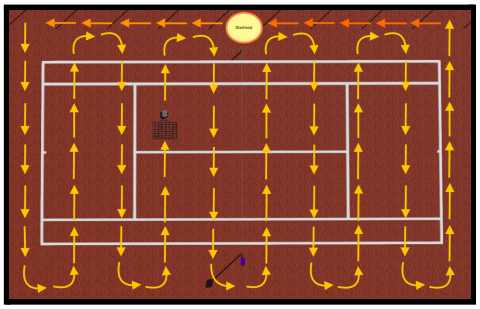
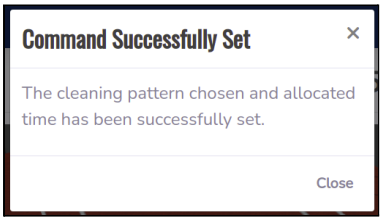
CLAYNCE USER GUIDE

Introduction

Welcome to Claynce! Our robot is designed to optimally sweep and level out the surface of a clay tennis court before, during, and after play - saving hours of menial labour. The robot comes in three parts, which are easy to assemble, and you don't need much technical knowledge to operate it! Once all parts are installed and combined, just register an account, link your robot to it, and start a program — the rest is done by the robot itself, meaning that you don't need to worry about anything.



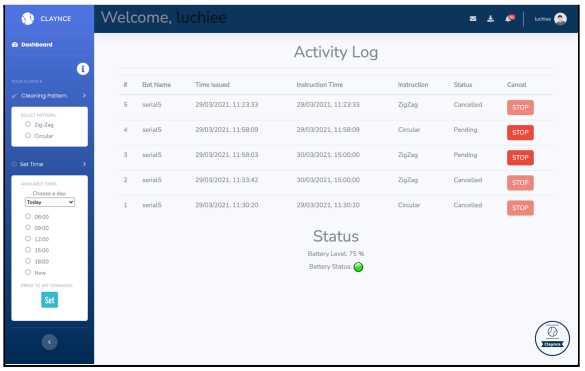


This guide will describe all the steps that need to be taken to properly install all the system parts, as well as create commands and take care of your robot. And if you still have questions, don't hesitate to contact our developers, who would be more than happy to assist you every step of the way!

Important safety information: don't place anything on top of the robot. Not suitable for any person under 16. If the court is outside, don't operate in wet weather and make sure the court is dry before use. Before using the robot, remove any objects lying on the court. Don't pour liquids on the robot.

Overview of the Interface:

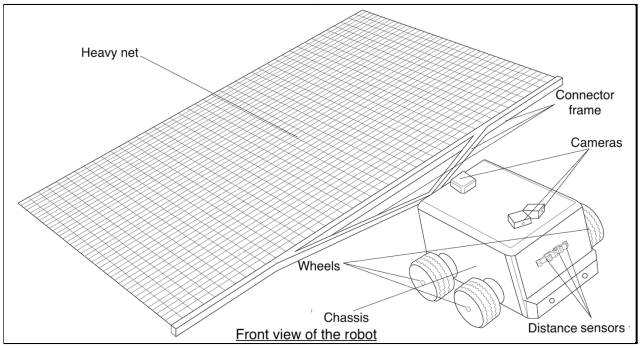
The main interaction with the robot takes place in a web application that can be viewed on both PCs and mobile devices. The web application has a login, register page, and dashboard page to give you an overview of the web application itself. The dashboard allows the user to control the robot's programs and gives the user the ability to choose the cleaning pattern and time to clean the court. All the labels and instructions for how to use the dashboard can be found in Appendix C.



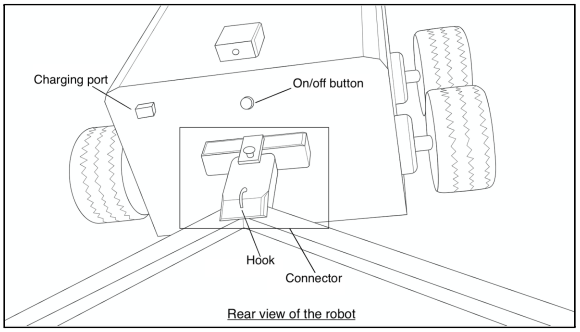
We ensured that the application is run with minimal installation required. Our team additionally believes in respecting each individual's privacy: each username and password is stored securely with a strong authentication protocol.

Overview of the Robot:

The robot's dimensions, while not connected to any attachment, are roughly 0.44x0.44x0.25m. While connected to the heavy net, the whole system takes up 1.71x1.5x0.25m. If the user chooses to remove the provided net and use their own, the robot with just the connector and no net takes up 0.72x0.90x0.25m.



Front view of the robot



Rear view of the robot

The labelled front-view diagram, top right, shows the components visible from the front:

- Chassis: all-sealed design ensures dust-proofing when working in a clay court environment.
- Wheels: suited with high-traction thread for sandy terrains. The wheels and axles are sealed inside tubes to adapt to sandy environments.
- Cameras (3pc): placed on top of the robot. They are used to detect ArUco markers installed on the court's walls for navigation purposes.
- Distance sensors (3pc): used to detect obstacles in front of Claynce to avoid a collision.
- On/off button: press once to power on Claynce and press twice to force a shutdown.
- Net: dragged behind Claynce to even out the clay court. Adapted from the traditional net to provide the tried and true clay texture. The net is 1.5x1m, with weight 5kg.
- Frame: connects the sweeping net with the robot body.

The major components labelled in the rear-view diagram are :

- Y-shaped connector: links the net frame to Claynce, allowing to attach more interchangeable parts in the future. The dimensions of the connector alone are 0.30x0.90x0.10m.
- Charging port: open the cover of the charging port and plug the cord in.
- Hook: can be used to hold the net's handle rope (if you're using your own net)

There is a QR code at the bottom of the robot that you need to scan to link it to your account. You can simply lift the robot to do it, as the robot by itself weighs less than 8kg.

Installation

Server Setup - Deploying a Claynce Server Instance

Installing Dependencies

In order to run the system, you'll need a number of dependencies. A setup script has been made that will download and install all of them and fetch the server's source code from our team's repository. Git, Python, cURL, PostgreSQL, Docker, Docker-Compose are the main dependencies that will be installed using this script. If you want to know more about them and what they do, see Appendix A.

The advantages of using a dockerized application shine through in this server application. When the system is started, the system will reach out to the Docker Hub to see if there is a newer version of the software. This ensures that wherever the system is deployed we are still able to push security or other patches.

The server can be set up by giving the necessary permissions to the install script and running it. You may be prompted for your password depending on the permissions of the user running the script. Make sure that no error messages have appeared while you were installing the dependencies, to ensure that all tools have been installed properly.

Download the installation script (>> shows the start of the line, and doesn't need to be typed):

```
>> wget https://raw.githubusercontent.com/SDP-G2/backend/master/install.sh
```

Give permission to the `install.sh` file:

```
>> chmod +x install.sh
```

Run the installation script

```
>> ./install.sh
```

If all steps have been done correctly, you should see the phrase "Do you want to continue?" at the bottom of your terminal window; type Y to complete the installation. If there are any error messages, contact support for help.

Running the Database Migrations

After running the `install.sh` script we can now move to the newly downloaded backend directory and perform the initial setup for the system.

```
>> cd backend
```

The following command performs three essential steps:

1. Starts running the database container in the background. This is done so that we have the database server running so we can connect to it.
2. Run the migrations for the database, which will create all of the required tables and constraints.
3. Finally, we can bring down the database container

Run the database migrations:

```
>> sudo make setup
```

During this process, you will be prompted for the database password. By default it is "password" (without quotes).

Running the Server Stack

After you run the migrations for the system, it can then be started in production mode. There are three commands that relate to starting, logging and stopping the system.

`>> sudo make run`

Uses `docker-compose` to start the SDP Container and the Database Server in the background. This allows the user to disconnect from the server and still have the system running, it also prevents multiple trivial issues such as broken pipes in an ssh connection from abruptly halting the system. This command will check that the container is the latest version of the image on the Docker Hub, if it is not it will pull down the latest version of the image - ensuring all instances of the software stay up to date and secure.

`>> sudo make logs`

Prints the server and database logs to the output (by default it is your shell/window), which allows the system to be debugged, or to monitor the traffic to the API and database services.

`>> sudo make stop`

Uses to safely spin down the containers and destroy them allowing the database to perform any required operations before it is backed to the volume.

Preparing the Court

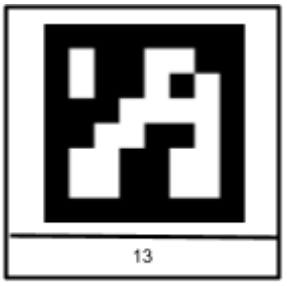
Choosing the Right Set of ArUcos

In order to use Claynce, you need to make a few adjustments to the surrounding of the tennis court on which you wish to use the product. Claynce's autonomous behaviour relies on the correct placement of the ArUco markers (sample marker on the right). They are captured by the cameras and used to determine the position on the court of the robot. For that reason, it is important to ensure that the layout of the markers on the court follows closely our guidelines.

You can choose one of the three sets of markers to be used with your robot:

- 40 pcs of 40cm x 40cm ArUco markers
- 40 pcs of 50cm x 50cm ArUco markers
- 40 pcs of 60cm x 60cm ArUco markets

The rule of thumb is simple: the bigger the marker is, the more efficiently your robot will clean the court for you. However, we understand that markers are a bit intrusive, hence we provide various size options.



Sample ArUco Marker

Placing ArUcos Around the Court

The system configuration procedure is the following:

1. Set markers around the tennis court following the guidelines outlined on the next page.
2. For each marker write down its coordinates with respect to the court centre
    - Note A: the layout of XY axes is presented in the image below
    - Note B: you can choose either side to correspond to the negative/positive coordinates. Just remember that the robot will start in the corner at negative XY coordinates and follow patterns in a positive Y direction

3. Contact us and pass us the coordinates that correspond to each of the markers. Our team will configure the robot for you and deliver it to your specified address.

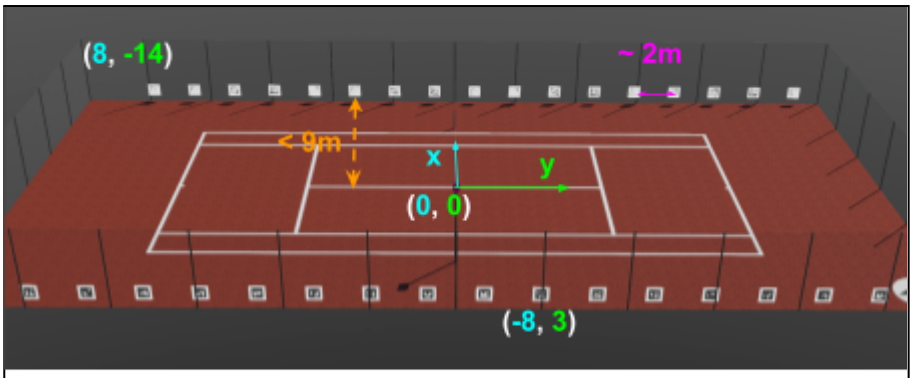
Recommendations for choosing the position of markers

1. Markers should be placed along with the court, on both sides
2. Markers should be placed, if possible, vertically. If you need to put them at an angle make sure to notify us about it so we can implement necessary adjustments.
3. The marker's number should be at the bottom of the marker.
4. Markers should be placed 60 centimetres above the ground.
5. If the robot is standing against one wall, it should be able to see the markers on that same wall
6. The markeres shouldn't be placed further than 9 meters away from the court's centerline
7. Consecutive markers should be 2 meters away from each other

Following those guidelines will ensure that the robot will sweep your court as efficiently as possible. However, don't worry if you can't meet all of them exactly. E.g. if you need to have a gap of 3 meters between two markers or it would be easier for you to place them 45 centimetres above the ground - it will be fine. The system was built with robustness in mind and the proposed values are the most optimal ones based on our empirical testing.

The most important part is to get the position measurements for each marker right. A 5-centimetre error for one marker will not break the system, but accumulated errors across many markers might cause the faulty behaviour of the robot.

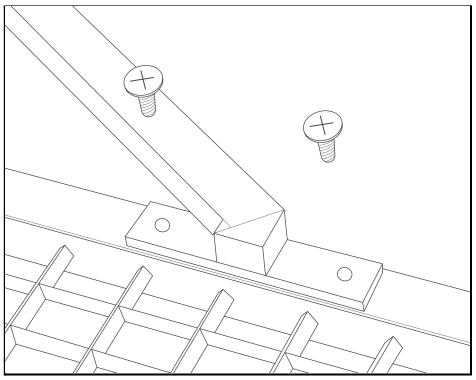
On the right you can see an example layout of ArUco markers around the court. It also visualises some of the aforementioned guidelines.

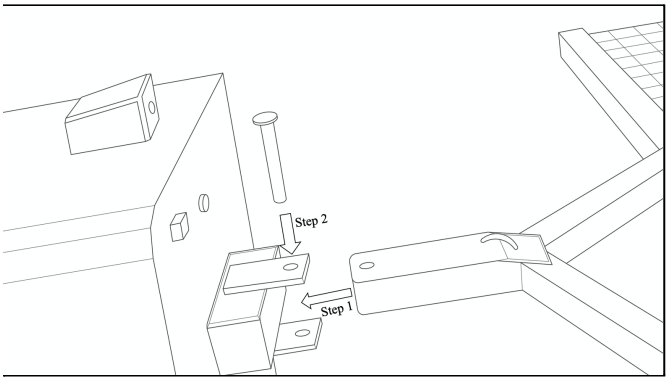


Recommended layout of ArUco markers around the court.

Robot Unboxing

The main body of our robot is pre-assembled and ready to operate. The robot allows you to either use our net, or your own if you prefer. Whatever the case, simply drill the holes and screw the Y-frame provided to the bar of the net, following the instructions below (the screws are provided; make sure the Y-frame is in the centre of the net). For best results, we advise the net to have length 0.6m-1.4m and width 1.2m-1.7m and weigh no more than 5kg.





To connect the sweeping net to Claynce, lift the hinge's pin up and align the hole on the Y-frame with the connector on the robot body. Push the pin back down to secure the connector in place.

Account setup

After you have setup the server, you can go onto the web browser to register your user details: **http://localhost:8000/static/register.html** (more information on layout can be found in Appendix C)

New User:

1. Click on "New User", the website would redirect you to the register page
2. In the Register Page, fill in your username and password details
3. Fill in a UUID that corresponds to the cleaning robot (or scan a QR code at the bottom of the robot itself)
4. This would then redirect you to the dashboard page automatically.

If you have already registered your user details, you can go to the following URL: **http://localhost:8000/static/login.html**

Existing User:

1. Fill in your existing username and password
2. This would then redirect you to the dashboard, if your details are correct.

Operation

1. Turn on the robot by pressing the button on the back of it
2. The robot will perform the initial localisation
3. Create a command either for now or for the future
4. The robot will start the command at the appropriate time

Launching the robot:
Turn on the robot by pressing the ON/OFF button on the back of the robot. In order to ensure that the robot orients itself in the court properly, put it next to the markers outside the court. Once it is turned on, feel free to move the robot around when it is not carrying out the task, as long as it's placed in the proximity of the markers when it is about to start the task. When the scheduled start of the task comes, the robot will start the sweeping process from half of the court that it is facing when the task starts.

Creating a command:
After you login to the app at http://localhost:8000/static/login.html, you are allowed to select either a zig-zag or circular cleaning pattern and even set a specific time to allow the robot to clean the court up to three days in advance. Once both configuration options are selected, you then instruct the robot to clean by clicking on the "Set" button, and it will start either now, or at a later specified time.

Obstacles:
The robot is designed to stop and pause its task if it encounters a reasonably large obstacle in front of it. If you see this happening, come up to the robot from the side that hasn't been cleaned yet (to avoid messing up the freshly-swiped court). Slowly remove the obstacle. The robot should start moving automatically after that. If you notice a small obstacle (e.g. racket) on the robot's path and don't want to cancel a command, simply walk up to the robot from the front. It should notice you and stop before you and the obstacle, giving you enough space and time to remove it.

Finishing the cleaning problem:
If you see that the robot has completed a program (it has stopped close to the net on the side of the court and the program is listed as complete in your application), check the battery level. If you prefer, you can then remove the robot inside by gently lifting and carrying it; otherwise, you can leave it on the court — the robot's algorithm allows it to go back to its starting position by itself once a new command is issued.

Interrupting the pattern / Stopping the robot:
If the robot is about to crash into something and doesn't seem to be stopping, or you accidentally start a command that you didn't want to, you can press the red STOP button to immediately stop the robot from moving or delete a pending command.

Maintenance

Charging the robot:
The robot is designed to work with the usual 220V sockets. Just plug the cord into the robot's charging port and wait until the website shows a full charge (or for a couple of hours). If you see that the robot holds the charge less and less over time, consider changing the battery inside the robot - contact support for replacement instructions.

Cleaning the robot's parts:
The robot's sensors and joints are susceptible to dust and particles that can accumulate over time. To ensure Claynce performs at its best, we recommend our users to carry out maintenance weekly or after extended uses. Before maintenance, make sure the robot is powered off.

Wipe the two front and one back cameras with a clean, dry cotton or microfiber cloth. Use a cotton swab to remove any particles caught by the distance sensors in front. For resilient stains, gently apply pressure with a cloth sprayed with alcohol. DO NOT spray cleaning solution directly onto the devices.

Remove the wheels by pulling on them firmly. Clean the wheel cavity, paying attention not to scratch the plating. Occasionally, or if you feel heavy friction when you try to spin the wheels, apply oil-based lubricants to the axles. Reinstall the wheels to their axles and make sure they are secured in place.

Troubleshooting

If your problem is not resolved after consulting this section, please contact our support.

Problem	Solution
Marker Setup Problems	
Markers are too big	If you have 50x50 markers, feel free to contact us and we will arrange the return of the markers. Instead, we will send you a set of 40x40 markers to try out. Otherwise, unfortunately, there is not much we can do to help at this stage of the development.
I changed the position of a marker and now the robot is not sweeping the court correctly	After changing the position of any marker, you should contact us, so we can manually update the robot's software. In the future, we hope to automate this process using OTA updates or a simple configuration panel on our websites so you can do it anytime yourself.
Hardware (robot-related) Problems	
The robot has problems following the pattern	This indicates problems with marker detection. Try cleaning all the camera lenses and wiping the dirt/dust from the markers - especially the parts containing the ArUco code.
The robot doesn't turn on	Check battery level, charge overnight.
Wheels are not moving OR the robot is stuck in one place	Check if something is jammed underneath the robot or between a wheel and the body. Remove any debris, clean and oil the wheel joints (see Maintenance).
The robot doesn't detect an obstacle	Dirty sensors. Cancel the program, and gently clean the sensors (see Maintenance). Restart the robot
Application-related Problems	
Can't create a command	Ensure that one cleaning pattern is selected. If you have selected a fixed time and date, ensure the respective time and date is not set in the past. If the settings on the frontend are valid then we must check that the device connected to the internet and the connection to the backend server is not obstructed by a firewall or other means. If the issues continue, please contact support.
Stop button not working	In case of emergency, manually intervene to turn off the robot (press the OFF button - see Introduction - Overview of the robot) and immediately contact customer service.
The task does not execute when scheduled	Ensure that the robot is powered on (if the robot detects an internal error it will power off, as a safety precaution). If the robot is powered on and the task is still not executing, restart the robot in order to ensure that there is not a connection issue between the server and robot. If the issues continue, please contact support.

Contacts[1]

Robot and everything to do with it: robot-help@claynce.com
Web application and everything to do with it: web-help@claynce.com

[1] Please note that these email addresses are for demonstration purposes only, and are not active or monitored

References

https://prod-help-content.care.irobotapi.com/files/s_Series/s9/ownersGuide/ownersGuide_enUS.pdf
https://github.com/SDP-G2/backend
https://github.com/SDP-G2/frontend
https://github.com/SDP-G2/hardware

Appendix A: Local Server Dependencies

Git:
One of the first dependencies to be installed is git, which allows us to later download the source code for the Backend System and check its integrity

Python:
Python is also a dependency of the Backend System, it is used only for a small script that simply performs a get request to the Docker Hub and gets the Latest Tag for our SDP Backend System. This ensures that if a change is made to the system when the system administrator updates the software they will get the latest version.

cURL:
The next dependency to be installed is cURL. This command-line utility is again used by the install script to install other essential software

PostgreSQL:
One of the most essential dependencies of the system is PostgreSQL, we are only using this for the psql client. This allows us to later connect to the running database container and run migrations in a convenient manner, without having to attach to the database container, or another utility container - none of these options are considered good practice, since they create a considerably larger attack surface for the system with no real benefit.

Docker:

Docker is then installed onto the system using the provided install script, which is very convenient. docker-compose is installed in the exact same way. These tools are used to run our SDP Backend System Image, using docker-compose to orchestrate the startup, logging, stopping, etc for our backend container and database.

Appendix B: Local Server Setup (expected outputs)

```
>> chmod +x install.sh
>> ./install.sh
```

```
kyle@kylw:~/Desktop$ chmod +x install.sh
kyle@kylw:~/Desktop$ ./install.sh
[sudo] password for kyle:
Reading package lists... Done
Building dependency tree
Reading state information... Done
Note, selecting 'python-is-python2' instead of 'python'
The following packages were automatically installed and are no longer required:
  libfprint-2-tod1 linux-headers-5.4.0-42 linux-headers-5.4.0-42-generic linux-image-5.4.0-42-generic linux-modules-5.4.0-42-generic
  linux-modules-extra-5.4.0-42-generic
Use 'sudo apt autoremove' to remove them.
The following additional packages will be installed:
  liberror-perl libpq5 libpython2-stdlib libpython2.7-minimal libpython2.7-stdlib postgresql-12 postgresql-client-12 postgresql-client-common
  postgresql-common python2 python2-minimal python2.7 python2.7-minimal sysstat
Suggested packages:
  git-daemon-run | git-daemon-sysvinit git-doc git-el git-email git-gui gitk gitweb git-cvs git-mediawiki git-svn postgresql-doc postgresql-doc-12
  libjson-perl python2-doc python-tk python2.7-doc binfmt-support isag
The following NEW packages will be installed
  curl git git-man liberror-perl libpq5 libpython2-stdlib libpython2.7-minimal libpython2.7-stdlib postgresql postgresql-12 postgresql-client-12
  postgresql-client-common postgresql-common python-is-python2 python2 python2-minimal python2.7 python2.7-minimal sysstat
0 to upgrade, 19 to newly install, 0 to remove and 0 not to upgrade.
Need to get 24.7 MB of archives.
After this operation, 103 MB of additional disk space will be used.
Do you want to continue? [Y/n]
```

Figure 1: Installing all of the required dependencies.

```
>> sudo make setup
```

```
kyle@kylw:~/Desktop$ cd backend/
kyle@kylw:~/Desktop/backend$ sudo make setup
docker-compose up -d sdp_db
WARNING: The LAST_TAG variable is not set. Defaulting to a blank string.
Pulling sdp_db (postgres:)...
latest: Pulling from library/postgres
6f28985ad184: Pull complete
163a60947b3a: Pull complete
1791984387e5: Pull complete
ccf9c39579c4: Pull complete
1d8dd50a5ee9: Pull complete
3991abc55a94: Pull complete
4cf2cdef0857: Pull complete
ed1bec410498: Pull complete
0930368b9a14: Pull complete
a[illegible]2936fdb5: Pull complete
bb3d505cd0cb: Pull complete
4f1bb2dd6f16: Pull complete
8d3f6ff7b2da: Pull complete
687caf1b1f9b: Pull complete
Digest: sha256:c83014a2b46834ef6d17f64c8e4a70089901a8c0dee158f1ca5ccae032ea32e5
Status: Downloaded newer image for postgres:latest
Creating sdp_db ... done
psql -U postgres -d sdp -h localhost --single-transaction -a -f database/up.sql -f database/robots.sql
Password for user postgres:
```

Figure 2: Pulling the latest database container image.

```
CREATE TABLE IF NOT EXISTS Robot (
       robot_serial_number VARCHAR PRIMARY KEY,
       battery_level BIGINT NOT NULL DEFAULT 0,
       assigned BOOLEAN NOT NULL DEFAULT FALSE
);
CREATE TABLE
CREATE TABLE IF NOT EXISTS Users (
       user_id BIGSERIAL PRIMARY KEY,
       user_name VARCHAR NOT NULL UNIQUE,
       password_hash VARCHAR NOT NULL,
       robot_serial_number VARCHAR NOT NULL UNIQUE REFERENCES Robot(robot_serial_number)
);
CREATE TABLE
CREATE TABLE IF NOT EXISTS Commands (
       command_id BIGSERIAL PRIMARY KEY,
       robot_serial_number VARCHAR NOT NULL REFERENCES Robot(robot_serial_number),
       time_issued timestamptz NOT NULL,
       time_instruction timestamptz NOT NULL,
       instruction VARCHAR NOT NULL,
       status VARCHAR NOT NULL DEFAULT 'Status::Pending'
);
CREATE TABLE
INSERT INTO Robot(robot_serial_number, assigned) VALUES
('serial1', false),
('serial2', false),
('serial3', false),
('serial4', false),
('serial5', false),
('serial6', false),
('serial7', false),
('serial8', false),
('serial9', false),
('serial10', false)
INSERT 0 10
docker stop `docker ps -aq`
1e8446239de8
```

Figure 3: Performing the database migrations.

```
>> sudo make run
```

```
kyle@kylw:~/Desktop/backend$ sudo make run
LAST_TAG=`python last_tag.py` docker-compose up -d
Pulling sdp_backend (kylecotton/sdp-backend:bede1e9)...
bede1e9: Pulling from kylecotton/sdp-backend
d94d38b8f0e6: Pull complete
1cfcc1bb8434: Pull complete
6f78ac1091ad: Pull complete
7733e081174e: Pull complete
2288c628e1a2: Pull complete
Digest: sha256:7769b95011e862a7d5a3391a0b979b78c3eda523b4daf9631de400ef331bc5d2
Status: Downloaded newer image for kylecotton/sdp-backend:bede1e9
Starting sdp_db      ... done
Creating sdp_backend ... done
```

Figure 4: Starting the entire backend system.

```
>> sudo make logs
```

```
kyle@kylw:~/Desktop/backend$ sudo make logs
docker-compose logs
Attaching to sdp_db, sdp_backend
sdp_db         |
sdp_db         | PostgreSQL Database directory appears to contain a database; Skipping initialization
sdp_db         |
sdp_db         | 2021-03-25 01:58:54.080 UTC [1] LOG:  starting PostgreSQL 13.2 (Debian 13.2-1.pgdg100+1) on x86_64-pc-linux-gnu, compiled by gcc (Debia
n 8.3.0-6) 8.3.0, 64-bit
sdp_db         | 2021-03-25 01:58:54.081 UTC [1] LOG:  listening on IPv4 address "0.0.0.0", port 5432
sdp_db         | 2021-03-25 01:58:54.081 UTC [1] LOG:  listening on IPv6 address "::", port 5432
sdp_db         | 2021-03-25 01:58:54.084 UTC [1] LOG:  listening on Unix socket "/var/run/postgresql/.s.PGSQL.5432"
sdp_db         | 2021-03-25 01:58:54.090 UTC [27] LOG:  database system was shut down at 2021-03-25 01:58:42 UTC
sdp_db         | 2021-03-25 01:58:54.094 UTC [1] LOG:  database system is ready to accept connections
```

Figure 5: Display all of the logs for the entire backend system.

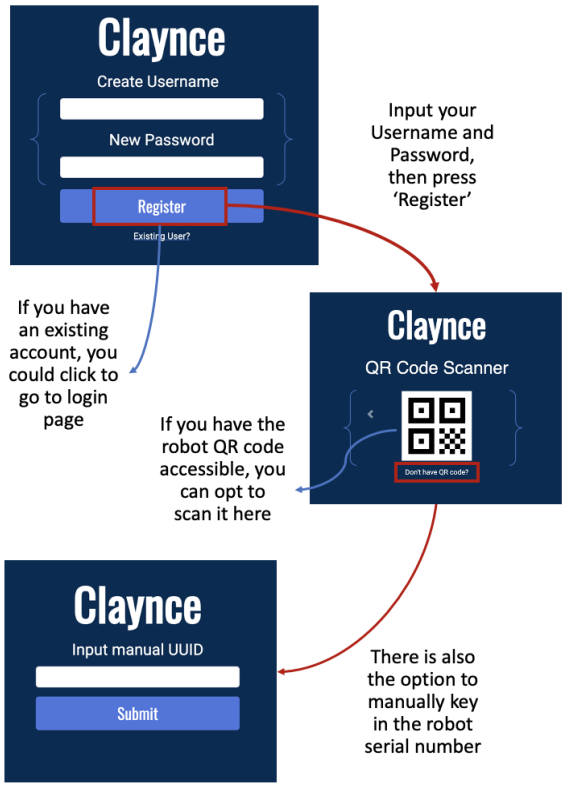
```
>> sudo make stop
```

```
kyle@kylw:~/Desktop/backend$ sudo make stop
docker-compose down
WARNING: The LAST_TAG variable is not set. Defaulting to a blank string.
Stopping sdp_db      ... done
Stopping sdp_backend ... done
Removing sdp_db      ... done
Removing sdp_backend ... done
Removing network backend_default
```

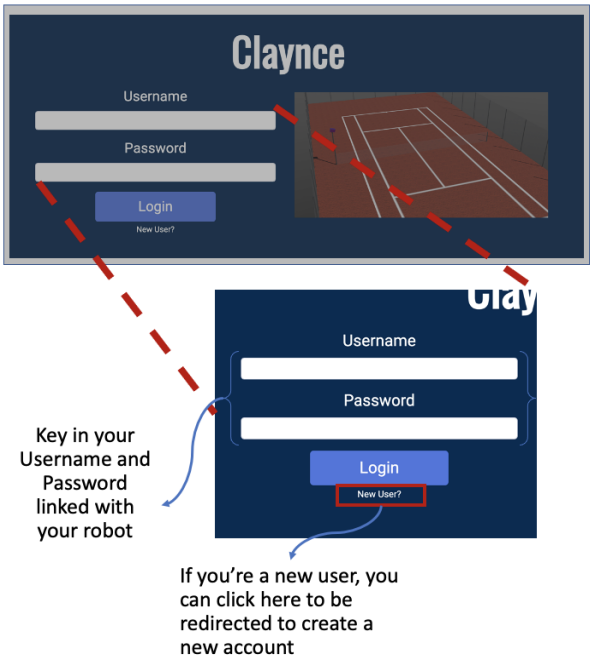
Figure 6: Stop all of the running services.

Appendix C: Visual User Guide for UI

REGISTRATION



LOGIN



BOT COMMAND CREATION

DASHBOARD