Virtual Development Environment in a Box

Students: Dylan McDougall (dmcdougall2019@my.fit.edu)

and Ian Orzel (iorzel2019@my.fit.edu)

Faculty: Ryan Stansifer (ryan@fit.edu)

Client: Ryan Stansifer (ryan@fit.edu)

Progress of current Milestone:

Task	Completion	Ian	Dylan	Todo
Compare and Select Technical Tools	100%	66%	33%	
“Hello World” Demos	100%	33%	66%	
Resolve Technical Challenges	100%	66%	33%	
Compare and Select Collaboration Tools	100%	55%	45%	
Requirement Document	100%	50%	50%	
Design Document	100%	10%	90%	
Test Plan	100%	90%	10%	

Discussion:

- **Compare and Select Technical Tools**:
    - For user interaction, we considered technical tools to support either a command line interface or a graphical user interface. For graphical user interfaces, we considered using Node.js and PyGUI, but we decided that this project would be more effective as a command line interface. We first planned to use python as a command line interface, but we quickly realized that this would prove inconvenient for the user as they would have to start every command with

"python" and then a path to a file. Thus, we decided to create the command line interface using shell and batch scripts (depending on the operating system).

- For emulating virtual environments, we considered using QEMU, Docker, and VirtualBox. QEMU was the only viable option for our project as it is the only one that allows us to emulate a wide variety of architectures with little hassle. Docker and VirtualBox were not viable for this project as they are not capable of emulation, only virtualization.
- We have decided to use a web server for container distribution as it would be more reliable and would put less work on the clients.

Tool	Advantages	Disadvantages
Web Server	A web server would be very convenient to use in order to obtain containers. This option would be a lot more reliable for obtaining containers.	This would take a lot of time to set up. The server would require upkeep in order to stay up.
Local File Transfer	This would require minimal effort from us to set up. This requires no upkeep other than professors sharing files with students.	This would require more effort from students and faculty to set up and use.

- **"Hello World" Demos**:
  - For the containerization demo, I installed Debian 4 on a SPARC virtual machine. Once Debian was installed, I booted into it, logged into the shell, and ran 'echo Hello, World!' from inside the container.
  - For the container sharing demo, I transferred a pre-built qemu image from one computer to another to ensure that these images can be easily transferred between

computers. I took the .qcow2 file along with a json file configuration that contains command line arguments. Once I transferred these files to another computer, I started using the related commands. I ensured that the image started and worked the same way as it did on the previous computer.

    - For the development tools demo, I installed gcc onto the previously mentioned Debian SPARC virtual machine. I wrote a simple 'Hello World' program in C and compiled it using the virtual machine's gcc compiler. After it successfully compiled, I ran the resulting SPARC binary.

- **Resolve Technical Challenges**:
    - QEMU offers various different 'systems' for emulation. For example, there is an x86_64 system, a SPARC system, an ARM system, etc. These systems all have the same command line interface so commands between them are interchangeable. The qemu-img command can be used to create virtual hard disk images of different sizes using standard formats (qcow, raw, vdi, etc.). In order to boot from a hard disk, one must point QEMU to the virtual hard disk and use command line flags to configure the machine. These flags will be stored in a configuration file along with the hard disk image as described in the design document. In order to install an operating system onto a QEMU virtual machine, you must do all the previously mentioned things as well as use the -cdrom from to give it an installation ISO to boot off of and the -boot flag to tell it to boot from the installation disk. QEMU also provides a -nographic flag, which allows the system to run within the terminal.

- The compiler theory course requires an environment that has support for basic developer tools, including gcc, gdb, and malloc. It also needs to contain the ability to compile and run Java programs (thus must have the JVM set up). Another important aspect of this image is that it must be set up in SPARC architecture so that SPARC binaries can be run natively on it.
- In order to support this on multiple platforms, we first need to create installers or installation guides that work for every platform that a student can use (Windows, MacOS, and Debian). To do this, we would need to create exe, pkg, and deb files that will install all of the proper tools.

  Next, we need to talk about the CLI for the tool. To support the CLI for all platforms, it needs to be created in a shell script and a batch script. Thus, we need to be able to create a program using both of these. Luckily, all other features of this program will be written in Python, so it will be automatically supported by all platforms assuming that they have Python installed.

- **Compare and Select Collaboration Tools**:
  - For software development we will be using Git for version control, GitHub for sharing our code, and Visual Studio Code for programming. Git is a given since it is an industry standard, and both of us are familiar with GitHub so it was only natural that we use it for collaboration. Visual Studio Code has good support for Python, which we will primarily be using, as well as shell scripting languages. Here are a few other options we considered:

Editor	Python Support	CMD Support	Bash Support	JSON Support	Not defunct
VSCode	Yes	Yes	Yes	Yes	Yes
Atom	Yes	No	No	Yes	No
PyCharm	Yes	No	No	No	Yes

- We have decided to mostly use Latex and Google Docs for this project. We will use latex for a lot of the documents, but we will use Google Docs for the presentations and certain documents where it is more convenient.

Tool	Advantages	Disadvantages
Google Docs	Convenient for multiple people to use at once.	Has less functionality than other options.
Microsoft Office	Requires minimal technical knowledge. Has an expanse of features.	Difficult to use for multiple people at once.
Latex	Has huge functionality, more so than any of the others. Easy to use with two people at once.	Requires technical knowledge in advance to use.

- We decided to use Discord for communication due to our large experience with the platform.

Tool	Advantages	Disadvantages
Slack	Has a lot of expansive features for many types of communication.	Minimal experience with this tool.
Discord	We are extremely experienced with Discord.	Is typically associated with gaming rather than technical communication.

Microsoft Teams	Has a lot of expansive features for many types of communication.	No experience with this tool.

- For the task calendar we decided between Trello and Jira. Dylan is more familiar with Trello than with Jira, so we decided that we will use Trello.

	Advantages	Disadvantages
Trello	Intuitive Interface Familiar to all members of the project team Good for small projects	Not as feature-rich as Jira Too simple for large projects
Jira	Provides more functionality Good for large projects	More difficult to learn Cumbersome for small projects

- **Requirement Document**: We constructed a requirement document that lists the various requirements for the different parts of the project. We have included functional, performance, and logical types of requirements.
- **Design Document**: This document describes the overall architecture of the project and provides an appropriate UML diagram. It goes into detail about how various aspects of the system will function and provides proper definitions for them, such as the usage of containers, repositories, etc. It also provides a mock-up for the command line interface.
- **Test Plan**: This document features a list of various tests that we plan to use in order to ensure that the system we have constructed is working properly. These tests test all of the included requirements to ensure that each of them are satisfied by the system.

Member Discussion:

- **Ian Orzel**: For this milestone, I compared technical tools for user interfacing and for container sharing. Then, I performed a hello world demo of sharing containers. I also resolved the technical issues of what Dr. Stansifer needed in an image for Compiler Theory and with how to support multiple platforms. I also compared tools for documentation and messaging. Finally, I wrote half of the requirement document as well as the entire test document. I proofed and made small changes to the design document.
- **Dylan McDougall**: For this milestone I compared technical tools for emulating virtual environments and created a hello world demo for using QEMU, which is the tool I chose for this aspect of the project. I decided that we would be using Git and GitHub for version control and code sharing, and Visual Studio Code for development. I wrote half of the requirements document as well as the entire design document, and suggested minute changes to Ian's test document.

Next Milestone Matrix:

Task	Ian	Dylan
Basic Qemu Image for Compiler Theory	0%	100%
Run Commands and Provide/Receive Standard Output/Input	50%	50%
Import and Export Files from an Image	75%	25%

Discussion of Planned Tasks for Next Milestone:

- **Basic Qemu Image for Compiler Theory**: The purpose of this task is to create a qemu image that satisfies all of the requirements needed for the Compiler Theory course.
- **Run Commands and Provide/Receive Standard Output/Input**: The Python program should be able to take a container and start and stop it. Then, it should be able to run shell

commands on the container. Finally, the program should be able to provide standard input and obtain standard output.

- **Import and Export Files from an Image**: The Python program should be able to transfer files from the local file system to the file system of the container. The Python program should be able to transfer files from the file system of the container to the local file system.

Dates of Meeting with Client: (See Faculty Meeting Times)

Client Feedback for Milestone: (See Faculty Feedback)

Dates of Meeting with Faculty:

- September 14th
- September 28th
- October 5th

FacultyFeedback for Milestone:

- **Compare and Select Technical Tools**:
- **“Hello World” Demos**:
- **Resolve Technical Challenges**:
- **Compare and Select Collaboration Tools**:
- **Requirement Document**:
- **Design Document**:
- **Test Plan**:

Faculty Advisor Signature: _____ Date: _____

Score for each member:

Ian	0	0.5	1	1.5	2	2.5	3	3.5	4	4.5	5	5.5	6	6.5	7	7.5	8	8.5	9	9.5	10
Dylan	0	0.5	1	1.5	2	2.5	3	3.5	4	4.5	5	5.5	6	6.5	7	7.5	8	8.5	9	9.5	10

Faculty Advisor Signature: _____ Date: _____