Script for Product Design Overview

**[SCRIPT FOR PRODUCT DESIGN OVERVIEW PRESENTATION]**

**[Design Process Slide]**

**So before running through the product and sub-system designs, I want to provide a refresher on the design process methodology that I am incorporating into this project. This methodology is broken down into the six phases shown in this schematic. Having begun incorporating feedback from project collaborators on requirements and design ideas into my product system designs, I am currently progressing through the ‘Understand’ and ‘Ideate’ phases to develop the framework of my product design before translating that software-application/architecture agnostic product design into working preliminary designs that will be tested in the next step itemized as the ‘Evaluate’ step.**

**[Project Objectives 1]**

**SKIP THE FOLLOWING FOR PRESENTATION TO CRIOS GROUP**

**The next thing I wanted to briefly recap the project’s objectives. As a refresher, the product objectives are broken up into two categories,first, SMART goals that are specific, measurable, achieveable, and time-insensitive that will serve as a baseline for end project goals, and secondly, stretch goals that add complexity or introduce new features to product design that require additional research, time, and collaboration.**

**The Smart goals are to**

1. **Develop portable units of software (ideally containers) that can be leveraged to build, and run computationally inexpensive MITgcm model problems with installation of MITgcm, ECCO libraries and their dependencies and without mpi libraries**

1. **Configure support for these units of software on the most frequently used computing platforms, architectures, and operating systems according to CRIOS group users**

1. **Document the software products in code repository and image registry and present results to MITgcm-ECCO master repository managers**

**[Project Objectives 2]**

**SKIP THE FOLLOWING FOR PRESENTATION TO CRIOS GROUP**

**The Stretch Goals then include the following**

1. **Implement a graphical-user-interface for end users to input MITgcm model problem parameters and specifications ( such as step size, spatial and temporal domain boundaries, etc) along with desired post-processing and data analysis tasks such as invoking ECCO to render data visualizations and render output on a GUI in the portable unit of software.**
2. **Scale up support for MITgcm model problems and ECCO visualization tasks to enable more computationally expensive configurations**
3. **Add support for ASTE regional domain model problems**

**4. Scale up support for product to run on most-widely adopted computing platforms, architectures, and OS according to general purpose MITgcm community use.**

[Product Design CRIOS Environment]

Depicted in this slide, is a high-level flowchart of my intended CRIOS environment release product design that addresses the SMART objectives mentioned earlier. The first two steps necessary to run the containers involve the end user initializing the singularity containerization module on their host machine followed by pulling the containers down from the public image registry. Once the containers are pulled down to the end user's host machine, the end user will run a shell script that initializes the automation of the MITgcm build and run workflow that begins with running a single container. This container will prompt the end user to select a model problem and input desired configurations through a GUI. Once this information is stored locally, a job scheduling script will be built to schedule the compilation and execution of the model problem. Then a second container will be initialized. For load balancing purposes, multiple containers will be run in the workflow to process different tasks. Through this, the first container will relay the job script to the second container through a TCP port and terminate. The second container will then submit the script to Slurm to initialize job execution on compute nodes and will compile the model into an executable. From here, the executable will be run and output will be written out to a file in a specified directory on the end user's host machine. The user will then be prompted to run another model problem or terminate the container and quit the product.

[Product Design MITgcm-ECCO Community Release]

Transitioning to the next slide, we have a similar flowchart that depicts the intended product design of a release intended for general end users of MITgcm-ECCO communitiy software with the stretch objectives accounted for in the elements bounded in red. As shown in the schematic the design is identical up to the stage where a third container is initialized following the communication of the job submission script from the first to second container. This third container will receive the model output from the second container and then prompt the user to input or select desired data-processing and analysis methods on the output data. From here, the container will invoke the ecco library and perform these post-processing and data analysis tasks in the background and both render the output on the GUI for the end user to visualize, and write the output out to a file in a directory specified on the end user's host machine. This design will also be comprised of recurring prompts to run additional model problems. The next step will involve developing class diagrams and sub-system flowcharts for the automation tasks and incorporate those into the high-level product design.