M0: General Project Requirements

Overview

This document outlines expectations for students' projects and presentations, which will be completed by teams.

Project Requirements

Project development and submission must follow specific guidelines depending on the nature of the project. Deviation from these criteria will result in a grade of ***zero***. All projects must build and run via Python 3.10 / Rust 1.77; in some cases, this may require using an alpha / beta build of frameworks.

General

The following requirements apply to all projects, regardless of type or target platform.

1) Projects must build and run via Python 3.10 / Rust 1.77, even if pre-production framework builds are needed.
2) Project submissions must be entirely self-contained; databases should be submitted in a form utilizing SQLite or another local database file that is installed with the rest of the application / server.
3) Projects as submitted must not depend on external servers unless cleared by instructors in advance of development. There must be strong justification for any exception (e.g., access to proprietary technology).
4) Projects may not use JavaScript, CoffeeScript, or any scripting language other than Python. Project utilizing a web-based interface should consider Python and/or WebAssembly frameworks.

Project Type

Project requirements will vary in part depending on the type of project. Where there is overlap, teams should target the primary objective of the project when considering requirements.

Application

1) Intended for interaction directly with the end-user
2) Primary purpose is to accomplish one or more tasks (vs entertainment)
3) Build specifications will vary depending on platform
4) Has target functionality and aesthetic theme

Games

1) Intended for interaction directly with the end-user
2) Primary purpose is entertainment (vs accomplishing tasks)
3) Build specifications will vary depending on platform
4) Has a story, mechanics, and aesthetic theme

Frameworks

1) Intended for use by engineers (not the end-user)
2) Primary purpose is to facilitate the development of other software
3) Build specifications are ***universal***, regardless of target platform
4) Has an application programming interface (API)

Build Specifications

The build specifications for projects vary depending on project type and, in some cases, target platform. These build specifications must be met for all project milestones.

Frameworks – All Platforms

1) Each build should be prepared as a source distribution **buildable and installable** via a single command.
2) Projects should build / install via pip (e.g., "**`pip install project`**") or via cargo ("**`cargo run`**").
3) The build and installation should build and install documentation in **HTML** format via Sphinx or rustdoc.
4) Frameworks must include at least one sample application to demonstrate functionality as proof of concept.
5) Frameworks must build and run on target platform for Applications or Games, depending on use case.
6) Application(s) must run via a single command from any directory after installation (e.g., "**`appname`**").
7) Any feature not clearly laid out in the project's documentation will not be credited.

Mobile Projects – Apps & Games

- All mobile application code must be written to target an accessible, cross-platform mobile framework.
- All mobile projects should target Android and must run in one of LineageOS 14.5 / Bliss OS 15.8.6 from **`osboxes.org`** or the latest Reptilian OS. Ensure your project is practical in this context before proceeding!
- Each release's source distribution should be buildable via a single command (e.g., "**`python setup.py`**" or "**`cargo run`**") from WSL2. This should result in an Android Package file (APK).

Personal Computers - Applications

1) Applications must build and run within WSL2 on Windows or Ubuntu 22.04 / Mint 21.3 Cinnamon VMs.
2) Each release's source distribution should be **buildable and installable** via a single command (e.g., "**`pip install project`**" or "**`cargo run`**").
3) Application must run via a single command from any directory after installation (e.g., "**`appname`**").

Personal Computers - Games

1) Applications must build and run on Windows 10 October 2022 and Windows 11 October 2023 release.
2) Each release should be buildable via a single command (e.g., "**`python setup.py`**" or "**`cargo run`**") and **also** should be prepared as a application binary installer (e.g., MSI).
3) Application must run via a single command from any directory after installation (e.g., "**`gamename`**").

Servers (or Server Elements)

1) Servers must build and run on the latest WSL2 release (currently Ubuntu 22.04).
2) Each release should be buildable via a single command (e.g., "**`pip install project`**" or "**`cargo run`**").
3) Servers must run via a single command from any directory after installation (e.g., "**`appname`**"). If the server requires initialization the first time, it should include a check to see if this is the first execution.

Team Coordination

The following expectations apply to each team and its members' work and coordination with one another.

Contributions

1) All students must contribute, in approximately equal proportion, to technical aspects of the project, with most work completed in Rust and Python.
2) Serving as a Team Lead does not absolve a team member from the expectation of technical contributions.
3) Instructors of the course may elect to make grade adjustments for unequal contributions.
4) Team members are obligated to complete peer review documentation honestly and with candor.
5) Teams must meet weekly and must keep a meeting log of attendance. This log should be in the repository. This meeting must be distinct from the client meeting.

Code Repository

1) Team must use both a repository system (e.g., GitHub) and issue tracking system (e.g., Trello). The issue tracking system must include the capability to assign tasks to members, and it must be utilized.
2) Repository commits should be made for all code and resources developed, without exception. Git LFS support can be used for very large files.
3) All source code written should be committed by the individual writing the code; code should be merged within the repository system directly to create a clear trail of effort and contribution.
4) Any contribution should be placed in the repository, including planning documents that are developed, in order to facilitate identification of time and effort spent.
5) All code must be merged to "master" or "main" by project milestone deadlines.
6) Projects must have a "Contributions" file in the repository that outlines work on all non-source tasks (which are therefore not reflected in the repository). This should include the name of the person who completed the task, date(s) of work, and a short description of the task itself.

Presentations

1) All students must contribute to the presentation directly – i.e., everyone must present.
2) All students must take part in the planning and preparation of the presentations to receive credit.
3) All team members should have approximately equal speaking time during the presentations.