1. Create a virtual environment and install Django in it - attach a screen shot

Step 1: Create a virtual environment

python -m venv env

Step 2: Activate the virtual environment

env\Scripts\activate

**Step 3: Install Django**

Once the virtual environment is activated, you can install Django using:

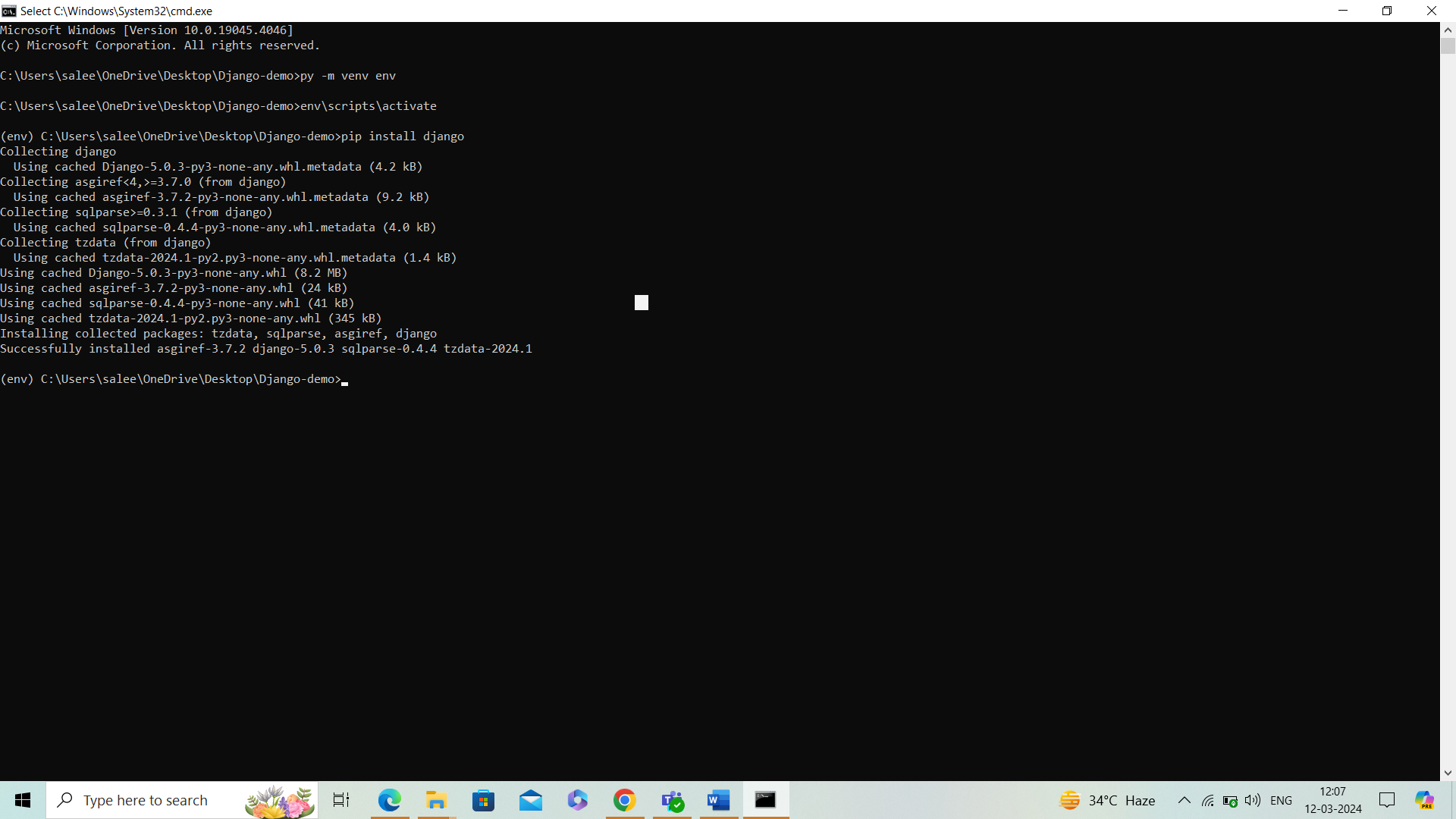
bashCopy code

pip install django

**Step 4: Verify the installation**

You can verify the installation by checking the Django version:

django-admin –version



2) Create a Django project and app , explain the files of the project and app folders.

**Creating a Django Project and App:**

Step 1: Create a Django Project

To create a Django project, you use the **django-admin** command-line utility:

**django-admin startproject projectname**

Replace **projectname** with the desired name of your Django project.

Step 2: Create a Django App

Inside your Django project directory, you create Django apps. Apps are like components that perform specific functions within your project:

cd projectname

python manage.py startapp appname

Replace **appname** with the desired name of your Django app.

**Files and Folders in the Project Directory:**

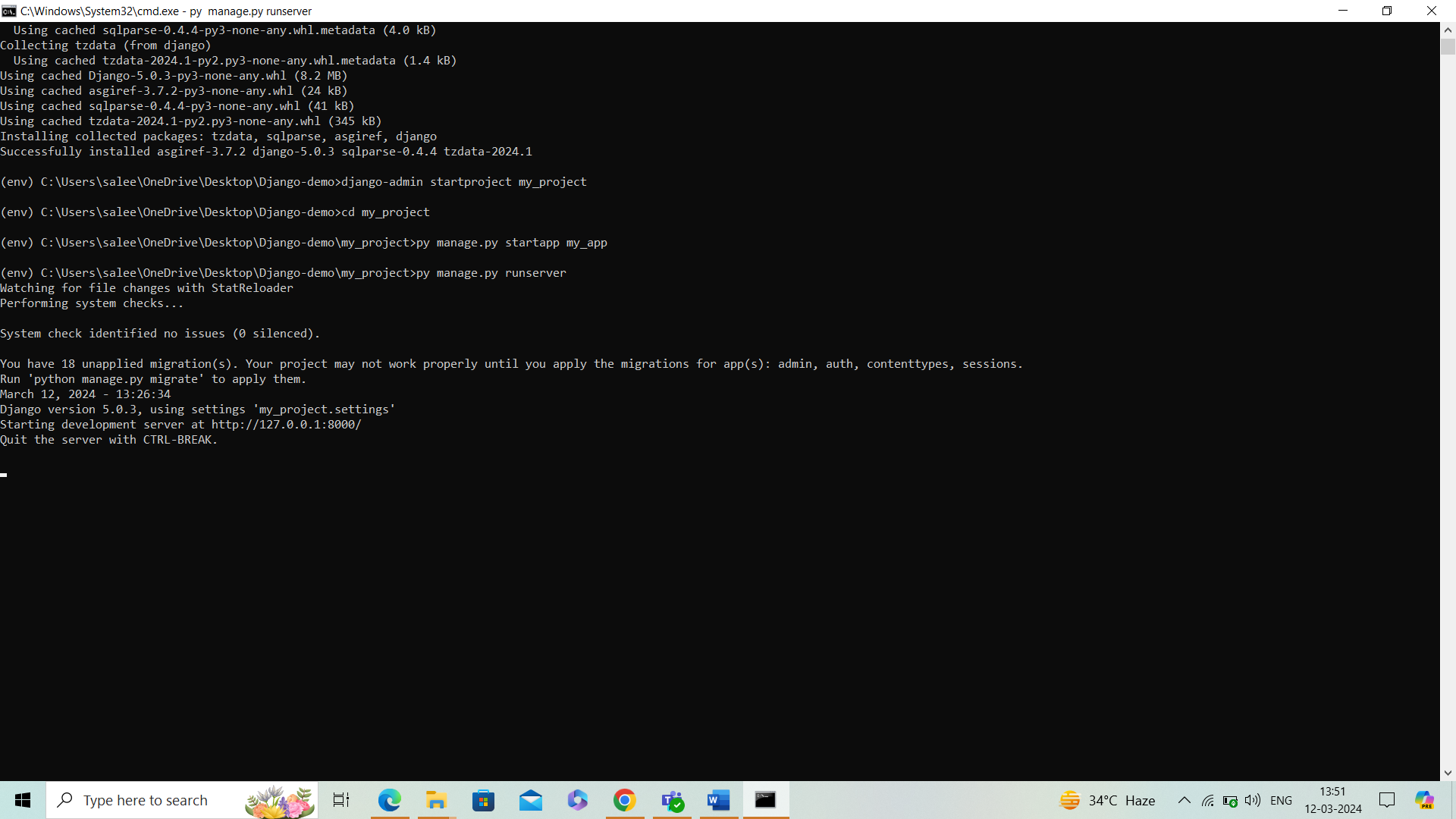
1. **manage.py:** This is a command-line utility that lets you interact with your Django project. You can use it to run development servers, create database migrations, and perform other administrative tasks.
2. **projectname/ (directory):** This directory contains the settings, URLs, and other configurations for your Django project.
   * **\_\_init\_\_.py:** This file is an empty Python file that tells Python that this directory should be considered a Python package.
   * **settings.py:** This file contains all the configuration settings for your Django project, including database configuration, installed apps, middleware, static files, and more.
   * **urls.py:** This file contains the URL patterns for your Django project. You define the mappings between URLs and views (functions that handle web requests) here.
   * **wsgi.py:** This file contains the configuration for the WSGI (Web Server Gateway Interface) server, which is used to deploy your Django project to production.
   * **asgi.py:** This file contains the configuration for the ASGI (Asynchronous Server Gateway Interface) server, which is used for asynchronous web servers like Daphne.
3. **venv/ (directory):** This directory typically contains the Python virtual environment for your project, which isolates your project's dependencies from other projects.
4. **db.sqlite3:** This is the default SQLite database file that Django uses during development.

**Files and Folders in the App Directory:**

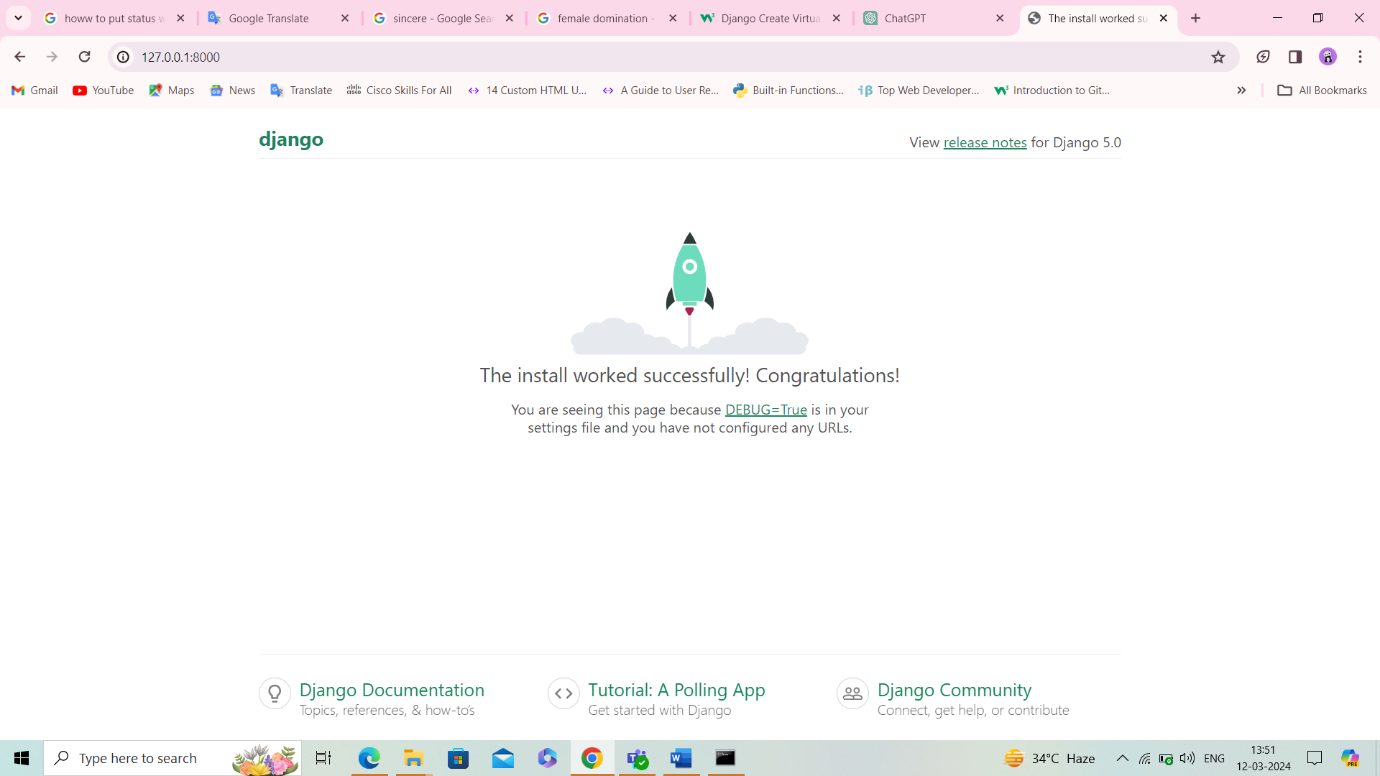
1. **models.py:** This file contains the definitions of your Django models, which are Python classes that represent database tables.
2. **views.py:** This file contains the view functions, which are Python functions that handle web requests and return web responses.
3. **urls.py:** This file contains the URL patterns for your Django app. You define the mappings between URLs and views specific to this app here.
4. **admin.py:** This file allows you to register your models with the Django admin interface, allowing you to manage your data through the admin interface.
5. **apps.py:** This file contains the configuration for the app, such as the app's name.
6. **migrations/ (directory):** This directory contains database migration files generated by Django's migration system. Migrations are used to manage changes to your database schema over time.
7. **\_\_init\_\_.py:** This file, like in the project directory, indicates that this directory should be considered a Python package.
8. Run the Django project - attach screen shots of the Django admin panel and SITE.

To Run the development server:

**python manage.py runserver**

****

Look server has started and can be accessed at localhost with port 8000. Let's access it using the browser, it looks like the below



1. **Run the createsuperuser command:** Run the following command to create an admin user:

python manage.py createsuperuser

You will be prompted to enter a username, email address, and password for the new superuser.

1. **Enter superuser details:**
   * Enter a desired username when prompted.
   * You'll be asked for an email address; this can be left blank if you don't want to provide one.
   * Enter a secure password. Note that the password will not be visible as you type.
2. **Confirm creation:** After entering the details, you'll be asked to confirm the information:

Superuser created successfully.

1. **Run the development server:** Ensure your development server is running:

python manage.py runserver

1. **Access the admin panel:** Open your web browser and navigate to **http://localhost:8000/admin/** (or the appropriate URL based on your server configuration).
2. **Log in with the superuser credentials:** Enter the username and password you just created for the superuser.

