M3: Production Release (Framework)

Overview

As a team, you will complete and deliver a ***production release***, implementing all the features of your project, as your final deliverable. As the name implies, the production release must be a complete product with all final features integrated, functional, and polished. Implementation of all aspects of every feature must be completed, and user testing should have been completed. The production release is the final version of the application that is delivered to users, and as such it should be usable by a typical developer. The production release should represent approximately 100 hours of effort ***per person***, cumulatively (including time spent on the prototype).

Specification

The deliverable should meet the following requirements.

User Experience

The user feedback system, including any UI and sensory response elements, should complete and polished in the production release. All elements of user control should be completed.

Interface Design

The application programming interface (API) should be usable and polished. Options should be intuitive and consistent. Additionally, any persistent state of the framework must be easily accessible via the interface.

Navigation

The API should be functional, well documented, and polished. There should be no bugs. Use of API should not require significant time for acclimation; calls should be predictable and easy to use. Features should be discoverable, well-explained. They should not overwhelm new users.

User Perception

The API must be intuitive to users. Teams should have solicited user testing from non-team members to ensure general usability. Users should report an enjoyable experience. The release should help users attain stated goals with minimal frustration.

Responsiveness

The API should be responsive and should not waste computing resources. For any operation that requires significant I/O operations, the call should yield the CPU while awaiting response – i.e., there should be no busy wait loops. Additionally, if any call may have a delayed return, there should be a non-blocking option within the API call. Success, failure, and exceptional occurrences must be communicated (by exception, error code, or other mechanism) to the calling procedure. Changes in state should be immediately accessible.

Build Quality

Projects receiving a perfect build score will have no detectable bugs. All content, including art and code systems, should be present and polished.

Robustness

There should be no crashes or glitches in the production release of the project.

Consistency

Except where explicitly by design, the framework should act predictably – for the same call / state, it should yield the same result. When / if behavior is unpredictable, it must be for a clear and compelling reason.

Aesthetic Rigor

No cosmetic issues should be present. Under no circumstances should aesthetic issues cause the framework to be unusable. All assets (e.g., images and sounds) should be present and functional within the context of the project. Assets and their integration within the framework should be polished.

Vertical Features

All features should be complete, with edge cases tested and accounted for, and with failure modes anticipated and handled.

Front-End

For each possible use-case, at all variants of application calls must be implemented within the API of the framework. State changes must be reflected in future calls. This must connect to the *persistent state*.

Persistent State

For each possible use case, the release should make proper use of data structures and update the state of the framework. This must connect to the *front-end* and the *back-end*.

Back-End

For each possible use case, all data processing step must be implemented. This must connect to the *persistent state*.

Submissions

Your submission will be a tgz (gzipped tar) archive including the following:

- Source code and assets for submission
- README file

Source & Assets

Submissions should include all source code, including build configurations and scripts (e.g., setup.py and/or Cargo.toml) and run scripts as necessary. The project should be runnable using an executable command without parameters and should be prepared by build scripts. For server applications, a command to initialize the server state for first run may be included. Submissions should also include all pre-built assets necessary for application function (e.g., pre-built database collections necessary for initial function and/or visual/audio assets).

README File

A README file, either in plaintext or markdown (UTF-8) should be included. The README should have, at a minimum, the following information:

- A description of the project
- One-line command for building and installation
- Link to the repository for the project (e.g., GitHub) [Note: if private, you must add instructors]
- List of packages included in framework
- Executable command(s)