

Got it 👍 You want a **complete Expenses Tracker project** using **KivyMD** (for mobile UI), **Django** (as backend API), and **MySQL** (as database). I'll break it into steps so you can set it up smoothly.

◆ Project Overview

- **Frontend** → KivyMD app (cross-platform, Android/iOS/desktop)
 - **Backend** → Django REST API (handles users, expenses, categories, authentication)
 - **Database** → MySQL (stores data)
-

◆ Step 1: Setup Django + MySQL

1. Install dependencies:
2. `pip install django djangorestframework mysqlclient`
3. Create Django project:
4. `django-admin startproject expenses_backend`
5. `cd expenses_backend`
6. In `settings.py`, configure **MySQL**:
7. `DATABASES = {`
8. `'default': {`
9. `'ENGINE': 'django.db.backends.mysql',`
10. `'NAME': 'expenses_db',`
11. `'USER': 'root',`
12. `'PASSWORD': 'your_password',`
13. `'HOST': 'localhost',`
14. `'PORT': '3306',`
15. `}`
16. `}`
17. Create expenses app:
18. `python manage.py startapp expenses`
19. Define **Expense model** (`expenses/models.py`):
20. `from django.db import models`
21. `from django.contrib.auth.models import User`
- 22.

```
23. class Expense(models.Model):
24.     user = models.ForeignKey(User, on_delete=models.CASCADE)
25.     title = models.CharField(max_length=100)
26.     category = models.CharField(max_length=50)
27.     amount = models.DecimalField(max_digits=10, decimal_places=2)
28.     date = models.DateField(auto_now_add=True)
29.
30.     def __str__(self):
31.         return f"{self.title} - {self.amount}"
32. Create serializers (expenses/serializers.py):
33. from rest_framework import serializers
34. from .models import Expense
35.
36. class ExpenseSerializer(serializers.ModelSerializer):
37.     class Meta:
38.         model = Expense
39.         fields = '__all__'
40. Create views (expenses/views.py):
41. from rest_framework import viewsets, permissions
42. from .models import Expense
43. from .serializers import ExpenseSerializer
44.
45. class ExpenseViewSet(viewsets.ModelViewSet):
46.     queryset = Expense.objects.all()
47.     serializer_class = ExpenseSerializer
48.     permission_classes = [permissions.IsAuthenticated]
49.
50.     def get_queryset(self):
51.         return Expense.objects.filter(user=self.request.user)
52.
53.     def perform_create(self, serializer):
```

```
54.     serializer.save(user=self.request.user)
55. Setup URLs (expenses_backend/urls.py):
56. from django.contrib import admin
57. from django.urls import path, include
58. from rest_framework.routers import DefaultRouter
59. from expenses.views import ExpenseViewSet
60.
61. router = DefaultRouter()
62. router.register(r'expenses', ExpenseViewSet, basename='expense')
63.
64. urlpatterns = [
65.     path('admin/', admin.site.urls),
66.     path('api/', include(router.urls)),
67. ]
68. Run migrations:
69. python manage.py makemigrations
70. python manage.py migrate
71. Create superuser:
72. python manage.py createsuperuser
```

◆ Step 2: Setup KivyMD App (Frontend)

```
1. Install KivyMD:
2. pip install kivy kivyMD requests
3. Example main.py:
4. from kivy.lang import Builder
5. from kivyMD.app import MDApp
6. import requests
7.
8. KV = """
9. BoxLayout:
10.     orientation: "vertical"
```

```

11.
12. MDToolbar:
13.     title: "Expense Tracker"
14.     md_bg_color: app.theme_cls.primary_color
15.     specific_text_color: 1,1,1,1
16.
17. ScrollView:
18.     MDList:
19.         id: expenses_list
20.
21. MDFloatingActionButton:
22.     icon: "plus"
23.     pos: "20dp", "20dp"
24.     md_bg_color: app.theme_cls.primary_color
25.     on_release: app.add_expense()
26. """"
27.
28. class ExpenseApp(MDApp):
29.     API_URL = "http://127.0.0.1:8000/api/expenses/"
30.
31.     def build(self):
32.         self.theme_cls.primary_palette = "Blue"
33.         return Builder.load_string(KV)
34.
35.     def on_start(self):
36.         self.load_expenses()
37.
38.     def load_expenses(self):
39.         response = requests.get(self.API_URL, headers={"Authorization": "Token
your_api_token"})
40.         if response.status_code == 200:

```

```

41.     expenses = response.json()
42.     for exp in expenses:
43.         self.root.ids.expenses_list.add_widget(
44.             self.create_expense_item(exp["title"], exp["amount"], exp["category"])
45.         )
46.
47.     def create_expense_item(self, title, amount, category):
48.         from kivymd.uix.list import OneLineListItem
49.         return OneLineListItem(text=f"{title} - {category}: ₹{amount}")
50.
51.     def add_expense(self):
52.         # Example: Add expense via POST
53.         data = {"title": "Tea", "category": "Food", "amount": "20.00"}
54.         requests.post(self.API_URL, data=data, headers={"Authorization": "Token
your_api_token"})
55.         self.root.ids.expenses_list.clear_widgets()
56.         self.load_expenses()
57.
58. ExpenseApp().run()

```

◆ Step 3: Authentication

- Install token auth:
 - `pip install django-rest-framework-simplejwt`
 - Update Django to use JWT tokens so KivyMD app can login & get token.
-

◆ Step 4: Deployment

- **Backend** → Deploy Django on server (Heroku, Railway, VPS, etc.)
 - **Frontend** → Package KivyMD into APK using **Buildozer**.
-

👉 Do you want me to prepare a **ready-to-run zipped project structure with Django + KivyMD + MySQL** (with authentication included), or just the **code snippets** like above in a PDF?

You said:

it is not installing mysqlclient

ChatGPT said:

That's a very common issue 🛑 when working with **mysqlclient**, especially on Windows. Don't worry — I'll walk you through the fixes and alternatives.

◆ **Why mysqlclient Fails to Install**

- It needs **C compiler & MySQL development libraries** installed.
 - On Windows, you need **Visual Studio Build Tools** and the **MySQL Connector/C**.
-

✅ **Fix 1: Install Precompiled mysqlclient Wheel (Windows only)**

1. Go to Unofficial Python Binaries.
 2. Download the .whl file matching your Python version (e.g., mysqlclient-2.2.0-cp311-cp311-win_amd64.whl for Python 3.11 x64).
 3. Install it:
 4. `pip install path\to\mysqlclient-2.2.0-cp311-cp311-win_amd64.whl`
-

✅ **Fix 2: Use mysql-connector-python Instead (Simpler, pure Python)**

Instead of mysqlclient, you can use:

`pip install mysql-connector-python`

Update **Django settings** (settings.py):

```
DATABASES = {  
    'default': {  
        'ENGINE': 'mysql.connector.django',  
        'NAME': 'expenses_db',  
        'USER': 'root',  
        'PASSWORD': 'your_password',  
        'HOST': 'localhost',  
        'PORT': '3306',  
    }  
}
```

⚠️ Downside: mysql-connector-python is slower than mysqlclient, but works without extra compilation.

✅ **Fix 3: Use PyMySQL (Another Alternative)**

pip install pymysql

Then add this to your Django project's `__init__.py` (inside the project folder where `settings.py` is):

```
import pymysql
```

```
pymysql.install_as_MySQLdb()
```

And in `settings.py` keep:

```
'ENGINE': 'django.db.backends.mysql',
```

👉 Question for you:

Do you want me to **rebuild the expense tracker setup using mysql-connector-python or PyMySQL** (so it installs easily), instead of sticking with mysqlclient?