SOLVER:

Given schematic in KICAD, the solver is responsible to assign position of components on the breadboard. It can operate for a given range of components ( resistors, opamps………………). The users is expected to use our platform with a certain library of components already created. From KICAD, the solver takes in the netlist, an xml script, which has details about the circuit schematic. The solver has been broken into 2 components: kicad.py, to handle kicad specifically; and solver.py which takes a common netlist input to solve for the positions of the different components around the breadboard.

Kicad.py:

The script contains around 70 lines of code and makes use of libraries like argparse, os and xml parser. The code reads into the netlist file generated from kicad and structures it into the netlist data format which is need by the solver. It also creates a list of the values of all the components that are used in the circuit. Finally after structuring the required data, it calls solver.py to initiate the process.

Solver.py:

The script contains around 580 lines of code and makes use of libraries: math, json, copy, random. The solver knows the size of the board that the system uses and follows the following steps to serve the purpose:

1. The solver first allocates row 1 and 2 for GND and 17 and 18 for pwr. This is done because users prefer to have designated section for power and gnd.
2. Priority is given to components like IC, op-amps, relays which are supposed to be placed in the middle of the board. The placement is started from the top of the board in this case. It places such components one by one. Also it takes care of the sizing of such components and the space that it would occupy on the board. For eg: the relay is a 6 pin component but takes up the pace of a 10 pin component
3. After the positioning of the priority components. The different nets of the circuit are allotted the adequate number of rows depending on the size of the net. For eg. net with <5 components gets one row; those having 6-10 get 2 rows and so on. While making such placement, the solver tries to put all the rows connected to the same net together. However, if it is unable to do so, then it jumps to another free space on the breadboard. This is done because users reported that they prefer the rows of the same net to be clogged together.
4. After assigning rows to each net, the solver assigns columns to each component present in each net. For optimality, it makes complete utilization of each row before moving to the next. It also considers the number of rows that have been assigned to a particular net to make sure that there is enough space to make wired connection between the different rows of the same net.
5. All this information about the positioning of the components and the nets is then printed out into a structured json file which is later used by the unity interface to guide the user.