

Welcome to the SecureMe Password Generator Developer's Guide!

This guide equips you with the technical know-how to set up, run, and contribute to the SecureMe Password Generator application. Whether you're a seasoned Django developer or just getting started, this guide will provide a roadmap for navigating the codebase and understanding its functionalities.

Prerequisites:

To embark on this development journey, you'll need a few essential tools:

- **Python:** SecureMe Password Generator leverages the power of Python. Ensure you have a compatible version installed on your system. You can find download information and installation instructions here: <https://www.python.org/downloads/>
- **Django Web Framework:** This application is built upon the robust Django framework. Download and install Django following the official guide: <https://www.djangoproject.com/download/>

Getting Started:

Once you have Python and Django set up, we can dive into the development environment!

1. **Open Your Project Folder:** Fire up your preferred Integrated Development Environment (IDE) – Visual Studio Code, PyCharm, or any code editor you're comfortable with – and navigate to the SecureMe Password Generator project directory.
2. **Activate Virtual Environment (if applicable):** If you're using a virtual environment to manage project dependencies, activate it using the appropriate command (e.g., `source venv/bin/activate` for a virtual environment named `venv`).
3. **Install Dependencies:** The application relies on additional libraries specified in a file named `requirements.txt`. Run the following command in your terminal to install these dependencies:
 4. Bash
 1. `pip install -r requirements.local.txt`
 - 5.
 - 6.

7. Run the Development Server: With everything in place, it's time to bring the application to life! Execute the following command in your terminal to start the development server:

8. Bash

2. `python manage.py runserver`

9.

10.












This command launches the development server, typically providing a local URL (including a port number) where you can access the SecureMe Password Generator application in your web browser.

By following these steps, you'll have a fully functional development environment set up and be ready to delve into the SecureMe Password Generator codebase!

Module Descriptions

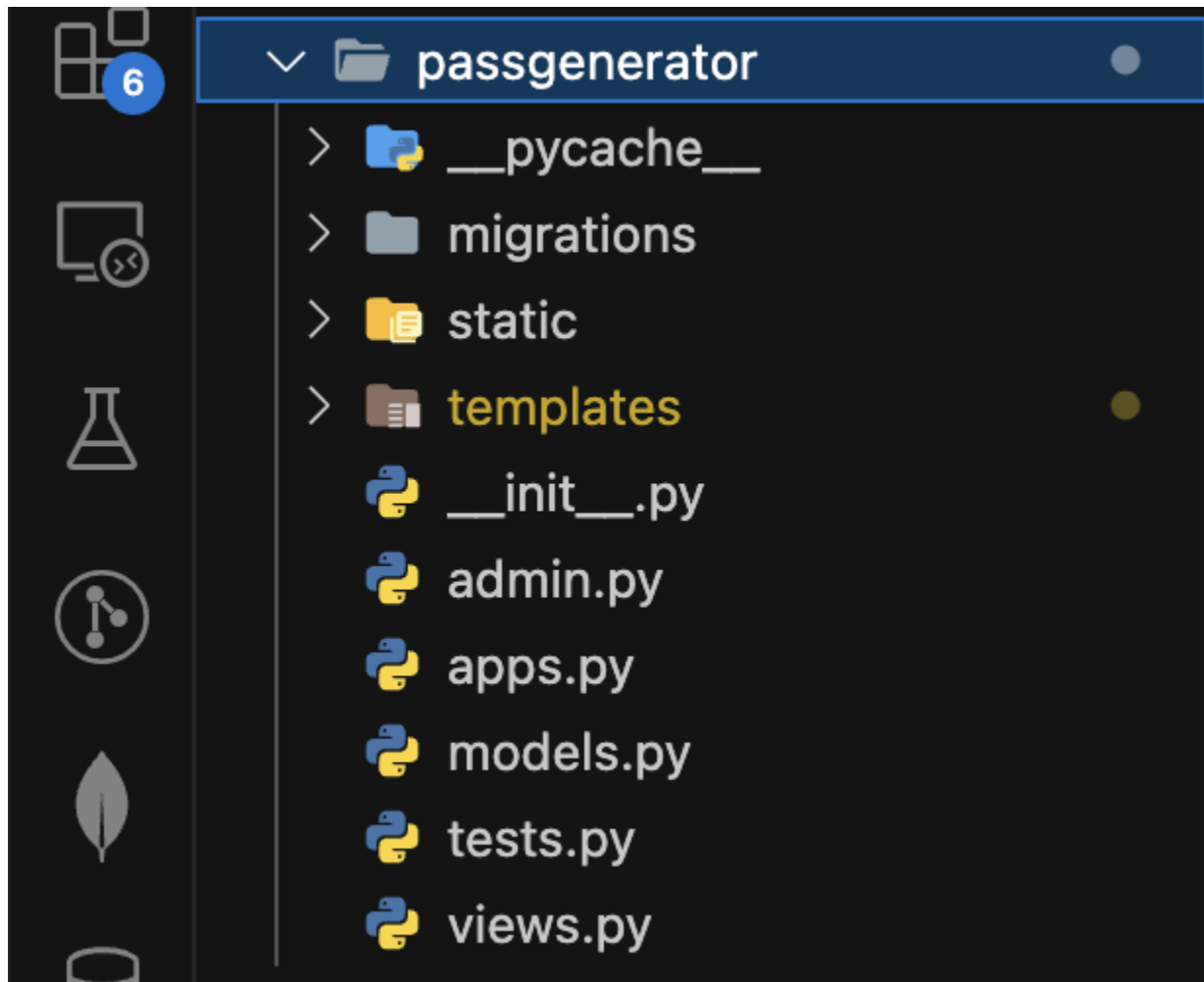
Now let's get into web application module descriptions
Let show you what our project structure looks like



- >  css
- >  media
- >  **passgenerator**
- >  SecureMe
- >  static
- >  venv
-  db.sqlite3
-  manage.py
-  requirements.txt
-  SecureMeDfd.jpg
-  User Guide SecureMe.pdf

The passgenerator is our Project and the SecureMe directory is our app.

Our project has the following files inside



Now lets take a look at the most important ones here which is the views, templates and static dir

The views

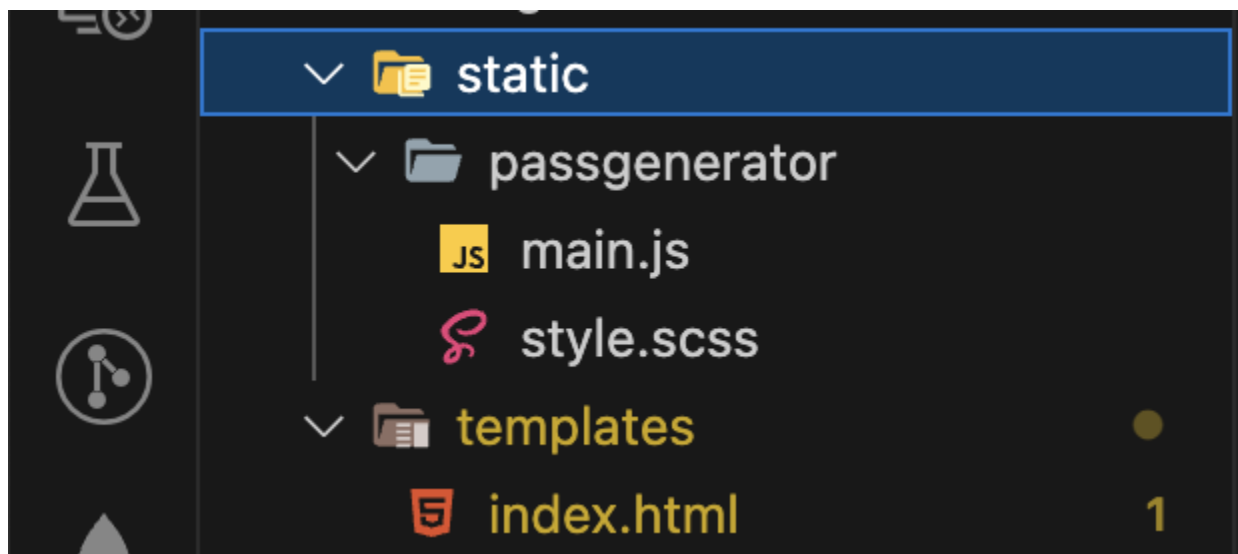
The views.py acts as the bridge between your application's data (models) and its presentation layer (templates or APIs) by handling user requests, processing data, and generating appropriate responses.

Templates dir

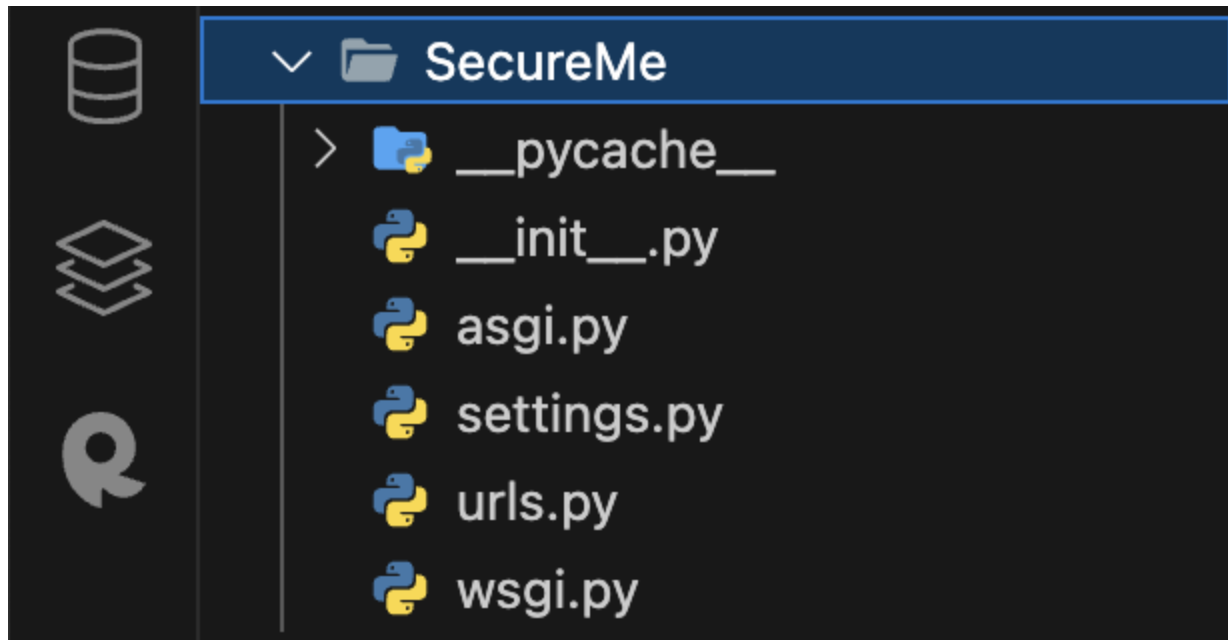
The template directory stores your application's HTML templates that define the structure and layout of your web pages, while the

Static dir

The Static directory directory holds static assets like images, JavaScript, and CSS that enhance the visual presentation of your application.



Now lets move in the SecureMe app dir and see what files are inside and the most important ones



The most important ones here are the settings.py urls.py based on the project requirement

Settings.py

The Settings.py Acts as the central configuration file for your Django project. It stores various settings that govern the application's operation, including:

- **Database configuration:** Specifies the database engine (e.g., PostgreSQL, MySQL) and connection details used by your application to store and retrieve data.
- **Secret keys:** Defines cryptographic keys used for various security purposes within Django (e.g., session management, password hashing).
- **Installed apps:** Lists all the Django apps and third-party libraries that are part of your project.
- **Template and static directories:** Specifies the locations of your application's template files (templates) and static assets (static).
- **Email configuration:** Defines settings for sending emails from your application (optional).

- **Debug mode:** Enables additional debugging features during development (should be disabled in production).
- **Authentication settings:** Configures user authentication mechanisms if your application requires user logins (optional).

Urls.py

The `Url.py` Defines the URL patterns for your Django application. It acts as a roadmap, directing incoming user requests (based on URLs) to the appropriate view functions in your application for handling.

- **Content:** Uses URL patterns typically written using regular expressions to match incoming URLs. Each URL pattern is mapped to a specific view function defined in your application's `views.py` file.
- **Example:** A URL pattern like `path("", views.home, name='home')` might map the root URL (`/`) of your application to the home view function defined in `views.py`. This view function would then be responsible for rendering the homepage of your application.

Now one more last thing. The `requirements.txt` file is a plain text file that lists all the external Python libraries (packages) your Django application relies on to function properly.

- It serves as a central location to specify the exact versions of these dependencies.