Code Workstation

Status Report 3

10/7/2015 - 10/29/2015

https://github.com/lrickard/CodeWorkstation/wiki

Lukas Rickard - Project Leader - l_rickard@u.pacific.edu
Jake MacMillan - Integrations Lead - j_macmillan1@u.pacific.edu

Project Schedule

Current Schedule

9/11/2015 - 9/23/2015 - Create outlines for all documents
9/11/2015 - 9/23/2015 - Figure out likely licensing
9/11/2015 - 9/23/2015 - Have considerable content in Requirements Analysis Document
9/11/2015 - 10/7/2015 - Figure out some likely APIs and Libraries
9/24/2015 - 10/7/2015 - Test out one of the APIs or Libraries
9/24/2015 - 10/28/2015 - Begin sketching plan of attack
9/24/2015 - 10/28/2015 - Create Content for Design Document
9/24/2015 - 12/2/2014 - Gather data from peers on which features would be most appreciated
10/8/2015 - 11/13/2015 - Create draft of Design Document
10/8/2015 - 11/13/2015 - Test remaining likely APIs
10/29/2015 - 11/13/2015 - Begin Final draft for Design document
11/13/2015 - 12/2/2015 - Finish 75% of test plan
11/13/2015 - 12/2/2015 - Revise Requirements Analysis documents based on Design
10/23/2015 - 12/2/2015 - Testing API/Library integration and feature compatibility
10/23/2014 - 12/2/2015 - Create plan of attack for implementation based on Design and integration testing

Code Workstation Gantt Chart	9/11	9/18	9/25	10/2	10/9	10/16	10/23	10/30	11/6	11/13	11/20	11/27	12/4
Create docs outlines													
Figure out likely licensing													
Begin filling in RAD													
Figure out likely APIs and libraries													
Begin testing APIs and/or libraries													
Begin sketching plan of attack													
Create content for design doc													
Gather data on appreciated features													
Create draft of design doc													
Test remaining likely APIs													
Begin final draft for design doc													
Finish 75% of test plan													
Revise RAD based on design doc													
Testing API/library integration													
Create plan for implementation													

Schedule Changes

We found it difficult to create the design doc without first testing the PyQt API and making proofs of concept. We decided to test the operability of our planned features with PyQt and test a plan of attack to assess the feasibility of our idea and to get an understanding of what classes and methods we need to create. As can be seen above, we stretched back “Testing API” and “Create plan for implementation” to reflect that we have started on it. We have also pushed back our planned finish time for the design document.

Current Status

We have each read a great amount on the Qt library. We are testing our features with PyQt and we think we can create all of our desired UI features by mixing existing Qt features. We are reasonably confident in our planned approach, but still need to dig into the details of implementation. We have created code for testing features and importing ‘.ui’ files’ but we still do not have anything we would like to commit to our repository or submit as actual project code. We are still not sure if Qt supports the kind of auto-scaling we want for our project, but we have a plan in case Qt does not. With the remaining time we have we should be able create a full plan of implementation and at least start a test plan for when we are actually implementing.

Plan for 10/30/2015 - 11/11/2015

Start on:

1. Test Plan
2. Redraft RAD

Work on most:

1. Design Document
2. Testing feature integration

Finish:

1. First Draft of Design Document

Challenges and Concerns

It has been a challenge to work on the Design document without sufficient testing, but overall I think we are moving forward. We are concerned that Qt will not provide the scaling features that we want, but we have an idea for a coroutine which will fix box widths as the user scales the different docks. We are concerned about making it look nice and populating new features/UI elements at run time, but this pends more testing. We are hopeful that we can create these dynamic elements and attach universal event handlers to their features. Our idea is to have one event handler for each UI feature in a command box, and have the same set of event handlers for every command box, but have each one send a parameter saying which command box the event belongs to. This sounds doable, but may require naming convention assumptions or some other hackery.

Updated Artifacts

Included with Update

Project Proposal (Unchanged)
Requirements Analysis Document
- Updated Sequence Diagrams

Updated Wiki and Repository (Linked, not included in zip)
- Added Design Document to repository

Status Report 1, 2, and 3 (These will not change)
System Design Document
- System Architecture, Requirements, Integrations, and other stuff
- Beginning of Class Diagrams

NOT Included in Update

Test Plan - Not started
Implementation Code - Nothing exists we want to share