Getting it running

1. Install python (most new versions will work but i have 3.11 so probably use that)
a. https://www.python.org/downloads/release/python-3110/
b. Make sure you add it to your path
2. Open the finecore-django directory
3. Create a virtual environment. (my venv is in the repo but just ignore it)
a. run : python -m venv myenv
4. Activate virtual environment
a. Windows: myenv/scripts/activate
b. Linux: source myenv/bin/activate
5. Install requirements
a. Run: pip install -r requirements.txt
b. Ensure they all work. If some fail try going into the requirements file and remove the version. Eg change “django==5.1.6” to “django”
6. Go into the finecore_api directory
a. cd finecore-api
7. Start the django app on port 8000
a. Python manage.py runserver 8000

Navigating the code

1. Finecore-api folder
    a. This is the django project folder
    b. It is the entry point for the app and i’ve mostly configured it
    c. urls.py directs to the other routes in the project
    d. settings.py file alters the settings and is important for things like middleware, cors, attaching a DB.
        i. This is set up okay right now so don’t worry about it yet.
2. ninjaAPI folder
    a. This is a django app folder.
    b. You can have multiple
    c. urls.py file directs to routes of the app. - not particularly important now it is set up
    d. Most important files are ninjaAPI.py and models.py

Models.py

- ORM and database stuff
- Copy and paste into chat gpt to understand what each bit does
- Each model is a table in a DB
- If you make any changes, you must run two commands to apply them to the DB
    - python manage.py makemigrations
        - Creates the code to change the DB in the migrations folder
    - Python manage.py migrate

ninjaAPI.py

- Where all endpoints are.

- Import ORM models at the top

```
from .models import UserApiKey, Transaction, UserWallet
```

- Uses library called django-ninja which is a copy of fastapi built on the django framework
    - https://django-ninja.dev/
- Ninja requires you to define the datatypes at each endpoint
    - Schemas
    - Can also be done in the function definition but can get too long
- Example of definition:
    - Takes input in the form of schema “CreateUserID”
    - Checks secret key for security
    - Generates an api key with a function i defined in the utils file
    - Creates the row in the DB using the ORM
        - The model has built in automatic fields to create the datetime and UUID
    - Returns the database row

```
#creates new userid with api key and time created
@api.post("/newuserid")
def create_api_key(request, createuserid: CreateUserID):

    if createuserid.secret_key != SECRET_KEY:
        return {"error": "incorrect secret key"}

    # creates the user uuid, an initial API key and the time
    new_key = generate_api_key()

    api_model = UserApiKey.objects.create(key=new_key, created_at=time_now)
    # will row of data in ApiKey table for this UUID
    return {
        "uuid": str(api_model.uuid),  # Convert UUID to string
        "api_key": api_model.key,
        "created_at": api_model.created_at.isoformat()  # Convert datetime to string
    }
```

Postman call:

POST http://localhost:8000/apis/ninjaAPI/newuserid

```
{
  "secret_key": "jHnHG86273nKgD77012aas125miihbVRGpPjGfRrrgjnmPmmQD0d83"
}
```

POSTMAN

- There is a file called finecore_django.postman_collection.json in the root directory.
- To run endpoints in postman, import this file as a collection into postman.