Summer Fellowship Report

On

eSim on Cloud

Submitted by

Darsh Patel
Geddam Saigirishwar Rohit
Darshan Sudake
Faizal Ahmed

Under the guidance of

Prof. Kannan Moudgalya
Chemical Engineering Department
IIT Bombay

May 2020

Acknowledgement

We, the summer interns of the **FOSSEE - eSim Cloud Project** are overwhelmed in all humbleness and gratefulness to acknowledge our deep gratitude to all those who have helped us put our ideas to perfection and have assigned tasks well above the level of simplicity and into something concrete and unique. We wholeheartedly thanks **Prof. Kannan M. Moudgalya** for having faith in us, selecting us to be a part of his valuable project and for constantly motivating us to do better. We thanks **Mr. Nagesh Karmali** and **Ms. Firuza Aibara** for providing us the opportunity to work on this project. We are also very thankful to our mentors for their valuable suggestions. They were and are always there to show us the right track when needed help. With help of their brilliant guidance and encouragement, we all were able to complete our tasks properly and were up to the mark in all the tasks assigned. During the process, we got a chance to see the stronger side of our technical and nontechnical aspects and also strengthen our concepts. Last but not the least, we sincerely thank all our other colleagues working in different projects under **Prof. Kannan M. Moudgalya** for helping us evolve better with their critical advice.

Declaration

We declare that this written submission represents our ideas in our own words and whenever others' ideas or words have been included, We adequately cited and referenced the original sources. We declare that We have properly and accurately acknowledged all sources used in the production of this thesis.

We also declare that We have adhered to all principles of academic honesty and integrity and have not misrepresented or fabricated or falsified any idea/data/fact/source in our submission. We understand that any violation of the above will be a cause for disciplinary action by the Institute and can also evoke penal action from the sources which have not been properly cited or from whom proper permission has not been taken when needed.

Darsh Patel
Geddam Saigirishwar Rohit
Darshan Sudake
Faizal Ahmed

Contents

List of Figures

Chapter 1

Introduction

1.1 Problem Statement

Design and develop a web-based platform to draw different types of Electronic Circuit and Simulate them by providing simulation parameters which will be hosted on the cloud.

1.2 Project Objective

This project aims to provide an easy to use EDA application for students to complement their undergraduate electronic courses and a collaborative tool for authors. Since there is lack of open source web-based circuit simulator application this will fulfill the requirements with easy access and other useful functionalities.

1.3 Project Outcome

The contributor will be able to draw schematic diagram of circuits using drag and drop facility for provided components from the left pane onto the schematic grid. The components on the grid can be connected using virtual wires. The basic ERC check (i.e. Electrical Rule Check) facilitate users to find out if there are any errors. Then he can simulate that circuit by passing parameters under different simulation modes (DC Solver, DC Sweep, Transient analysis, and AC analysis).

1.4 Project Requirements

Following Major Technologies have been used during development.

- Django (v2.2.12)
- React (v16.13.1)
- mxGraph (v4.1.1)
- ngSpice (v31)
- Docker Containers
- MongoDB
- PostgreSQL
- Celery
- Redis
- Nginx

Chapter 2

Project Overview

This system allows the users to draw analog and digital circuits and simulate them. The users have a facility to drag and drop components from the left pane onto the schematic grid on the right pane. The components on the grid are connected using wires. The circuit can then be simulated using the different simulation parameters (DC Solver, DC Sweep, Transient analysis, and AC analysis). The basic ERC check enables the users to find out errors if any. The size of the schematic grid can be changed from A1 to A5 paper sizes along with portrait and landscape modes. The users can also print the circuit or save it in pdf format for documentation purposes.

2.1 Features

The schematic editor is divided into 3 panes. The left pane consists of the **Component List** and a facility to search components. The middle pane consists of the grid on which the components will be dropped and the circuit will be designed. The right pane consists of the grid properties, description of the circuit, and components position. More details are given below.

2.1.1 Grid size and Orientation

The size of the grid can be changed from A1 to A5 and offers Portrait and Landscape mode.

2.1.2 Schematic Description

A text area in which one can write the description about the circuit.

2.1.3 Component Categories

The kicad components are categorized as follows, where each component has **Name, Description, Keywords** and **Datasheet**.

- Analog
- Device
- Triac Thyristor
- Transistor IGBT
- Diode
- Transistor FET
- pspice
- Oscillator
- eSim Sources
- eSim Hybrid
- Motor
- LED
- Transistor BJT
- power
- 4xxx

2.1.4 Searching Component

Rather than going through categories and locating the component symbol, one can also search a component by typing in the textbox given, using the filters like **Name, Keyword, Description, Component Library** and **Prefix**.

2.1.5 Components Position

Using the components position box, one can access and view the circuit which do not fit onto the specified grid size. Its like accessing another page. This situation arises when one has a large circuit and changes the grid from a larger size to a smaller one.

2.1.6 Basic Editor Operations

- Undo
- Redo
- Rotate
- Delete
- Zoom in
- Zoom out
- Clear All
- Default size
- Print Preview

2.1.7 ERC Check

Basic ERC check used to find out if there are any errors in a circuit. For example, if the wires are connected or not.

2.1.8 Generate Netlist

Based on the circuit a ngSpice compatible netlist is generated. User can download the generated netlist for command line simulation with ngSpice. The internal process of generating a netlist is described in the section 3.3.

2.1.9 Simulate

There are four simulation modes as follows

- **DC Solver :** A DC simulation attempts to find a stable DC solution of your circuit.
- **DC Sweep :** A DC Sweep will plot the DC solution of your circuit across different values of a parameter of a circuit element. You can sweep any numerical parameter of any circuit element in your circuit.
- **Transient Analysis :** A Transient analysis does a Time-Domain Simulation of your circuit over a certain period of time.
- **AC Analysis :** AC Analysis does a small signal analysis of your circuit. The input can be any voltage source or current source.

2.1.10 Saving and Re-Opening

The circuits are saved only for an authenticated user and are viewed on the user dashboard. The same can be reopened as well for further modification or simulation.

2.1.11 Export

- **Image Export :** The circuit can be exported as jpeg, png, and svg. This is useful for documenting and printing.
- **JSON :** The circuit can be exported as JSON so as to open it again using the Upload feature.

2.1.12 Sharing

Using the **Share** button one can get sharing URL for the circuit generated by the system; with others using the link. The link can be opened and viewed by anyone. However, to make changes one would need to login and the changes saved will be associated with the new user.

2.1.13 Dashboard

A place where the authenticated user can view the different circuits designed by him/her. Then user can open the saved circuit into the editor by clicking on **Launch in editor**.

2.1.14 Gallery

A set of example projects (circuit design) which can be referred by the users. This is very much useful for the novice users who need to get a feel of the system and the circuit design.

2.1.15 Spice Simulator

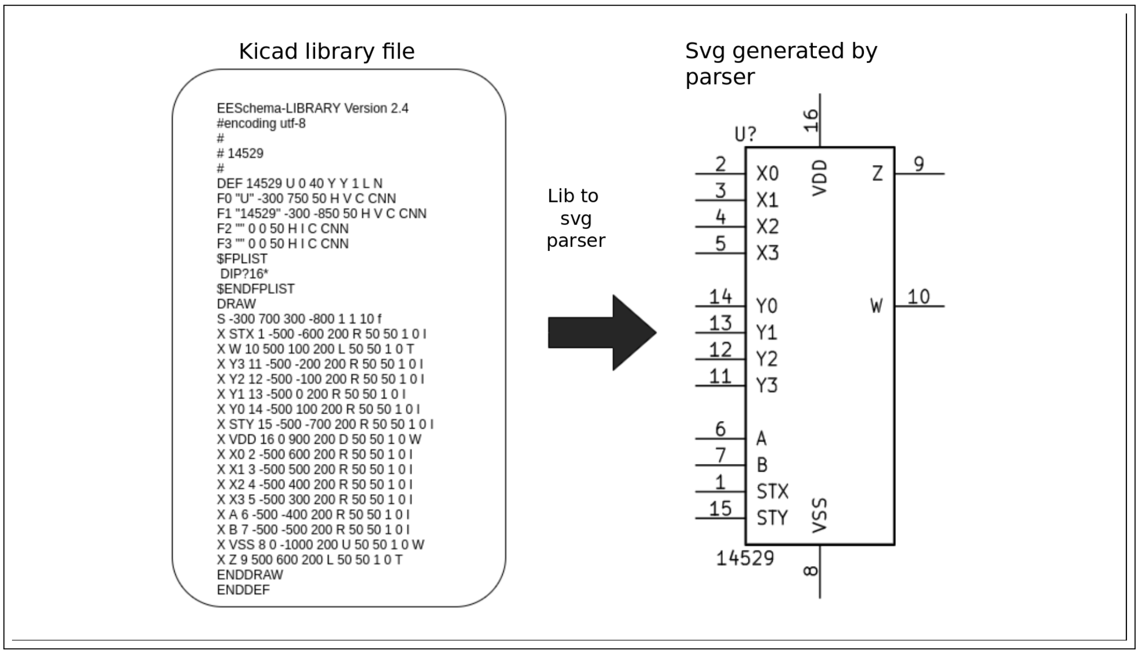
This is a particularly handy feature of this web application, which allows anyone to enter the netlist in the code editor provided and simulate without drawing the circuit. Simulation result window will popup displaying the result.

Chapter 3

eSim Development Flow

3.1 Generating component images from KiCAD symbol library files

The component Symbols used in this web application are generated by parsing the **.lib** and **.dcm** files from kicad symbol library files using an inhouse parser. These components are generated only once and are cached. In the frontend these component symbols are displayed under the component list on the left sidebar of the editor.



```
EESchema-LIBRARY Version 2.4
#encoding utf-8
#
# 14529
#
DEF 14529 U 0 40 Y Y 1 L N
F0 "U" -300 750 50 H V C CNN
F1 "14529" -300 -850 50 H V C CNN
F2 "" 0 0 50 H I C CNN
F3 "" 0 0 50 H I C CNN
$FPLIST
 DIP?16*
$ENDFPLIST
DRAW
S -300 700 300 -800 1 1 10 f
X STX 1 -500 -600 200 R 50 50 1 0 I
X W 10 500 100 200 L 50 50 1 0 T
X Y3 11 -500 -200 200 R 50 50 1 0 I
X Y2 12 -500 -100 200 R 50 50 1 0 I
X Y1 13 -500 0 200 R 50 50 1 0 I
X Y0 14 -500 100 200 R 50 50 1 0 I
X STY 15 -500 -700 200 R 50 50 1 0 I
X VDD 16 0 900 200 D 50 50 1 0 W
X X0 2 -500 600 200 R 50 50 1 0 I
X X1 3 -500 500 200 R 50 50 1 0 I
X X2 4 -500 400 200 R 50 50 1 0 I
X X3 5 -500 300 200 R 50 50 1 0 I
X A 6 -500 -400 200 R 50 50 1 0 I
X B 7 -500 -500 200 R 50 50 1 0 I
X VSS 8 0 -1000 200 U 50 50 1 0 W
X Z 9 500 600 200 L 50 50 1 0 T
ENDDRAW
ENDDEF
```

Figure 3.1: Kicad symbol library to svg generation example

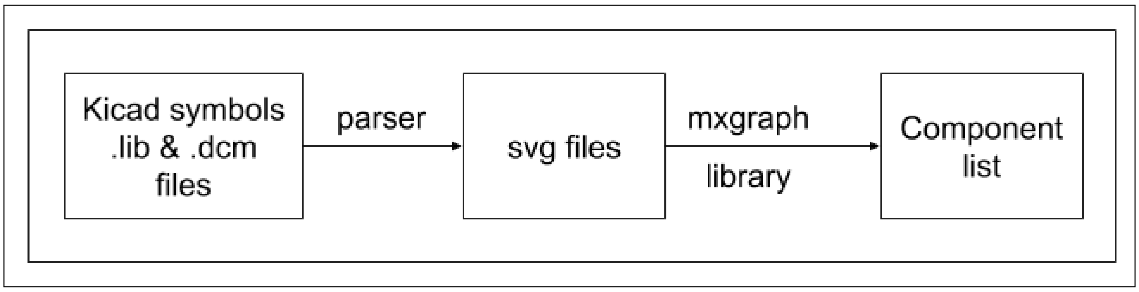


Figure 3.2: Kicad symbol library to svg generation

3.2 Generating XML files

The components from the left pane are dropped onto the schematic grid. By default, the size of the grid is A4, which can be changed from A5 to A1. The components connected by wires are converted to XML format using mxgraph inbuilt function, whenever the circuit is saved by the user. This XML is used to save and re-open the saved circuits.This XML is also used to auto annotate the circuit as well as in performing ERC checks.

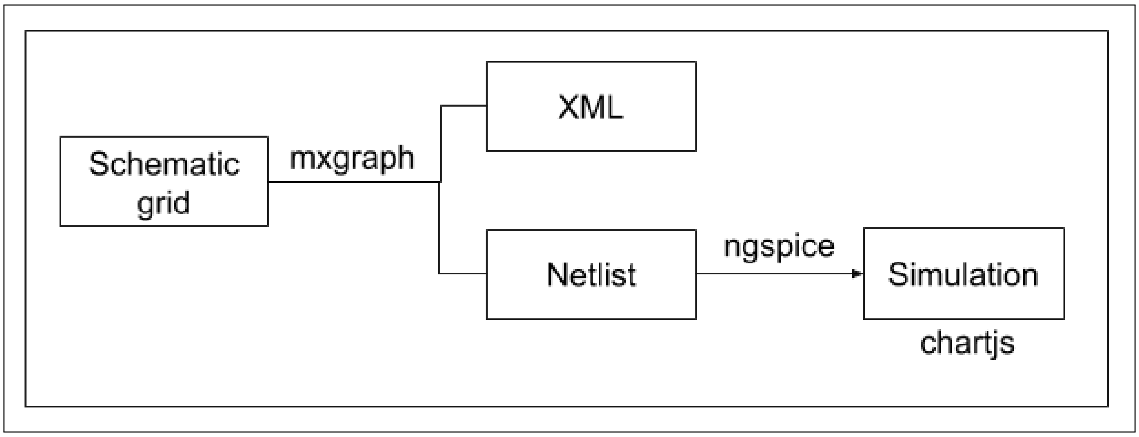


Figure 3.3: Generation of Netlist and Storing XML

3.3 Generating Netlist

The netlist is broken down into seperate sections.

- **Title :** title of the schematic diagram

```
RC Circuit
```

- **Model :** All spice models given by users will be listed here. These are extra parameters which are not delivered with ngspice. They are device manufacturer specific and may be obtained from their web sites or from other sites

  ```
  .model BC546B npn ( IS=7.59E-15 VAF=73.4 .........)
  ```

- **Netlist :** Text description of circuit. It has all components listed with connecting nodes, parameters and spice model (if specified by the user). This is generated with the help of mxgraph object. An example is shown below

  ```
  r1  in out 1k
  c1  out gnd 10u
  v1  in gnd pwl(0m 0 0.5m 5 50m 5 50.5m 0 100m 0)
  Q1 intc intb 0 BC546B
  ```

- **Control Line :** It has all simulation parameters. It is generated depending on the type of simulation and the parameters specified by user

  ```
  .tran 10e-03 100e-03 0e-03    // Transient analysis
  .ac dec 10 10 1Meg            // AC Analysis
  ```

- **Control Block :** All Interactive commands to actually produce output for given schematic.

  ```
  .control
  run
  print all > data.txt
  .endc
  .end
  ```

Using the mxgraph object, a netlist is generated (compatible with ngspice simulator) when the user clicks on the **Simulate** or **Generate Netlist** button. The simulation parameters are supplied by the user based on the simulation type chosen by the user.

3.4 Simulation Output

When the simulate button is clicked the ERC checks are performed and the netlist generated is sent to the backend services where it is kept in a queue. The queue manager used is celery. At backend this netlist is supplied as input to ngspice which outputs a text file with all the coordinates required to plot the graph. This text file is then parsed using an inhouse parser to convert the data in the text file into an organised data structures in JSON format. At the frontend the graph is plotted using the data returned in this JSON.

3.5 JSON format returned by parser

```
{
 total_number_of_tables: <int>,
 isGraph: <bool>,
    data:[
     {
     labels : [ ], x : [ ], y : [ [ ] , [ ] ] ,
    }
    ]
}
```

- **total-number-of-tables :** this property tells how many tables will be present.
- **isGraph :** this property tell if the data is a graph or just a table of data
- data - this is an array which contains one or more objects depending on the input provided to the parser.
- **labels :** this is an array which contains all the labels that have to be present on the graph. Eg. [“time”,”vin”,”vout”].
- **x :** this is an array containing all the x co-ordinates for a set of graphs. Eg. Time on x-axis.this is a linear array as the x coordinates will be same for different set of y coordinates.
- **y :** this is a 2d array. Containing y co-ordinates for different graphs.

Chapter 4

Frontend

4.1 UI Components

4.1.1 Home UI

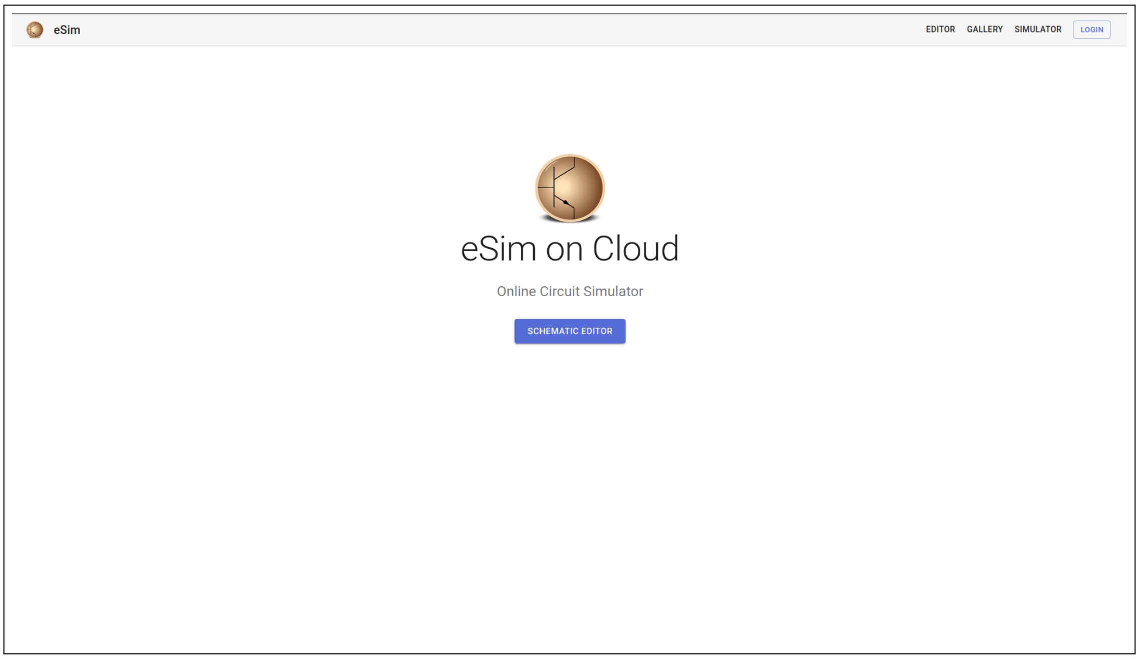


Figure 4.1: Home page of eSim on cloud

Home page of eSim on cloud contains links to various public component of app such as

- Editor
- Gallery
- Simulator
- Login

4.1.2 Schematic Edior UI

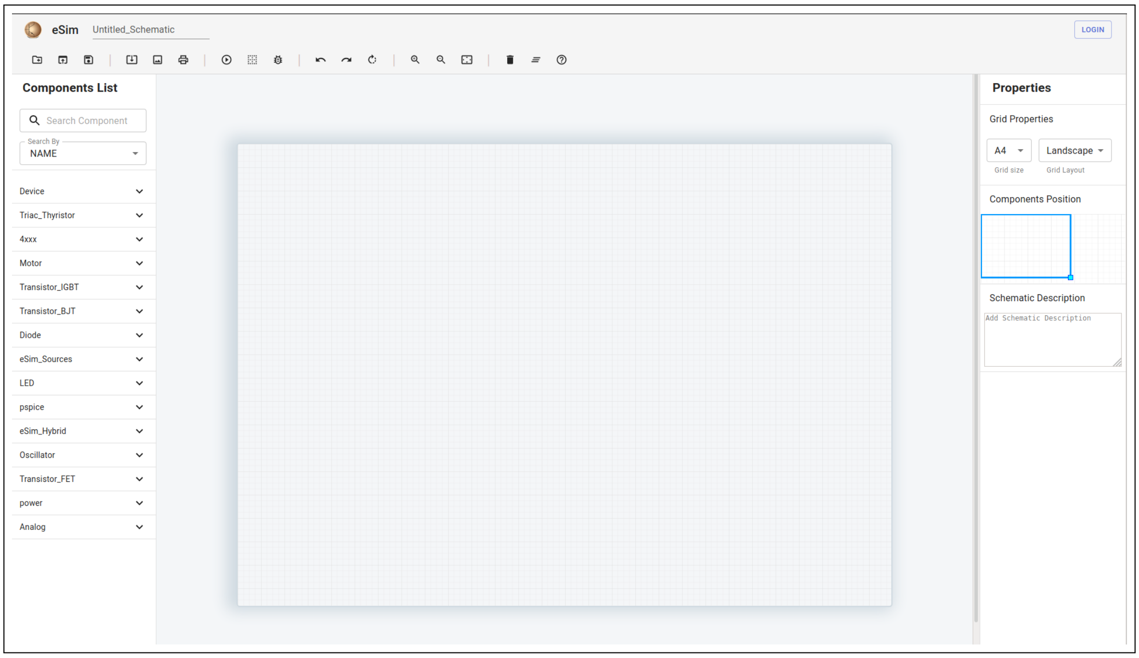


Figure 4.2: Schematic Editor of eSim on cloud

Schematic Editor is where the user can draw his circuits using provided components list on left side pane and simulate the circuit by providing simulation parameters. Editor provide facility to connect the component's terminal using virtual wires. Non-Authenticated user can also draw and simulate the circuit, but for saving and sharing of circuit user has to login. It is mainly divided into four sections

- Toolbar (top)
- Component Sidebar (left)
- Grid (center)
- Properties Sidebar (right)

Various tools in Toolbar are listed below in user guide section

4.1.3 Spice Simulator UI

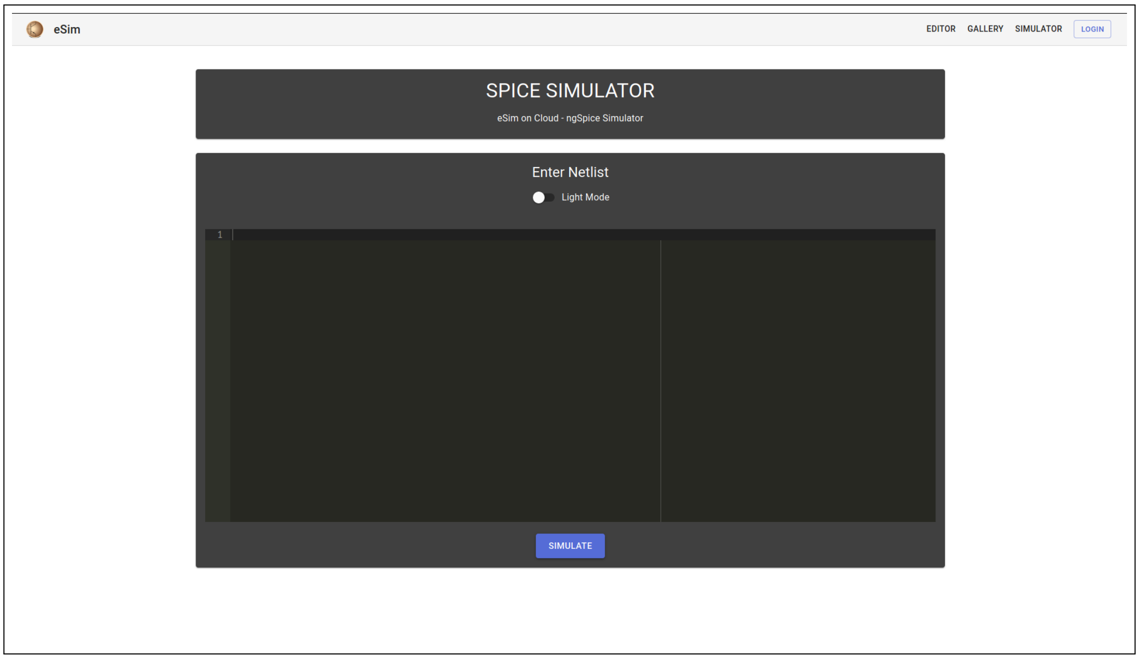
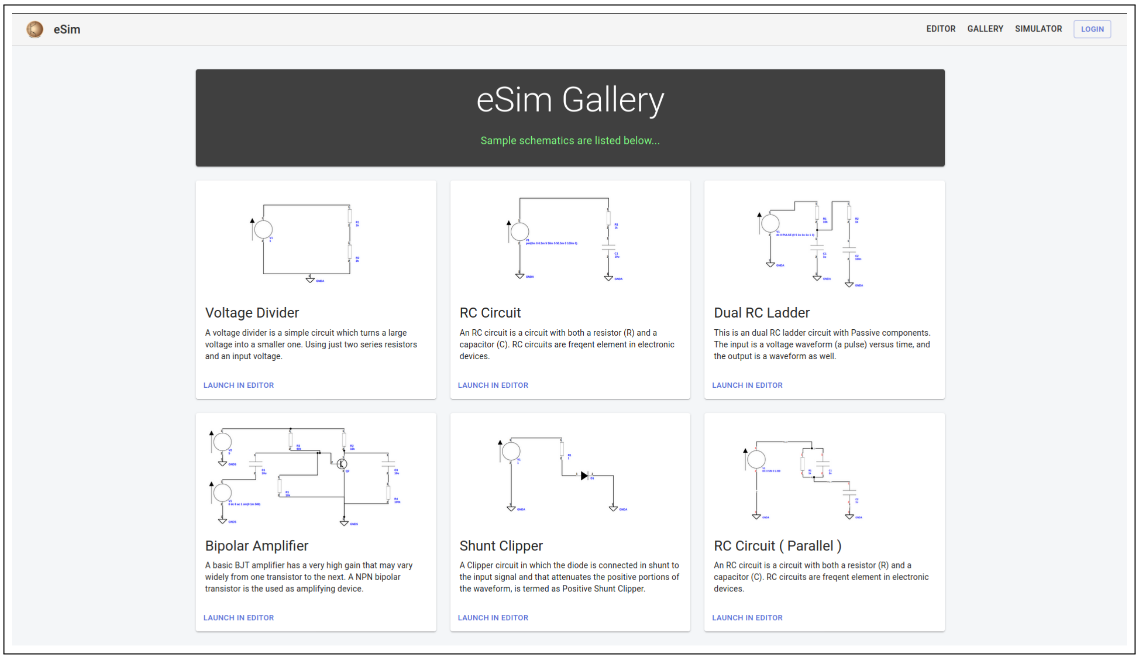


Figure 4.3: Simulator page of eSim on cloud

This is a particularly handy feature where the user can directly Enter/ Type the ngSpice compatible netlist to simulate without the need of drawing circuit. The user has to Enter/ Type the netlist in the code editor provided and click on simulate. User can change theme of code editor using switch on top. The simulation graph or text result appear in a popup window(i.e. Simulation Result Screen).

4.1.4 eSim Gallery UI



eSim Gallery

Sample schematics are listed below...

Voltage Divider

A voltage divider is a simple circuit which turns a large voltage into a smaller one. Using just two series resistors and an input voltage.

LAUNCH IN EDITOR

RC Circuit

An RC circuit is a circuit with both a resistor (R) and a capacitor (C). RC circuits are freqent element in electronic devices.

LAUNCH IN EDITOR

Dual RC Ladder

This is an dual RC ladder circuit with Passive components. The input is a voltage waveform (a pulse) versus time, and the output is a waveform as well.

LAUNCH IN EDITOR

Bipolar Amplifier

A basic BJT amplifier has a very high gain that may vary widely from one transistor to the next. A NPN bipolar transistor is the used as amplifying device.

LAUNCH IN EDITOR

Shunt Clipper

A Clipper circuit in which the diode is connected in shunt to the input signal and that attenuates the positive portions of the waveform, is termed as Positive Shunt Clipper.

LAUNCH IN EDITOR

RC Circuit (Parallel)

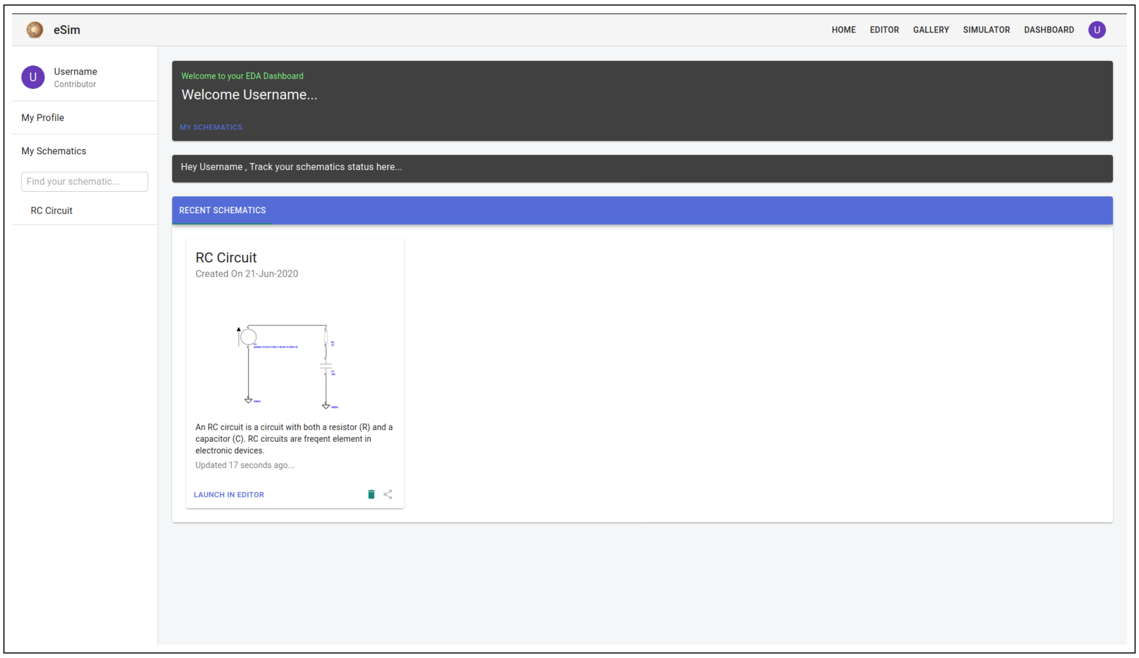
An RC circuit is a circuit with both a resistor (R) and a capacitor (C). RC circuits are freqent element in electronic devices.

LAUNCH IN EDITOR

Figure 4.4: Gallery page of eSim on cloud

This page provides some of the example circuit for the user to try out. User can click on **LAUNCH IN EDITOR** button in card of specific circuit to open that circuit in the schematic editor where the user can modify or simulate it. This is very much useful for the novice users who need to get a feel of the system and the circuit design.

4.1.5 Dashboard UI



Welcome to your EDA Dashboard

Welcome Username...

Hey Username , Track your schematics status here...

RC Circuit

Created On 21-Jun-2020

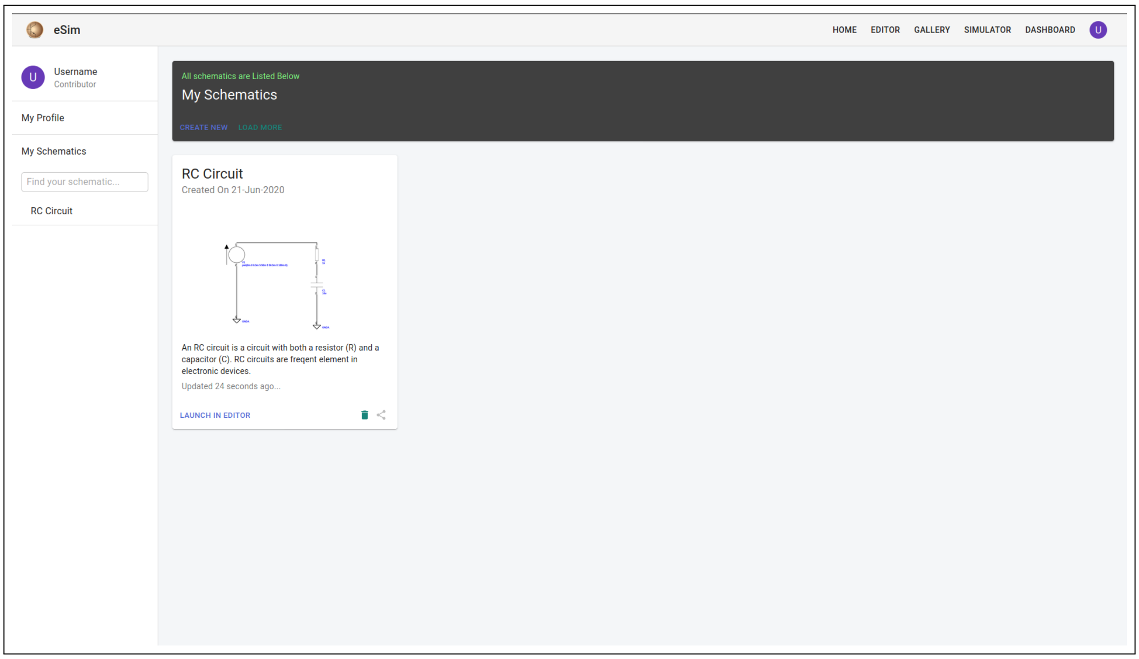
An RC circuit is a circuit with both a resistor (R) and a capacitor (C). RC circuits are freqent element in electronic devices.

Updated 17 seconds ago...

Figure 4.5: Dashboard page of eSim on cloud

This is dashboard home page for authenticated users where under Recent Schematics tab user will get to see recently created circuit.

On each schematic circuit card information about Created Date and Last updated status is displayed. User can open the circuit in editor to modify or simulate. Delete button on each card can be used to delete the unwanted circuit. Share logo displays the sharing status of the schematics.



Username
Contributor

All schematics are Listed Below
My Schematics

RC Circuit
Created On 21-Jun-2020

An RC circuit is a circuit with both a resistor (R) and a capacitor (C). RC circuits are freqent element in electronic devices.
Updated 24 seconds ago...

Figure 4.6: Dashboard My Schematics page of eSim on cloud

In My Schemaics section of dashboard all the schematics specific to user are listed.

4.2 Basic Workflow of eSim on Cloud

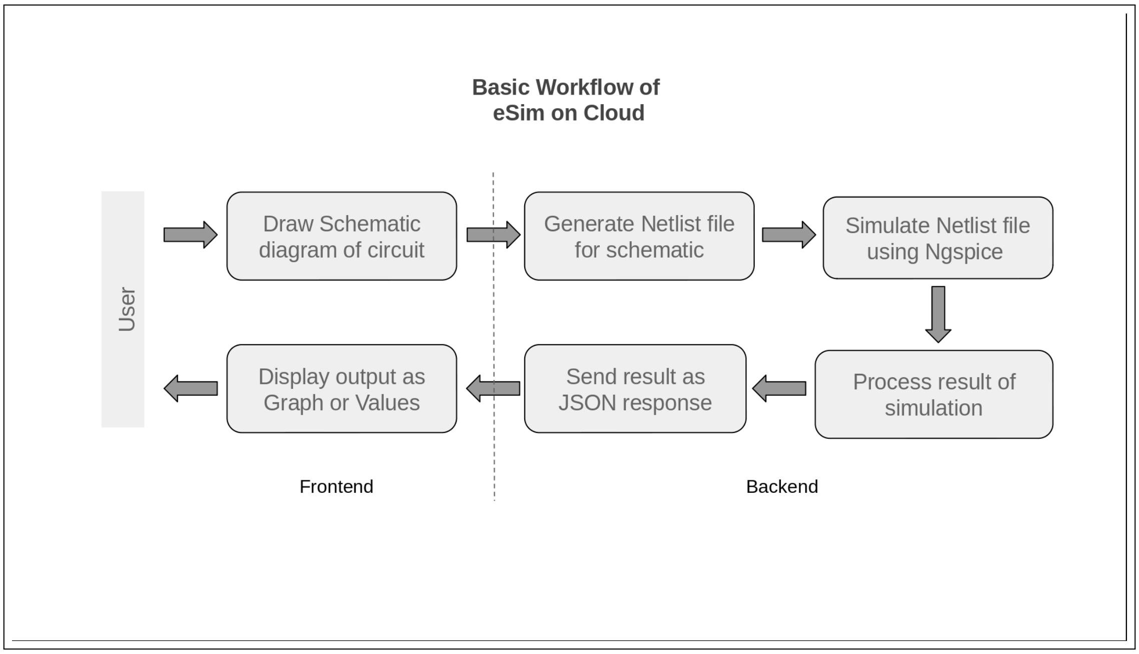

Basic Workflow of
eSim on Cloud
User
Draw Schematic diagram of circuit
Generate Netlist file for schematic
Simulate Netlist file using Ngspice
Process result of simulation
Send result as JSON response
Display output as Graph or Values
Frontend
Backend

Figure 4.7: Basic Workflow of eSim on cloud

Chapter 5

Architecture

5.1 Overview

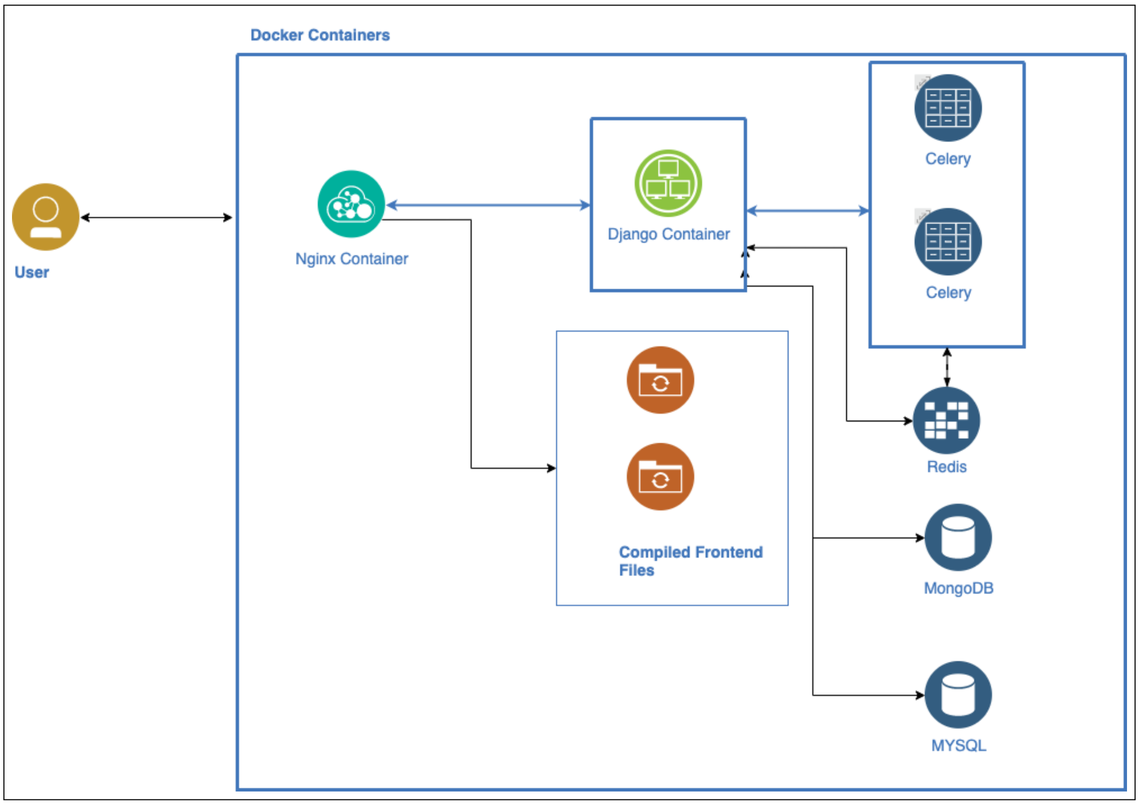


Figure 5.1: Architecture Diagram

The project relies on `docker-compose` [1] to orchestrate docker containers. As described in Figure 5.1, The users query is handled by Niginx[2, 3], and it is routed to the Application Interface (API) Endpoints created by Django Web Framework[4] (discussed in detail in 8) or the Front-End files , the Django Container then connects to Celery[5], Redis[6] and the databases MongoDB[7] and MYSQL[8]. These services have all been individually containerized so as to allow scaling with a single command in future. These services and their role in the project have been described in detail in the subsequent subsections.

5.2 Nginx

Nginx [3, 2] has been utilized for Caching, Load-Balancing, Reverse-Proxying. Essentially, when a user tries to access the project, it is Nginx which is handling the requests. It does the following things:

- Cache Images, Cascading Style Sheet(CSS) files and API requests for faster responses
- Route requests on the same domain but on different Uniform Resource Locator (URL) to the required web server
- Load Balance requests between multiple docker containers using Round-Robin Algorithm [9]

5.3 Django

Django Framework along with Django-Rest-Framework [10] have been utilized to create API endpoints required for the Frontend. Some APIs have also been utilized by the `Arduino Project`. The endpoints have been discussed in detail in 8. `Gunicorn`[11] has been used as a WSGI Server to serve the Django APIs, it can be configured to use multiple threads and workers. The Django Container installs the required python dependencies, and also system dependencies necessary for the python dependencies. The built docker images are also served in the project's home page.

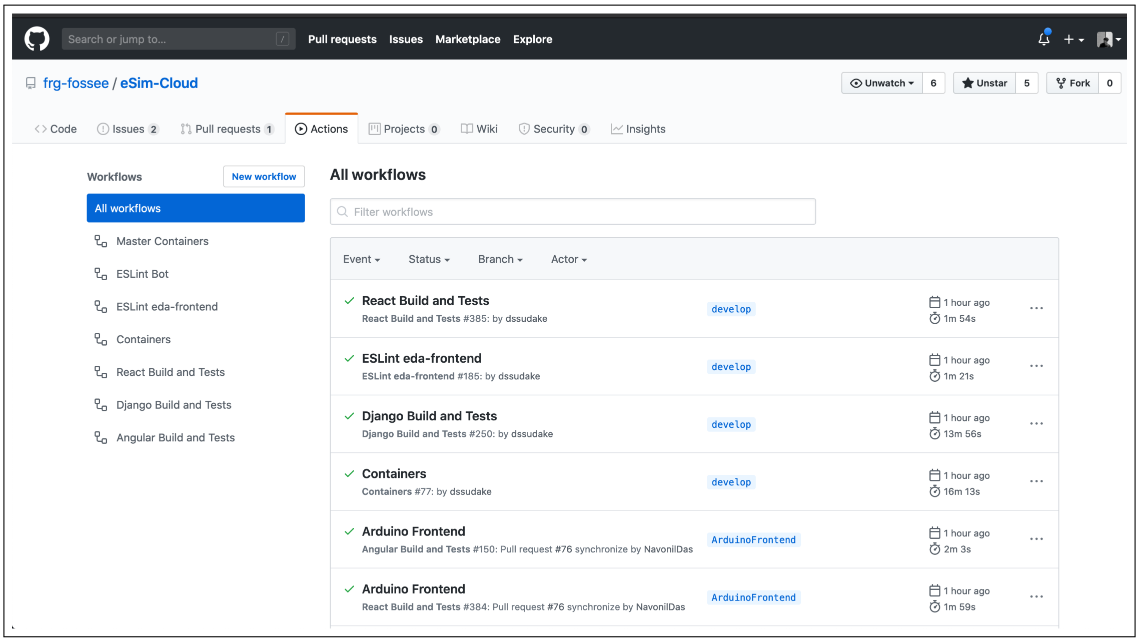
5.4 Celery and Redis

Celery uses the same docker container as Django, but runs the celery daemon with multiple workers to handle processing heavy tasks in the background asynchronously. It is also worth noting that multiple celery containers can 'Discover' each other on the network and share tasks amongst themselves. Celery also requires a Task Queue, for which Redis has been utilized. Redis is essentially a In memory Key-Value Store, this is utilized to store the task details and it's data which is later fetched and processed by a Celery worker.

Chapter 6

DevOps

6.1 Continuous Integration



✓	**React Build and Tests** React Build and Tests #385: by dssudake	develop	1 hour ago 1m 54s
✓	**ESLint eda-frontend** ESLint eda-frontend #185: by dssudake	develop	1 hour ago 1m 21s
✓	**Django Build and Tests** Django Build and Tests #250: by dssudake	develop	1 hour ago 13m 56s
✓	**Containers** Containers #77: by dssudake	develop	1 hour ago 16m 13s
✓	**Arduino Frontend** Angular Build and Tests #150: Pull request #76 synchronize by NavonilDas	ArduinoFrontend	1 hour ago 2m 3s
✓	**Arduino Frontend** React Build and Tests #384: Pull request #76 synchronize by NavonilDas	ArduinoFrontend	1 hour ago 1m 59s

Figure 6.1: Github Workflows

6.1.1 Github Actions - Workflows

Multiple github workflows have been setup to run Linting and Testing operations on various parts of the project, they have been described below:

- eslint bot ensures that the JavaScript files in the project have been linted according to the eslint standards, it even annotates the lines with linting errors
- React Tests - Simple tests to ensure the project compiles successfully and there are not any syntax errors present. Coverage needs to be significantly improved

- PEP8Speaks Bot - It is utilised to ensure that Python code in the repository meets the PEP8 standards
- Django Build - Ensures that the Django container sucessfully builds
- Containers Publish - Builds all the containers defined in the docker-compose configuration and publishes them on github packages as discussed in 6.1.2

6.1.2 Container Images

Pre-Compiled docker images are being published in Github's docker registry once code is merged to the develop or master branch. There are two tags, 'dev' and 'latest', images with the 'dev' tag are built from the develop branch while images with the 'latest' branch are published with code from master branch of the project. Using a published image saves time and resources to build a docker container locally on one's system.

6.2 Deployment

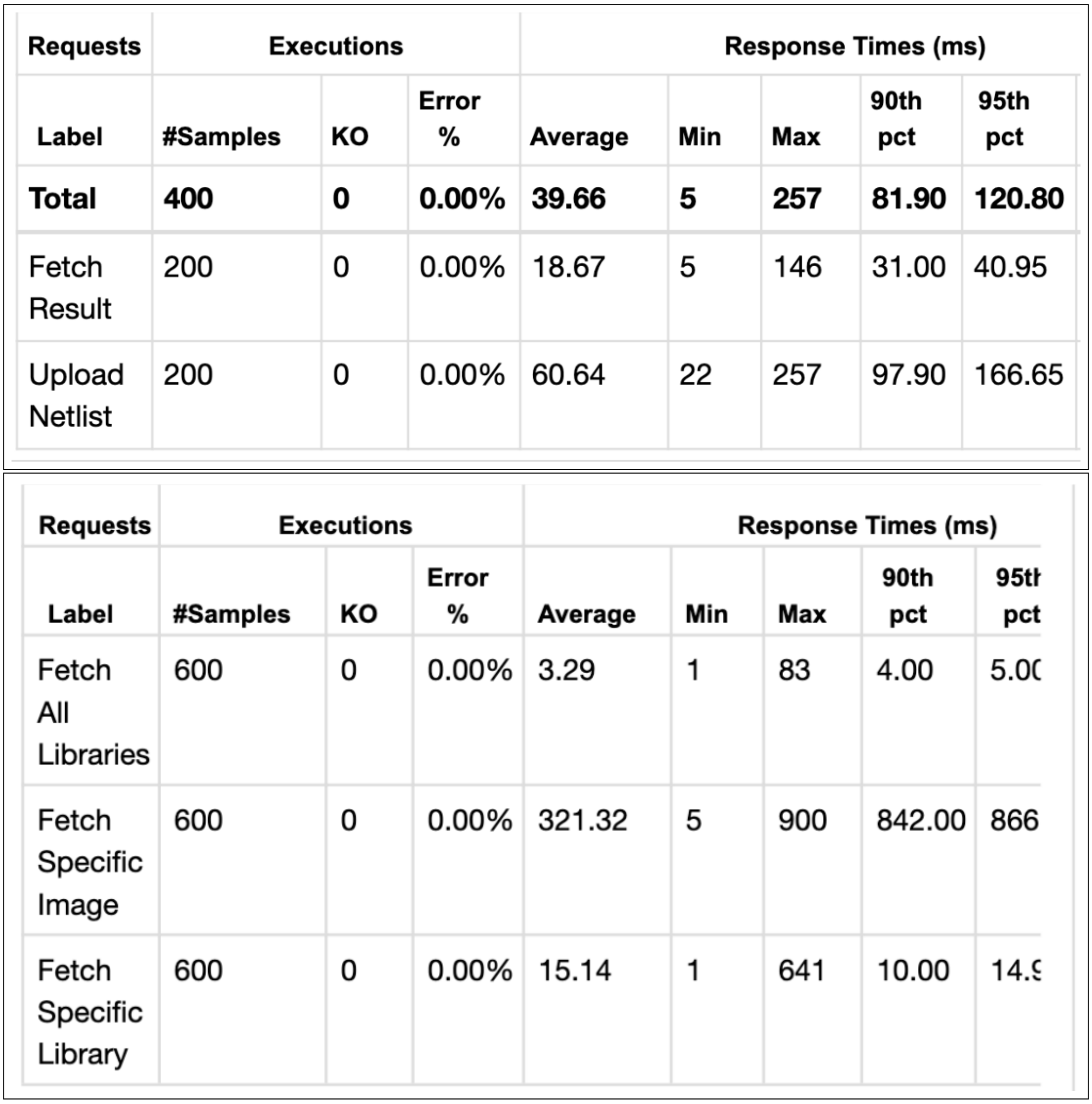
6.2.1 Ansible Scripts

Ansible[12] is a IT Orchestration tool developed by RedHat to manage servers and deployments across multiple machines. To make it easier to deploy the project on a production server or even a developers local machine ansible scripts have been provided with the project. There are essentially two scripts, the first configures the machine with the correct versions of docker and docker-compose and the latter clones the project repository, builds and runs the necessary docker containers.

6.2.2 Development Installation Script

To make it easier for developers contributing to the project to setup the development environment quickly, `first-run.dev.sh` has been written to allow single command installation of the development environment. This script deletes any residual folders which can conflict with the installation of the project , then builds the docker images for all the services and then runs database migrations and even seeds the KiCAD libraries to the database using a `Django Management Command` which was also written to ease seeding KiCAD libraries to the project.

6.3 Performance Testing



Requests	Executions			Response Times (ms)				
Label	#Samples	KO	Error %	Average	Min	Max	90th pct	95th pct
Total	**400**	**0**	**0.00%**	**39.66**	**5**	**257**	**81.90**	**120.80**
Fetch Result	200	0	0.00%	18.67	5	146	31.00	40.95
Upload Netlist	200	0	0.00%	60.64	22	257	97.90	166.65

Requests	Executions			Response Times (ms)				
Label	#Samples	KO	Error %	Average	Min	Max	90th pct	95th pct
Fetch All Libraries	600	0	0.00%	3.29	1	83	4.00	5.00
Fetch Specific Image	600	0	0.00%	321.32	5	900	842.00	866
Fetch Specific Library	600	0	0.00%	15.14	1	641	10.00	14.9

Figure 6.2: Jmeter Tests

JMeter[13] is Java application designed to load test functional behavior and measure performance of web applications. Jmeter tests have also been written to evaluate the performance of the most APIs.

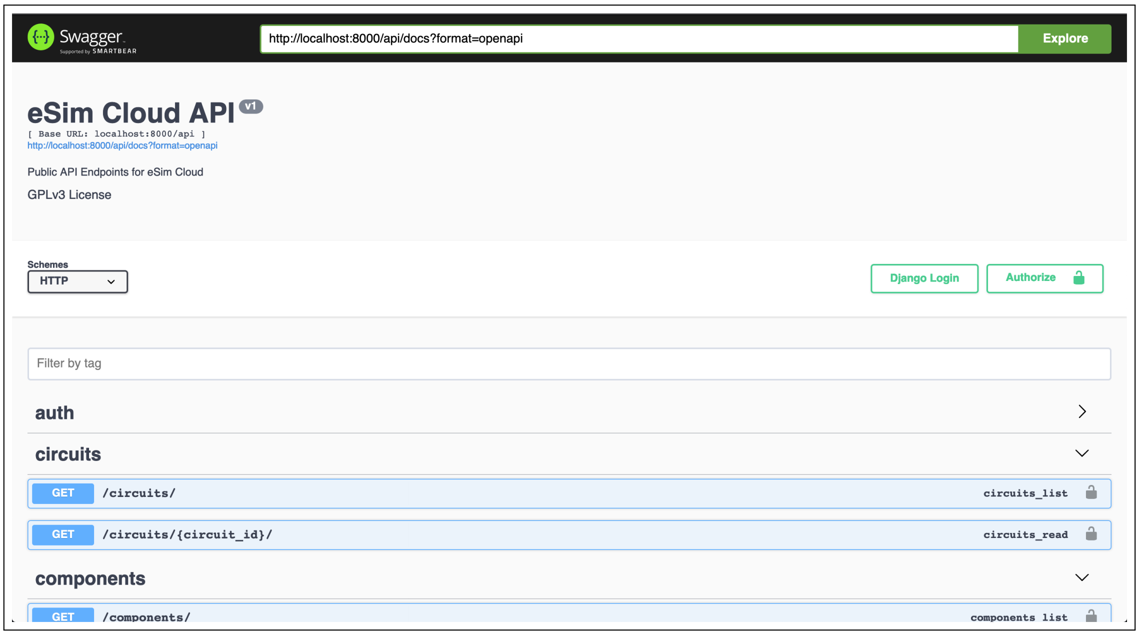
Test Setup

JMeter configuration to run 200 simultaneous threads (can be perceived as individual users) ramping up over a period of 2 seconds was used for all the tests. The tests were performed on Apple MacBook Pro with 8GB of RAM , SSD Storage and Intel's i5-8279U Processor. The tests were performed using the Production Config of the Project.

Test Results

Figure 6.2 shows test results, Label column describes the API action being tested and Response time describe response times for the same. It is worth noting that all endpoints tested provide sub 400ms response times.

6.3.1 Automatic OpenAPI Compliant API Documentation



http://localhost:8000/api/docs?format=openapi

eSim Cloud API v1

[Base URL: localhost:8000/api]
http://localhost:8000/api/docs?format=openapi

Public API Endpoints for eSim Cloud

GPLv3 License

Schemes HTTP

auth

circuits

GET /circuits/ circuits_list

GET /circuits/{circuit_id}/ circuits_read

components

GET /components/ components_list

Figure 6.3: API Documentation Screenshot

Swagger[14] has been integrated to automatically generate OpenAPI Compatible schema and also a dashboard documenting the endpoints. This also provides easy to try out various API endpoints from the browser itself.

Chapter 7

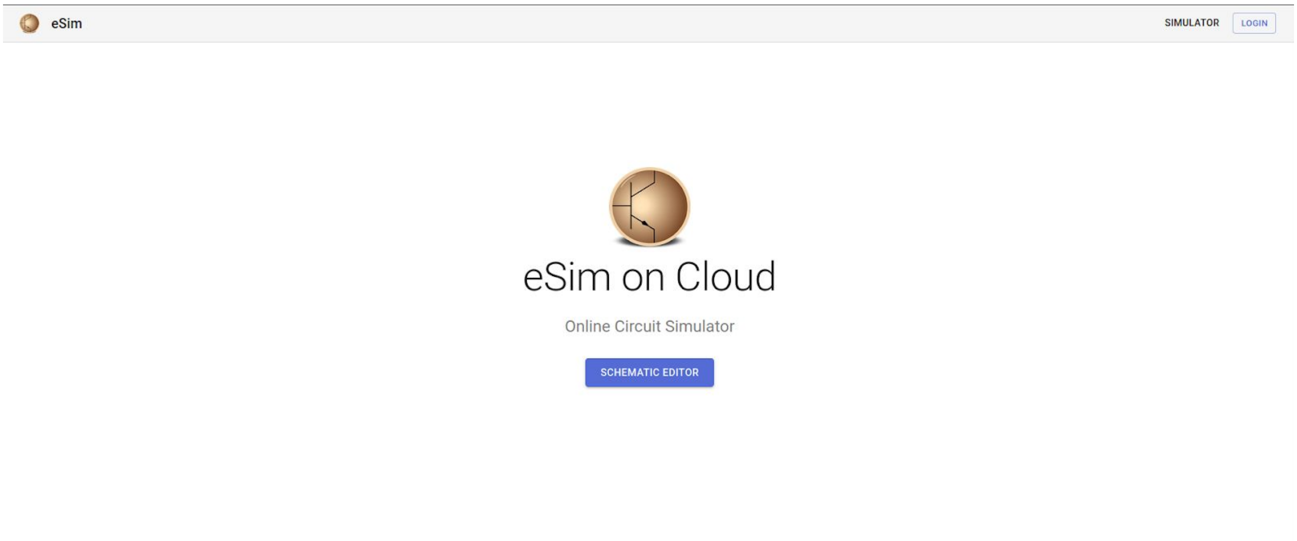
eSim On Cloud User Guide

The following are basic instructions for the user about how to use the application for drawing the schematic diagram of circuit by using provided components and Simulate the drawn circuit under specific simulation modes. Instructions for spice simulator are listed below. Parameters for sample circuits in eSim gallery page are listed too.

THE ESIM ON CLOUD USER GUIDE

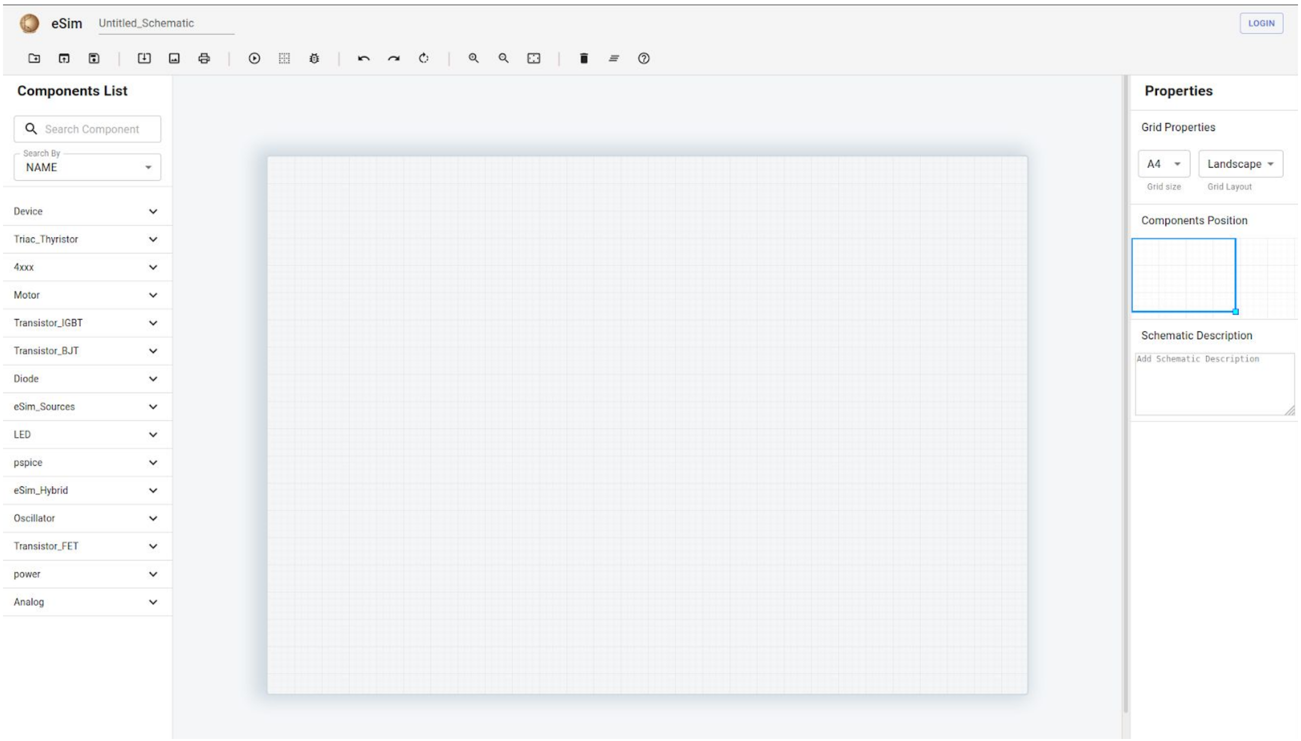
HOME PAGE :

This is Home page of eSim on cloud

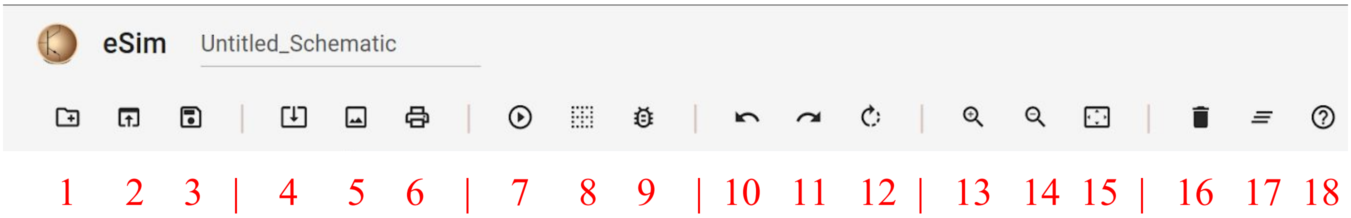


SCHEMATIC EDITOR :

Here you can draw schematic diagram with help of components

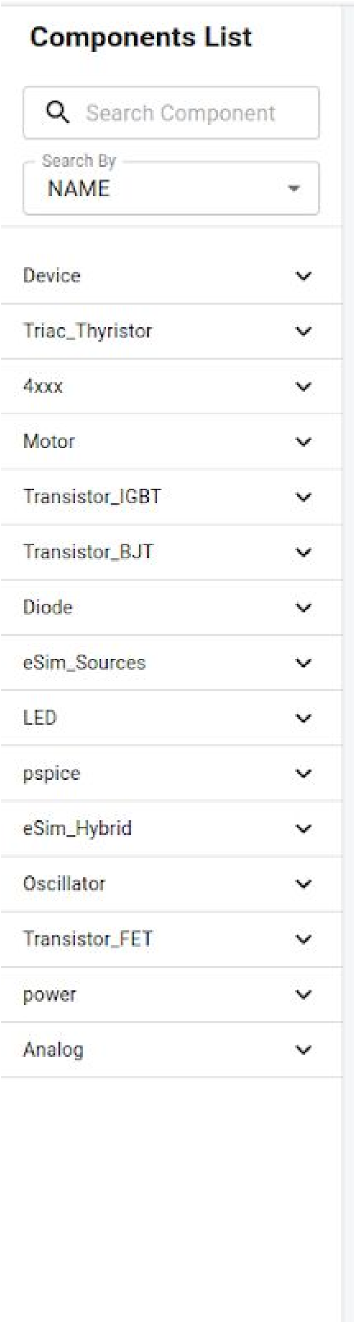


TOOLBAR :

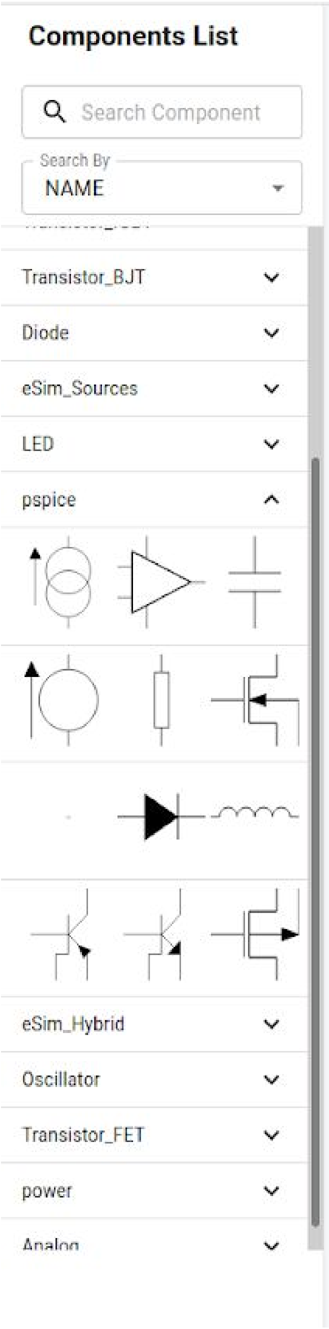


1	**New**	To open a New Editor
2	**Open**	To open the existing saved project
3	**Save**	To save existing drawn schematic
4	**Export**	To export the circuit as JSON
5	**Image Export**	To export circuit as jpeg, png, and svg images
6	**Print Preview**	Opens up print preview to print the circuit
7	**Simulate**	Opens simulate modes on the left side bar
8	**Generate Netlist**	Generates a netlist for the circuit on the grid
9	**ERC check**	Runs basic erc checks on the circuit on the grid
10	**Undo**	Undo the latest changes
11	**Redo**	Redo the undone changes
12	**Rotate**	Rotate component by 90 degree
13	**Zoom In**	Zooms in the whole circuit
14	**Zoom out**	Zooms out the whole circuit
15	**Default Size**	Reset’s to the default grid size
16	**Delete**	Delete the component selected
17	**Clear All**	Clear the grid with schematic drawn
18	**Help**	Shows the help screen

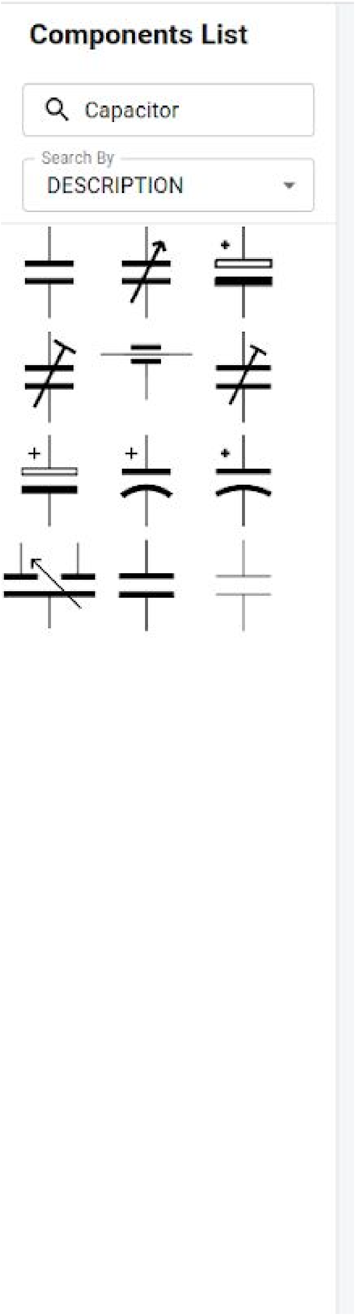
COMPONENT SIDEBAR : [Left Side Panel]


Components List

Search By
NAME


Components List

Search By
NAME

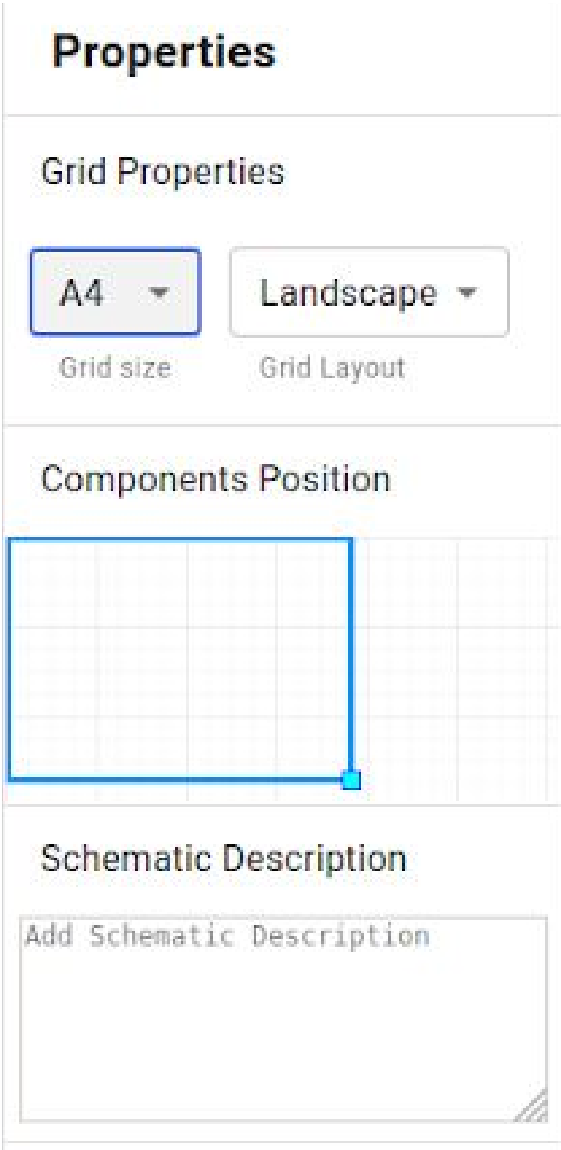

Components List

Capacitor

Search By
DESCRIPTION

You can either select components from the component sidebar or
You can search for the component in the search bar. There are various search filter options to sort components. To see all the search filters click on the **[Search By]** dropdown below the search box.

PROPERTIES SIDEBAR: [Right Side Panel]

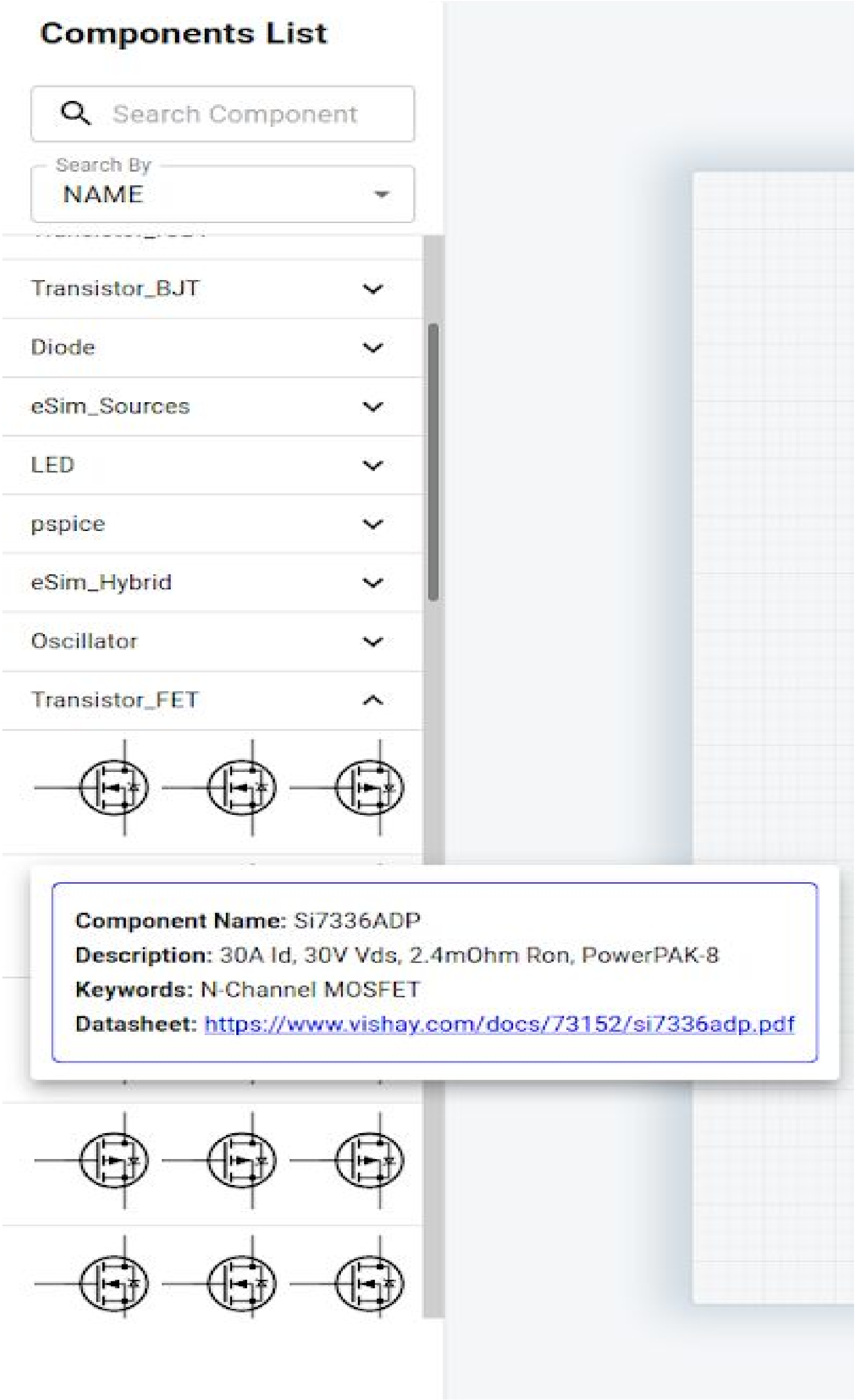


Grid size: A4

Grid Layout: Landscape

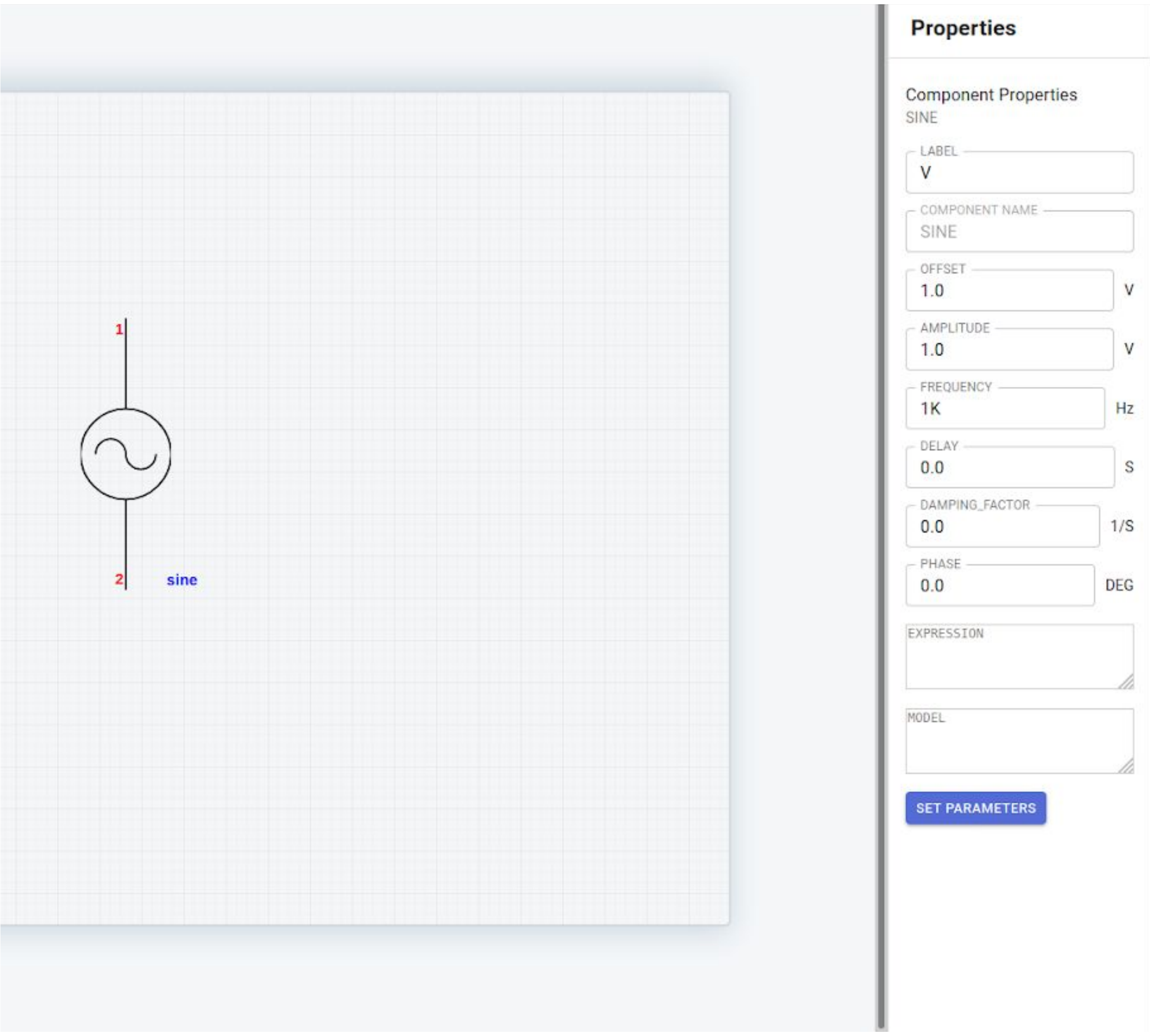
To change the size of the grid, select the grid size from the **[Grid size]** dropdown menu
To change the orientation of the grid, Select the orientation from
[Grid Layout] dropdown menu
On the properties sidebar.
To move over the grid (bird’s eye view) Use the Blue box.
To provide description to your circuit while saving, type the description in the **[Schematic Description]** text box.

DRAWING THE CIRCUIT :



Component Name: Si7336ADP
Description: 30A Id, 30V Vds, 2.4mOhm Ron, PowerPAK-8
Keywords: N-Channel MOSFET
Datasheet: https://www.vishay.com/docs/73152/si7336adp.pdf

Hover on the component to see the name and for details **Single Click** on that component. To place a component on the grid, **Drag and drop** that component onto the grid.



Properties

Component Properties
SINE

LABEL: V

COMPONENT NAME: SINE

OFFSET: 1.0 V

AMPLITUDE: 1.0 V

FREQUENCY: 1K Hz

DELAY: 0.0 S

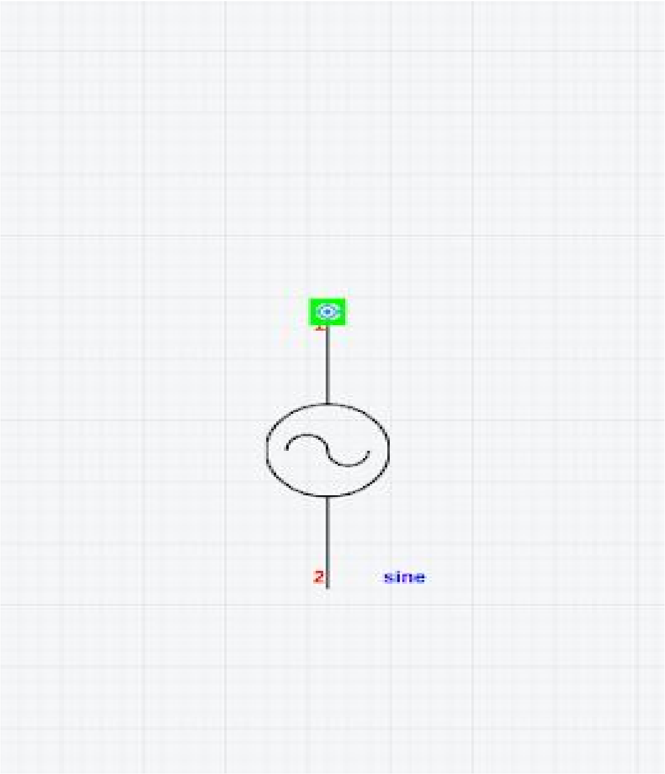
DAMPING_FACTOR: 0.0 1/S

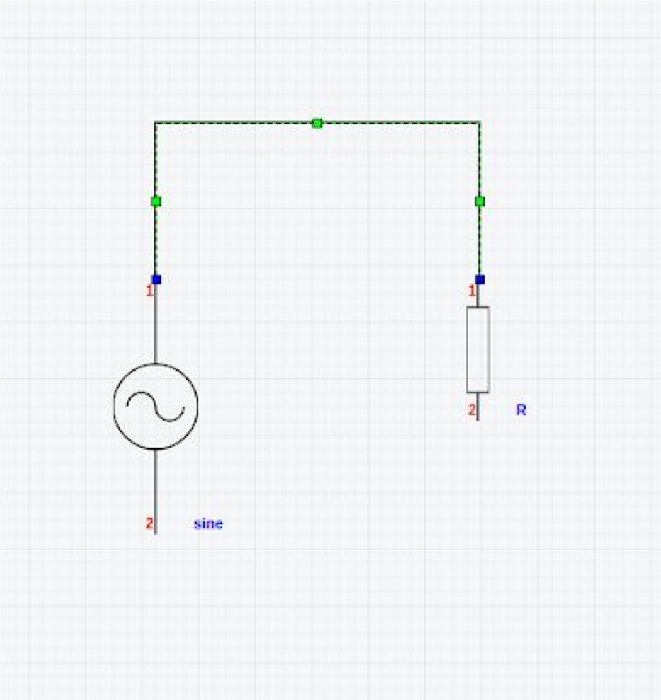
PHASE: 0.0 DEG

EXPRESSION

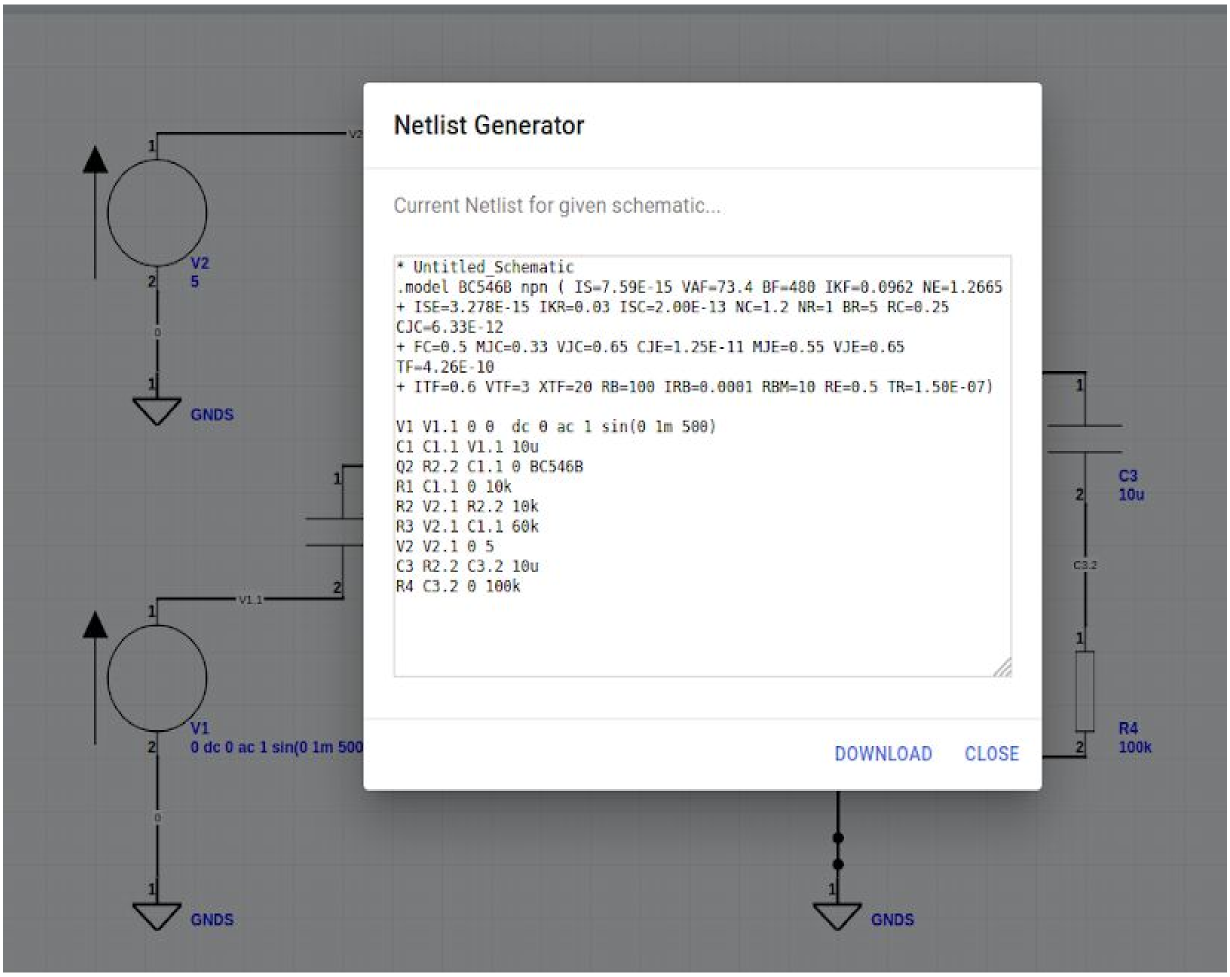
MODEL

To Enter/ Edit parameters of a component **Double Click** on the component.
You can see all the parameters of that component on the properties sidebar on the Right side panel of the screen.
After entering the values, click on **[SET PARAMETER]** button to set the values.





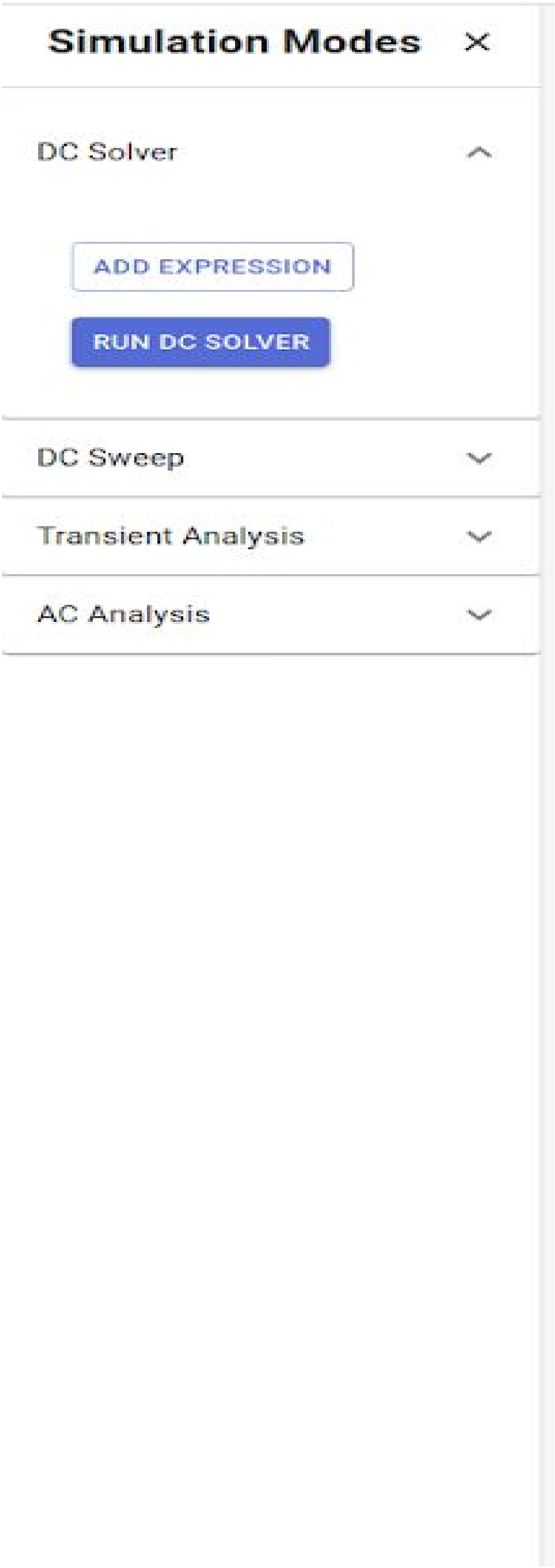
For Connecting wire between two pins simply hover over the 1st pin then you will see that a green box appears click on it and drag to reach the 2nd pin, Release the mouse to connect the pins. Now the two pins are
connected successfully.

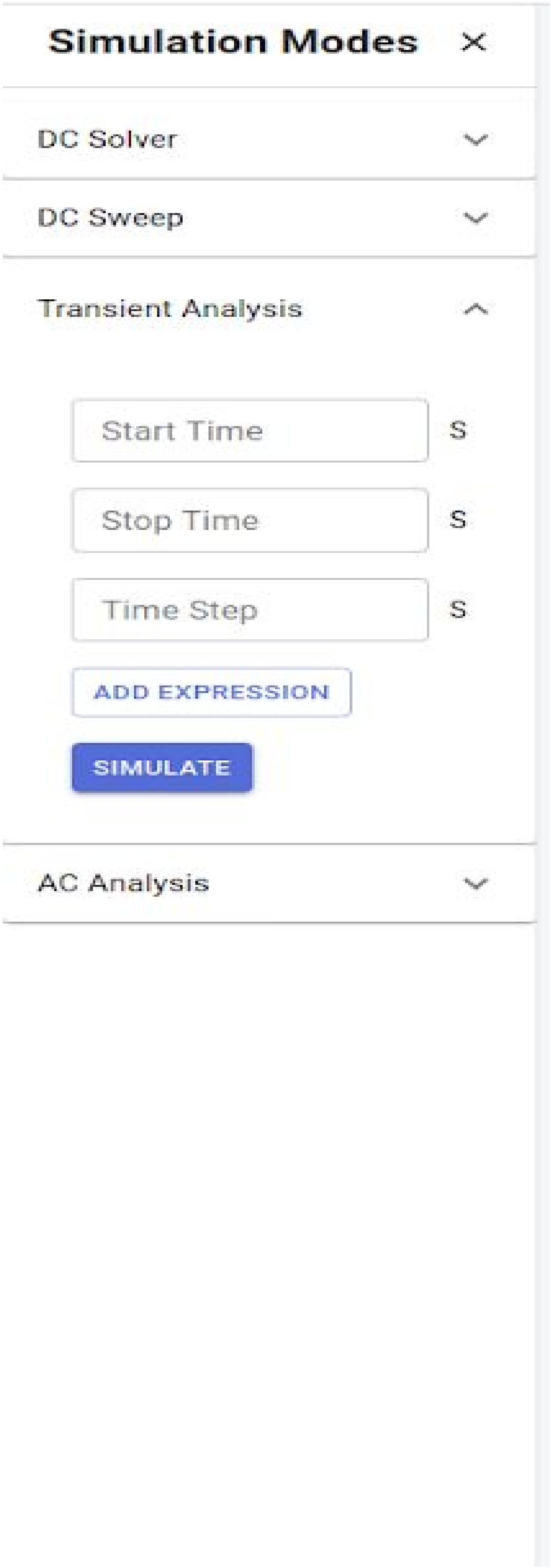


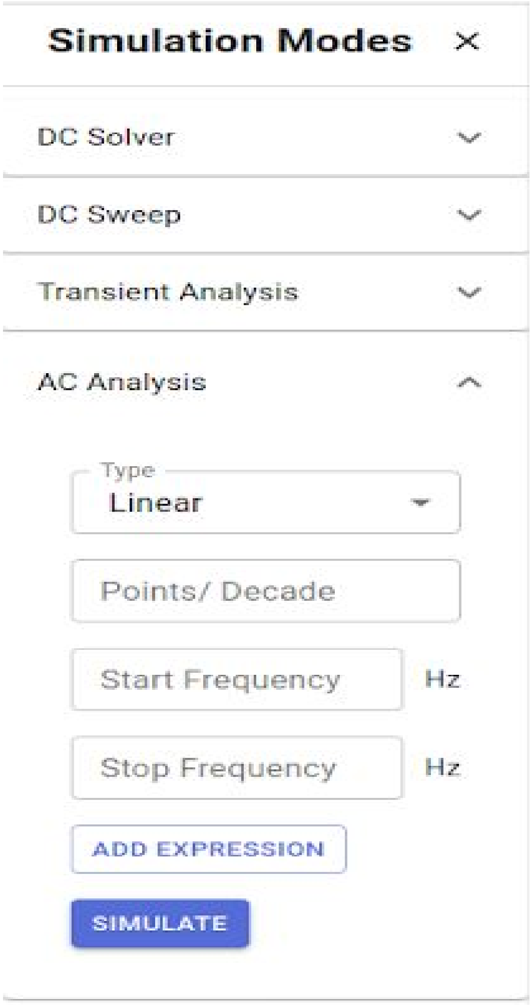
```
* Untitled_Schematic
.model BC546B npn ( IS=7.59E-15 VAF=73.4 BF=480 IKF=0.0962 NE=1.2665
+ ISE=3.278E-15 IKR=0.03 ISC=2.00E-13 NC=1.2 NR=1 BR=5 RC=0.25
CJC=6.33E-12
+ FC=0.5 MJC=0.33 VJC=0.65 CJE=1.25E-11 MJE=0.55 VJE=0.65
TF=4.26E-10
+ ITF=0.6 VTF=3 XTF=20 RB=100 IRB=0.0001 RBM=10 RE=0.5 TR=1.50E-07)

V1 V1.1 0 0  dc 0 ac 1 sin(0 1m 500)
C1 C1.1 V1.1 10u
Q2 R2.2 C1.1 0 BC546B
R1 C1.1 0 10k
R2 V2.1 R2.2 10k
R3 V2.1 C1.1 60k
V2 V2.1 0 5
C3 R2.2 C3.2 10u
R4 C3.2 0 100k
```

After you are done with drawing the circuit you can view the netlist of the circuit by clicking the **[Generate Netlist]** button on the toolbar.
A popup window will appear displaying the netlist.
You can also download the netlist by clicking on the download button.

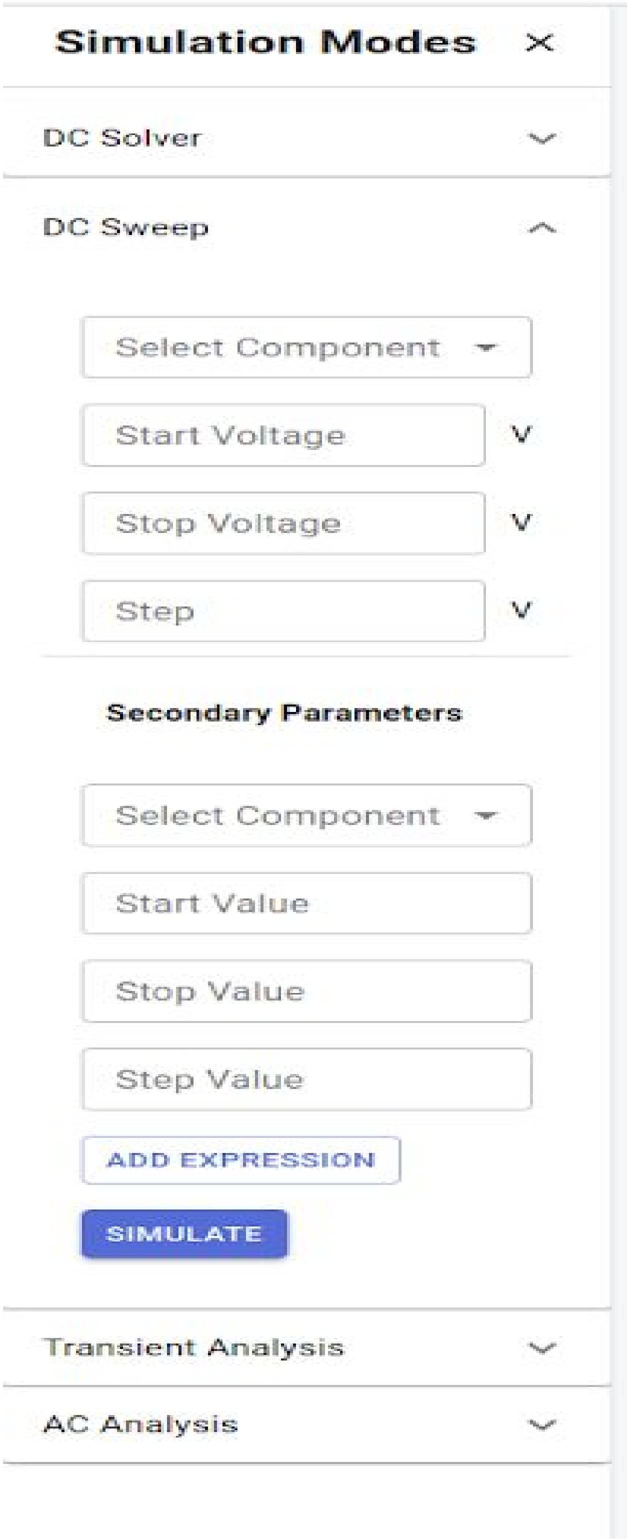
SIMULATION MODES :







Type
Linear



After you are done with drawing the circuit, To simulate the circuit click on the **[Simulate]** button on the toolbar.

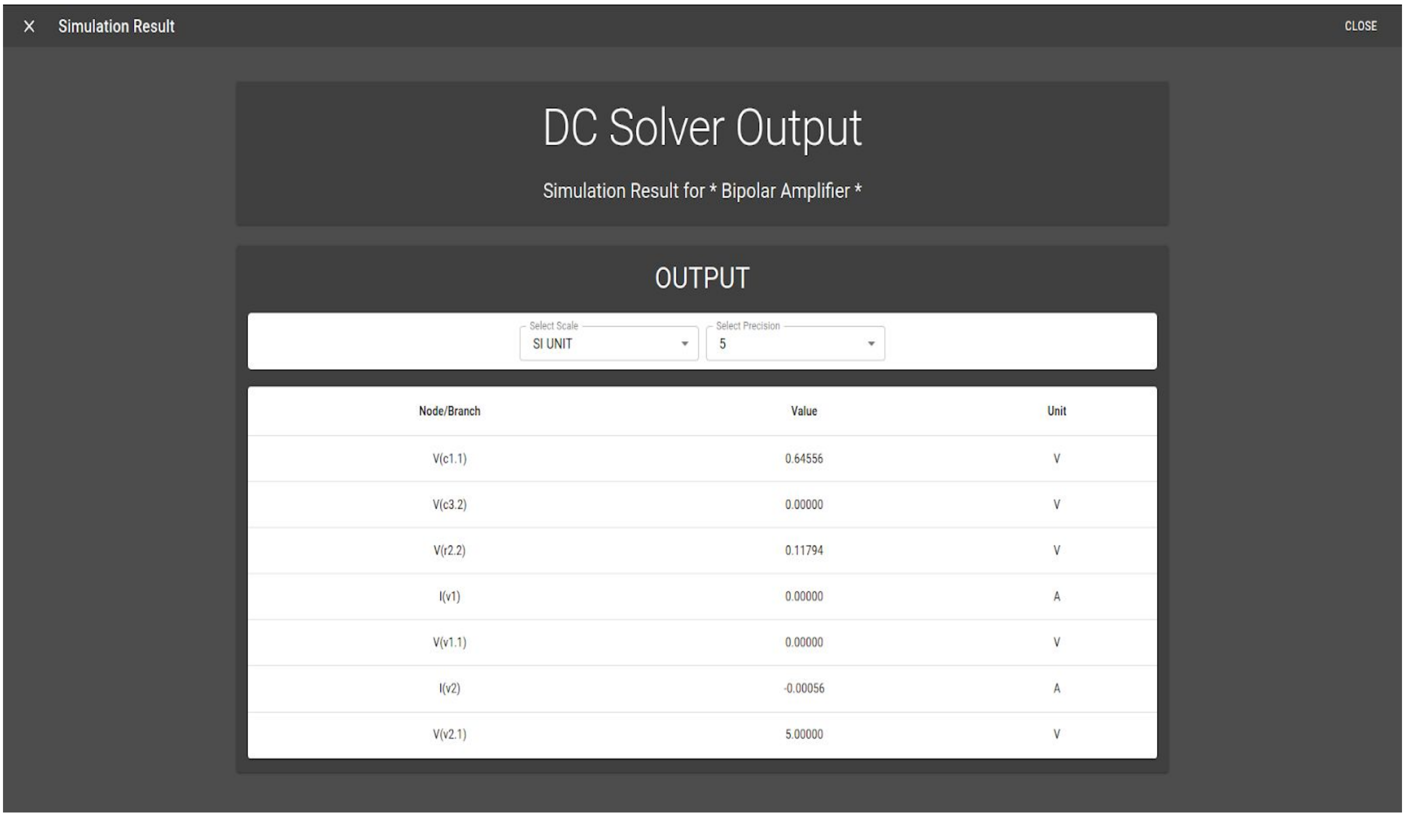
You will notice the left sidebar replaces components list to display Simulation Modes.

Click on the simulation mode you need, To display the simulation parameter for the simulation. Simply fill in the simulation parameters and click on the
[Simulate] button to run the simulation.

<u>Note</u> - if you only want to calculate values at a particular node you need to click on "Add expression" and enter the node name there before clicking on the simulate button.
Add Expression follows ngspice syntax therefore to indicate current
Add the postfix "#branch" at the end of node name.

After you click on the **[Simulate]** button a new screen will pop up where you can see the simulation result.
<u>Note</u> : The simulation output can be either a table or a graph depending on the simulation mode.

SIMULATION RESULT SCREEN :



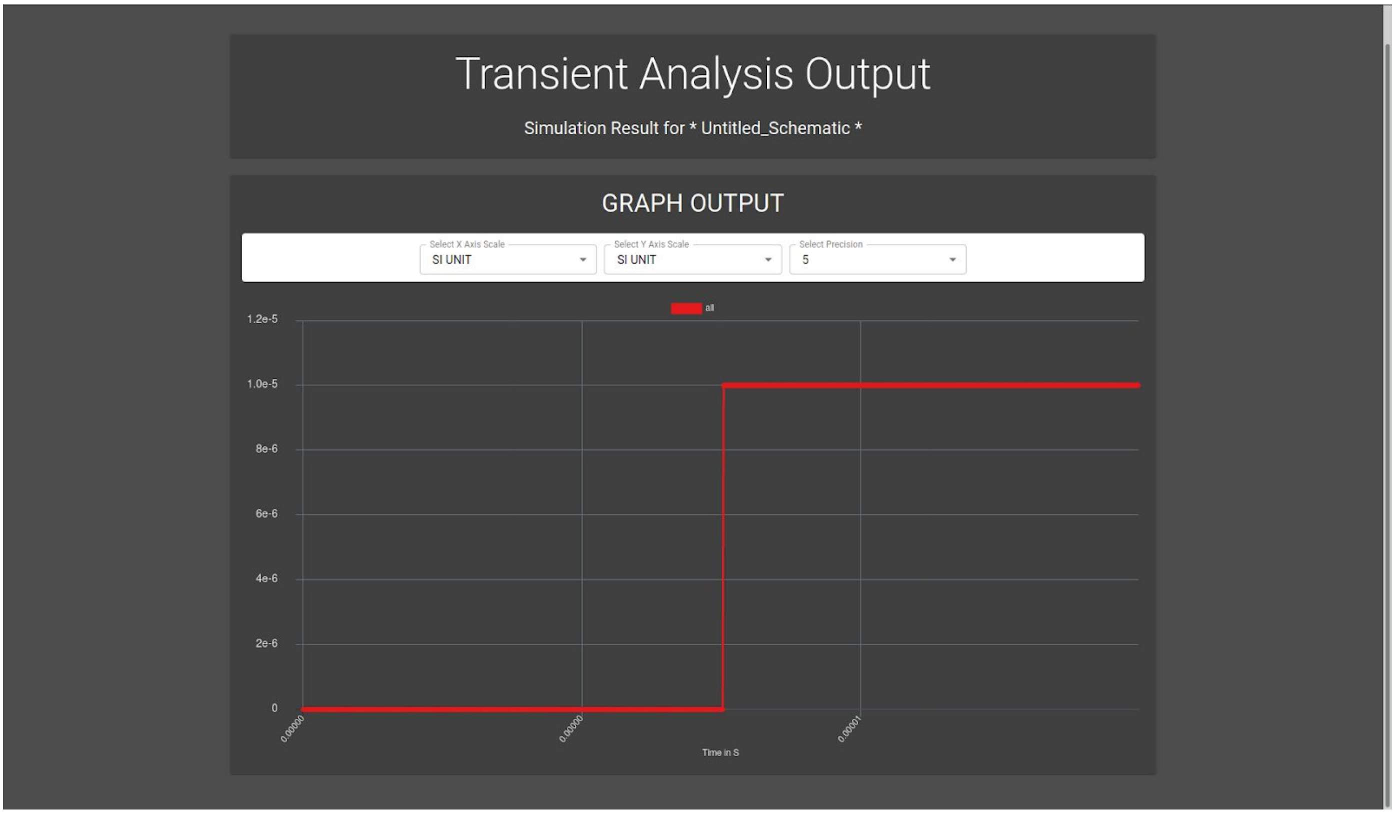
DC Solver Output

Simulation Result for * Bipolar Amplifier *

OUTPUT

Select Scale: SI UNIT
Select Precision: 5

Node/Branch	Value	Unit
V(c1.1)	0.64556	V
V(c3.2)	0.00000	V
V(r2.2)	0.11794	V
I(v1)	0.00000	A
V(v1.1)	0.00000	V
I(v2)	-0.00056	A
V(v2.1)	5.00000	V



Transient Analysis Output

Simulation Result for * Untitled_Schematic *

GRAPH OUTPUT

Select X Axis Scale: SI UNIT

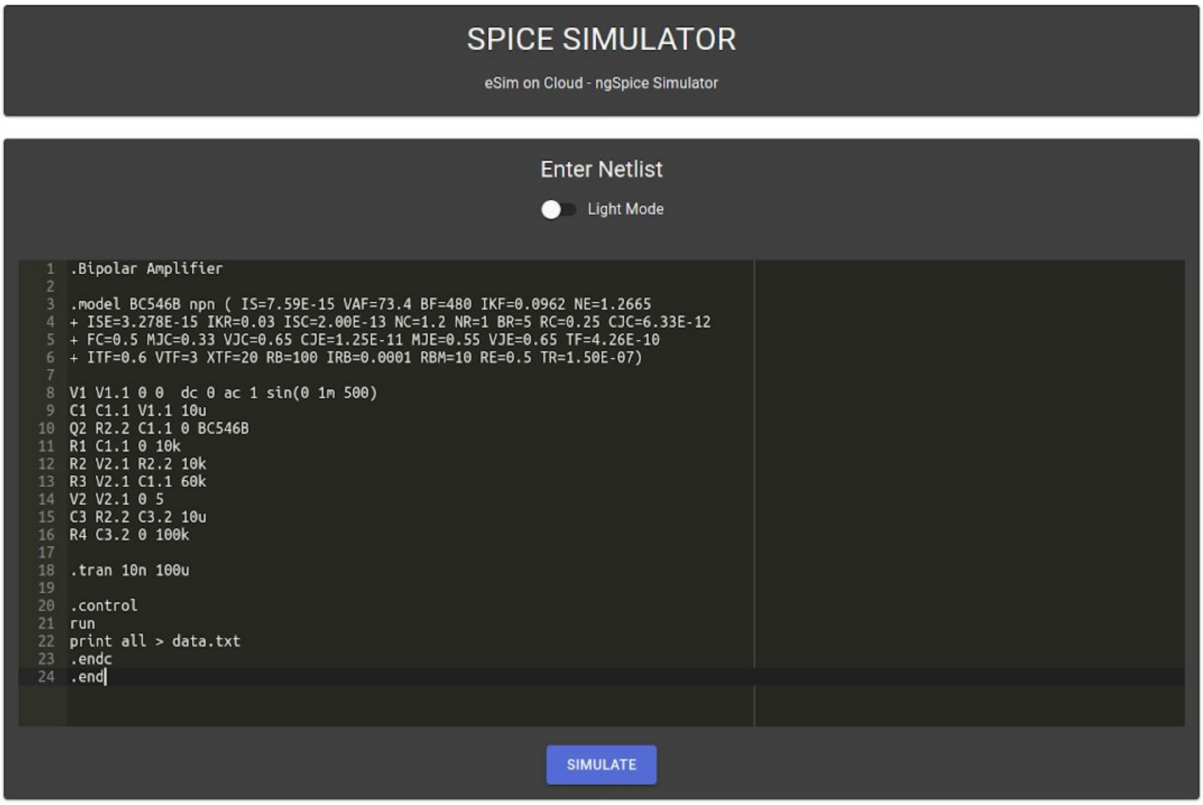
Select Y Axis Scale: SI UNIT

Select Precision: 5

On the simulation result screen you can change the scale of the x-axis and the y-axis by selecting an option from the **[Select x-axis scale]** and **[Select y-axis scale]** dropdown menu. To change the precision of the values select an option from the **[Select Precision]** dropdown menu.

You can also select or deselect the checkbox on top of the graph to display graphs related to a particular node.

SPICE SIMULATOR :



```
.Bipolar Amplifier

.model BC546B npn ( IS=7.59E-15 VAF=73.4 BF=480 IKF=0.0962 NE=1.2665
+ ISE=3.278E-15 IKR=0.03 ISC=2.00E-13 NC=1.2 NR=1 BR=5 RC=0.25 CJC=6.33E-12
+ FC=0.5 MJC=0.33 VJC=0.65 CJE=1.25E-11 MJE=0.55 VJE=0.65 TF=4.26E-10
+ ITF=0.6 VTF=3 XTF=20 RB=100 IRB=0.0001 RBM=10 RE=0.5 TR=1.50E-07)

V1 V1.1 0 0  dc 0 ac 1 sin(0 1m 500)
C1 C1.1 V1.1 10u
Q2 R2.2 C1.1 0 BC546B
R1 C1.1 0 10k
R2 V2.1 R2.2 10k
R3 V2.1 C1.1 60k
V2 V2.1 0 5
C3 R2.2 C3.2 10u
R4 C3.2 0 100k

.tran 10n 100u

.control
run
print all > data.txt
.endc
.end
```

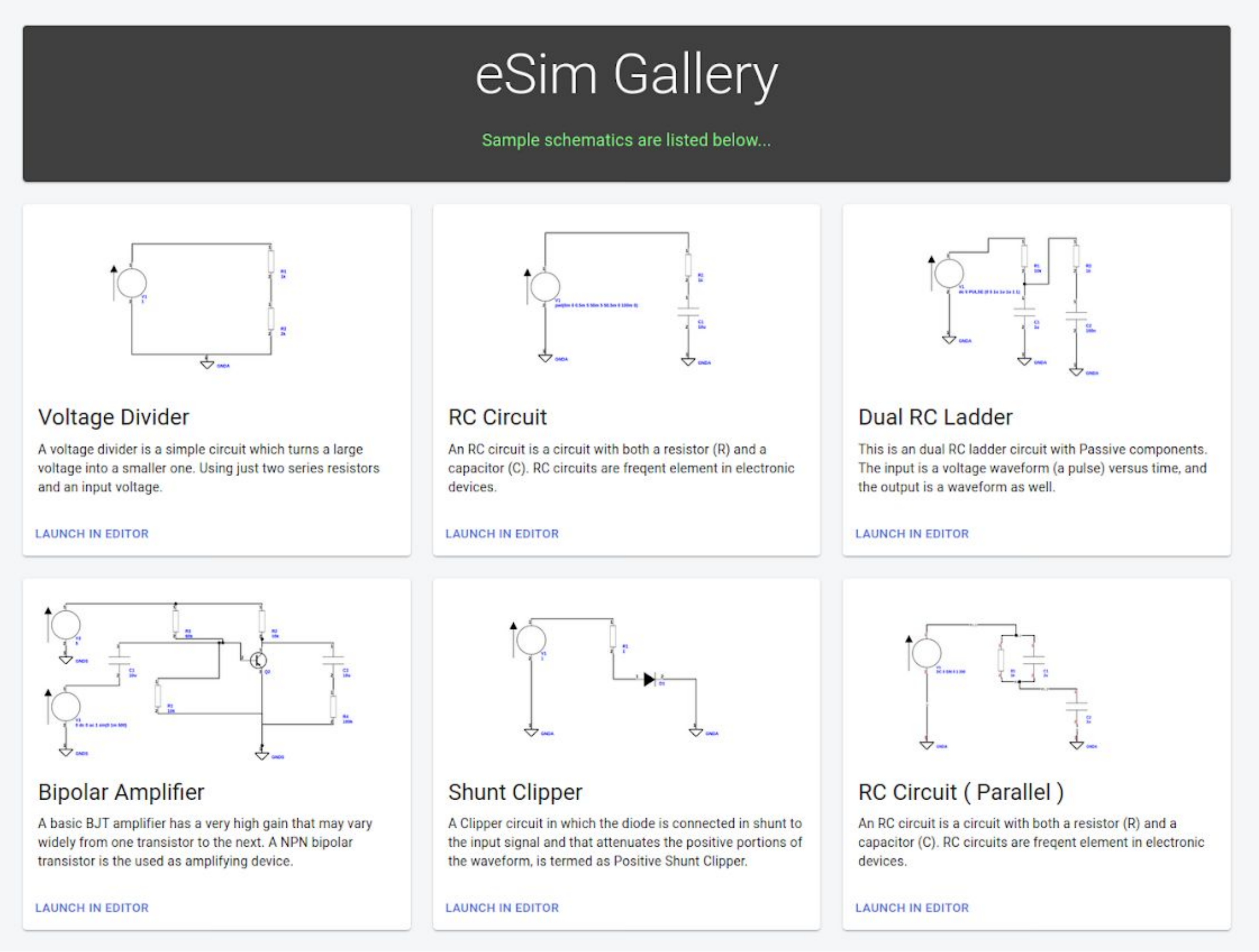
In the spice simulator page you can type your netlist in the code editor box. You can change the theme of the editor to light by clicking the switch on top. After you are done typing netlist you can click on the **[Simulate]** button to view the result. Simulation result window will popup displaying the result.

NOTE - The user needs to add " > data.txt " at the end of the control line.

E.g

.control
run
print all ***> data.txt***
.endc
.end

The eSim Gallery : [Sample schematic examples]



eSim Gallery

Sample schematics are listed below...

Voltage Divider

A voltage divider is a simple circuit which turns a large voltage into a smaller one. Using just two series resistors and an input voltage.

LAUNCH IN EDITOR

RC Circuit

An RC circuit is a circuit with both a resistor (R) and a capacitor (C). RC circuits are freqent element in electronic devices.

LAUNCH IN EDITOR

Dual RC Ladder

This is an dual RC ladder circuit with Passive components. The input is a voltage waveform (a pulse) versus time, and the output is a waveform as well.

LAUNCH IN EDITOR

Bipolar Amplifier

A basic BJT amplifier has a very high gain that may vary widely from one transistor to the next. A NPN bipolar transistor is the used as amplifying device.

LAUNCH IN EDITOR

Shunt Clipper

A Clipper circuit in which the diode is connected in shunt to the input signal and that attenuates the positive portions of the waveform, is termed as Positive Shunt Clipper.

LAUNCH IN EDITOR

RC Circuit (Parallel)

An RC circuit is a circuit with both a resistor (R) and a capacitor (C). RC circuits are freqent element in electronic devices.

LAUNCH IN EDITOR

Step 1 : From the Home page of “eSim on cloud” navigate to **Gallery** page from the navbar at the top.

Step 2 : Open a circuit in the editor by clicking on **[LAUNCH IN EDITOR]** The circuit is launched in the editor with preconfigured component Properties [however users can change it].

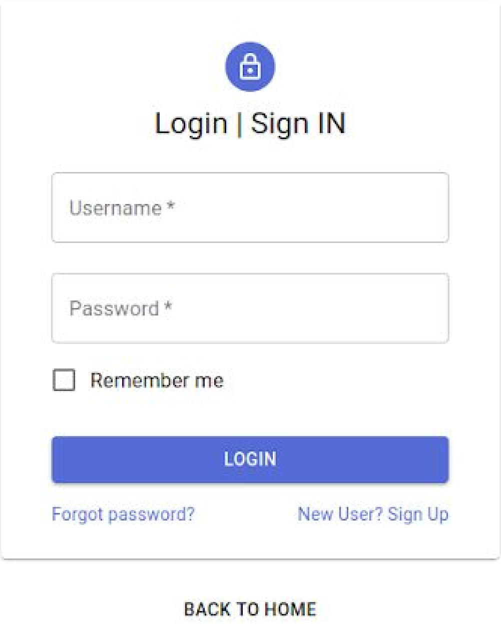
Step 3 : Click on the simulate button on the toolbar on top, select the simulation mode and fill in the simulation parameters, then click on the simulate button.

Simulation properties for respective examples in the gallery are listed below
[however you can also use different values than those written below].

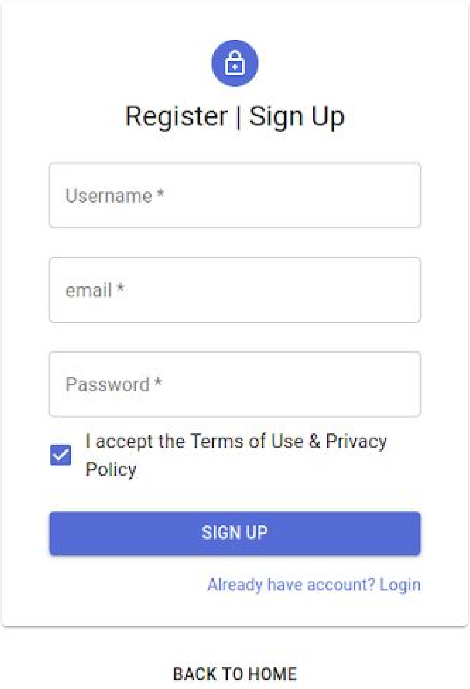
SAMPLE VALUES FOR REFERENCE :

S.NO	NAME	SIMULATIO N TYPE	SIMULATION PARAMETRS	ADD EXPRESSION
1	Voltage Divider	DC SOLVER	NONE	NONE
2	RC Circuit	TRANSIENT ANALYSIS	Start Time - 0 Stop Time - 100m Step Time - 10m	NONE
3	Dual RC Ladder	TRANSIENT ANALYSIS	Start Time - 0 Stop Time - 50m Step Time - 50u	NONE
4	Bipolar Amplifier	TRANSIENT ANALYSIS	Start Time - 0s Stop Time - 10m Step Time - 10u	NONE
		AC ANALYSIS	Type - DECADE Points - 10 Start Freq - 10 Stop Freq - 10Meg	NONE
5	Shunt Clipper	DC SWEEP	Component - V1 Start Voltage - 0 Stop Voltage - 1 Step Voltage - 1m	**-v1#branch**
6	RC Circuit (Parallel)	TRANSIENT ANALYSIS	Start Time - 0 Stop Time - 30m Step Time - 10u	NONE

LOGIN FORM :



SIGN UP FORM :


Register | Sign Up
Username *
email *
Password *
☑ I accept the Terms of Use & Privacy Policy
SIGN UP
Already have account? Login
BACK TO HOME

Chapter 8

API Endpoints

In following API documentation we have listed API used in this project.

eSim Cloud API

Overview

Public API Endpoints for eSim Cloud

Version information

Version : v1

License information

License : GPLv3 License
Terms of service : null

URI scheme

Host : localhost
BasePath : /api
Schemes : HTTP

Consumes

- `application/json`

Produces

- `application/json`

Security

Basic

Type : Token Authorization

Paths

GET /auth/google-callback

Description

Creates user if OAuth token valid

Responses

HTTP Code	Schema
200	No Content

Tags

- auth

POST /auth/o/{provider}/

Parameters

Type	Name	Schema
Path	**provider** *required*	string
Body	**data** *required*	ProviderAuth

Responses

HTTP Code	Schema
201	ProviderAuth

Tags

- auth

GET /auth/o/{provider}/

Parameters

Type	Name	Schema
Path	**provider** *required*	string

Responses

HTTP Code	Schema
200	ProviderAuth

Tags

- auth

POST /auth/token/login/

Description

Use this endpoint to obtain user authentication token.

Parameters

Type	Name	Schema
Body	**data** *required*	TokenCreate

Responses

HTTP Code	Schema
201	TokenCreate

Tags

- auth

POST /auth/token/logout/

Description

Use this endpoint to logout user (remove user authentication token).

Responses

HTTP Code	Schema
201	No Content

Tags

- auth

POST /auth/users/

Parameters

Type	Name	Schema
Body	**data** *required*	UserCreate

Responses

HTTP Code	Schema
201	UserCreate

Tags

- auth

GET /auth/users/

Responses

HTTP Code	Schema
200	< User > array

Tags

- auth

POST /auth/users/activation/

Parameters

Type	Name	Schema
Body	**data** *required*	Activation

Responses

HTTP Code	Schema
201	Activation

4

Tags

- aut h

GET /auth/users/me/

Responses

HTTP Code	Schema
200	< User > array

Tags

- aut h

PUT /auth/users/me/

Parameters

Type	Name	Schema
Body	**data** *required*	User

Responses

HTTP Code	Schema
200	User

Tags

- aut h

DELETE /auth/users/me/

Responses

HTTP Code	Schema
204	No Cont ent

Tags

- aut h

PATCH /auth/users/me/

Parameters

Type	Name	Schema
Body	**data** *required*	User

Responses

HTTP Code	Schema
200	User

Tags

- aut h

POST /auth/users/resend_activation/

Parameters

Type	Name	Schema
Body	**data** *required*	SendEmailReset

Responses

HTTP Code	Schema
201	SendEmailReset

Tags

- aut h

POST /auth/users/reset_password/

Parameters

Type	Name	Schema
Body	**data** *required*	SendEmailReset

Responses

HTTP Code	Schema
201	SendEmailReset

Tags

- aut h

POST /auth/users/reset_password_confirm/

Parameters

Type	Name	Schema
Body	**data** *required*	PasswordReset Confirm

Responses

HTTP Code	Schema
201	PasswordReset Confirm

Tags

- aut h

POST /auth/users/reset_username/

Parameters

Type	Name	Schema
Body	**data** *required*	SendEmailReset

Responses

HTTP Code	Schema
201	SendEmailReset

Tags

- aut h

POST /auth/users/reset_username_confirm/

Parameters

Type	Name	Schema
Body	**data** *required*	UsernameReset Confirm

Responses

HTTP Code	Schema
201	UsernameReset Confirm

Tags

- aut h

POST /auth/users/set_password/

Parameters

Type	Name	Schema
Body	**data** *required*	Set Password

Responses

HTTP Code	Schema
201	Set Password

Tags

- aut h

POST /auth/users/set_username/

Parameters

Type	Name	Schema
Body	**data** *required*	Set Username

Responses

HTTP Code	Schema
201	Set Username

Tags

- aut h

GET /auth/users/{id}/

Parameters

Type	Name	Description	Schema
Path	**id** *required*	A unique int eger value ident ifying t his user.	int eger

Responses

HTTP Code	Schema
200	User

Tags

- aut h

PUT /auth/users/{id}/

Parameters

Type	Name	Description	Schema
Path	**id** *required*	A unique integer value identifying this user.	integer
Body	**data** *required*		User

Responses

HTTP Code	Schema
200	User

Tags

- auth

DELETE /auth/users/{id}/

Parameters

Type	Name	Description	Schema
Path	**id** *required*	A unique integer value identifying this user.	integer

Responses

HTTP Code	Schema
204	No Content

Tags

- auth

PATCH /auth/users/{id}/

Parameters

Type	Name	Description	Schema
Path	**id** *required*	A unique integer value identifying this user.	integer
Body	**data** *required*		User

Responses

HTTP Code	Schema
200	User

Tags

- auth

GET /circuits/

Description

Listing Published Circuits

Responses

HTTP Code	Schema
200	< Circuit > array

Tags

- circuits

GET /circuits/{circuit_id}/

Description

Listing Published Circuits

Parameters

Type	Name	Description	Schema
Path	**circuit_id** *required*	A UUID string identifying this circuit.	string (uuid)

Responses

HTTP Code	Schema
200	Circuit

Tags

- circuit s

GET /components/

Description

List ing All Library Det ails

Parameters

Type	Name	Schema
Query	**component_library** *optional*	st ring
Query	**component_library*library_name*contains** *optional*	st ring
Query	**description*contains*** opt ional__	st ring
Query	**keyword*contains*** opt ional__	st ring
Query	**name*contains*** opt ional__	st ring
Query	**symbol_prefix** *optional*	st ring

Responses

HTTP Code	Schema
200	< LibraryComponent > array

Tags

- components

GET /components/{id}/

Description

Listing All Library Details

Parameters

Type	Name	Description	Schema
Path	**id** *required*	A unique integer value identifying this library component.	integer

Responses

HTTP Code	Schema
200	LibraryComponent

Tags

- components

GET /libraries/

Description

Listing All Library Details

Parameters

Type	Name	Schema
Query	**library_name** *optional*	string

Responses

HTTP Code	Schema
200	< Library > array

Tags

- libraries

GET /libraries/{id}/

Description

List ing All Library Det ails

Parameters

Type	Name	Description	Schema
Path	**id** *required*	A unique int eger value ident ifying t his library.	int eger

Responses

HTTP Code	Schema
200	Library

Tags

- libraries

POST /publish/circuit/

Description

CRUD for viewing unpublished / published circuit s (Permission Groups)

Parameters

Type	Name	Schema
Body	**data** *required*	Circuit

Responses

HTTP Code	Schema
201	Circuit

Tags

- publish

GET /publish/circuit/

Description

CRUD for viewing unpublished / published circuit s (Permission Groups)

Responses

HTTP Code	Schema
200	< Circuit > array

Tags

- publish

GET /publish/circuit/{circuit_id}/

Description

CRUD for viewing unpublished / published circuit s (Permission Groups)

Parameters

Type	Name	Description	Schema
Path	**circuit_id** *required*	A UUID st ring ident ifying t his circuit .	st ring (uuid)

Responses

HTTP Code	Schema
200	Circuit

Tags

- publish

PUT /publish/circuit/{circuit_id}/

Description

CRUD for viewing unpublished / published circuit s (Permission Groups)

Parameters

Type	Name	Description	Schema
Path	**circuit_id** *required*	A UUID st ring ident ifying t his circuit .	st ring (uuid)
Body	**data** *required*		Circuit

Responses

HTTP Code	Schema
200	Circuit

Tags

- publish

DELETE /publish/circuit/{circuit_id}/

Description

CRUD for viewing unpublished / published circuit s (Permission Groups)

Parameters

Type	Name	Description	Schema
Path	**circuit_id** *required*	A UUID st ring ident ifying t his circuit .	st ring (uuid)

Responses

HTTP Code	Schema
204	No Cont ent

Tags

- publish

PATCH /publish/circuit/{circuit_id}/

Description

CRUD for viewing unpublished / published circuit s (Permission Groups)

Parameters

Type	Name	Description	Schema
Path	**circuit_id** *required*	A UUID st ring ident ifying t his circuit .	st ring (uuid)
Body	**data** *required*		Circuit

Responses

HTTP Code	Schema
200	Circuit

Tags

- publish

POST /publish/publishing/

Description

Publishing CRUD Operat ions

Parameters

Type	Name	Schema
Body	**data** *required*	Publish

Responses

HTTP Code	Schema
201	Publish

Tags

- publish

GET /publish/publishing/

Description

Publishing CRUD Operat ions

Responses

HTTP Code	Schema
200	< Publish > array

Tags

- publish

GET /publish/publishing/{id}/

Description

Publishing CRUD Operat ions

Parameters

Type	Name	Description	Schema
Path	**id** *required*	A unique int eger value ident ifying t his publish.	int eger

Responses

HTTP Code	Schema
200	Publish

Tags

- publish

PUT /publish/publishing/{id}/

Description

Publishing CRUD Operat ions

Parameters

Type	Name	Description	Schema
Path	**id** *required*	A unique int eger value ident ifying t his publish.	int eger
Body	**data** *required*		Publish

Responses

HTTP Code	Schema
200	Publish

Tags

- publish

DELETE /publish/publishing/{id}/

Description

Publishing CRUD Operat ions

Parameters

Type	Name	Description	Schema
Path	**id** *required*	A unique int eger value ident ifying t his publish.	int eger

Responses

HTTP Code	Schema
204	No Cont ent

Tags

- publish

PATCH /publish/publishing/{id}/

Description

Publishing CRUD Operations

Parameters

Type	Name	Description	Schema
Path	**id** *required*	A unique integer value identifying this publish.	integer
Body	**data** *required*		Publish

Responses

HTTP Code	Schema
200	Publish

Tags

- publish

POST /save

Description

API to save the state of project to db which can be loaded or shared later

Parameters

Type	Name	Schema
FormData	**data_dump** *required*	string
FormData	**owner** *optional*	integer
FormData	**save_id** *optional*	string (uuid)

Type	Name	Schema
FormData	**shared** *optional*	boolean

Responses

HTTP Code	Schema
201	StateSave

Consumes

- `application/x-www-form-urlencoded`

Tags

- save

GET /save/list

Description

Returns Saved data for given username, Only user who saved the state can access it THIS WILL ESCAPE DOUBLE QUOTES

Responses

HTTP Code	Schema
200	StateSave

Tags

- save

POST /save/{save_id}

Description

Returns Saved data for given save id , Only user who saved the state can access / update it THIS WILL ESCAPE DOUBLE QUOTES

Parameters

Type	Name	Schema
Path	**save_id** *required*	string

Responses

HTTP Code	Schema
200	StateSave

Tags

- save

GET /save/{save_id}

Description

Returns Saved data for given save id , Only user who saved the state can access / update it THIS WILL ESCAPE DOUBLE QUOTES

Parameters

Type	Name	Schema
Path	**save_id** *required*	string

Responses

HTTP Code	Schema
200	StateSave

Tags

- save

POST /save/{save_id}/sharing/{sharing}

Description

Enables sharing for the given saved state

Parameters

Type	Name	Schema
Path	**save_id** *required*	string
Path	**sharing** *required*	string

Responses

HTTP Code	Schema
200	StateSave

Tags

- save

GET /simulation/status/{task_id}

Description

Returns Simulation results for 'task_id' provided after uploading the netlist /api/task/<uuid>

Parameters

Type	Name	Schema
Path	**task_id** *required*	string

Responses

HTTP Code	Schema
200	No Content

Tags

- simulation

API for NetlistUpload

```
POST /simulation/upload
```

Description

Requires a multipart/form-data POST Request with netlist file in the 'file' parameter

Responses

HTTP Code	Schema
201	No Content

Consumes

- `multipart/form-data`
- `application/x-www-form-urlencoded`

Tags

- simulation

POST /tags/

Description

CRUD for Tags

Parameters

Type	Name	Schema
Body	**data** *required*	Circuit Tag

Responses

HTTP Code	Schema
201	Circuit Tag

Tags

- tags

GET /tags/

Description

CRUD for Tags

Responses

HTTP Code	Schema
200	< Circuit Tag> array

Tags

- tags

GET /tags/{id}/

Description

CRUD for Tags

Parameters

Type	Name	Description	Schema
Path	**id** *required*	A unique integer value identifying this circuit tag.	integer

Responses

HTTP Code	Schema
200	Circuit Tag

Tags

- tags

PUT /tags/{id}/

Description

CRUD for Tags

Parameters

Type	Name	Description	Schema
Path	**id** *required*	A unique integer value identifying this circuit tag.	integer
Body	**data** *required*		Circuit Tag

Responses

HTTP Code	Schema
200	Circuit Tag

Tags

- tags

DELETE /tags/{id}/

Description

CRUD for Tags

Parameters

Type	Name	Description	Schema
Path	**id** *required*	A unique integer value identifying this circuit tag.	integer

Responses

HTTP Code	Schema
204	No Content

Tags

- tags

PATCH /tags/{id}/

Description

CRUD for Tags

Parameters

Type	Name	Description	Schema
Path	**id** *required*	A unique integer value identifying this circuit tag.	integer
Body	**data** *required*		Circuit Tag

Responses

HTTP Code	Schema
200	Circuit Tag

Tags

- tags

Definitions

Activation

Name	Description	Schema
token *required*	**Minimum length** : 1	string
uid *required*	**Minimum length** : 1	string

Circuit

Name	Description	Schema
author *optional*		integer

Name	Description	Schema
base64_image *optional* *read-only*		st ring (uri)
circuit_id *optional* *read-only*		st ring (uuid)
data_dump *required*	**Minimum length** : `1`	st ring
description *required*	**Minimum length** : `1`	st ring
last_updated *optional* *read-only*		st ring (dat e-t ime)
publish_request_time *optional* *read-only*		st ring (dat e-t ime)
sub_title *optional*	**Maximal length** : `200`	st ring
title *required*	**Length** : `1 - 200`	st ring

CircuitTag

Name	Description	Schema
description *required*	**Length** : `1 - 200`	st ring
id *optional* *read-only*		int eger
tag *required*	**Length** : `1 - 100`	st ring

ComponentAlternate

Name	Description	Schema
dmg *required*	**Minimum value** : `0` **Maximum value** : `32767`	integer
full_name *required*	**Length** : `1 - 200`	string
id *optional* *read-only*		integer
part *required*	**Length** : `1`	string
svg_path *required*	**Length** : `1 - 400`	string

Library

Name	Description	Schema
id *optional* *read-only*		integer
library_name *required*	**Length** : `1 - 200`	string
saved_on *optional* *read-only*		string (date-time)

LibraryComponent

Name	Description	Schema
alternate_component *optional* *read-only*		< ComponentAlternate > array

Name	Description	Schema
component_library *required*		st ring (uri)
data_link *required*	**Length** : `1 - 200`	st ring (uri)
description *required*	**Length** : `1 - 400`	st ring
full_name *required*	**Length** : `1 - 200`	st ring
id *optional* *read-only*		int eger
keyword *required*	**Length** : `1 - 200`	st ring
name *required*	**Length** : `1 - 200`	st ring
svg_path *required*	**Length** : `1 - 400`	st ring
symbol_prefix *required*	**Length** : `1 - 10`	st ring
thumbnail_path *required*	**Length** : `1 - 400`	st ring

PasswordResetConfirm

Name	Description	Schema
new_password *required*	**Minimum length** : `1`	st ring
token *required*	**Minimum length** : `1`	st ring

Name	Description	Schema
uid *required*	**Minimum length** : 1	st ring

ProviderAuth

Name	Description	Schema
access *optional* *read-only*	**Minimum length** : 1	st ring
refresh *optional* *read-only*	**Minimum length** : 1	st ring
user *optional* *read-only*	**Minimum length** : 1	st ring

Publish

Name	Schema
circuit *optional*	Circuit
published *optional*	boolean
reviewed_by *optional*	st ring (uri)
tags *optional* *read-only*	< Circuit Tag> array

SendEmailReset

Name	Description	Schema
email *required*	**Minimum length** : 1	st ring (email)

SetPassword

Name	Description	Schema
current_password *required*	**Minimum length** : 1	string
new_password *required*	**Minimum length** : 1	string

SetUsername

Name	Description	Schema
new_username *required*	Required. 150 characters or fewer. Letters, digits and @/.//-/_ only. + **Length** : `1 - 150` + **Pattern** : `"^[\\w.@-]+$"`	string

StateSave

Name	Description	Schema
data_dump *required*	**Minimum length** : 1	string
owner *optional*		integer
save_id *optional*		string (uuid)
save_time *optional* *read-only*		string (date-time)
shared *optional*		boolean

TokenCreate

Name	Description	Schema
password *optional*	**Minimum length** : 1	string
username *optional*	**Minimum length** : 1	string

User

Name	Description	Schema
email *optional*	**Maximal length** : 254	string (email)
id *optional* *read-only*		integer
username *optional* *read-only*	Required. 150 characters or fewer. Letters, digits and @/./+/-/_ only. **Minimum length** : 1	string

UserCreate

Name	Description	Schema
email *optional*	**Maximal length** : 254	string (email)
id *optional* *read-only*		integer
password *required*	**Minimum length** : 1	string
username *required*	Required. 150 characters or fewer. Letters, digits and @/.//-/_ only. + **Length** : `1 - 150` + **Pattern** : `"^[\\w.@-]+$"`	string

UsernameResetConfirm

Name	Description	Schema
new_username *required*	Required. 150 characters or fewer. Letters, digits and @/.//-/_ only. + **Length** : `1 - 150` + **Pattern** : `"^[\\w.@-]+$"`	string

Bibliography

[1] https://docs.docker.com/compose/ (Last accessed 30 May 2020).

[2] W. Reese, “Nginx: the high-performance web server and reverse proxy,” *Linux Journal*, vol. 2008, no. 173, p. 2, 2008.

[3] https://www.nginx.com/ (Last accessed 30 May 2020).

[4] https://www.djangoproject.com/ (Last accessed 30 May 2020).

[5] https://www.celeryproject.org/ (Last accessed 30 May 2020).

[6] https://redis.io/ (Last accessed 30 May 2020).

[7] https://www.mongodb.com/ (Last accessed 30 May 2020).

[8] https://www.mysql.com/ (Last accessed 30 May 2020).

[9] R. Soni, “Load balancing with nginx,” in *Nginx*, pp. 153–171, Springer, 2016.

[10] https://www.django-rest-framework.org/ (Last accessed 30 May 2020).

[11] https://gunicorn.org/ (Last accessed 30 May 2020).

[12] https://www.ansible.com/ (Last accessed 30 May 2020).

[13] https://jmeter.apache.org/ (Last accessed 30 May 2020).

[14] hhttps://swagger.io/ (Last accessed 31 May 2020).