

---

# **Nadine Documentation**

*Release 1.8.1*

**Jacob Sayles**

**May 17, 2017**



---

## Getting Started

---

<b>1 License &amp; Copyright</b>	<b>3</b>
<b>2 Indices and tables</b>	<b>5</b>





Nadine is a Django web project which runs behind the scenes of coworking spaces.

Nadine has four applications: Members, Staff, Admin, and Tablet. These applications help your management team facilitate your coworking community.



# CHAPTER 1

---

## License & Copyright

---

Copyright 2016 Office Nomads LLC. Licensed under the Apache License, Version 2.0 (the “License”); you may not use this file except in compliance with the License.

You may obtain a copy of the License at <http://www.apache.org/licenses/LICENSE-2.0>

Unless required by applicable law or agreed to in writing, software distributed under the License is distributed on an “AS IS” BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.

See the License for the specific language governing permissions and limitations under the License.





- genindex
- modindex
- search

## 2.1 Requirements

- Python 2.7
- Virtualenv (Virtual environment)
- Homebrew if you are on Mac OS X (<http://brew.sh>)
- Postgresql

---

**Important:** Do not use SQLite.

---

### 2.1.1 Base System Installation

On Mac OS X

```
$ git # If you have not installed it, this will prompt you to download it.
$ brew update
$ brew install postgres python
$ pip install virtualenv
```

On Ubuntu/Debian

```
$ sudo apt-get update
$ sudo apt-get install postgresql postgresql-server-dev-all python-pip python-dev
↳ virtualenv libffi-dev git
```

---

Once that is ready, you can start the *quickstart*

## 2.2 Quickstart

Install the *required systems*

Setup the database

```
$ sudo su postgres -c "createuser -s $(whoami) "  
$ createdb nadinedb
```

Create a virtual environment for the python project

```
$ virtualenv nadine  
$ cd nadine  
$ source bin/activate
```

Download the nadine source code from Github

```
$ git clone https://github.com/nadineproject.nadine.github  
$ cd nadine
```

Install all the requirements

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

Run these scripts to setup nadine, install the database, and create your admin user

```
$ ./manage.py setup  
$ ./manage.py migrate  
$ ./manage.py createsuperuser
```

At this point you can run the server

```
$ ./manage.py runserver
```

Visit your installation of Nadine at <http://127.0.0.1:8000/>

## 2.3 Changing Application Settings

More information coming soon. If you need this information immediately, please file a [GitHub issue](#).

## 2.4 Nadine Structure

### 2.4.1 Overview

Nadine is comprised of four applications: Members, Staff, Admin, and Tablet.

**Members** is the member facing application. We consider members to be users with active resource allocations. **Staff** is the application used by staff members to help with the overall management of the community. **Admin** is the application in which only application administrators can access as it has absolute access to data. **Tablet** is the application used

for user sign-ins and for the interaction expected at the entrance of a space and greeting a new or returning user of the space.

## 2.4.2 Members Application

This application is what members will use to connect with each other, the coworking space staff, and greater community.

---

**Note:** Much of the information given by members and organizations is designated as either public (viewable by current members) or private (only viewable by staff and that specific member). This is clearly indicated to members and they opt in to sharing whatever of that information with which they are comfortable.

---

In the Members Application there are profiles for both members and organizations/companies. Each have a photo or logo, URLs they would like others to see, a short bio, and tags. While the listed items are public, privately the members and organizations can see emergency contacts, billing history, signed documents (such as a membership agreement), and their user activity.

A member and an organization can edit their profiles as often as they would like. One setting, though, that is set by application administrator is whether or not members are allowed to upload their own user photos. That setting is entitled ALLOW\_PHOTO\_UPLOAD. More information in *Changing Application Settings*.

Other features of the Members Application are the calendar of events, ability to subscribe and unsubscribe from mailing lists, Slack invitation (if allowed in Settings), and the general ability to see who the other members and organizations are and to make a request to connect.

One feature, tags, are interests they can share which are then sortable and searchable by other members and organizations. This allows a space to see how organizations self identify their industries and also what their members are doing.

The Members Application is laid similarly in the views and the templates.

- connect
- core
- events
- organization
- profile
- tags

## 2.4.3 Staff Application

The Staff Application allows the staff of a space to best manage memberships and the tasks required to run the space. It is the application designed for staff to easily track space usage, review billing and deposits, and edit member information if/when needed.

The navigation of the Staff Application includes:

- Tasks/ToDo
- Member List
- Activity
- Billing

- Stats
- Logs
- Lists
- Settings

### Tasks/ToDo

This section of the Staff Application deals with tasks that staff must complete. Tasks can be assigned to specific staff members or left available for anyone to complete and mark as such. This is the default home page for a staff member.

### Member List

Member List shows all members in a sortable manner and then allows a staff member to edit any person's information as needs.

### Activity

Activity is for recording any members particular use of the space and to generate reports on past usage and membership levels.

### Billing

It is important to know that Nadine does not store any person's credit card information. Nadine in its current iteration uses a USAEpay integration for billing. In this section of the application, staff can track payments, generate reports, and run the daily billing.

### Logs

Logs show device and user logins to the local internet. To know the user, the system remembers devices after a member has logged into Nadine from that computer. This allows us to better track use of space and to make sure that a member is using the space per their membership.

### Lists

These pages are for the management of whatever mailing lists a space might have and the Slack channel, if a space has it.

## 2.4.4 Admin

Like most admin applications, this has absolute access to user and space data. As stated before, though, this does not include any credit card information. Only application administrators have access to this part of Nadine.

## 2.4.5 Tablet Application

The Tablet is designed for use on an iPad at the entry of a space as a sort of portal. The user has access to sign in and see who else is in the space from this. Additionally, a user can sign documents such as a membership agreement.

## 2.5 Themes

Nadine's Member application is designed to be changed to better serve each space. Want to include your logo, company color scheme, etc? We tried to make that easy for you.

Currently the Member application is styled to be rather generic and includes the Nadine Logo throughout. What logo is that? That's the cow on the index page of this documentation. You are welcome to stay with this design but we also welcome you to get in the sandbox and make the Member application unique to your coworking space.

### 2.5.1 Creating a Theme

It is easy to create your own theme and implement it with the Member application.

Create a new project with the following doc tree:

```
THEME_NAME/  
- theme_settings.py  
- static/  
  | - css/  
  |   | -members.css  
  | - img/  
  | - js/  
  | - fonts/  
- templates/  
  | - members/  
- .gitignore
```

#### Static Folder

The `static/` folder will contain all of your new styling(css), any particular javascript files you might need, new font files, and images. Here you can include the style sheets for any new CSS framework you might use and/or your own stylesheet.

The **members.css** file will be the most important for your new styling. This is where you can override the stylings from the default theme.

To completely override the layout of a page, you will need to write that page with DTL and HTML and include that in the `templates/members` folder.

#### Logo

In in the `img/` folder, you can include your logos which you will use. If you do not intend to change the HTML then you will need to include two versions of your logo and save them as **logo.png** and **logo-line.png**. The first one to be used on the homepage jumbotron and the other to be part of the top navigation throughout the app.

#### Theme Settings

In `theme_settings/` you can set the local settings for the application. The settings available include things such as the social media URLs, permissions for registration and photo uploads, and others.

For example, a `theme_settings` file for Office Nomads might look like:

```
ALLOW_ONLINE_REGISTRATION = False
ALLOW_PHOTO_UPLOAD = False

FACEBOOK_URL = "https://www.facebook.com/OfficeNomads"
TWITTER_URL = 'https://twitter.com/OfficeNomads'
YELP_URL = 'https://www.yelp.com/biz/office-nomads-seattle-2'
INSTAGRAM_URL = 'https://www.instagram.com/officenomads/'
```

### Implementing the Theme

First, copy your new theme folder into the themes/ folder. Then, in the terminal:

```
$ cd themes
$ ln -s THEME_NAME active
```

This command tells Nadine to prioritize your theme over the members.css that came with it. Reload the Member App and see how it all looks!

## 2.6 Django Templating

Nadine uses Django as our backend framework. With Django comes the awesome power of the Django templates. The templates are HTML with the Django template language (DTL) and any necessary JavaScript.

For more information on DTL and Django templates, check out [the documentation](#).

### 2.6.1 Nadine's Usage of Django Templates

To make the DRY-est html, we have used the templates to manage repeated code. You will notice a base.html in all of the applications. This is the file which will layout the head, navigation, and footer. From there, each of the pages will include a version of:

```
{% extends app_name/base.html %}
```

In the Staff App, the templates are divided up even more and each of those folders include their own base.html which extends the staff/base.html and then sets some basic navigation styling and brings in a stylesheet for that section.

Below the 'extends' code, there might also be such code as:

```
{% load static %}
```

This loading of static or settings or whatever is called is bringing in a variable from the backend to be used.

To maintain our DRY code, there are points in the HTML in which the template 'includes' another HTML page. An example of this would be the date\_range\_form.html which is repeatedly used in the Staff App with:

```
{% include "staff/date_range_form.html" %}
```

Again, for more info on Django templating, please see [the documentation](#).

## 2.7 Static Files

Per Django documentation, Nadine serves static files by setting a STATIC\_URL in the settings file and then loading the static shortcut then including it in a path like this:

```
{% load static %}

```

The static folder includes all images Nadine will use, stylesheets, JavaScript files, and fonts. Each application has its own static folder in which you must include any necessary item to be used in that application.

## 2.8 Testing

Nadine is written with front-end and back-end tests. You are welcome to run the tests locally. If you do run into any issues, please enter it as an issue in [Github](#).

### 2.8.1 Front-End Testing

For Front-En testing, Nadine uses CasperJS which is a ‘navigation scripting & testing utility for the PhantomJS (WebKit) and SlimerJS (Gecko) headless browsers, written in Javascript.’

To run these tests, you first must have CasperJS installed and make sure that PhantomJS is installed.

```
$ brew install casperjs
$ phantomjs --version
```

If you do not have phantomjs, then use brew to install it.

To run all tests:

```
$ ./manage.py runserver #make sure you have the server or running it will error out
$ casperjs test frontend-testing/tests --username='YOUR_USERNAME' --password='YOUR_
→PASSWORD' --path='/PAGE_TO_TEST/'
```

To run a singular test, include the filename after tests/ in the path. In particular, to run tests to verify all links return a status code of 200, we have a test for that. Include a new variable called ‘path’ and either assign it ‘/member/’ or ‘/staff/’.

```
$ ./manage.py runserver
$ casperjs test frontend-testing/tests/linktesting.js --username=YOUR_USERNAME --
→password=PASSWORD --path='/PAGE_TO_TEST/'
```

Suggested paths:

- ‘/member/’
- ‘/member/view/’
- ‘/staff/’
- ‘/staff/user/members/’
- ‘/staff/user/detail/USERNAME’

Some of the tests are checking for a lot of information so it might take a minute or so to run.

### 2.8.2 Back-End Testing

Django is wonderful and includes its own ability to run unit tests. According to the docs, ‘Django’s unit tests use a Python standard library module: unittest. This module defines tests using a class-based approach.’ For more detailed information on Django testing then please see [the documentation](#)

To run backend tests, you can be specific or more broad. To run all tests:

```
$ ./manage.py test nadine.tests
```

To run one specific test, like from the room booking test suite:

```
$ ./manage.py test nadine.tests.test_room.RoomTestCase.test_available_straddling
```

## 2.9 Mailing Lists

In the interest of shipping more quickly, we have made certain assumptions about the interlink mailing lists which may or may not suit everyone's needs.

- the reply-to address for mail from a list is the original sender, not the entire list
- attachments are neither saved nor sent to the list, but a removal note is appended to the message
- incoming messages are parsed for a single text message and a single html message (not multiple MIME messages)
- loops and bounces are silently dropped
- any email sent to a list which is not in a subscriber's user or membership record is moderated
- the sender of a message receives a copy of the message like any other subscriber