

James Small

SDEV-435-81: Applied Software Practice I

APRIL 25, 2021

# Dusseldorf – The Developer Nexus Project Documentation



SOURCE PIXABAY, PUBLIC DOMAIN

## Executive Summary

Often beginning to intermediate developers are not sure where to find the right resources or even what they need. He may be just learning about the craft. She may be a student or just entering the field. He may be looking to progress from a junior to a senior developer. Isn't this already addressed? There are certainly plenty of good resources available. However, what's missing is a map. Experienced developers just "know" where to go. Those entering the field though or looking to progress often do not. Where are the best resources? What does it take to be good? Where do you practice? How do you learn what you don't know? While it is possible to slowly figure this out over time, this site aims to provide the big picture guide straight away. Those looking for direction will quickly see what they need and where to get it from.

This web application will help developers map out necessary skills and resources to acquire those skills. The site will grow its community by allowing user feedback and comments. The site will also allow blogging, writing articles and tutorials to help developers master their craft while also teaching and informing their peers.

To deliver this solution, the Django web framework will be used along with a PostgreSQL database. To design the site, I will start with a model of site resources. This will show what information is available for each resource type (database table) and how the resources are related. I will then build the views that allow users to interact with the site. Finally, I will create the actual templates that are used to render the site pages that the user sees and interacts with. Each view will have a test to make sure it's working. Test users will be enlisted to make sure the site works and to provide feedback on its utility and ease of use.

This site will allow user registration and require it for active participation. Users will have the opportunity to provide feedback and leave comments on site resources. User activity will be tracked with a leaderboard showing the most active users. Resources will be tagged allowing for easy location of related or specific resource types.

## Project Status and Details

Project Completion Status: 100%

Project Details:

- Deployed project (Heroku): <https://developernexus.herokuapp.com/>
  - Note: Because this is hosted on Heroku's free tier, it can take up to 30 seconds to spin up.
- GitHub repository home page: <https://github.com/sockduct/dusseldorf>
- GitHub page for this project: <https://github.com/sockduct/dusseldorf/blob/master/README.md>
- Project documentation file – this file, also on GitHub:  
<https://github.com/sockduct/dusseldorf/blob/master/docs/Project%20Documentation.pdf>
- Video walk-through of the system in-use: <https://share.getcloudapp.com/NQuwj6GE>
- Video walk-throughs of architectural requirements
  - Video walk-through of Data requirement: <https://share.getcloudapp.com/L1ud9qWK>
  - Video walk-through of Verification and Validation:  
<https://share.getcloudapp.com/qGuExkLG>
  - Video walk-through of Re-usable Code: <https://share.getcloudapp.com/eDuj7Nb6>

## Scenario | Browsing Site, Unauthenticated

Most of the site is viewable without authentication, starting from the home page:



### The Developer Nexus - Posts

[Posts](#) [Paths](#) [Leaderboard](#) [About](#) [Contact Us](#)

[Sign Up](#) [Login](#)

[+ Add Post](#)

**Welcome to my Champlain Capstone Project** · Blog by sockduct · 1 month, 1 week ago

Hello and thank you for stopping by. This is my senior project for my Champlain College Software Development degree. Content is still a little light, but expect me to keep adding more. Please sign up for an account and feel free to share some content yourself. You can create an account using one of your e-mail addresses or with a GitHub account. The idea is to create a community for beginning to intermediate developers. We can share ideas and I am working on building out a map of interesting resources. This will chart the way toward a senior/architect level. I am leveraging resources from the ACM, IEEE, and various highly regarded sites.


vision · no comments

**Test post** · Blog by test · 1 month, 1 week ago

## Markdown? --- Does django sanitize the things? `<script>alert("test");</script>` Looks good.

example · 1 comment

Individual posts on the home page can be viewed revealing any comments:



## The Developer Nexus - Post Detail

---

[Posts](#) [Paths](#) [Leaderboard](#) [About](#) [Contact Us](#) [Sign Up](#) [Login](#)

---

[+ Add Post](#)

---

**Test post** · Blog by test · 1 month, 1 week ago

## Markdown? --- Does django sanitize the things? `<script>alert("test");</script>` Looks good.

example

---

Comments

sockduct · 1 month ago

Adding markdown is on my list...

[+ Add Comment](#)

Note – creating a new post or commenting require an account. Clicking on Add Post or Add Comment will prompt the user to login or create an account (see the following scenarios).

The Paths area of the site may also be viewed unauthenticated:



## The Developer Nexus - Paths

---

[Posts](#) [Paths](#) [Leaderboard](#) [About](#) [Contact Us](#) [Sign Up](#) [Login](#)

---

[Areas](#) [Subjects](#) [Resources](#)

---

### Algorithms and Complexity

ACM CS2013 BoK/KA - AL: The study, design, and selection of algorithms and data structures as well as computational complexity

2 supporting subjects

### Architecture and Organization

ACM CS2013 BoK/KA - AR

No supporting subjects

Each Path may be drilled into by clicking on its title. This reveals related subjects. All Subjects and Resources may also be viewed by clicking on their respective links under the title. As with Paths, each Subject and Resource may be drilled into revealing any supporting/related items.

The Leaderboard (covered in Reports), About, and Contact Us pages may also be viewed by clicking on the appropriate links. However, for the Contact Us page, unauthenticated users will be prompted to login or sign up to submit feedback.

## Scenario | User Sign Up

Users wishing to submit content/feedback or comment on posts require an account. Accounts can be created with an email address and password or by leveraging an existing GitHub account:



### The Developer Nexus - Sign Up

[Posts](#) [Paths](#) [Leaderboard](#) [About](#) [Contact Us](#)[Sign Up](#) [Login](#)

E-mail\*

Password\*

[Sign Up »](#)[Sign Up with a GitHub Account](#)

I already have an account:

[Login with Existing Account](#)

Once a user creates an account, he will receive an email address validation email with a link. Clicking on the link validates the account. The user will also now be authenticated:



## The Developer Nexus - Posts

[Posts](#) [Paths](#) [Leaderboard](#) [About](#) [Contact Us](#)

testuser ▾

[+ Add Post](#)

Confirmation e-mail sent to testuser@example.com. Successfully signed in as testuser.

✕

[Welcome to my Champlain Capstone Project](#) · Blog by sockduct · 1 month, 1 week ago

Hello and thank you for stopping by. This is my senior project for my Champlain College Software Development degree. Content is still a little light, but expect me to keep adding more. Please sign up for an account and feel free to share some content yourself. You can create an account using one of your e-mail addresses or with a GitHub account. The idea is to create a community for beginning to intermediate developers. We can share ideas and I am working on building out a map of interesting resources. This will chart the way toward a senior/architect level. I am leveraging resources from the ACM, IEEE, and various highly regarded sites.

vision · no comments

[Test post](#) · Blog by test · 1 month, 1 week ago

## Markdown? --- Does django sanitize the things? `<script>alert("test");</script>` Looks good.

example · 1 comment

Example validation email:

Subject: [The Developer Nexus] Please Confirm Your E-mail Address

From: admin@localhost

To: testuser@example.com

Date: Mon, 26 Apr 2021 15:09:46 -0000

Message-ID:

<161944978603.23372.13889555456954166169@MICDTL02JS646Y.ITServices.sbc.com>

Hi from The Developer Nexus!



You're receiving this e-mail because the user testuser has given this e-mail address for the account.

To confirm this is correct, please go to:

[http://localhost:8000/accounts/confirm-email/MTg:1lb2rq:3CAZGhvJFWr5Ak0O2p\\_JPv5wMPoKII0R6y6\\_\\_Kp1TNM/](http://localhost:8000/accounts/confirm-email/MTg:1lb2rq:3CAZGhvJFWr5Ak0O2p_JPv5wMPoKII0R6y6__Kp1TNM/)

If you've received this in error, no further action is necessary. Just delete this message.

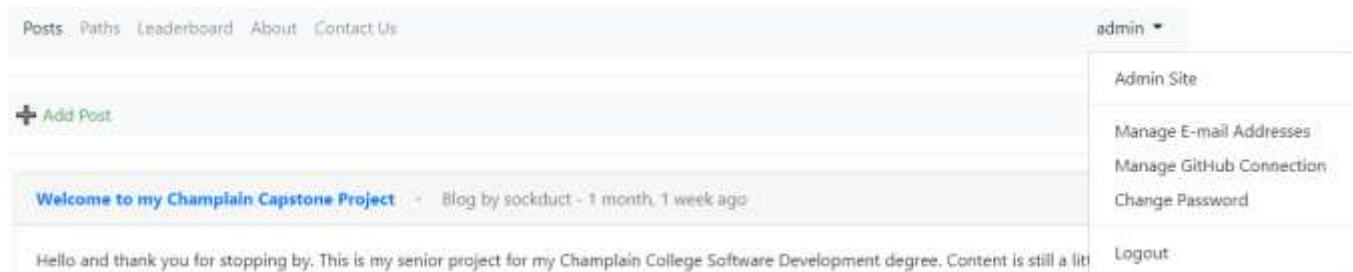
Thank you from The Developer Nexus!

[developer-nexus.herokuapp.com]

Once a user is authenticated, he will also have a menu of options to manage his email address and/or GitHub connection as well as change his password. Finally, the menu also allows the user to logout:



Note: Administrative users will have an additional option which allows access to the Django Admin site. This site allows managing database content:



Admin Site:

Django administration

WELCOME, **ADMIN**. [VIEW SITE](#) / [CHANGE PASSWORD](#) / [LOG OUT](#)

Site administration

ACCOUNTS

Email addresses

+ Add

Change

ACCOUNTS

Custom users

+ Add

Change

User types

+ Add

Change

AUTHENTICATION AND AUTHORIZATION

Groups

+ Add

Change

PATHS

Areas

+ Add

Change

Resource types

+ Add

Change

Resources

+ Add

Change

Subjects

+ Add

Change

POSTS

Comments

+ Add

Change

Post types

+ Add

Change

Posts

+ Add

Change

Tags

+ Add

Change


Recent actions

My actions

None available

## Scenario | User Login

Users returning to the site can login with an account email address or through their GitHub account:




# The Developer Nexus - Login

[Posts](#) [Paths](#) [Leaderboard](#) [About](#) [Contact Us](#) [Sign Up](#) [Login](#)

E-mail\*

Password\*

[Developer Nexus Login](#)

 [Login with GitHub](#)

[I forgot my password](#) [I need to sign up for an account](#)

In addition, a link to reset a user's password is also included.

## Reports

Any user can view the Leaderboard page. This shows all active users and includes a scoring system based on each user's contributions. Each post is worth ten points and each comment one:



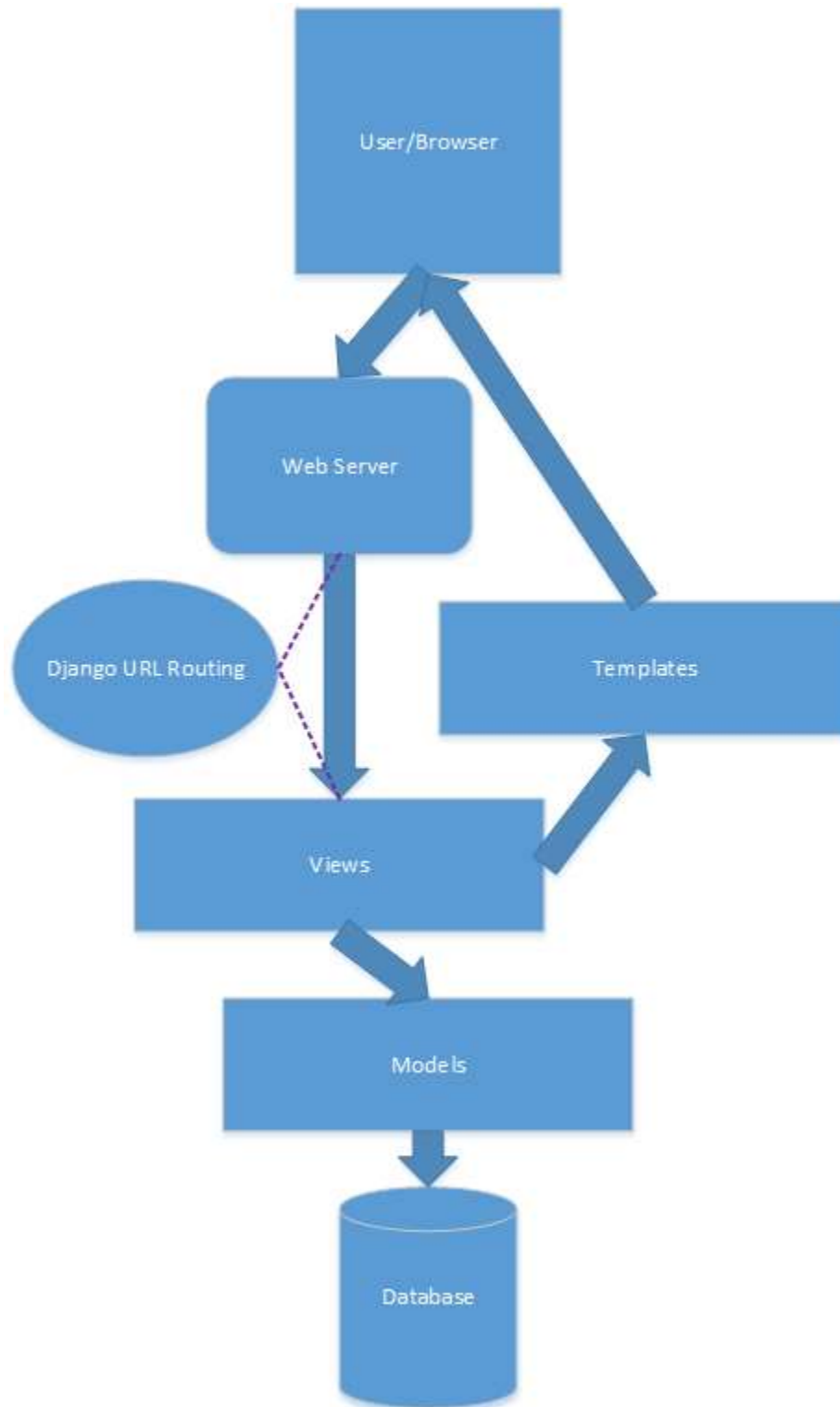
# The Developer Nexus - Leaderboard

[Posts](#) [Paths](#) [Leaderboard](#) [About](#) [Contact Us](#)

Username	Score	Member Since	Last On
admin	0	an hour ago	13 minutes ago
kendallhudson13	0	1 month, 1 week ago	1 month, 1 week ago
nathansenini	0	1 month, 1 week ago	1 month, 1 week ago
sockduct	11	1 month, 4 weeks ago	1 month ago
test	10	1 month, 1 week ago	1 month, 1 week ago
testuser	0	20 minutes ago	20 minutes ago

## System Architecture

The Developer Nexus is a Django project/web application. Django is built with Python as well as including a template language and an ORM which allows creating models of data and letting Django deal with all aspects of the backend database. SQLite was used as the development database and PostgreSQL is used in production. The Bootstrap Framework is used to augment the frontend.



## Source Code Structure

Source code structure introduction. The following is a summary of the source code directories and their contents:

Code Directory	
Directory	Usage
Accounts	Custom User model application, this user model is used for administrators and web site users.
Config	Houses overall Django project configuration.
Pages	Mostly static web pages – About, Contact Us – except Leaderboard which queries for user statistics. No models present in this application.
Paths	Dynamic web pages under Paths menu item. Paths have supporting Subjects which have supporting Resources. This application defines each of these, types, and how they're displayed/related. Only an administrator may change the content using the admin site. The admin site appears in the user menu of administrators only.
Posts	Dynamic web pages under the Posts (home page) menu item. Designed for user content submissions. Allows a variety of post types as well as tags.
Static	Static web page content including CSS, Fonts, Images, and JavaScript.
Staticfiles	Compiled static content for all static files used throughout the Django project. Used for production deployment.
Templates	HTML and email template files using Django's templating language. The templating language is rather simple, but still highlighting this cell as this is a type of code.

*Highlighted rows indicate directories containing source code.*

## Executables

The Django web framework is based on Python – an interpreted language. Python does compile its modules – these may be observed in the `__pycache__` directories. However, compiled byte code is not required to run a program. The entry points for a Django project are either `config/asgi.py` or `config/wsgi.py`. This application uses the later. The entry point for Heroku production deployment may be seen in `Procfile` in the root directory.

Note: Django project/application management is through `manage.py`:

- Run local test server: `python manage.py runserver`
- Run unit test suite: `python manage.py test`
- Enhanced shell access: `python manage.py shell_plus --ptpython`

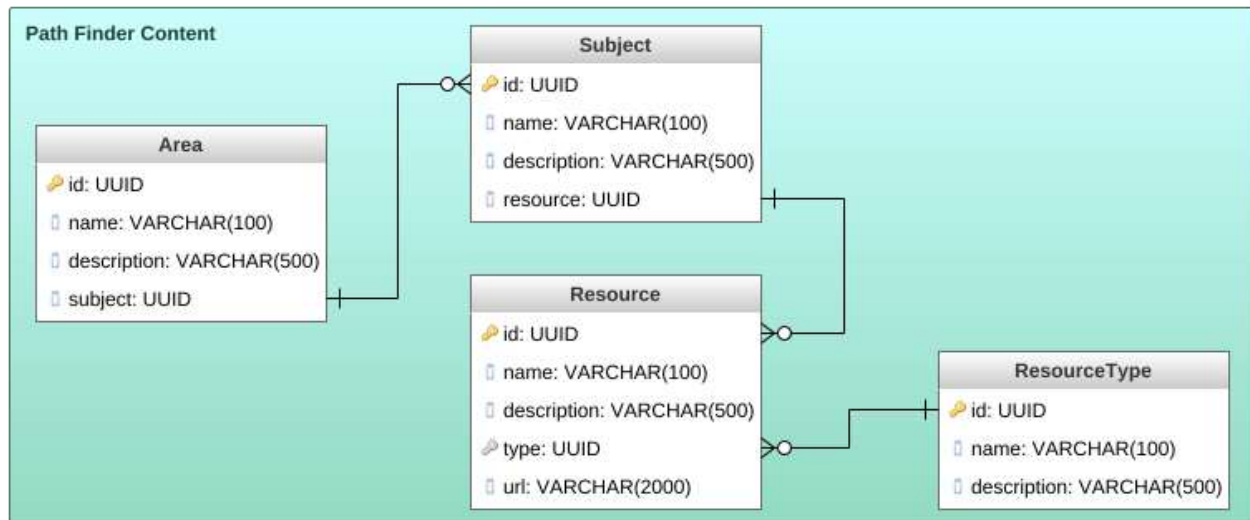
- Database access: `python manage.py dbshell`
- Other options: `python manage.py --help`

## Code Architecture

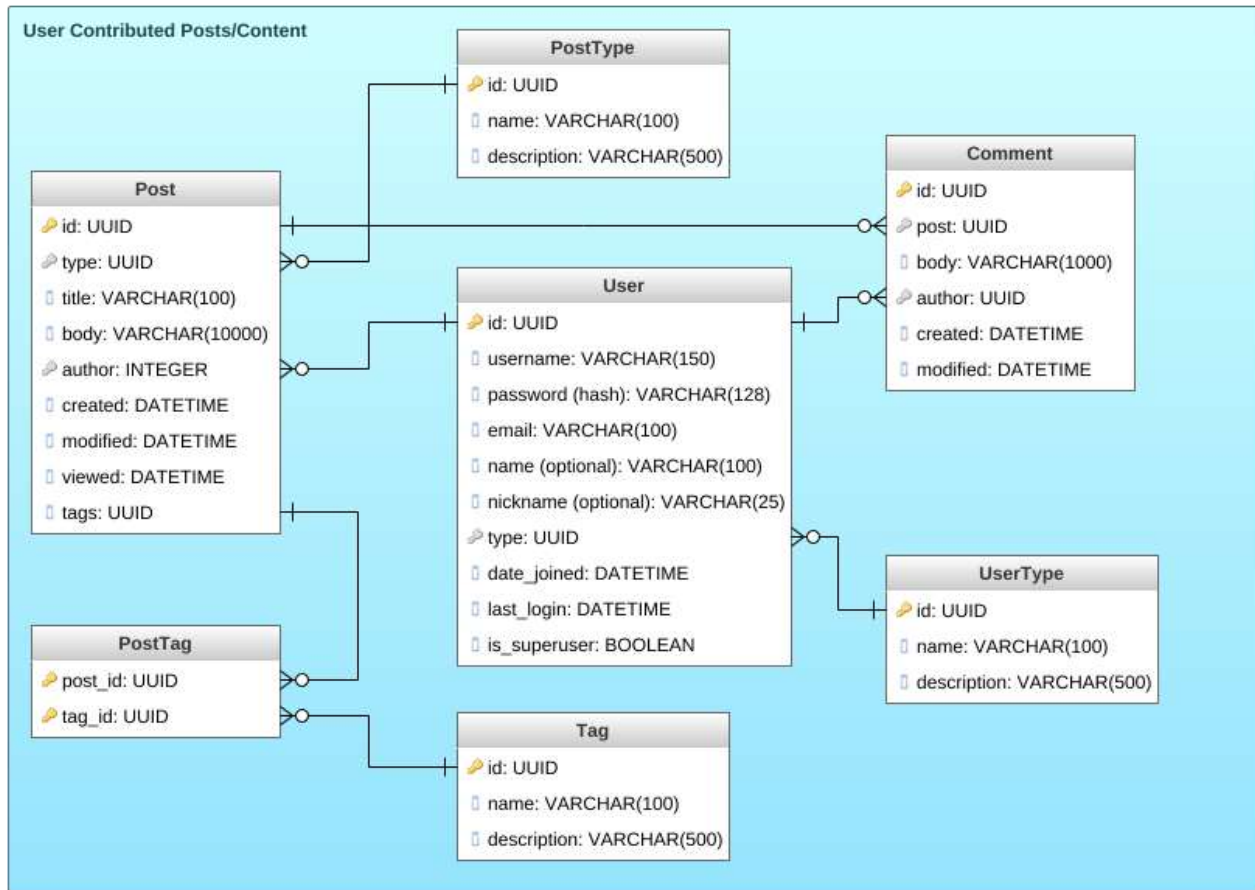
A Django project is created in a directory with a dedicated project configuration folder (config in this case). Applications are then created and registered within the configuration. Incoming user requests (from a client browser) are routed to the correct view through urls.py modules. The top-level urls.py lives in the configuration folder. It routes requests bound to specific applications to the urls.py in the application folder. In the target application, a specific view is chosen (a named class in the application views.py). This class then chooses the appropriate template to render (return) a response to the user. If a model is used, the class will pass the appropriate data from the model into the template in a context object. Templates are HTML pages with a templating language which can dynamically include/exclude page elements.

## Database or Data Store

As mentioned above, SQLite is used for local development with PostgreSQL used for production. The table layout is as follows:







Actual database table creation and maintenance is abstracted away by Django's ORM. Database management in Django:

- Models added/removed/changed: `python manage.py makemigrations <app>`
- Update database: `python manage.py migrate <app>`
  - Note: Optionally the specific application may be added. This is a best practice to limit changes to one application at a time.

## External Files & Data

Django projects/applications require some sensitive information such as API Keys, usernames/passwords, and a secret key. This information is stored in an environment file: `.env`

A sample file is provided in the project repo – `dotenv`. The repo README file includes directions for how to set this up correctly.

## Programming Language | Python

This project was created with the Django web framework using Python. Python v3.6 or later is required. The application was tested with Python v3.6 and v3.8. The repo README includes directions on obtaining Python and the libraries (dependencies) used for this application:

Top-Level Package	Version
cryptography	3.4.7
dj-database-url	0.5.0
dj-email-url	1.0.2
Django	3.1.8
django-allauth	0.44.0
django-cache-url	3.2.3
django-coverage-plugin	1.8.0
django-crispy-forms	1.11.2
django-debug-toolbar	3.2.1
django-extensions	3.1.3
environs	9.3.2
gunicorn	20.1.0
pipdeptree	2.0.0
psycpg2-binary	2.8.6
ptpython	3.0.17
werkzeug	1.0.1
wheel	0.36.2
whitenoise	5.2.0

In addition, Bootstrap v4.6.0 was used to augment the front end. For databases, v3.33.0 of SQLite was used and v13.2 of PostgreSQL is used in production on Heroku.

## Project Classes

Classes within the project are used to abstract re-usable pieces of code. Classes are also used to group related values, known as properties. The project utilizes these model and view classes:

### **Model Classes:**

UserType | accounts/models.py

Database representation of user types

CustomUser | accounts/models.py

Database representation of web site users and administrators

Area | paths/models.py

Database representation of web site path areas

Subject | paths/models.py

Database representation of web site path subjects

ResourceType | paths/models.py

Database representation of web site path resource types

Resource | paths/models.py

Database representation of web site path resources

Tag | posts/models.py

Database representation of web site post tags

PostType | posts/models.py

Database representation of web site post types

Post | posts/models.py

Database representation of web site user posts

Comment | posts/models.py

Database representation of web site user comments regarding posts

### **View Classes:**

AboutPageView | pages/views.py

Class responsible for creating response for user requests routed to it for “/about”

ContactPageView | pages/views.py

Class responsible for creating response for user requests routed to it for “/contact”

ErrorPageView | pages/views.py

Class responsible for creating response for user requests routed to it for “/error”

LeaderboardPageView | pages/views.py

Class responsible for creating response for user requests routed to it for “/success”

**SuccessPageView | pages/views.py**

Class responsible for creating response for user requests routed to it for “/leaderboard”

**AreaView | paths/views.py**

Class responsible for creating response for user requests routed to it for “/paths”

**AreaDetailView | paths/views.py**

Class responsible for creating response for user requests routed to it for “/paths/area/<UUID>”

**SubjectListView | paths/views.py**

Class responsible for creating response for user requests routed to it for “/paths/subjects”

**SubjectDetailView | paths/views.py**

Class responsible for creating response for user requests routed to it for “/paths/subject/<UUID>”

**ResourceListView | paths/views.py**

Class responsible for creating response for user requests routed to it for “/paths/resources”

**ResourceDetailView | paths/views.py**

Class responsible for creating response for user requests routed to it for “/paths/resource/<UUID>”

**CommentCreateView | posts/views.py**

Class responsible for creating response for user requests routed to it for “/post/comment/new”

**CommentDeleteView | posts/views.py**

Class responsible for creating response for user requests routed to it for “/post/comment/<UUID>/delete”

**CommentEditView | posts/views.py**

Class responsible for creating response for user requests routed to it for “/post/comment/<UUID>/edit”

**PostCreateView | posts/views.py**

Class responsible for creating response for user requests routed to it for “/post/new”

**PostDeleteView | posts/views.py**

Class responsible for creating response for user requests routed to it for “/post/<UUID>/delete”

**PostDetailView | posts/views.py**

Class responsible for creating response for user requests routed to it for “/post/<UUID>”

**PostEditView | posts/views.py**

Class responsible for creating response for user requests routed to it for “/post/<UUID>/edit”

**PostListView | posts/views.py**

Class responsible for creating response for user requests routed to it for “/”

## Project Modules

Modules are used for procedural based code that does not require state data like class modules do. Complete the introduction to modules.

migrations/000#\_generatedname.py | accounts, paths, posts folders  
database setup and configuration programs

admin.py | accounts, paths, posts folders  
register application components (classes) with Django administration site

apps.py | accounts, pages, paths, posts folders  
application registration and configuration – defaults used for this project

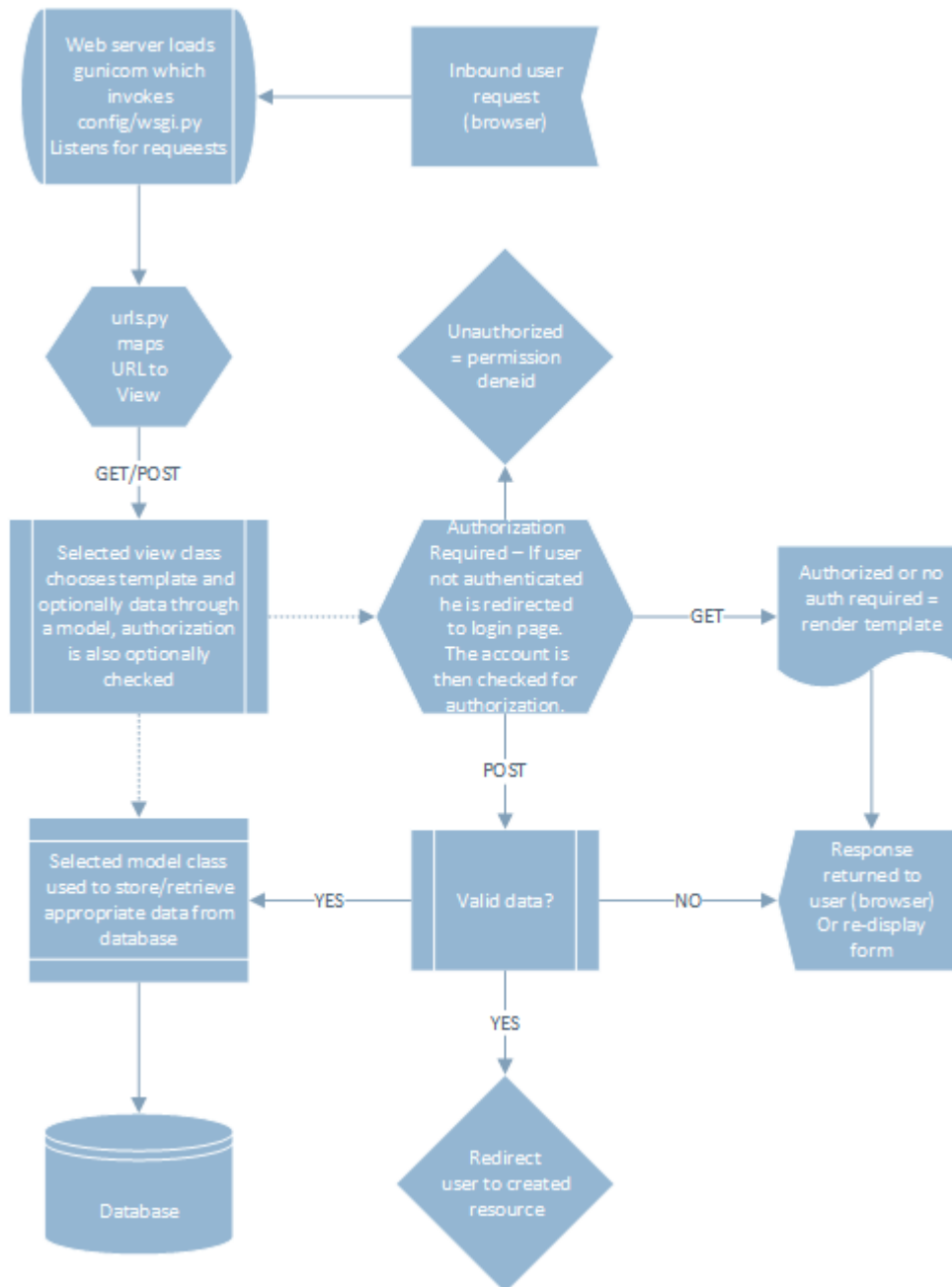
forms.py | accounts and pages folders  
custom input forms used by application

tests.py | accounts, pages, paths, posts folders  
unit testing for application

urls.py | pages, paths, posts folders  
used in tandem with config/urls.py to delegate routing to the application

## Program Start and End Flow

This application is hosted by a web server. The web server in turn runs an application server – gunicorn. This application server hosts the Django project entry point – in this case config/wsgi.py. Inbound user requests are then routed (urls.py) to the correct handler (view class selected by urls.py). The class then selects a template (from templates) and data if a model is included. This is then rendered and returned as a response to the user.



## Summary

The Developer Nexus is a web application for beginner to intermediate developers. This allows developers to contribute via blogs, articles, tutorials, and reviews. In addition, it provides roadmap resources to help developers plan their education. Using the popular Django web framework in tandem with Bootstrap allowed the creation of a polish site. Django's ORM allows using many popular databases for backend storage including SQLite and PostgreSQL. The application supports users, checks for authorization in restricted portions, and allows both convention and OAUTH sign up leveraging an existing GitHub account. The use of popular components allows this web application to be hosted almost anywhere with Heroku used for this project.



## APPENDIX A (TEST PLAN)

Django includes a unit testing framework. Each application created under Django leverages this with extensive testing performed. Testing is used to check routing, views, models, forms, and templates. Testing may be performed once the project is correct installed as follows:

```
python manage.py test
Creating test database for alias 'default'...
System check identified no issues (0 silenced).
.....
-----
Ran 125 tests in 47.116s

OK
Destroying test database for alias 'default'...
```

Manual testing was also performed:

- visit each page
- login/logout
- sign up – standard and GitHub-based
- Content creation
- Database inspection

Two friends were also enlisted to test the site and report any issues.

## APPENDIX B (BUILD AND RELEASE PROCESS)

### Local Development

Note: Assuming a [git client](#) is already installed

- Clone [the repo](#)
  - `git clone https://github.com/sockduct/dusseldorf`
  - `cd dusseldorf`
- Install [Python](#)
  - Note: Python v3.6 or later required
- Create a Python virtual environment and activate it
  - `python -m venv .venv`
  - Windows: `.venv\scripts\activate`
  - Linux: `source .venv/bin/activate`
- Install repo packages:
  - `pip install -r requirements.txt`
- Customize the environment file:
  - `cp dotenv .env`
  - Edit `.env` and tailor - must create at least a `DJANGO_SECRET_KEY` and a `DJANGO_ADMIN_URL_BASE`
  - For production deployments, `DJANGO_DEBUG` should be set to `False`, the DJANGO email variables need to be configured with legitimate values, and `DJANGO_ADMINS` should be set appropriately
- Make any desired changes
  - If any models were created/deleted/updated:
    - `python manage.py makemigrations`
  - If any static files were added/removed/changed:
    - `python manage.py collectstatic`
  - Commit any changes:
    - `git add --all`
    - `git commit -m "Update: <description>"`
- Run Django migrations:
  - `python manage.py migrate`
- Create at least one Django superuser:
  - `python manage.py createsuperuser`
- Run the local server:
  - `python manage.py runserver`

### Deploy to Heroku

Note: Assuming a local cloned repo of the project is present. See **Local** above for details.

- Install [the Heroku CLI](#)
- Login:
  - `heroku login`
- Create a new application:
  - `heroku create`
  - Note: Optionally specify a unique name

- Create a PostgreSQL instance:
  - `heroku addons:create heroku-postgresql:hobby-dev`
- Configure all environment variables (everything from `.env` except `DATABASE_URL` which Heroku will configure for you):
  - *# DJANGO\_DEBUG defaults to False and does not need to be set for production*
  - *# DO NOT deploy to production with DEBUG set to True!*
  - `heroku config:set DJANGO_SECRET_KEY='...'`
  - `heroku config:set DJANGO_ADMIN_EMAIL='...'`
  - ...
  - Note: To change the application, see Make any desired changes in Local above
- Push the repo to Heroku:
  - `git push heroku master`
- Setup the database:
  - `heroku run python manage.py migrate`
- Create an admin account:
  - `heroku run python manage.py createsuperuser`
- Open the application in a browser:
  - `heroku open`

## APPENDIX C (CLIENT INSTALLATION INSTRUCTIONS)

Client interaction only requires a modern browser. Browse to <https://developernexus.herokuapp.com> or wherever the application is running. Local development (python manage.py runserver) runs at <http://localhost:8000/>.

## APPENDIX D (DEVELOPER SETUP INSTRUCTIONS)

Appendix B describes the build process which includes all required packages. For editing the source code, an appropriate IDE or text editor can be used.

My setup:

- [Visual Studio Code](#)
- Visual Studio Code Extensions:
  - IntelliCode
  - Python
  - Pylance (Python)
  - Sourcery (Python)
  - HTML Boilerplate
  - HTML Preview
  - Intellisense for CSS class names in HTML
  - Markdownlint
  - SQLite