Deployment Instructions

Setting up a Database and Deploying the Server

** You will first need to purchase a host for your server to run on (a virtual computer to run your server continuously). The working prototype has been running on a Cybera Rapid Access Cloud instance.
** These instructions assume the host is a Ubuntu machine

1) To begin the process, you will download and install all of the items that you need from the Ubuntu repositories. Run the following commands:

```
sudo apt update
sudo apt install python3-venv python3-dev libpq-dev postgresql
postgresql-contrib nginx curl
```

2) Create a PostgreSQL Database and User for the django application. Run the following set of commands:

- **sudo -u postgres psql**
- **CREATE DATABASE daycaremgmt;**
- **CREATE USER postgres WITH PASSWORD 'password'; (Take password used in settings.py and replace in the command)**
- **ALTER ROLE postgres SET client_encoding TO 'utf8';**
- **ALTER ROLE postgres SET default_transaction_isolation TO 'read committed';**
- **ALTER ROLE postgres SET timezone TO 'UTC';**
- **GRANT ALL PRIVILEGES ON DATABASE daycaremgmt TO postgres ;**
- **\q**

3) Now that the database is set up, clone the repository into the VM (virtual machine), using:

```
git clone {github clone link}
```

4) Activate the virtual environment in the backend folder

```
cd daycare-mgmt
cd daycare-mgmt
source myprojectenv/bin/activate
```

5) Install all the required packages included in requirements.txt, by running the following command:

```
pip install -r requirements.txt
```

6) Make sure to add the ip address of your VM to ALLOWED_HOSTS section in the project's setting.py file

7) Now, you can migrate the initial database schema to our PostgreSQL database using the management script:

```
python manage.py makemigrations
python manage.py migrate
```

8) Create a super user to use the admin website:

```
python manage.py createsuperuser
```

9) You can collect all of the static content into the directory location that you configured by typing:

```
python manage.py collectstatic
```

10) You're now finished configuring your Django application. You can back out of our virtual environment by typing:

```
deactivate
```

Creating systemd Socket and Service Files for Gunicorn

11) Start by creating and opening a systemd socket file for Gunicorn with sudo privileges:

```
sudo nano /etc/systemd/system/gunicorn.socket
```

12) Copy and paste the following in the file:

```
[Unit]

Description=gunicorn socket

[Socket]

ListenStream=/run/gunicorn.sock

[Install]

WantedBy=sockets.target
```

13) Next, create and open a systemd service file for Gunicorn with sudo privileges in your text editor:

```
sudo nano /etc/systemd/system/gunicorn.service
```

14) Enter the following lines of code in the file:

```
[Unit]

Description=gunicorn daemon

Requires=gunicorn.socket

After=network.target



[Service]

User=ubuntu

Group=www-data
```

```
WorkingDirectory=/home/ubuntu/daycare-mgmt/daycare-mgmt

ExecStart=/home/ubuntu/daycare-mgmt/daycare-mgmt/venv/bin/gunicorn \

          --access-logfile - \

          --workers 3 \

          --bind unix:/run/gunicorn.sock \

          core.wsgi:application



[Install]

WantedBy=multi-user.target
```

15) Edit the /etc/nginx/nginx.conf file to increase the limit of the files to upload
    a) Add `client_max_body_size 50M;` to the http {...}
16) You can now start and enable the Gunicorn socket. This will create the socket file at `/run/gunicorn.sock` now and at boot. When a connection is made to that socket, systemd will automatically start the `gunicorn.service` to handle it:

```
sudo systemctl start gunicorn.socket
sudo systemctl enable gunicorn.socket
```

17) Reload the daemon and restart gunicorn, using:

```
sudo systemctl daemon-reload
sudo systemctl restart gunicorn
```

Configure Nginx to Proxy Pass to Gunicorn

18) Now that Gunicorn is set up, you need to configure Nginx to pass traffic to the process. Start by creating and opening a new server block in Nginx's `sites-available` directory:

```
sudo nano /etc/nginx/sites-available/daycare-mgmt
```

19) Add the following lines of code to the file:

```
server {
    listen [::80];
    server_name [your_ip_address];


    location = /favicon.ico { access_log off; log_not_found off; }
    location /static/ {
        root /home/ubuntu/daycare-mgmt/daycare-mgmt;
    }


    location / {
        include proxy_params;
        proxy_pass http://unix:/run/gunicorn.sock;
    }
}
```

20) Now, you can enable the file by linking it to the sites-enabled directory:

```
sudo ln -s /etc/nginx/sites-available/daycare-mgmt /etc/nginx/sites-enabled
```

21) Test your Nginx configuration for syntax errors by typing:

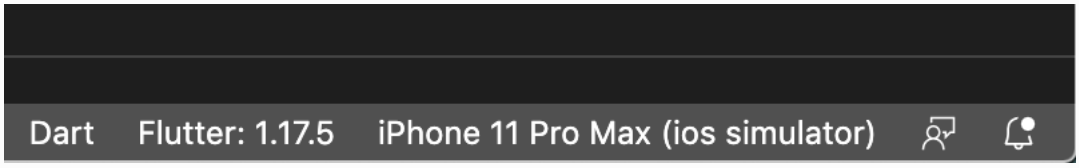
```
sudo nginx -t
```

22) If there are no errors, go ahead and restart Nginx by typing:

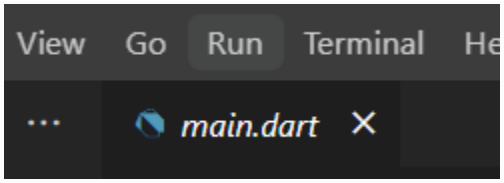
```
sudo systemctl restart nginx
```

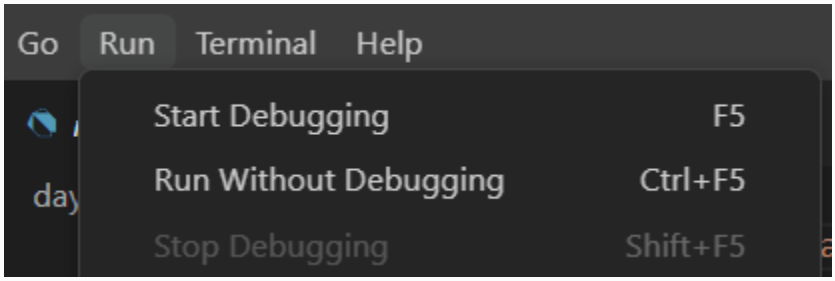
Running the Mobile App in Debug Settings Through Flutter and VS Code

1) Install Flutter here https://docs.flutter.dev/get-started/install
2) Set up VS code following the instructions here https://docs.flutter.dev/get-started/editor?tab=vscode
3) Set up a mobile emulator, or plug in your mobile device to run the app on.
4) Choose your device or emulator from the device selector, which you can open by clicking on the “iPhone 11 Pro Max” area (which may have a different device name to begin). This is at the bottom right of VS code.

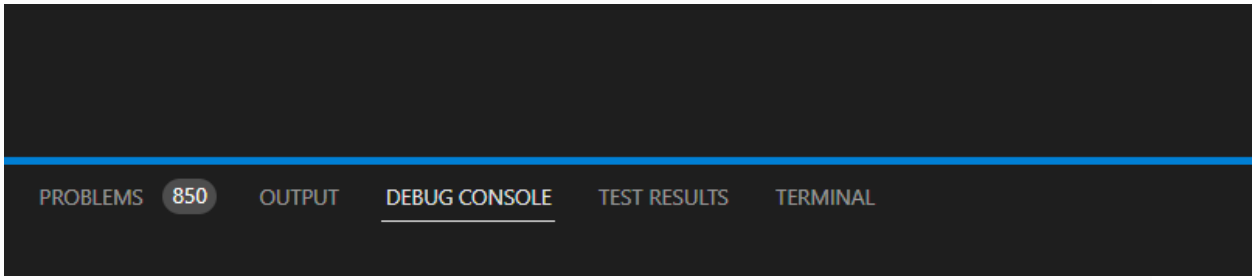


5) Ensure you are in a .dart file, the project files can be \daycare-mgmt\daycare_mgmt_frontend\lib
6) Click on the Run tab in VS code, then click start debugging.



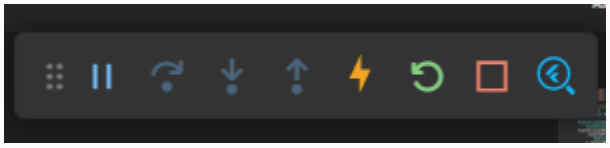


7) Debug your app.
   The app will take a few minutes to build, and you can open the “Debug Console” by opening a new terminal window, or pulling it up from the bottom.



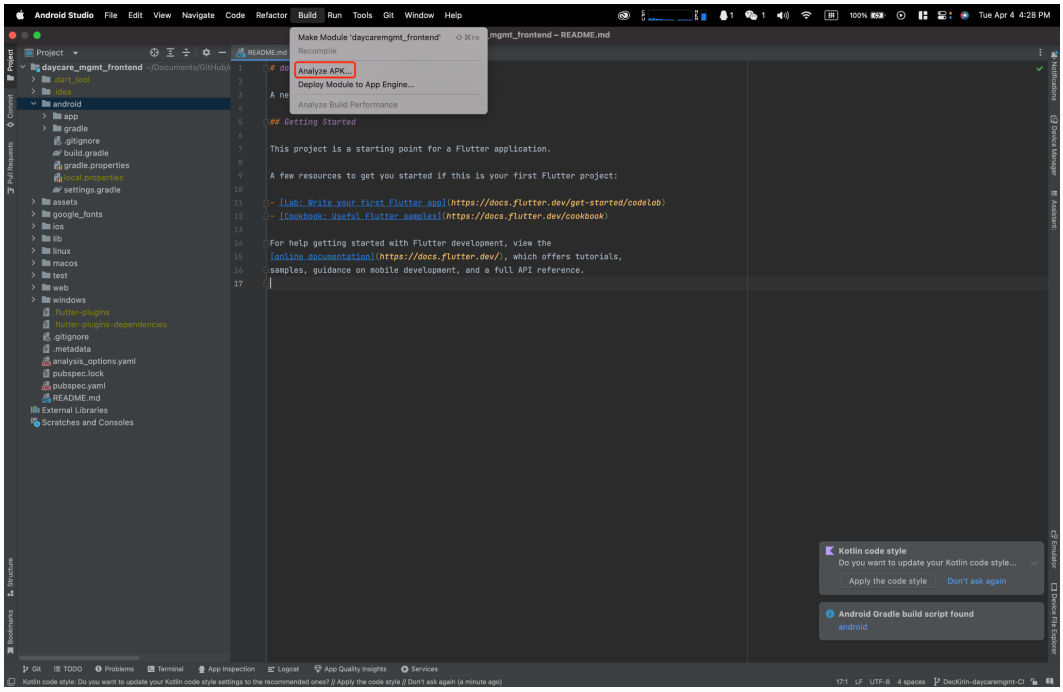
Any output or errors from the app will appear in the terminal while you are running the app and debugging.

If you come across a bug that breaks the app, or want to restart it after making changes you can rerun the app by clicking the green refresh arrow on the debugging plane that appears in the top right of VS code. You can also stop it at any time by clicking the red square.

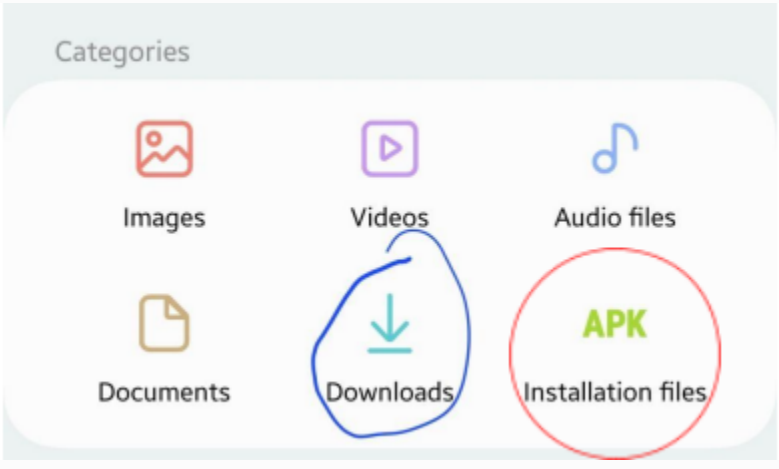


Building the APK With Android Studio and Installing it on a Mobile Device

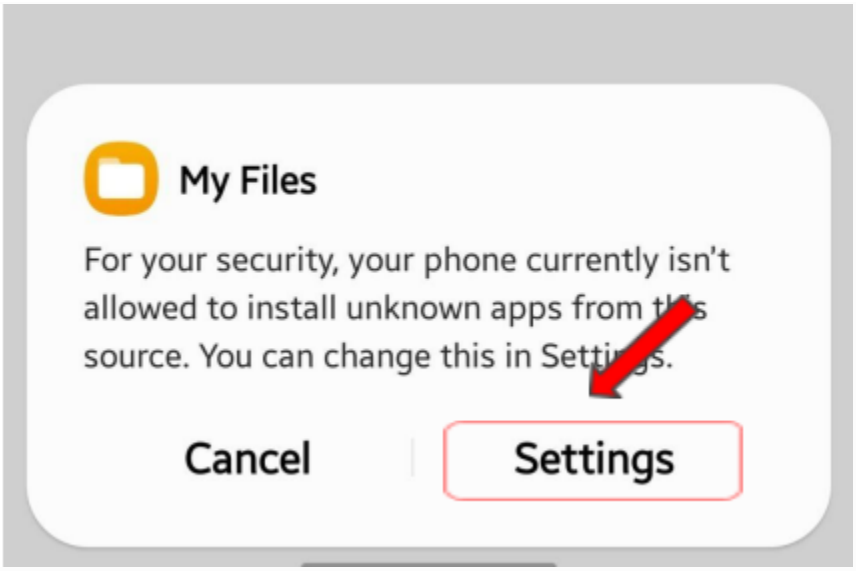
1) Open the Project daycare-mgmt/daycare_mgmt_frontend in Android Studio
2) Find Build Tab, choose build analyze APK

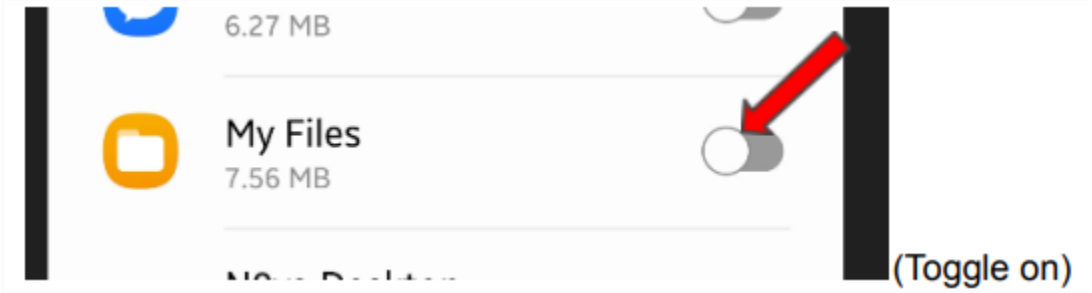


3) Download the APK file on your mobile device and open the file app on your phone.
4) Go to Download/Installation file



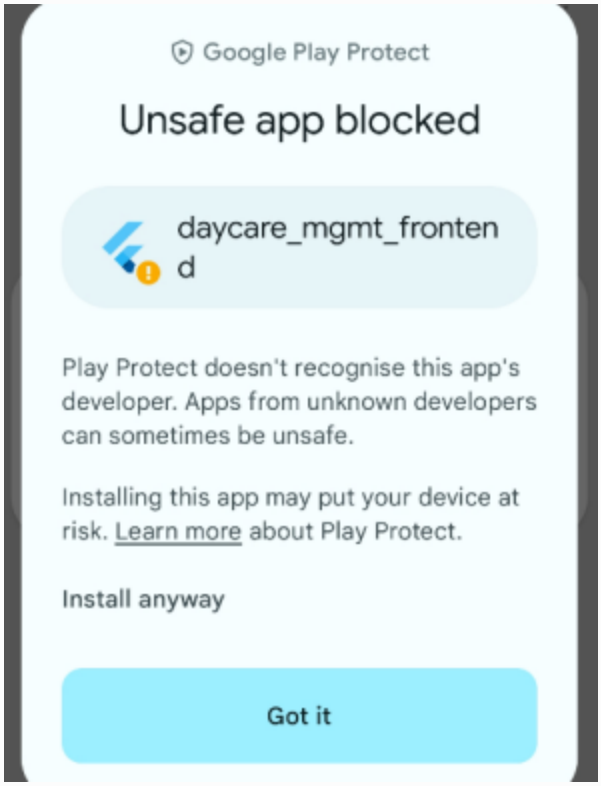
5) Tap the file to install the APK
6) Allow File app to install the APK by entering your settings and toggling the “My Files” to on.

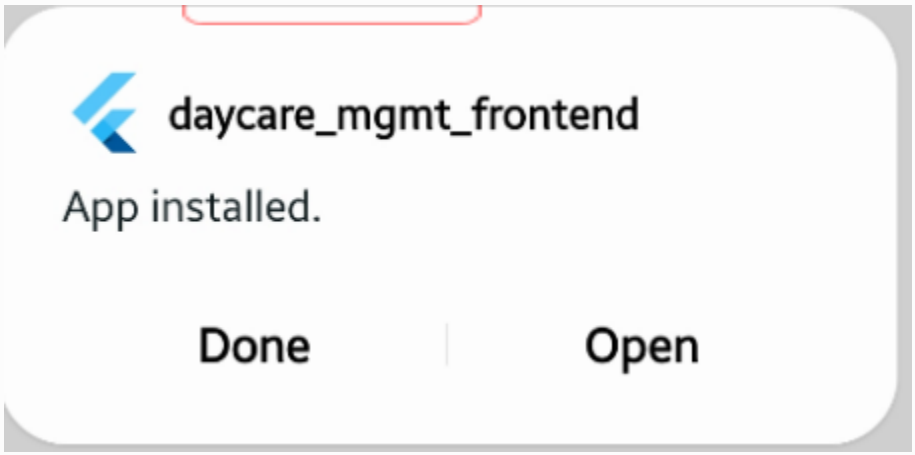




(Toggle on)

7) There might be a pop-up google protection window, because this is a debug build not on the play store yet, so that google will warn the user. But for our testing purpose it's ok to install it. Tap “Got it” and you shall continue





8) After you click Open, google might ask you to send this app for evaluation, please tap “Don’t send”.

PS: You might need to check if you have IPV6 internet access to connect to the server, use this link: https://test-ipv6.com/ if you see 10/10 ✔ then you are good to go.

Building a CI/CD Pipeline (Self-hosted Runner)

1) Setup a self-host runner following the github instructions
    a) Download packages: https://docs.github.com/en/actions/hosting-your-own-runners/adding-self-hosted-runners
    b) Set self-host runner runs in the background https://docs.github.com/en/actions/hosting-your-own-runners/configuring-the-self-hosted-runner-application-as-a-service
2) Setup Github Action for CI:
    a) Goto github -> action -> New Workflow
        i) Choose Django template (Under Suggested for this repository)
            (1) Alternatively, you can search for Django)

            (2) Press configure
        ii) Modify the yaml template

```yaml
name: Django CI


on:
 push:
   branches: [ "main" ]
 pull_request:
   branches: [ "main" ]


jobs:
 build:


   runs-on: self-hosted
```

```
    strategy:
      max-parallel: 4
      matrix:
        python-version: [3.8]

    steps:
    - uses: actions/checkout@v3
    - name: Set up Python ${{ matrix.python-version }}
      uses: actions/setup-python@v3
      with:
        python-version: ${{ matrix.python-version }}
    - name: Install Dependencies
      run: |
        python -m pip install --upgrade pip
        pip install -r daycare-mgmt/requirements.txt
    - name: Run Tests
      run: |
        python daycare-mgmt/manage.py test
```

iii) Start commit

3) Setup Github Action for CD:
   a) Go to Settings -> Action Secrets and Variable
   b) Add SEVER_HOST (Your own host name),SSH_PRIVATE_KEY(Your server key)
   c) Create YAML from following template

```
name: Django CD Pipeline

on:
 push:
   branches: [ "main" ]
```

```
jobs:
 deploy:
   runs-on: self-hosted
   steps:
   - name: Checkout code
     uses: actions/checkout@v2
   - name: Location
     run: |
       echo "$PATH"
       ls
   - name: Set Private Key
     env:
       SSH_PRIVATE_KEY: ${{ secrets.SSH_PRIVATE_KEY }}

     run: |
         mkdir -p ~/.ssh
         echo "$SSH_PRIVATE_KEY " > ~/.ssh/id_rsa
         chmod 600 ~/.ssh/id_rsa
         ssh-keyscan <your host ip> >> ~/.ssh/known_hosts


   - name: deploy
     run:  |
       ssh -i ~/.ssh/id_rsa <your username>@<your host ip> 'cd daycare-mgmt &&
git pull && source daycare-mgmt/venv/bin/activate && pip3 install -r
daycare-mgmt/requirements.txt && python3 daycare-mgmt/manage.py makemigrations
&& python3 daycare-mgmt/manage.py migrate && sudo systemctl daemon-reload &&
sudo systemctl restart gunicorn'
```

d) Start Commit

Set Up a firebase cloud messaging

1. Login to google cloud platform.
2. Search for Firebase Cloud Messaging API.
3. Set up Apps for Web and flutter.
4. Follow the instructions on the firebase page.