Task Report

I used django, python and sqllite3 that is connected to django admin to do the task. I implemented some models so that they will be reflected as tables in the database.
The created tables are:

1. Customer table: having id, email, password as attributes
2. Bank table: having id, email, password, total available funds as attributes
3. Provider table: having id, email, password as attributes
4. Loan table: having id, bid, cid, maxAmount, minAmount, interestRate, duration, amount, terms as attributes
5. Loan fund table: having id, pid, bid, maxAmount, minAmount, interestRate, duration, amount as attributes

I put the ids of banks and customers as bid and cid in loans as they can have many loans however the loan will belong to only one customer and one bank. I did the same for the ids of banks and providers as bid and pid in loan funds. I assumed that loan funds and loan are 2 different tables as I assumed that the total fund amount of the bank is collected from all loan funds that are provided, and the loans' amounts are taken from the total fund amount in general not from an explicit loan fund. I still didn't use the interestRate in anything as I think they will be used once the customer returns the amount of the loan to the bank and once the bank returns the funds to the providers.
The created APIs and how they can be tested:

1. For all users:
   1.1. Register to the app: using the url http://127.0.0.1:8000/register/
   When registering, the users are added to the corresponding tables according to their types.
   Here are JSON requests examples to test the API:
   For registering bank:
   ```
   {
      "userType":"bank",
      "email":"B1@gmail.com",
      "password":"1234",
      "totalFundAmount":0
   }
   ```
   For registering customer:
   ```
   {
      "userType":"customer",
      "email":"C1@gmail.com",
      "password":"1234"
   }
   ```

For registering provider:

```
{
   "userType":"provider",
   "email":"P1@gmail.com",
   "password":"1234"
}
```

1.2. Login to the website: using the url http://127.0.0.1:8000/login/
After the users login, their Id and type is stored in session so that we can make only a specific user types access a functionality and we saved also the ids so that we don't ask the users for their ids everytime they attempt to do a functionality.
Here is a JSON request example to test the API:

```
{
   "email":"C1@gmail.com",
   "password":"1234"
}
```

2. For banks:

2.1. Create loans: using the url http://127.0.0.1:8000/createLoan/
Whenever a bank creates a loan it will be stored in the loans table and the id of the customer as long as the details that the customer defines when he/she chooses a loan such as terms and amount will have the value of null until some customer chooses this loan.
Here is a JSON request example to test the API:

```
{
   "minAmount":10000,
   "maxAmount":50000,
   "interestRate":0.1,
   "duration":5
}
```

2.2. Create loan funds: using the url http://127.0.0.1:8000/createLoanFund/
Whenever a bank creates a loan fund it will be stored in the loan funds table and the id of the provider as long as the details that the provider defines when he/she chooses a loan fund such as the amount will have the value of null until some provider chooses this loan fund.

Here is a JSON request example to test the API:

```
{
    "minAmount": 10000.0,
    "maxAmount": 50000.0,
    "interestRate": 0.1,
    "duration": 5
}
```

3. For providers:

    3.1. Choose a loan fund: using the url http://127.0.0.1:8000/providerCreateFund/
    When choosing a loan fund given the loan fund id, the pid and amount of this loan fund change to the new values in the database and the total fund amount in the bank that owns the loan fund will increase.
    Here is a JSON request example to test the API:

    ```
    {
        "lid": 1,
        "amount": 20000
    }
    ```

4. For customers:

    4.1. Choose a loan: using the url http://127.0.0.1:8000/customerChooseLoan/
    When choosing a loan given the loan id, the cid, amount and terms of this loan change to the new values in the database and the total fund amount in the bank that owns the loan will decrease.
    Here is a JSON request example to test the API:

    ```
    {
        "lid": 1,
        "amount": 20000,
        "terms": "term1, term2"
    }
    ```

    4.2. Show their loans: using the url http://127.0.0.1:8000/customerViewLoans/
    Here all of the loans of this customer is shown once we request the url.

All of the above functionalities expect the register and login should be done after the login to function properly. You can also see the entries of each table as an admin from http://127.0.0.1:8000/admin/ after creating an admin using the command ‘python manage.py createsuperuser’ and login with it using the django interface.