# Kaleidoscope

Iteration 1 Progress Report

2019-03-28, version 1.0

## Document Revision History

**Rev. 1.0 2019-03-28: Initial Version**

## Summary

Starting out, we hoped to have a functioning program where the user could use the main application, create presets, open and close the overlay, and use the notepad widget. Coming to the end of iteration 1, we were only able to get the UI open with little to no functionality preset and widgets wise. The model, view, and controller components have yet to interact with each other. We were able to get a program to open based on keyboard shortcuts, but we don’t have an overlay to implement that with. As for widgets, we only have the notepad widget, but again, no overlay for the widget to be on.

## 

## Questions

1. What were the main difficulties so far?

For implementing the MVC architecture, it has proved difficult thus far to bind the software components together. At this point, actions taken in the View are registered and handled by the Controller, but the Controller still needs to be bound to the Model. It took a lot of time to figure out how to properly use event handlers so that Controller functions could be invoked.

For the UI, there has been some difficulties trying create the UI we wanted and correctly formatting everything. So far, nothing has been implemented with the model, so buttons that involve the model to work, do not work. The code for the UI originated on a Mac laptop. When opening up the application on a Windows, certain components wouldn’t align or looked differently, so we would have to compare the UI on a Mac and a Windows side-by-side.

As for background, when trying to get the key logging to work for the overlay (pressing a key combination to open or close the overlay) was difficult because throughout the process, the program would freeze or crash the computer.

1. Were there any features you did not implement as planned, and why? Are you pushing some features to later iterations, and if so, why?

With the MVC architecture, the original plan was to create object classes for the Model, View, and Controller. This approach was taken with the Model and Controller, but was unnecessary or could not work for the View. We were unsure of how to reference a created View object from an HTML file. Instead, a file was created for the View with static functions that act as event listeners. When an element is clicked, the event listener sends a flag to an event handler on the Controller side, which executes the required logic.

With the Background Process, the original goal was to allow our program to work across Operating Systems, but when implementing the keylogger to listen for the key combo, we came across the issue of each operating system registering keys a different way. Since our main focus it to get our application to run on a Windows machine, we had to implement windows virtual keys into our keylogger to allow the user to use any of the special keys in their own personal keyCombo. At this time, we have no expectation to include other operating systems into our program.

We were planning initially to have a mostly functional main application by the end of iteration one, but that turned out to be too ambitious of a goal. It has taken more time to develop than we anticipated, and finding time to pair program has proved crucial and difficult to setup. This is especially true for those of us working on the MVC components as those components are highly dependent on each other. Most of us are new to this tech stack, i.e., Javascript, HTML/CSS, Electron, so there has been a significant learning curve.

1. What tests did you prepare for this iteration, and what are they covering? Did the tests you wrote deviate from your plan? What features are you not testing yet? Did you use any test frameworks, such as JUnit, the Android Monkey, Selenium, etc.?

There are unit tests for the Controller of the MVC architecture. These cover various functions implemented for the Controller, such as handlers for actions taken in the View. These tests create a mock Model object so that the Controller can invoke Model functions in its logic. At this point, these do not deviate from the test plan.

So far, the Model, View, Background Process, and Overlay are not being tested. The Model is mostly implemented, so the tests should be created soon. The View is mainly HTML and CSS so we are unsure what needs to be tested there. For the Background Process, we will need to investigate testing libraries for C. The Overlay has not been implemented yet, so there are no tests.

The testing frameworks and libraries we have decided to use are Mocha, Chai, and Cucumber. Mocha is a Javascript framework for creating unit tests, and Chai is a library providing nice syntax for assertions. Cucumber with WebdriverIO will be used for system testing of the UI. Additionally, the project is hosted in Gitlab, so we have setup CI/CD pipelines which run the code in Docker containers. Currently we have nightly regression tests setup and tests are also run on every push to a branch.