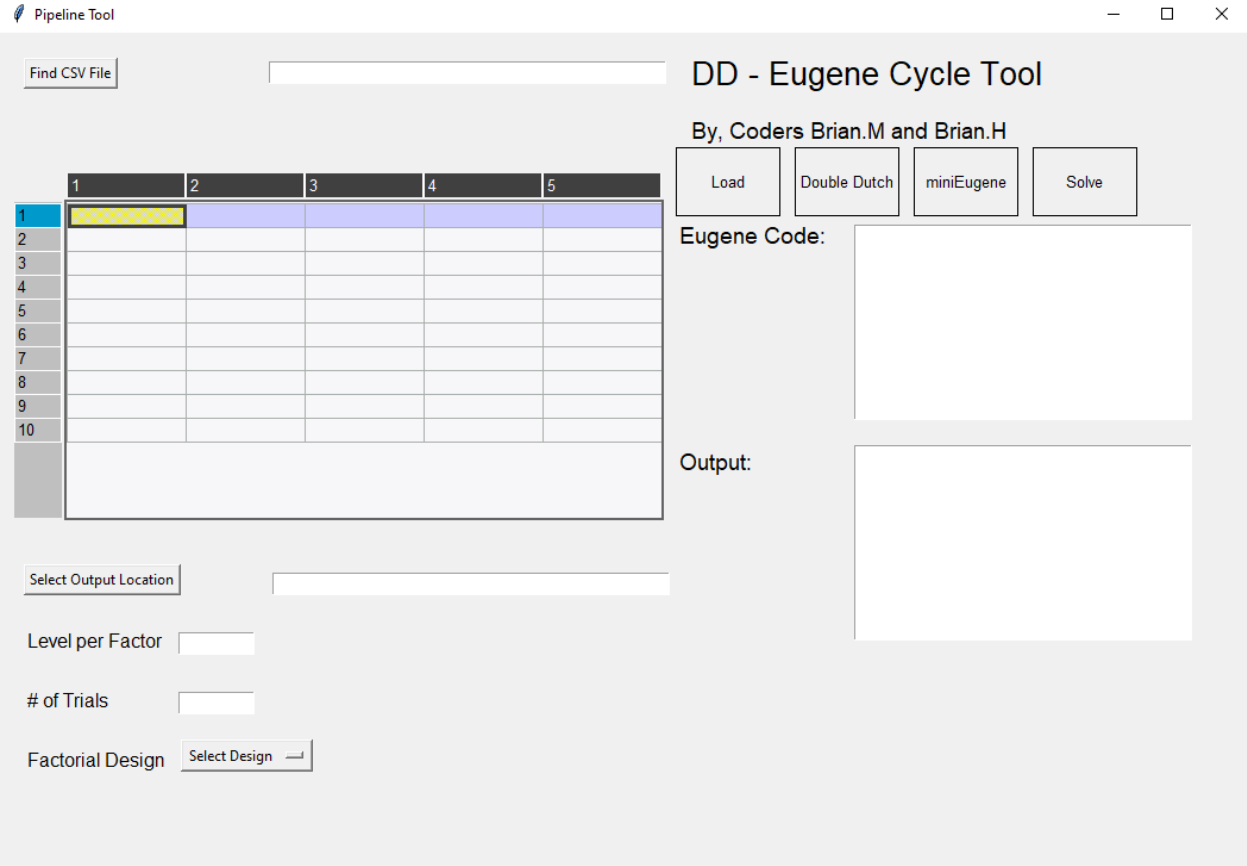
Final Project Report

Brian Mahabir, Landon Kushimi, Brian Jung, Zian Wang

Our project provides a streamlined process for giving users multiple design choices to choose from for Eugene using Double Dutch. We have designed a Graphical User Interface that interacts with Eugene and Double Dutch to give the user a clear comparison graph to showcase which designs might be useful for the user to pick.

The code is written in Python 3.7.9 and uses many preinstalled libraries to get the job done. The first major library the code uses is tkinter which is the main library to create the GUI. Then we have selenium which is arguably the most important library that our code uses. Selenium interfaces with front end graphical user interfaces to operate the code from the front end.

To preference why we ended up using selenium we have to delve a little bit into our trouble shooting problems. Originally, we were going to design the code to interact with Eugene and Double Dutch from the backend to both enter the user's inputs and extract the outputs. However, Eugene ended up being too cumbersome to sift through. We opted to go for miniEugene instead. However, miniEugene could not be run locally due to CORS errors surrounding jQuery and google analytics that we spent too much time trying to figure out. Double Dutch could not be run through a webserver since its server was down. We had to compromise with using selenium in tandem with python requests to get both Eugene and Double Dutch running the way we wanted.



First the user can code Eugene code in our built in text editor. Code mirroring is not supported on python and highlighting each technical word would be too hard so the text editor is in plain text.

The user should first pick a factorial design with level per factor number. The user should then pick a csv file and load it into the GUI with find csv file. The user will hit the Double Dutch button which will run the Double Dutch code locally on their machine. DD will run through the backend which will take an edited input csv file to get design outputs.

The user the will then hit miniEugene button to bring up the miniEugene website. The user can enter in there data and get an output.

From lines 0 – 75 The first part of the code imports the libraries and has a helper function that was taken from stack overflow to instantiate a local instance of javascript code.

From lines 77 – 200 that is the framework for the gui which creates all the buttons, entries, text blocks, etc. The code also has global variables to save paths for using with selenium.

From lines 203 – 213 that is the button function to load the csv file

From 215-225 this code selects a folder to output currently not being used

From 227 – 231 this is the code to open miniEugene on the web

From 233 – 273 this is the code that opens DD, inputs the csv and gets an output

From 277 – 331 the part of the code is to load a user csv edit it to match the format that DD wants and loads the csv into the gui

From 335 – 349 this is residual code for testing and the framework for the solve button

Final 350 – 353 that is the code to instantiate the gui