# AI ASSISTED CODING

## LAB TEST 3

**NAME: SRIRAMOJU AJAY**

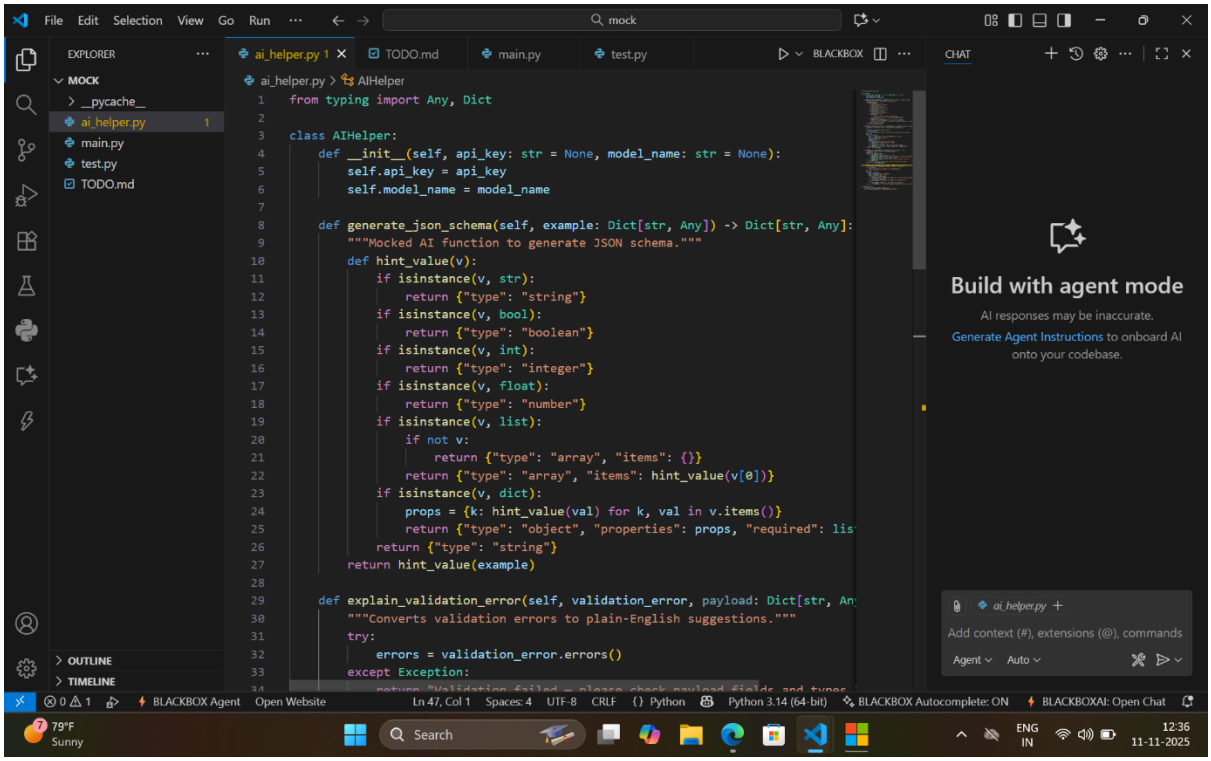
**HT:NO : 2503A51L25**

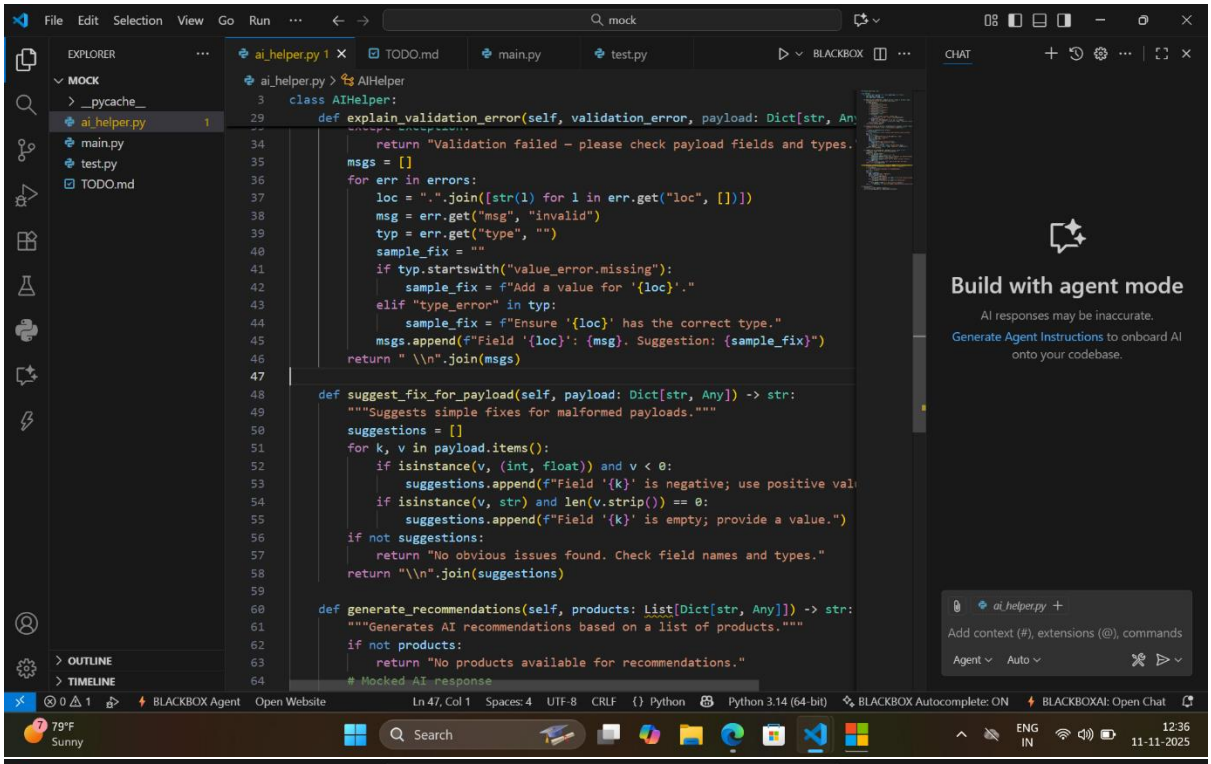
**BATCH:19**

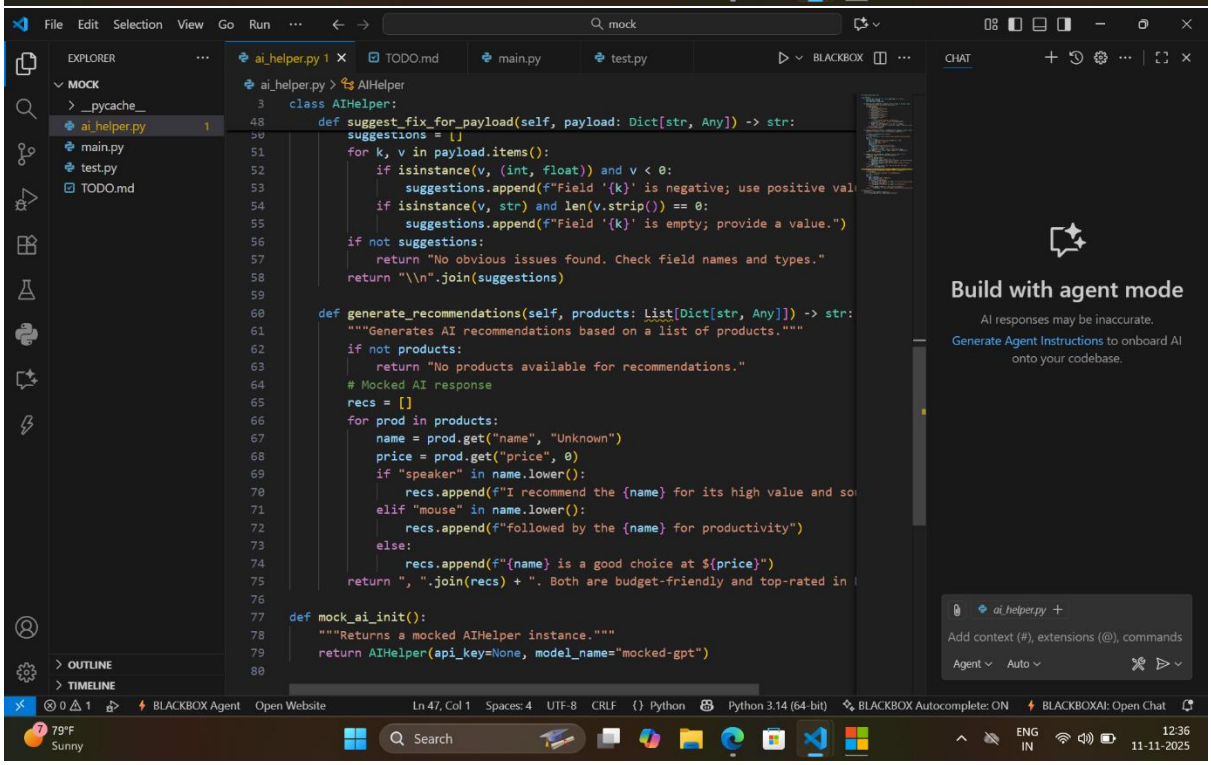
**SET : E7**

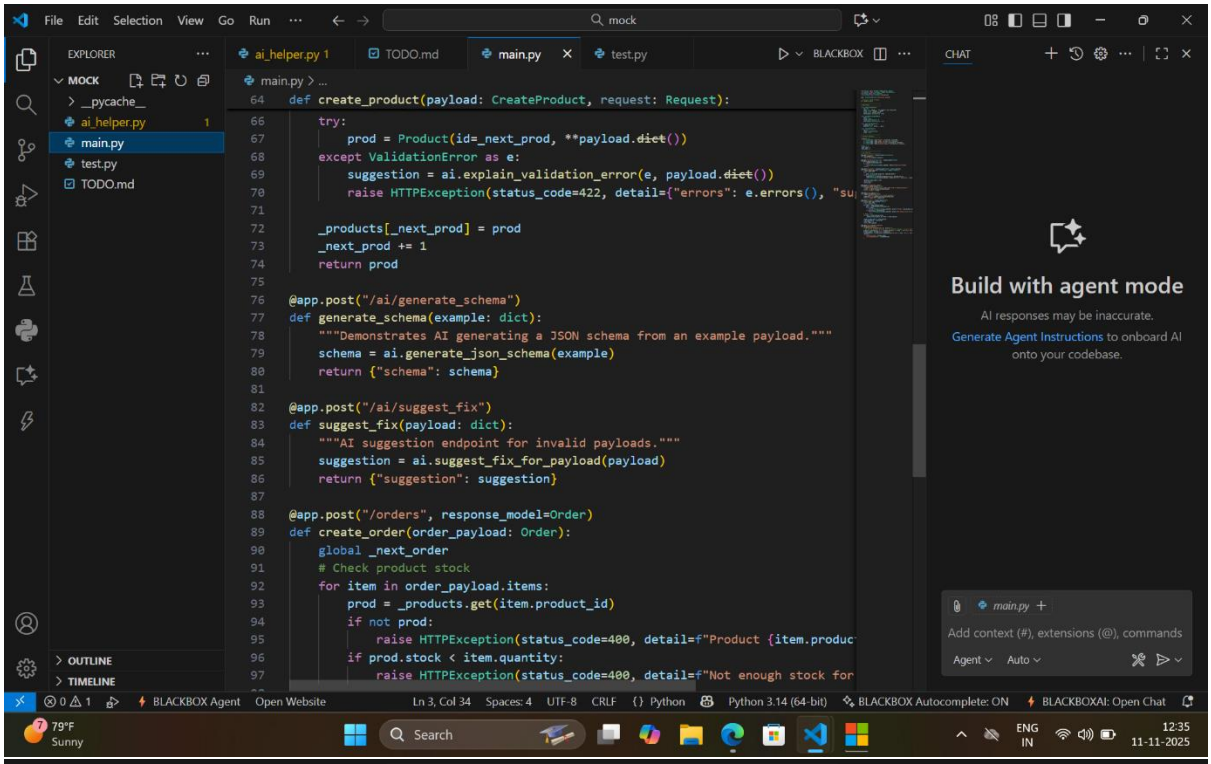
Q1: Scenario: In the domain of E-commerce, a company is facing a challenge related to backend api development. Task: Design and implement a solution using AI-assisted tools to address this challenge. Include code, explanation of AI integration, and test results. Deliverables: Source code, explanation, and output screenshots

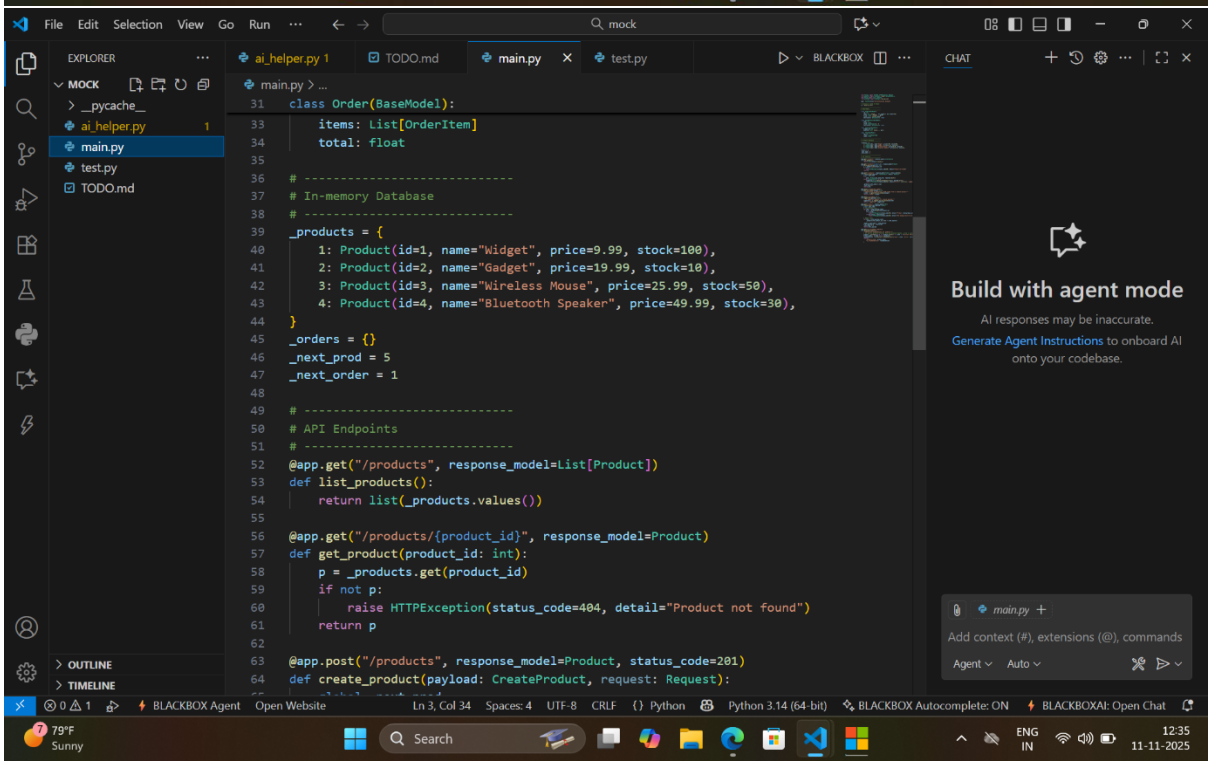
```python
from fastapi import FastAPI, HTTPException, Request
from pydantic import BaseModel, Field, ValidationError
from typing import List, Optional
from ai_helper import AIHelper, mock_ai_init

app = FastAPI(title="Ecom AI-Assisted Backend")

# Initialize mocked AI helper
ai = mock_ai_init()

# ----------------------------
# Data Models
# ----------------------------
class Product(BaseModel):
    id: int
    name: str = Field(..., min_length=2, max_length=200)
    price: float = Field(..., gt=0)
    stock: int = Field(0, ge=0)
    description: Optional[str] = None

class CreateProduct(BaseModel):
    name: str
    price: float
    stock: Optional[int] = 0
    description: Optional[str] = None

class OrderItem(BaseModel):
    product_id: int
    quantity: int = Field(..., ge=1)

class Order(BaseModel):
    id: int
    items: List[OrderItem]
    total: float
```

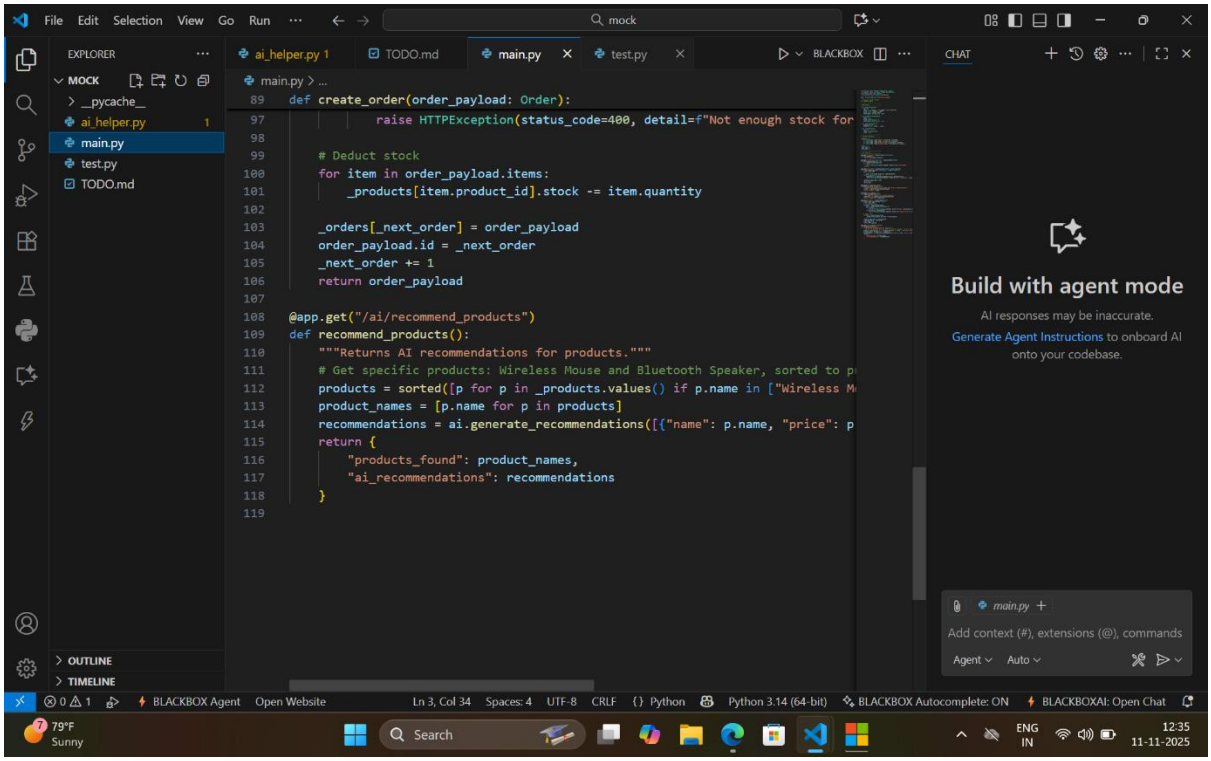

File Edit Selection View Go Run
mock
EXPLORER
MOCK
__pycache__
ai_helper.py 1
main.py
test.py
TODO.md
ai_helper.py 1
TODO.md
main.py
test.py
BLACKBOX
ai_helper.py > AIHelper
from typing import Any, Dict
class AIHelper:
def __init__(self, api_key: str = None, model_name: str = None):
self.api_key = api_key
self.model_name = model_name
def generate_json_schema(self, example: Dict[str, Any]) -> Dict[str, Any]:
"""Mocked AI function to generate JSON schema."""
def hint_value(v):
if isinstance(v, str):
return {"type": "string"}
if isinstance(v, bool):
return {"type": "boolean"}
if isinstance(v, int):
return {"type": "integer"}
if isinstance(v, float):
return {"type": "number"}
if isinstance(v, list):
if not v:
return {"type": "array", "items": {}}
return {"type": "array", "items": hint_value(v[0])}
if isinstance(v, dict):
props = {k: hint_value(val) for k, val in v.items()}
return {"type": "object", "properties": props, "required": lis
return {"type": "string"}
return hint_value(example)
def explain_validation_error(self, validation_error, payload: Dict[str, An
"""Converts validation errors to plain-English suggestions."""
try:
errors = validation_error.errors()
except Exception:
OUTLINE
TIMELINE
CHAT
Build with agent mode
AI responses may be inaccurate.
Generate Agent Instructions to onboard AI onto your codebase.
ai_helper.py
Add context (#), extensions (@), commands
Agent
Auto
0 1
BLACKBOX Agent
Open Website
Ln 47, Col 1
Spaces: 4
UTF-8
CRLF
Python
Python 3.14 (64-bit)
BLACKBOX Autocomplete: ON
BLACKBOXAI: Open Chat
79°F
Sunny
Search
ENG
IN
12:36
11-11-2025


File Edit Selection View Go Run
mock
EXPLORER
MOCK
__pycache__
ai_helper.py 1
main.py
test.py
TODO.md
ai_helper.py 1
TODO.md
main.py
test.py
BLACKBOX
CHAT
ai_helper.py > AIHelper
class AIHelper:
def explain_validation_error(self, validation_error, payload: Dict[str, An
return "Validation failed — please check payload fields and types.
msgs = []
for err in errors:
loc = ".".join([str(l) for l in err.get("loc", [])])
msg = err.get("msg", "invalid")
typ = err.get("type", "")
sample_fix = ""
if typ.startswith("value_error.missing"):
sample_fix = f"Add a value for '{loc}'."
elif "type_error" in typ:
sample_fix = f"Ensure '{loc}' has the correct type."
msgs.append(f"Field '{loc}': {msg}. Suggestion: {sample_fix}")
return " \\n".join(msgs)
def suggest_fix_for_payload(self, payload: Dict[str, Any]) -> str:
"""Suggests simple fixes for malformed payloads."""
suggestions = []
for k, v in payload.items():
if isinstance(v, (int, float)) and v < 0:
suggestions.append(f"Field '{k}' is negative; use positive val
if isinstance(v, str) and len(v.strip()) == 0:
suggestions.append(f"Field '{k}' is empty; provide a value.")
if not suggestions:
return "No obvious issues found. Check field names and types."
return "\\n".join(suggestions)
def generate_recommendations(self, products: List[Dict[str, Any]]) -> str:
"""Generates AI recommendations based on a list of products."""
if not products:
return "No products available for recommendations."
# Mocked AI response
Build with agent mode
AI responses may be inaccurate.
Generate Agent Instructions to onboard AI onto your codebase.
ai_helper.py
Add context (#), extensions (@), commands
Agent
Auto
OUTLINE
TIMELINE
BLACKBOX Agent
Open Website
Ln 47, Col 1
Spaces: 4
UTF-8
CRLF
Python
Python 3.14 (64-bit)
BLACKBOX Autocomplete: ON
BLACKBOXAI: Open Chat
79°F
Sunny
Search
ENG
IN
12:36
11-11-2025


File Edit Selection View Go Run
mock
EXPLORER
MOCK
__pycache__
ai_helper.py 1
main.py
test.py
TODO.md
ai_helper.py 1
TODO.md
main.py
test.py
BLACKBOX
CHAT
ai_helper.py > AIHelper
class AIHelper:
def suggest_fix_for_payload(self, payload: Dict[str, Any]) -> str:
suggestions = []
for k, v in payload.items():
if isinstance(v, (int, float)) and v < 0:
suggestions.append(f"Field '{k}' is negative; use positive val
if isinstance(v, str) and len(v.strip()) == 0:
suggestions.append(f"Field '{k}' is empty; provide a value.")
if not suggestions:
return "No obvious issues found. Check field names and types."
return "\\n".join(suggestions)
def generate_recommendations(self, products: List[Dict[str, Any]]) -> str:
"""Generates AI recommendations based on a list of products."""
if not products:
return "No products available for recommendations."
# Mocked AI response
recs = []
for prod in products:
name = prod.get("name", "Unknown")
price = prod.get("price", 0)
if "speaker" in name.lower():
recs.append(f"I recommend the {name} for its high value and so
elif "mouse" in name.lower():
recs.append(f"followed by the {name} for productivity")
else:
recs.append(f"{name} is a good choice at ${price}")
return ", ".join(recs) + ". Both are budget-friendly and top-rated in
def mock_ai_init():
"""Returns a mocked AIHelper instance."""
return AIHelper(api_key=None, model_name="mocked-gpt")
Build with agent mode
AI responses may be inaccurate.
Generate Agent Instructions to onboard AI onto your codebase.
ai_helper.py
Add context (#), extensions (@), commands
Agent
Auto
OUTLINE
TIMELINE
BLACKBOX Agent
Open Website
Ln 47, Col 1
Spaces: 4
UTF-8
CRLF
Python
Python 3.14 (64-bit)
BLACKBOX Autocomplete: ON
BLACKBOXAI: Open Chat
79°F
Sunny
Search
ENG
IN
12:36
11-11-2025

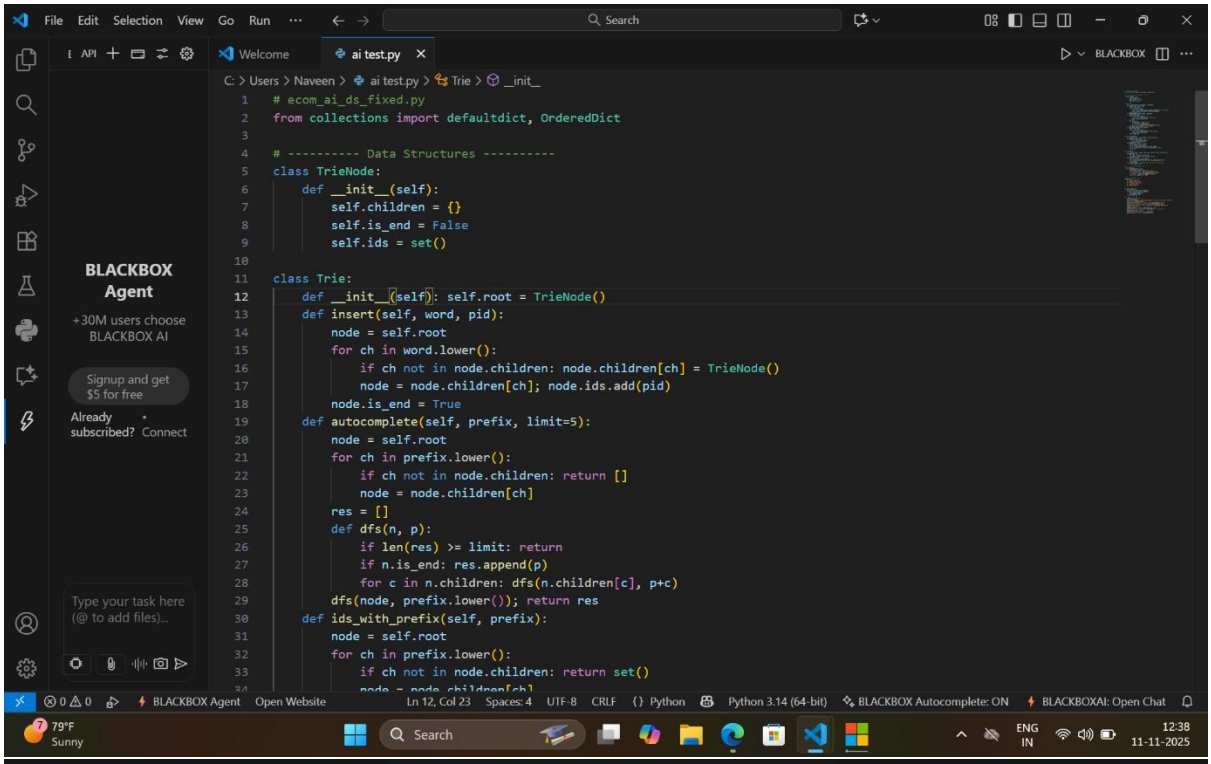

File Edit Selection View Go Run
mock
EXPLORER
MOCK
__pycache__
ai_helper.py 1
main.py
test.py
TODO.md
ai_helper.py 1
TODO.md
main.py
test.py
BLACKBOX
CHAT
main.py > ...
def create_product(payload: CreateProduct, request: Request):
try:
prod = Product(id=_next_prod, **payload.dict())
except ValidationError as e:
suggestion = ai.explain_validation_error(e, payload.dict())
raise HTTPException(status_code=422, detail={"errors": e.errors(), "su
_products[_next_prod] = prod
_next_prod += 1
return prod
@app.post("/ai/generate_schema")
def generate_schema(example: dict):
"""Demonstrates AI generating a JSON schema from an example payload."""
schema = ai.generate_json_schema(example)
return {"schema": schema}
@app.post("/ai/suggest_fix")
def suggest_fix(payload: dict):
"""AI suggestion endpoint for invalid payloads."""
suggestion = ai.suggest_fix_for_payload(payload)
return {"suggestion": suggestion}
@app.post("/orders", response_model=Order)
def create_order(order_payload: Order):
global _next_order
# Check product stock
for item in order_payload.items:
prod = _products.get(item.product_id)
if not prod:
raise HTTPException(status_code=400, detail=f"Product {item.produc
if prod.stock < item.quantity:
raise HTTPException(status_code=400, detail=f"Not enough stock for
Build with agent mode
AI responses may be inaccurate.
Generate Agent Instructions to onboard AI onto your codebase.
main.py
Add context (#), extensions (@), commands
Agent
Auto
OUTLINE
TIMELINE
BLACKBOX Agent
Open Website
Ln 3, Col 34
Spaces: 4
UTF-8
CRLF
{} Python
Python 3.14 (64-bit)
BLACKBOX Autocomplete: ON
BLACKBOXAI: Open Chat
79°F
Sunny
Search
ENG
IN
12:35
11-11-2025

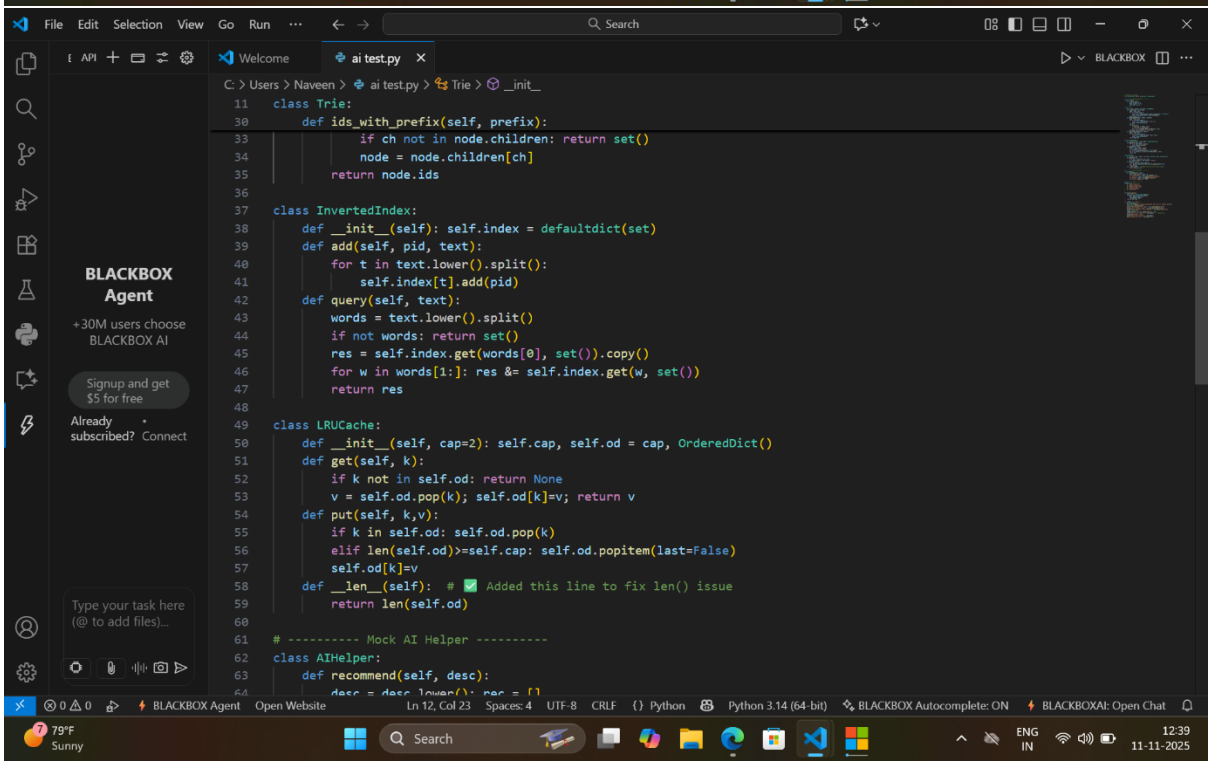

File Edit Selection View Go Run
mock
EXPLORER
MOCK
__pycache__
ai_helper.py 1
main.py
test.py
TODO.md
ai_helper.py 1
TODO.md
main.py
test.py
BLACKBOX
CHAT
main.py > ...
class Order(BaseModel):
items: List[OrderItem]
total: float
# ----------------------------
# In-memory Database
# ----------------------------
_products = {
1: Product(id=1, name="Widget", price=9.99, stock=100),
2: Product(id=2, name="Gadget", price=19.99, stock=10),
3: Product(id=3, name="Wireless Mouse", price=25.99, stock=50),
4: Product(id=4, name="Bluetooth Speaker", price=49.99, stock=30),
}
_orders = {}
_next_prod = 5
_next_order = 1
# ----------------------------
# API Endpoints
# ----------------------------
@app.get("/products", response_model=List[Product])
def list_products():
return list(_products.values())
@app.get("/products/{product_id}", response_model=Product)
def get_product(product_id: int):
p = _products.get(product_id)
if not p:
raise HTTPException(status_code=404, detail="Product not found")
return p
@app.post("/products", response_model=Product, status_code=201)
def create_product(payload: CreateProduct, request: Request):
Build with agent mode
AI responses may be inaccurate.
Generate Agent Instructions to onboard AI onto your codebase.
main.py
Add context (#), extensions (@), commands
Agent
Auto
OUTLINE
TIMELINE
BLACKBOX Agent
Open Website
Ln 3, Col 34
Spaces: 4
UTF-8
CRLF
{} Python
Python 3.14 (64-bit)
BLACKBOX Autocomplete: ON
BLACKBOXAI: Open Chat
79°F
Sunny
Search
ENG
IN
12:35
11-11-2025

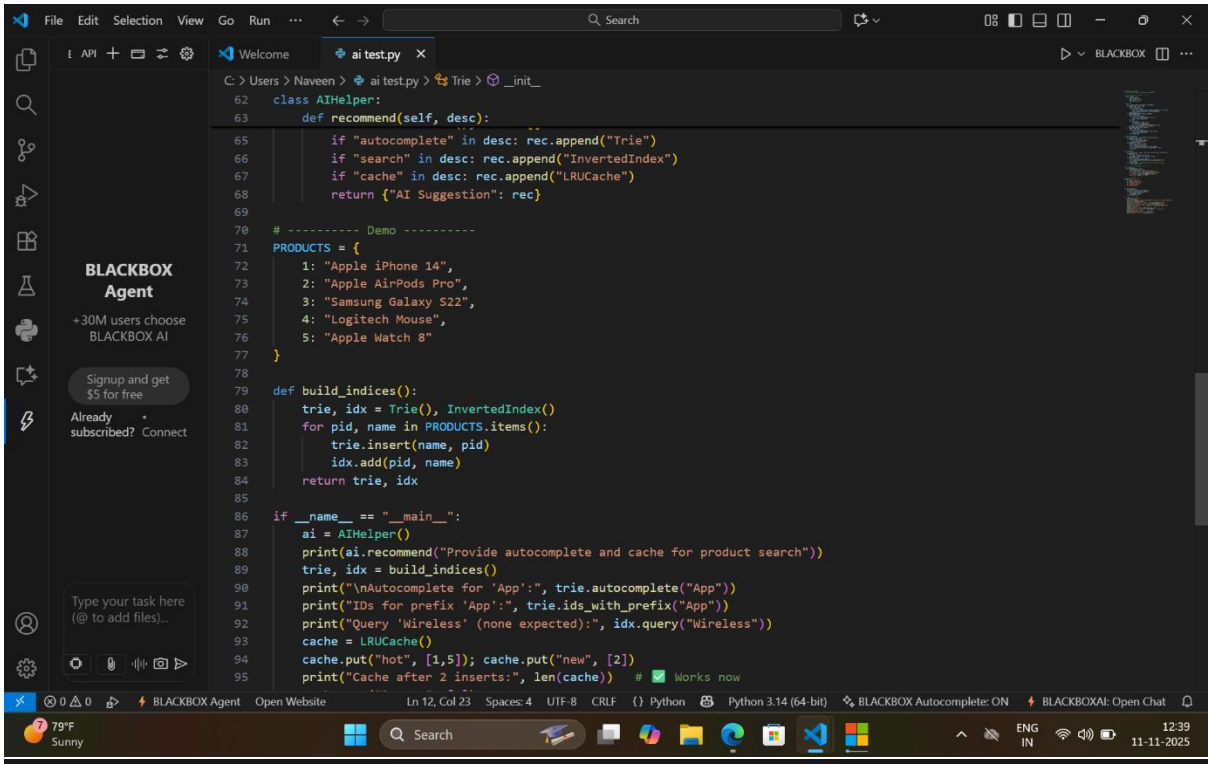

File Edit Selection View Go Run
mock
EXPLORER
MOCK
__pycache__
ai_helper.py 1
main.py
test.py
TODO.md
ai_helper.py 1
TODO.md
main.py
test.py
BLACKBOX
main.py > ...
def create_order(order_payload: Order):
raise HTTPException(status_code=400, detail=f"Not enough stock for
# Deduct stock
for item in order_payload.items:
_products[item.product_id].stock -= item.quantity
_orders[_next_order] = order_payload
order_payload.id = _next_order
_next_order += 1
return order_payload
@app.get("/ai/recommend_products")
def recommend_products():
"""Returns AI recommendations for products."""
# Get specific products: Wireless Mouse and Bluetooth Speaker, sorted to p
products = sorted([p for p in _products.values() if p.name in ["Wireless M
product_names = [p.name for p in products]
recommendations = ai.generate_recommendations([{"name": p.name, "price": p
return {
"products_found": product_names,
"ai_recommendations": recommendations
}
OUTLINE
TIMELINE
CHAT
Build with agent mode
AI responses may be inaccurate.
Generate Agent Instructions to onboard AI onto your codebase.
main.py
Add context (#), extensions (@), commands
Agent
Auto
BLACKBOX Agent
Open Website
Ln 3, Col 34
Spaces: 4
UTF-8
CRLF
Python
Python 3.14 (64-bit)
BLACKBOX Autocomplete: ON
BLACKBOXAI: Open Chat
79°F
Sunny
Search
ENG
IN
12:35
11-11-2025

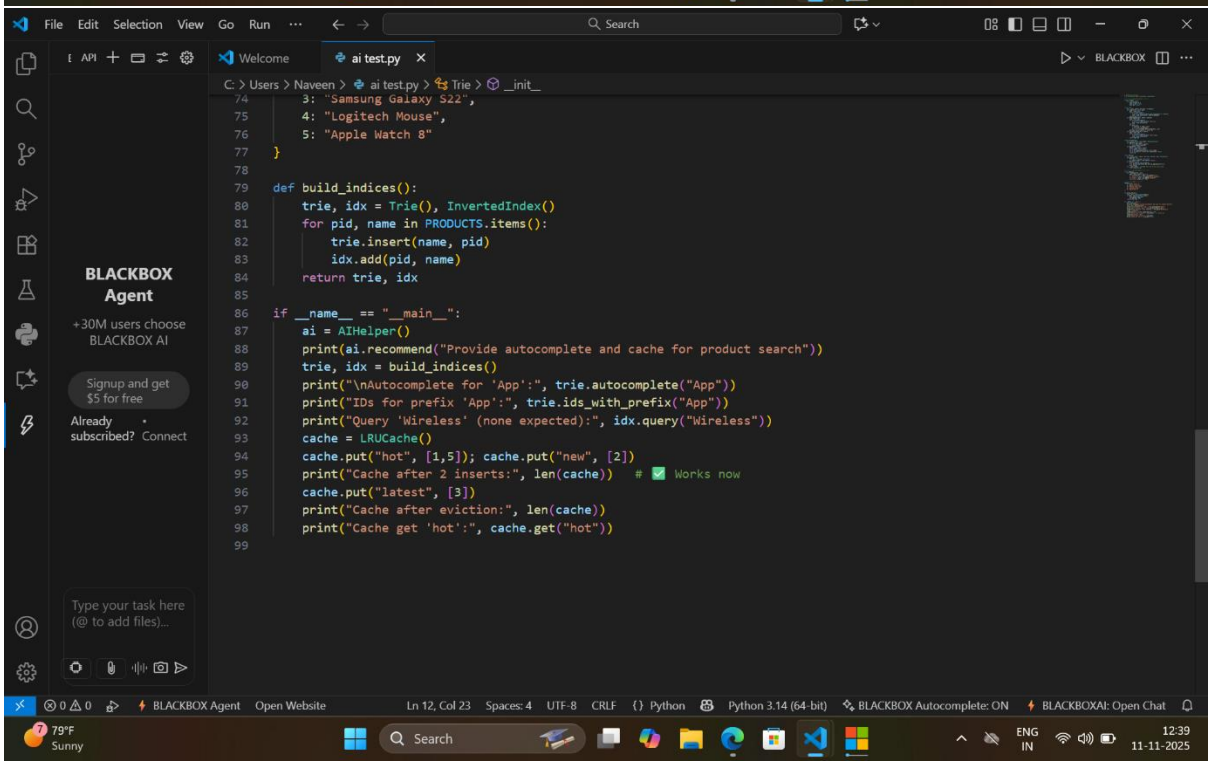
OUT PUT

```
StatusCode        : 200
StatusDescription : OK
Content           : {"products_found":["Wireless Mouse","Bluetooth
                    Speaker"],"ai_recommendations":"followed by the Wireless
                    Mouse for productivity, I recommend the Bluetooth Speaker
                    for its high value and sound quality. ...
```


File Edit Selection View Go Run
Search
Welcome
ai test.py
BLACKBOX
C: > Users > Naveen > ai test.py > Trie > __init__
# ecom_ai_ds_fixed.py
from collections import defaultdict, OrderedDict
# ---------- Data Structures ----------
class TrieNode:
def __init__(self):
self.children = {}
self.is_end = False
self.ids = set()
class Trie:
def __init__(self): self.root = TrieNode()
def insert(self, word, pid):
node = self.root
for ch in word.lower():
if ch not in node.children: node.children[ch] = TrieNode()
node = node.children[ch]; node.ids.add(pid)
node.is_end = True
def autocomplete(self, prefix, limit=5):
node = self.root
for ch in prefix.lower():
if ch not in node.children: return []
node = node.children[ch]
res = []
def dfs(n, p):
if len(res) >= limit: return
if n.is_end: res.append(p)
for c in n.children: dfs(n.children[c], p+c)
dfs(node, prefix.lower()); return res
def ids_with_prefix(self, prefix):
node = self.root
for ch in prefix.lower():
if ch not in node.children: return set()
BLACKBOX Agent
+30M users choose BLACKBOX AI
Signup and get $5 for free
Already subscribed? Connect
Type your task here (@ to add files)...
BLACKBOX Agent Open Website
Ln 12, Col 23 Spaces: 4 UTF-8 CRLF {} Python Python 3.14 (64-bit) BLACKBOX Autocomplete: ON BLACKBOXAI: Open Chat
79°F Sunny
Search
ENG IN
12:38 11-11-2025


File Edit Selection View Go Run
Search
Welcome
ai test.py
BLACKBOX
C: > Users > Naveen > ai test.py > Trie > __init__
class Trie:
def ids_with_prefix(self, prefix):
if ch not in node.children: return set()
node = node.children[ch]
return node.ids
class InvertedIndex:
def __init__(self): self.index = defaultdict(set)
def add(self, pid, text):
for t in text.lower().split():
self.index[t].add(pid)
def query(self, text):
words = text.lower().split()
if not words: return set()
res = self.index.get(words[0], set()).copy()
for w in words[1:]: res &= self.index.get(w, set())
return res
class LRUCache:
def __init__(self, cap=2): self.cap, self.od = cap, OrderedDict()
def get(self, k):
if k not in self.od: return None
v = self.od.pop(k); self.od[k]=v; return v
def put(self, k,v):
if k in self.od: self.od.pop(k)
elif len(self.od)>=self.cap: self.od.popitem(last=False)
self.od[k]=v
def __len__(self): # ✅ Added this line to fix len() issue
return len(self.od)
# ---------- Mock AI Helper ----------
class AIHelper:
def recommend(self, desc):
BLACKBOX Agent
+30M users choose BLACKBOX AI
Signup and get $5 for free
Already subscribed? Connect
Type your task here (@ to add files)...
BLACKBOX Agent Open Website
Ln 12, Col 23 Spaces: 4 UTF-8 CRLF {} Python Python 3.14 (64-bit) BLACKBOX Autocomplete: ON BLACKBOXAI: Open Chat
79°F Sunny
Search
ENG IN
12:39 11-11-2025


File Edit Selection View Go Run
Search
Welcome
ai test.py
BLACKBOX
C: > Users > Naveen > ai test.py > Trie > _init_
class AIHelper:
def recommend(self, desc):
if "autocomplete" in desc: rec.append("Trie")
if "search" in desc: rec.append("InvertedIndex")
if "cache" in desc: rec.append("LRUCache")
return {"AI Suggestion": rec}
# ---------- Demo ----------
PRODUCTS = {
1: "Apple iPhone 14",
2: "Apple AirPods Pro",
3: "Samsung Galaxy S22",
4: "Logitech Mouse",
5: "Apple Watch 8"
}
def build_indices():
trie, idx = Trie(), InvertedIndex()
for pid, name in PRODUCTS.items():
trie.insert(name, pid)
idx.add(pid, name)
return trie, idx
if __name__ == "__main__":
ai = AIHelper()
print(ai.recommend("Provide autocomplete and cache for product search"))
trie, idx = build_indices()
print("\nAutocomplete for 'App':", trie.autocomplete("App"))
print("IDs for prefix 'App':", trie.ids_with_prefix("App"))
print("Query 'Wireless' (none expected):", idx.query("Wireless"))
cache = LRUCache()
cache.put("hot", [1,5]); cache.put("new", [2])
print("Cache after 2 inserts:", len(cache)) # Works now
BLACKBOX Agent
+30M users choose BLACKBOX AI
Signup and get $5 for free
Already subscribed? Connect
Type your task here (@ to add files)...
BLACKBOX Agent Open Website
Ln 12, Col 23 Spaces: 4 UTF-8 CRLF {} Python Python 3.14 (64-bit)
BLACKBOX Autocomplete: ON
BLACKBOXAI: Open Chat
79°F Sunny
Search
ENG IN
12:39 11-11-2025


File Edit Selection View Go Run
Search
Welcome
ai test.py
BLACKBOX
C: > Users > Naveen > ai test.py > Trie > _init_
3: "Samsung Galaxy S22",
4: "Logitech Mouse",
5: "Apple Watch 8"
}
def build_indices():
trie, idx = Trie(), InvertedIndex()
for pid, name in PRODUCTS.items():
trie.insert(name, pid)
idx.add(pid, name)
return trie, idx
if __name__ == "__main__":
ai = AIHelper()
print(ai.recommend("Provide autocomplete and cache for product search"))
trie, idx = build_indices()
print("\nAutocomplete for 'App':", trie.autocomplete("App"))
print("IDs for prefix 'App':", trie.ids_with_prefix("App"))
print("Query 'Wireless' (none expected):", idx.query("Wireless"))
cache = LRUCache()
cache.put("hot", [1,5]); cache.put("new", [2])
print("Cache after 2 inserts:", len(cache)) # Works now
cache.put("latest", [3])
print("Cache after eviction:", len(cache))
print("Cache get 'hot':", cache.get("hot"))
BLACKBOX Agent
+30M users choose BLACKBOX AI
Signup and get $5 for free
Already subscribed? Connect
Type your task here (@ to add files)...
BLACKBOX Agent Open Website
Ln 12, Col 23 Spaces: 4 UTF-8 CRLF {} Python Python 3.14 (64-bit)
BLACKBOX Autocomplete: ON
BLACKBOXAI: Open Chat
79°F Sunny
Search
ENG IN
12:39 11-11-2025

Out put

```
{'AI Suggestion': ['Trie', 'InvertedIndex', 'LRUCache']}

Autocomplete for 'App': ['apple iphone 14', 'apple airpods pro', 'apple watch 8']
IDs for prefix 'App': {1, 2, 5}
Query 'Wireless Bluetooth' (none expected): {6, 7}
Cache after 2 inserts: 2
Cache after eviction: 2
Cache get 'hot': None
PS C:\Users\Naveen>
```