2025-10-24
    - The Architecture documentation models the component subsystems of the application and the interfaces and interactions that will exist between them and links those subsystems and interactions to the requirements they fulfill.
    - A live discussion and/or whiteboarding session will be used to sketch the basics of the architecture, then team members will split to focus on one or two specific subsystems, communicating with the member(s) responsible for related subsystems to ensure a sensible stable interface exists between the two.
    - Detailed designs of each subsystem will later be created based on the broader system architecture.
    - Expected time: 5-10 person-hours.
- Detailed Design Documentation - Due 2025-11-07
    - Detailed designs will define the structure of the code to be written during development using mockups of user interfaces and diagrams of code structures and interactions.

Designs will be linked both to the architectural components to which they belong and the requirements the design will fulfill.
    - Team members will be assigned to one or more specific subsystems of the architecture. Each member will be responsible for defining the design of their subsystem according to project and architectural requirements and communicating with other members responsible for the same subsystems to ensure a uniform design.
    - Code written during development will adhere to the structure and sequencing described in the detailed design.
    - Expected time: 20-30 person-hours.
- Test Plan - Due 2025-11-21
    - Test plans describe how defects in the application and design will be discovered and will inform the next iteration of application development.
    - Team members will collectively decide on a testing methodology and how testing should be used to inform further development iterations. Individuals assigned to each architectural subsystem are expected to inform the rest of the team of any accommodations needed to support the testing of their subsystem(s).
    - Test cases will be linked to use cases and requirements, describing in detail the relationship between the two.
    - Expected time: 10-20 person-hours.
- Code Development - Due 2025-12-02
    - The code will be developed in line with the design documents and incorporating testing methodologies, resulting after each iteration in an executable artifact that can undergo testing and review processes.
    - Team members will write code to implement the designs they created for their subsystem. Multiple members working on a single subsystem will communicate to avoid duplicate work in translating their designs to implementation.
    - The completion of development across all iterations necessitates that all requirements be fulfilled and testing completed for the team to be confident that the application is ready for a production environment.
    - Expected time: 20-40 person-hours.
- Final Project Presentation Slides - Due 2025-12-02
    - The final presentation slides will incorporate a summary of the project, references and inclusions of information and graphics from the project's documentation, and a demonstration of the final product functioning as if in a production environment. The demonstration will include a cursory view of common use cases and alternative outcomes to show off the product and verify that its final form is acceptable to the project sponsor.
    - Team members will each produce a portion of the presentation - according to those parts of the project they were responsible for - to be used as a basis for the final slideshow. The final slideshow will be reviewed by the team as a whole for correctness and cohesion across its content.
    - Expected time: 5-10 person-hours.
- Final Project Report - Due 2025-12-05
    - The Final Project Report will include all prior documentation, summarizing the entire product development within a single document. Code, executables, and other artifacts will be archived for inclusion in the final submission.
    - Team members will each produce a portion of the report - according to those parts of the project they were responsible for - to be used as a basis for the final report. The final report will be reviewed by the team as a whole for correctness and cohesion across its content.
    - The Final Project Report cannot be completed until every other activity has also completed.

    - Expected time: 2 person-hours.

MONITORING, REPORTING, AND CONTROLLING MECHANISMS

Management reports of different levels of granularity should be produced consistently throughout the lifetime of the project. These include:

1. Weekly meeting logs: Every week (by end of day Saturday) the team should log an attendance report that records who in the team was and was not able to meet with the sponsor and discuss progress.
2. Requirements documentation: By the end of 26 September, the team should have submitted a requirements document that outlines the comprehensive functional and non-functional requirements for the project.
3. Architecture documentation: By the end of 24 October, the team should have submitted an architecture document that outlines the architectural styles, models, technology, and rationales that will be used when building the product.
4. Detailed design documentation: By the end of 7 November, the team should have submitted a document detailing the GUI design, static and dynamic models, and their rationale that will be used when building the product.
5. Test plan: By the end of 21 November, the team should have submitted a document that outlines the specifications-based system level test cases, techniques for generating the tests, and the traceability of said test cases to use cases for use in ensuring the product meets the specified requirements.
6. Midpoint demonstration: On 31 October, the team should present a demonstration for the corporate stakeholders. It should include the progress of the application up to that point, as well as future plans and any other pertinent information.
7. Final project slides and report: By 2 and 5 December respectively, the team should have submitted a comprehensive report of the project aimed to debrief the class on the previously submitted documents, as well as a slideshow to present the information in a more digestible manner.
8. Final sponsor demonstration: On 10 December, the team should present a slideshow, comprehensive report, and the final product to the corporate sponsors aimed to debrief them on the product and its creation.

To elaborate on the rationale behind forming and submitting each of these documents:

Regarding weekly updates, it is important to keep stakeholders up-to-date on our participation, as well as to hold ourselves accountable to ensure we are making progress. For documents 2-5, they are important as they allow the team to establish a consensus on these important factors in the software lifecycle, as well as to provide strict and comprehensive documentation for implementing each phase of the lifecycle. For the midpoint demonstration, the stakeholders will be able to evaluate our progress and course-correct if needed. For the final report and slides, these will allow stakeholders and the class to view the processes and decisions that went into the creation of the product, and will allow the team to get a good understanding of what went well and what could be improved for future products; in effect, a retrospective document.

PROFESSIONAL STANDARDS

- Academic Integrity
    - All code, documents, and diagrams we submit must be our own or properly attributed.
    - No falsification of progress or results.
        - Rationale: Maintain honesty, learning outcomes, and project credibility.

- Meetings & Participation
    - Weekly meetings; attend on time and prepared.

    - Communicate planned or urgent absences as early as possible.
        - Rationale: Keep everyone aligned and accountable.
- Task Ownership & Quality
    - Each task has an owner, a clear due date, and acceptance criteria.
    - Raise risks or blockers quickly.
    - Deliverables must meet agreed quality: functional, tested, documented, and logged to syslog.
        - Rationale: Ensures predictable progress and reliable deliveries.
- Collaboration & Communication
    - Respectful and professional behavior at all times.
    - Use team chat for any updates, issues for technical changes, and PRs for reviews
        - Rationale: Promotes teamwork, transparency, and knowledge sharing.
- Escalation of issues (Appendix A)
    - First issue → Discuss and document.
    - Second issue → Notify the instructor and meet.
    - Third issue → Instructor notified again; possible removal from team with pro-rated grade.
        - Rationale: Provides fairness while protecting team progress.

EVIDENCE THE DOCUMENT HAS BEEN PLACED UNDER CONFIGURATION MANAGEMENT

- See GitHub repository [here] where you can find this document and its version history/changes under the "docs" folder.

ENGINEERING STANDARDS AND MULTIPLE CONSTRAINTS

- IEEE Std 1058-1998: Software Project Management Plans
- PMBOK® Guide: Project Management Body of Knowledge
- IEEE Std 12207: Software Life Cycle Processes
- IEEE Std 15939: Measurement Process
- ISO/IEC/IEEE Std 29148-2018: Systems and Software Engineering

ADDITIONAL REFERENCES

- Larson, E. and Gray, C., 2014. Project Management: The Managerial Process. McGraw Hill
- Humphrey, W.S. and Thomas, W.R., 2010. Reflections on Management: How to Manage Your Software Projects, Your Teams, Your Boss, and Yourself. Pearson Education

Appendix A.

The following provides a professional standards guideline for the team.

Guideline:

On the first occurrence of unacceptable behavior, determine the circumstances involved, resolve the problem, and document the event in the meeting minutes.

On a second occurrence, notify the instructor of the problem. A meeting will be set up to evaluate the situation and resolve the problem.

On a third occurrence, again notify the instructor of the problem. A meeting will be set up to evaluate the situation and resolve the problem. At this point, the team will have the *option* of removing the team

member. If removed, then the team member receives a pro-rated grade based on the number of weeks they have participated in the group.

Examples of unacceptable behavior may include not delivering on time, delivering poor quality work, missing team meetings, being unprepared for team meetings, disrespectful or rude behavior, etc. Reasons such as “too busy” or “I forgot”, or “my dog ate my design model” are unacceptable.

Valid reasons that must be considered include those listed for obtaining an incomplete standing in a course (illness, death in the family, travel for business or academic reasons, etc.)