

**Assignment 3**

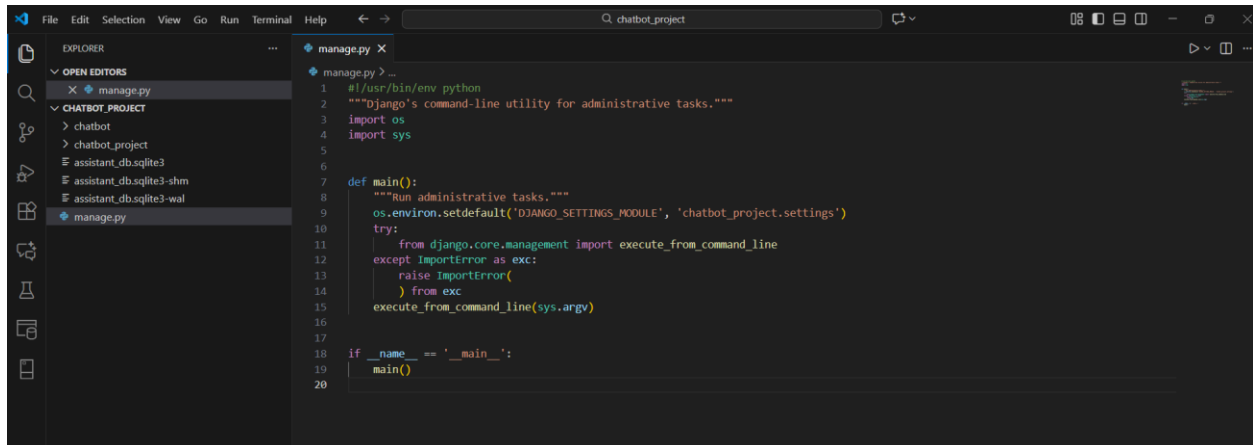
Prashanna Acharya

Advance Artificial Intelligence (MSCS-633-B01)

University of the Cumberlands

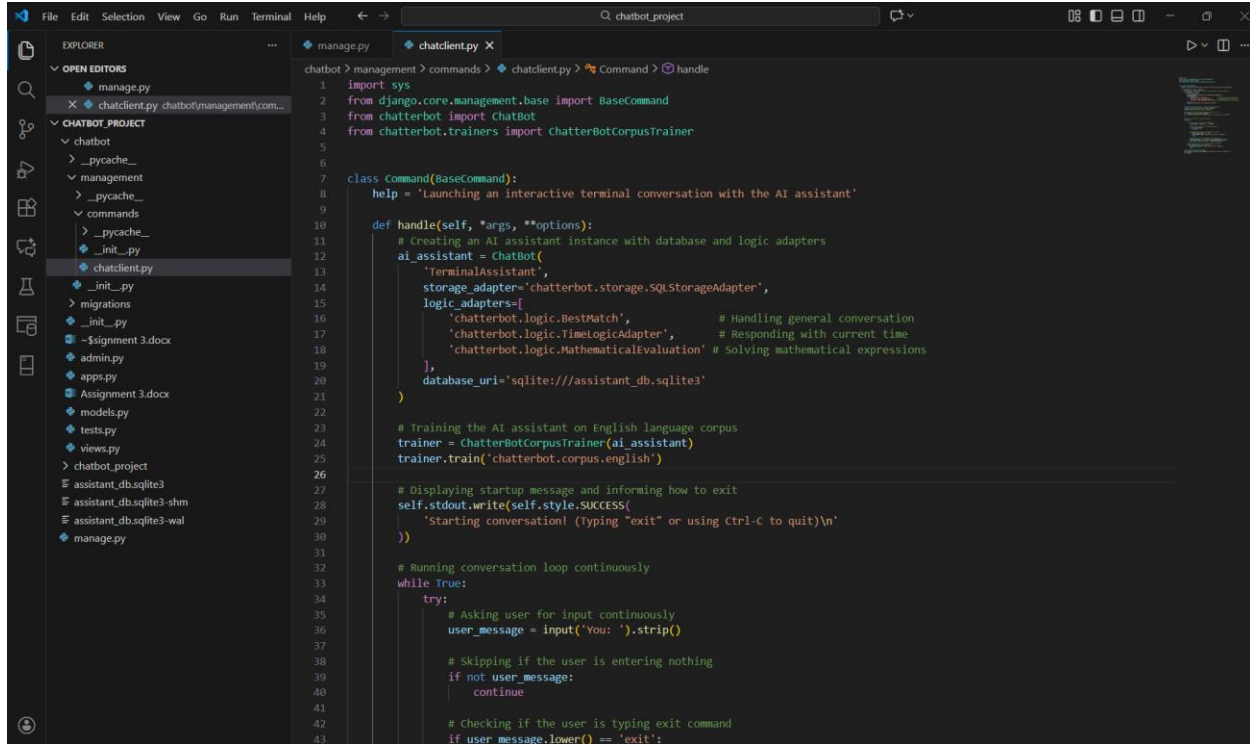
Dr. Ulrich Vouama

## manage.py



The `manage.py` file is a script that Django automatically generates when a new project is created. Its role is to provide a command-line tool for controlling and working with the Django application. When executed, it sets the appropriate environment variable, so Django knows which settings configuration to load. After that, it brings in Django's command-handling machinery and passes control to it. This setup enables to run various project-related commands, such as launching the development server, managing database migrations, creating apps, or executing custom commands. Essentially, `manage.py` acts as the main gateway for performing administrative operations in a Django project and ensures everything is properly configured before those tasks run.

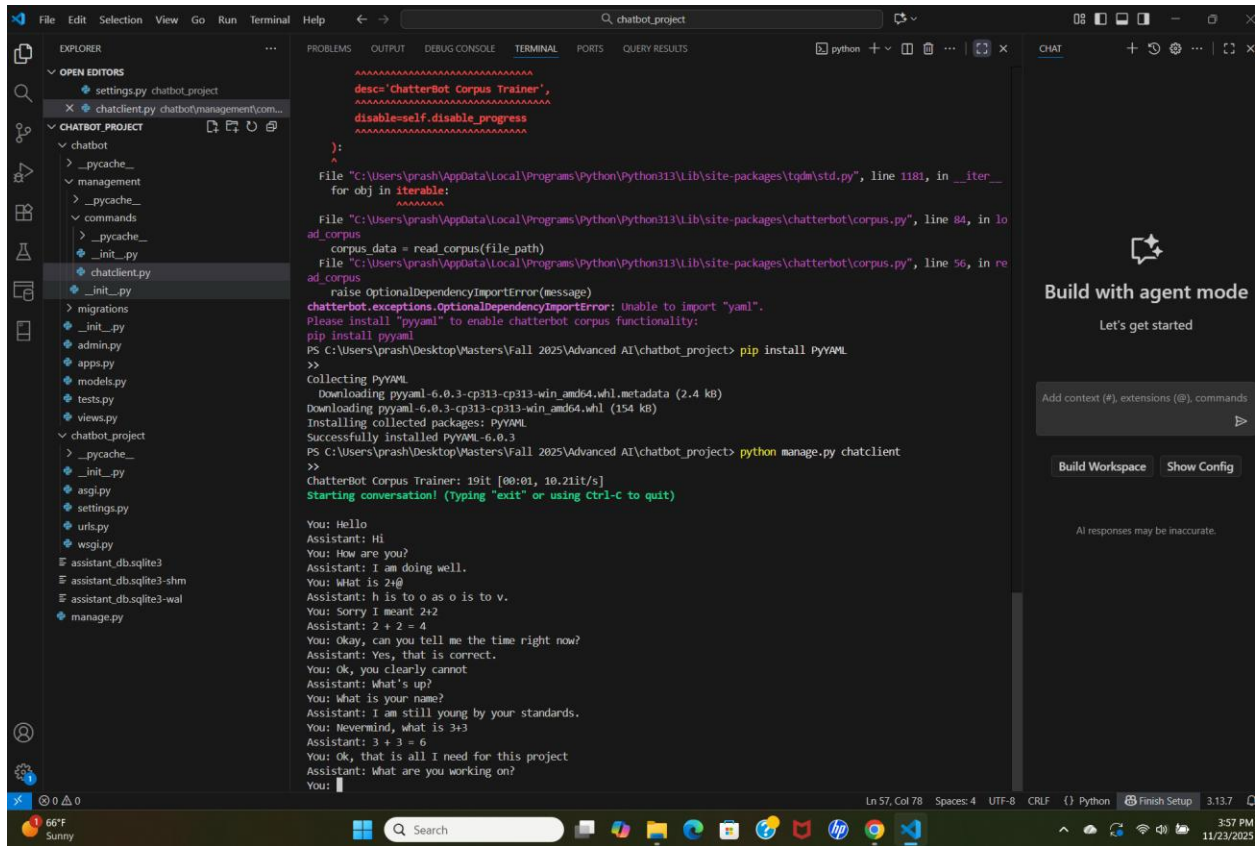
## chatclient.py



```
1 import sys
2 from django.core.management.base import BaseCommand
3 from chatterbot import ChatBot
4 from chatterbot.trainers import ChatterBotCorpusTrainer
5
6
7 class Command(BaseCommand):
8     help = 'Launching an interactive terminal conversation with the AI assistant'
9
10     def handle(self, *args, **options):
11         # Creating an AI assistant instance with database and logic adapters
12         ai_assistant = ChatBot(
13             'TerminalAssistant',
14             storage_adapter='chatterbot.storage.SQLiteStorageAdapter',
15             logic_adapters=[
16                 'chatterbot.logic.BestMatch',          # Handling general conversation
17                 'chatterbot.logic.TimeLogicAdapter',   # Responding with current time
18                 'chatterbot.logic.MathematicalEvaluation' # Solving mathematical expressions
19             ],
20             database_uri='sqlite:///assistant_db.sqlite3'
21         )
22
23         # Training the AI assistant on English language corpus
24         trainer = ChatterBotCorpusTrainer(ai_assistant)
25         trainer.train('chatterbot.corpus.english')
26
27         # Displaying startup message and informing how to exit
28         self.stdout.write(self.style.SUCCESS(
29             'Starting conversation! (typing "exit" or using Ctrl-C to quit)\n'
30         ))
31
32         # Running conversation loop continuously
33         while True:
34             try:
35                 # Asking user for input continuously
36                 user_message = input('You: ').strip()
37
38                 # Skipping if the user is entering nothing
39                 if not user_message:
40                     continue
41
42                 # Checking if the user is typing exit command
43                 if user_message.lower() == 'exit':
```

The chatclient.py file is implemented as a Django management command that is intended to operate a ChatterBot-driven chatbot within the terminal. A Command class is defined by extending Django's BaseCommand, allowing the script to be detected and executed through the command python manage.py chatclient. Inside the command, a ChatBot instance is configured with a storage adapter, multiple logic adapters, and an SQLite database in which conversational data is stored. Training is then carried out using ChatterBot's English corpus so that a basic level of conversational ability is provided. Once training has been completed, an interactive loop is initiated, through which user input is continuously received and responses from the chatbot are generated in real time. The session is maintained until the user types "exit" or interrupts the program. Through this design, the Django project is enabled to function as a terminal-based chatbot system controlled by a custom management command.

## Output on Terminal



```
File "C:\Users\prash\AppData\Local\Programs\Python\Python313\Lib\site-packages\tqdm\std.py", line 1181, in __iter__
for obj in iterable:

File "C:\Users\prash\AppData\Local\Programs\Python\Python313\Lib\site-packages\chatterbot\corpus.py", line 84, in io
ad_corpus
corpus_data = read_corpus(file_path)
File "C:\Users\prash\AppData\Local\Programs\Python\Python313\Lib\site-packages\chatterbot\corpus.py", line 56, in re
ad_corpus
raise OptionalDependencyImportError(message)
chatterbot.exceptions.optionaldependencyimporterror: Unable to import "yaml".
Please install "pyyaml" to enable chatterbot corpus functionality:
pip install pyyaml
PS C:\Users\prash\Desktop\Masters\Fall 2025\Advanced AI\chatbot_project> pip install PyYAML
>>
Collecting PyYAML
  Downloading pyyaml-6.0.3-cp313-cp313-win_amd64.whl.metadata (2.4 kB)
  Downloading pyyaml-6.0.3-cp313-cp313-win_amd64.whl (154 kB)
Installing collected packages: PyYAML
Successfully installed PyYAML-6.0.3
PS C:\Users\prash\Desktop\Masters\Fall 2025\Advanced AI\chatbot_project> python manage.py chatclient
>>
ChatterBot Corpus Trainer: 19it [00:01, 10.21it/s]
Starting conversation! (Typing "exit" or using Ctrl-C to quit)

You: Hello
Assistant: Hi
You: How are you?
Assistant: I am doing well.
You: What is 2+2?
Assistant: h is to o as o is to v.
You: Sorry I meant 2+2
Assistant: 2 + 2 = 4
You: Okay, can you tell me the time right now?
Assistant: Yes, that is correct.
You: Ok, you clearly cannot
Assistant: What's up?
You: What is your name?
Assistant: I am still young by your standards.
You: Nevermind, what is 3+3
Assistant: 3 + 3 = 6
You: Ok, that is all I need for this project
Assistant: What are you working on?
You: 
```