# Backend Developer Screening Assignment

**Scalable REST API Structure**

When creating a backend infrastructure, consider the following for structuring files and managing APIs, as well as handling PostgreSQL (PSQL) and NoSQL (Redis) instances that should be shared across all endpoints:

Ensure that object creation occurs only once when the server starts, and share the same object across all endpoints instead of creating database objects repeatedly.

1) Develop a system that provides test data through the /v1/hello and /v2/hello endpoints. Create a template for this sample project to allow for scalability.

2) Make more endpoints
    - /login: This endpoint should perform authentication using PostgreSQL (PSQL), generate a unique UUID, save it as a cache in Redis, and return it to the user as a new authorization header.
    /setName : set update users name (authorisation header required just any sample auth token)
    /getName : get the users name (return any sample json)

Note: Please avoide writing duplicate code again and again, Don’t make DB instance in all files it should be created in main.py and should be shared in the entire app.

Project using: fastapi, uvicorn, gunicorn

**Please utilize the "Main.py" file that was sent to you via email.**

Below is a basic structure provided for reference. Let's see how effectively it can be done.

```
└── API/
    ├── static/
    │   └── assets/
    │       └── img/
    │           └── happy.png
    ├── utils/
    │   ├── helper/
    │   │   └── helpme.py
    │   ├── database/
    │   │   └── psql.py
    │   └── redis/
    │       └── redis.py
    ├── web/
    │   ├── __init__.py
    │   ├── v1/
    │   │   ├── __init__.py
    │   │   └── somefile.py
    │   └── v2/
    │       ├── __init__.py
    │       └── somefile.py
    ├── constants.py
    ├── version.py
    ├── main.py
    ├── gunicorn.py.ini
    ├── .env
    ├── .env.local
    └── .env.prod
```

## Instructions for Candidates

1. Clone this and work on the tasks described above.
2. Please utilize the "Main.py" file that was sent to you via email
3. Organize your code and project structure following the provided reference structure.
4. Ensure that your code is well-documented and include a `readme.md` file with instructions on how to run the project.
5. Test your API endpoints using tools like Postman or `curl` and make sure they work as expected.
6. Once you have completed the tasks, share your code repository or project files with us for review.

Good luck with the assignment! If you have any questions or need clarification on any task, please feel free to reach out.