Setup the Pi:

1. **user**: magicmirror
2. **password**: raspberry
3. The app bar is hidden by default, after logging in, either move the cursor to the top of the screen or move your finger to the top for the bar to appear
4. Open terminal and run the command `onboard` to start the on screen keyboard
5. To fix the orientation of the touch screen, enter `./portrait.sh` in the terminal (this will open a vi window, you can close it)
6. In the terminal run command `sudo raspi-config`, then select `general`, then `audio` to change the default sound device to the speaker (USB 2.0 Device, not HDMI)
7. You can adjust the volume (recommend 100%) using the `alsamixer` command

Run the Magic Mirror Application:

1. Open up the terminal on the Pi
2. Open new terminal and type in `cd magicmirror`
3. Run the command `npm run start` to start up the application

How it Works:

- Upon starting the application you will be presented with the following screen illustrated in figure 1
- In the bottom right corner, you will see a lock icon, once you click that, you will be presented with the screen in figure 2
- Enter the passcode **1234** and hit the unlock button
- Once unlock is successful, you will be shown a similar screen as figure 3
- Here you are able to input medications into containers and schedule medications
- Figure 4 is labeled to show you how to navigate the UI

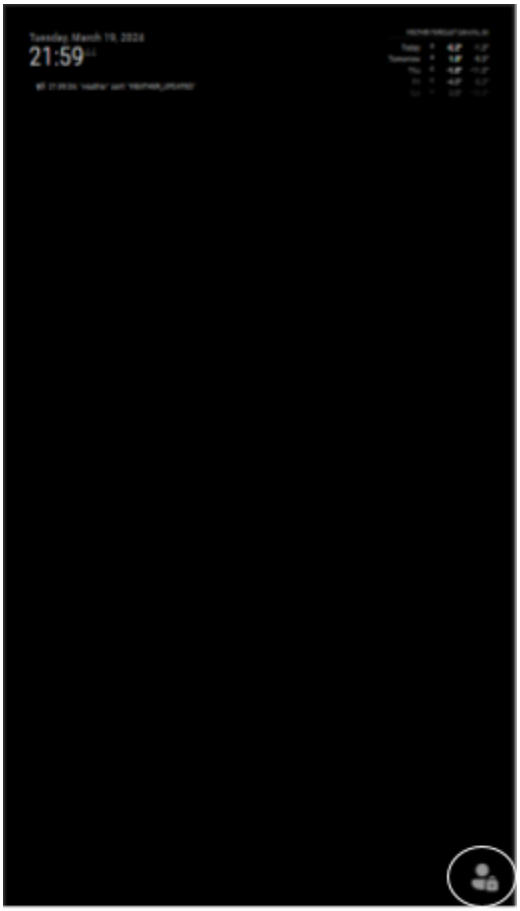


Figure 1

Figure 2

Figure 3

Figure 4

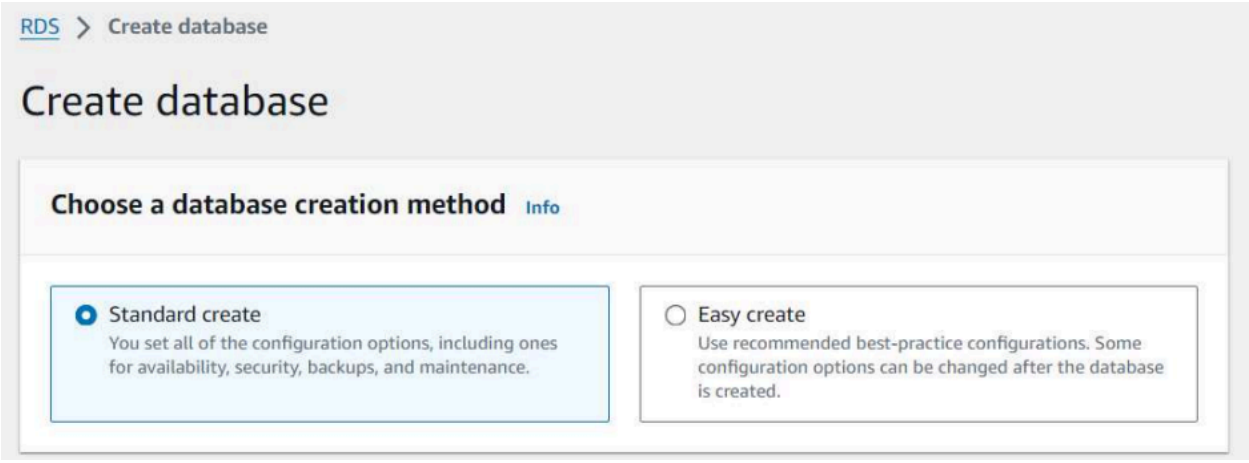
1. Search the brand name, generic name or NDC for the medication you want to add
2. Select from the options that pop up, if the medication is not listed, enter more characters in 1
3. Enter the quantity of pills to be stored in the pillbox
4. Select the pillbox to store the medication in
5. Add medication and repeat 1-5 for all medications
6. Select from the dropdown the medication that you want to schedule for the patient
7. Select all the days that the patient takes that medication
8. Select all the times of the day the patient has to take that medication
9. Schedule medication and repeat 6-9 for all the other medications
10. Click the lock button to lock the modules

- The cabinet is set up and ready to be used by the patient
- Medication Alarm will be triggered when there is a scheduled medication and pill will be dispensed from the set pillbox accordingly

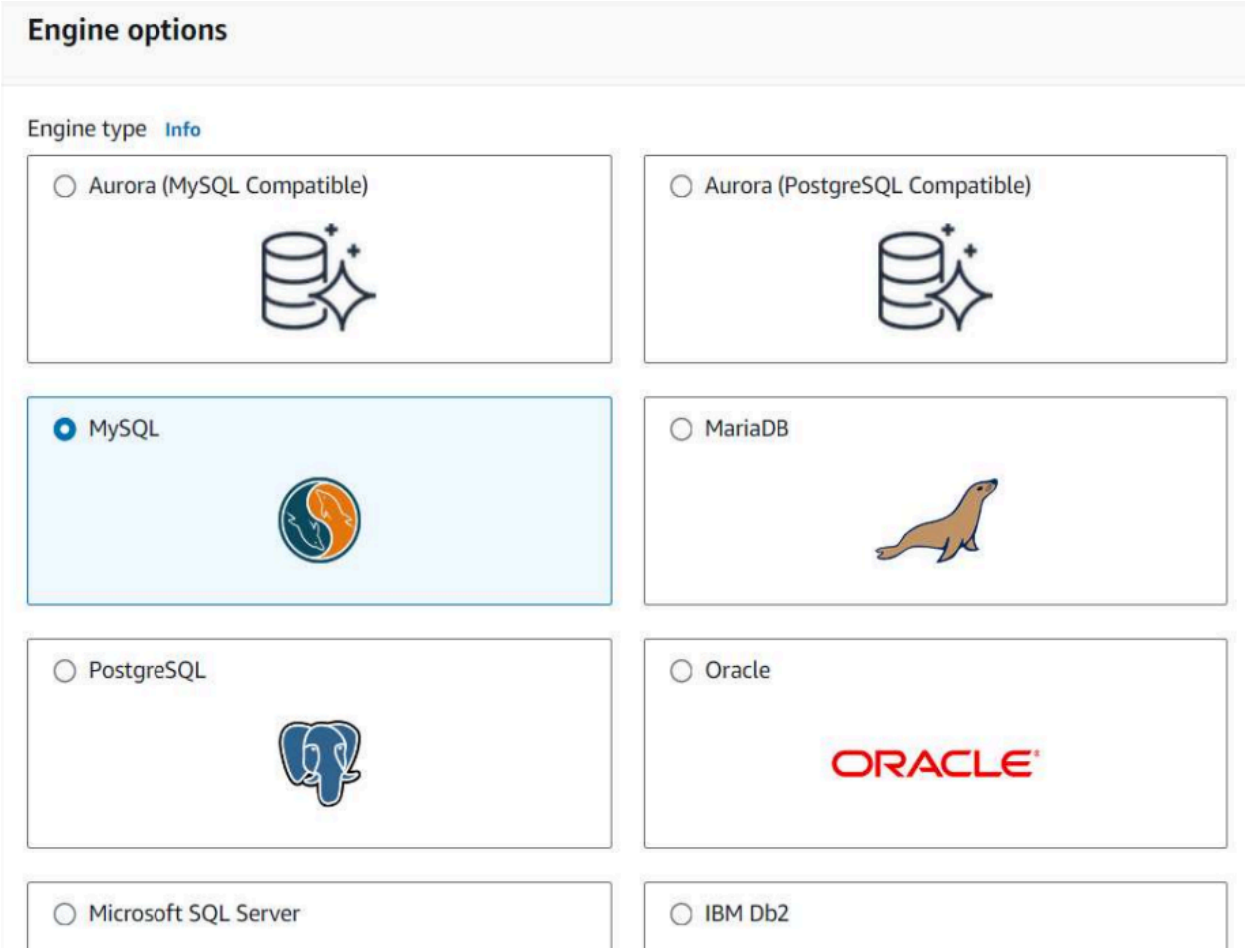
All the source code can be found at: [Smart Medicine Cabinet]
Any changes to the UI, adding additional modules or changing the passcode can be done by cloning the repository.

Setting up AWS RDS

1. All the current accounts for the portal are not functioning anymore due to costs. The github repo has the code required for one to deploy on their own server. Once registered with an account and logged in, navigate to the "RDS" service by either typing "RDS" in the search bar or locating it under the "Database" section.
2. Click on the "Create database" button.



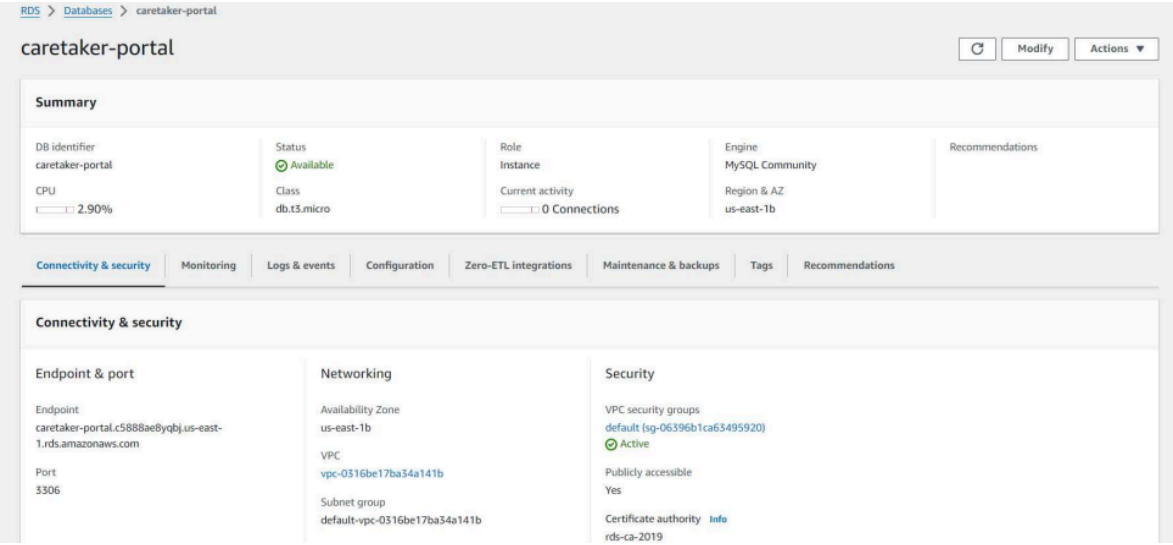
3. Choose the MySQL database engine



4. Select the appropriate engine version

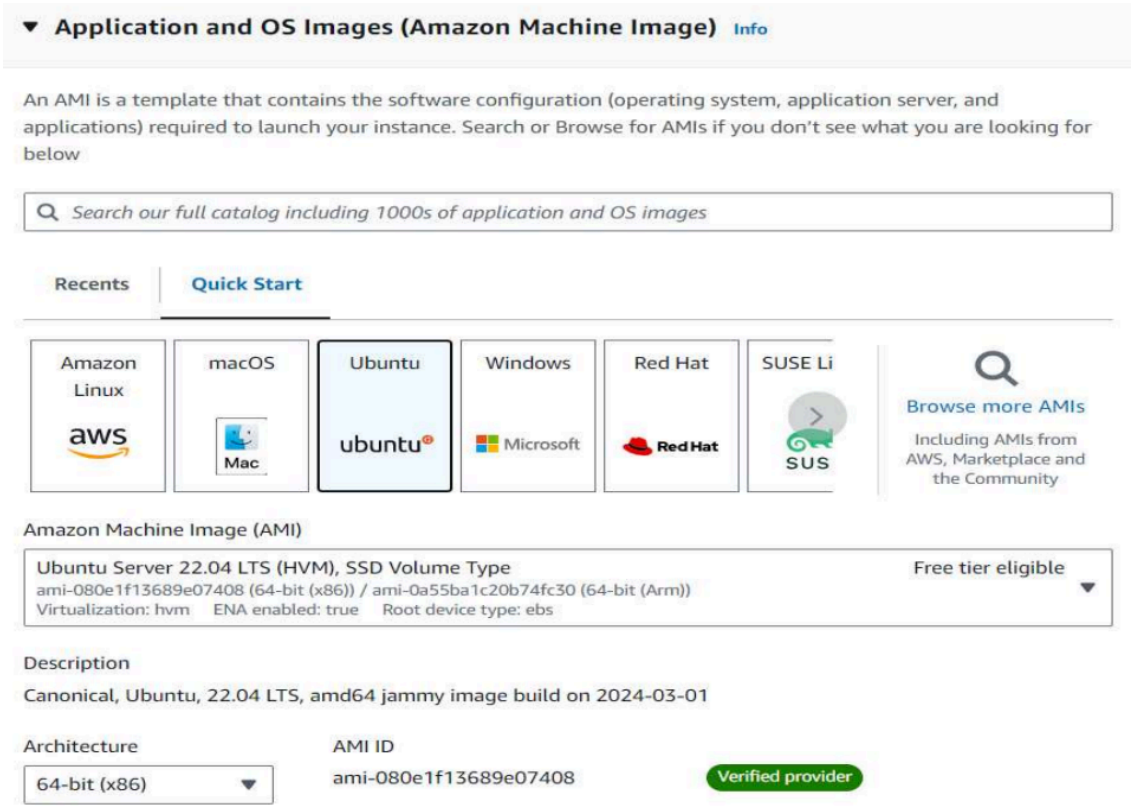
6. Provide a name for your database instance.
7. Set the username and password for the master user. Ensure it is a strong password.
8. Select the DB instance class based on your performance and capacity requirements.
9. Specify the amount of allocated storage.
10. Configure other optional settings such as VPC, subnet group, etc. as per your requirements.
11. Set up backup retention period according to your data recovery needs.
12. Choose whether to enable automatic backups and specify the preferred backup window.
13. Review all the configurations you've made.
14. Click on "Create database" to launch the RDS instance.

Once the RDS instance is created, copy the endpoint & port, username & password.

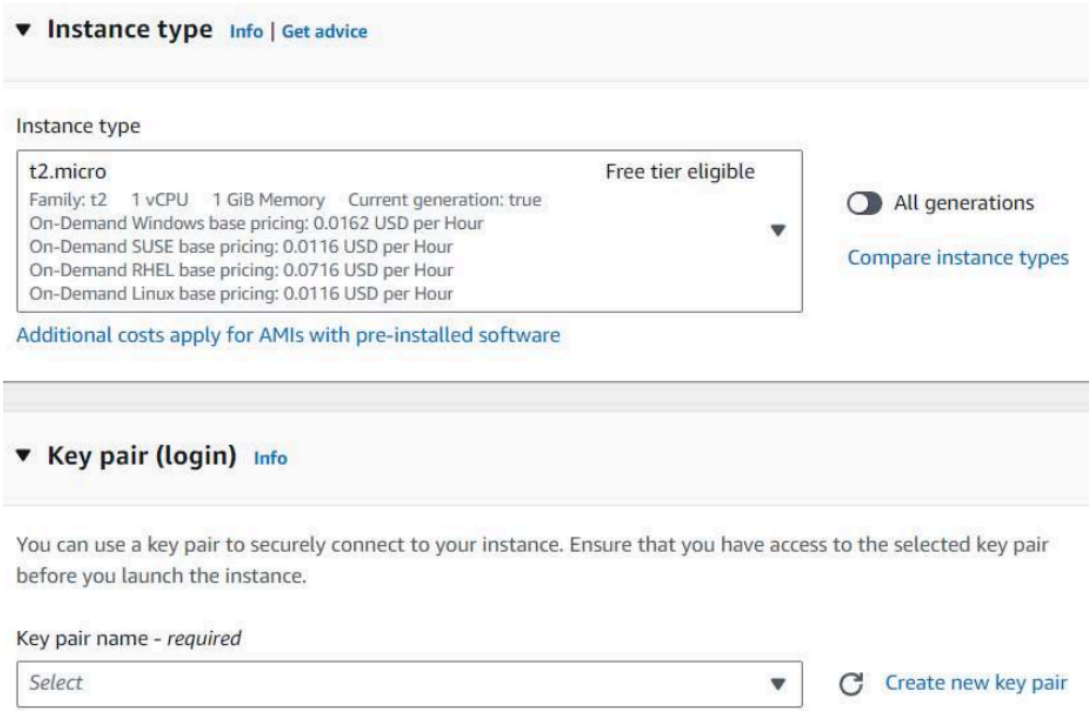


Creating AWS EC2 Instance

1. Once logged in, navigate to the "EC2" service by either typing "EC2" in the search bar or locating it under the "Compute" section.
2. Click on the "Launch Instance" button to begin the process of creating a new EC2 instance.
3. Select an AMI based on your requirements. This could be a pre-configured image provided by AWS or a custom AMI.
4. Choose the AMI based on the operating system and software stack you want to use.



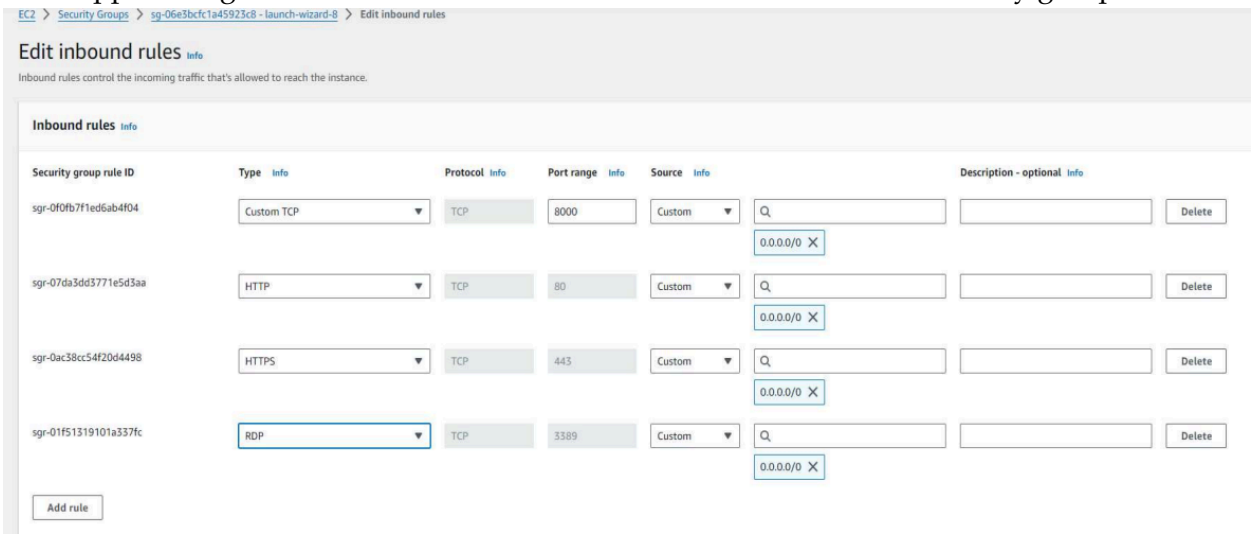
5. Select the instance type based on your workload requirements. Instance types vary in terms of CPU, memory, storage, and network capacity.
6. Create a new security group or select an existing one.



7. Define inbound and outbound rules to control the traffic to and from your instance. At least SSH (port 22) should be open if you plan to access the instance via SSH.

For Example:

If our app is using 8081 then it should be added to inbound rules of the security group.



8. Review all the configurations you've made for your instance.
9. Choose an existing key value pair or create a new key pair for SSH access to your instance
10. Click "Launch Instances" to create and launch your EC2 instance.
11. Here a .pem file will be downloaded automatically. By using that file we're going access the file system of that instance.
12. Once the instance state changes to "running" state, then you can connect to it using SSH or other remote access methods.

Connect to EC2 host using SSH host

1. Install the "Remote - SSH" extension in VS Code. You can find and install it from the Extensions view, then search for "Remote - SSH" and click install.
2. Click on Connect to Host, and "+ Add new SSH Host…" This popup will be displayed.
4. Here paste the SSH Command that present in Connect to Instance page, and there are list of steps to connect with Instance.
5. Now open the Configuration file and make sure the path of the .pem file is same as .pem file's location.
6. When you click on that Host, we are now connected with EC2 instance from VS Code. We can start developing our website in that workspace.

Updating Environment Variables on the Raspberry Pi

The credentials associated with the database can be found in the .env file in the root of the magicmirror folder. Update these variables with the new ones you created above.

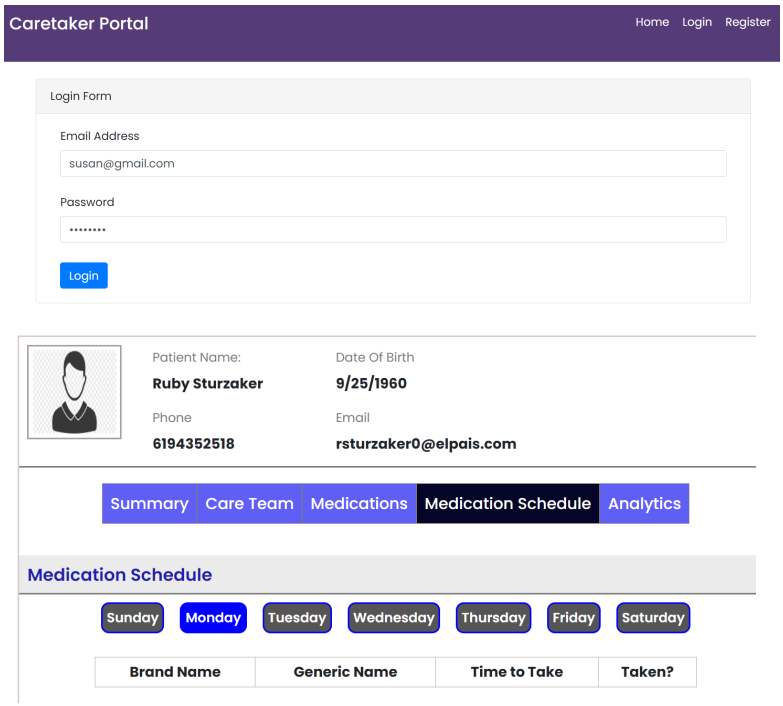
If changing the domain, the `cloud_url` variable will need to be updated in the `magicmirror/modules/cloud/node_helper.js` file. The default is *caretakerportal.com*.

Currently, the magicmirror installation has hard coded `patient_id` and `cabinet_id` variables, also found in the previously mentioned `cloud/node_helper.js` file. If updating

to support multiple patients or wish to change the associated patient, these will need to be manually changed to the UUIDs created in the database.

Developing Caretakerportal.com

1. Create a directory for our application using mkdir command.
2. Open cmd in that directory and perform the following commands to setup our application.
   `npm init -y`
3. Now install the required libraries using the npm i command.
   `npm install mysql axios nodemon`
4. Create a file called app.js or server.js file which will be the main entry point of your application.
5. Inside that file you can start writing your application code.
6. To start the application, use this command
   `npm run start` (or) `npm start`



How it Works:

- Once the Magic Mirror application runs, after the patient takes their medication, the quantity of the medication reduces under medications table
- Caretaker sets alarms for different medications
- Updates in the cloud portal to verify if the patient has taken medication or not
- Caretaker can access and review detailed graphs for analytical purposes

All the source code can be found at: Cloud Portal

Any changes to the UI, or adding additional functionalities can be done by forking the repository.

Setting up a new Raspberry Pi

To set up a new raspberry pi, the following steps must be followed:

1. Clone the magicmirror repository: https://github.com/madlitch/magicmirror
2. Install OpenCV (follow this tutorial: https://github.com/Qengineering/Install-OpenCV-Raspberry-Pi-64-bits)
3. Set up the camera: https://docs.arducam.com/Raspberry-Pi-Camera/Native-camera/Quick-Start-Guide/
4. Install onboard: https://launchpad.net/onboard
5. Copy over the portrait.sh file from the user folder on the provided Pi
6. GPIO pins are set up like this: (HWPin is the physical pin number, GPIO ## is the GPIO output, and pinID is the number associated with the relay/dispenser, as in the following guide: https://pinout.xyz/pinout/pin2_5v_power/). The LED is connected to pinID 8.

```
GPIOPin( pinID: 1,  HWPin: 16)); //GPIO 23
GPIOPin( pinID: 2,  HWPin: 11)); //GPIO 17
GPIOPin( pinID: 3,  HWPin: 13)); //GPIO 27
GPIOPin( pinID: 4,  HWPin: 15)); //GPIO 22
GPIOPin( pinID: 5,  HWPin: 29)); //GPIO 5
GPIOPin( pinID: 6,  HWPin: 31)); //GPIO 6
GPIOPin( pinID: 7,  HWPin: 36)); //GPIO 16
GPIOPin( pinID: 8,  HWPin: 37)); //GPIO 26
```

7. The relay board is connected to both +5v and ground on the Pi, and so is the fan. Both have their own independent connections, use the link to the pinout provided above.