

# Feedback | Group 9

---

## Milestone 1

### Problem Definition | 20 points

The problem is defined correctly, and the structure is kept. However, I would recommend to concentrate only on **Testing**.

- Broad Area of Interest
- Preliminary Research
  - Current trends
  - Opportunities
- Solution with Methodology
  - Data Collection
  - Analytical Techniques
  - Implementation Plan
- Expected Outcomes
- Evaluation Metrics

Make sure to change the `.docx` file to `pdf`. As soon as you do so, let me know, and I will change the grade to **20**

Grade: 18

### Roadmap | 10 points

The roadmap seems realistic but not user-friendly.

Grade: 10

### Administrative Tasks | 5 points

- Roles are assigned
- Preliminary discussion with me was done
- Slack channel is create
- Github Repo is created

Grade: 5

### Technical Tasks | 5 points

- Proper `.gitignore` file is available; however, Python track wasn't selected
- The Requirments.txt file is available, indicating that `venv` was created
- The first chapter of the Package Development course is done by **everyone**

Grade: 4

## Grade

Final Grade: 39/40

## Milestone 2 | Tasks

### Product and Project Manager | 40 points

1. Name your Python package: register to [pypi](#)
2. Install `mkdocs` package to start with the documentation
3. Database schema: Provide your product database structure (ERD)
4. Transform your project file structure according to the below tree

```
PythonPackageProject/ #github repo
├── yourpackagename/
│   ├── __init__.py
│   ├── submodule1/ #database related
│   │   ├── __init__.py
│   │   └── submodule1_1.py
│   ├── submodule2/ #model related
│   │   ├── __init__.py
│   │   └── submodule1_2.py
│   └── submodule3/ # api related
│       ├── __init__.py
│       └── submodule1_2.py
├── tests/
│   ├── __init__.py
│   ├── test_module1.py
│   └── test_module2.py
├── example.ipynb # showing how it works
├── run.py # in order to run an API
├── docs/ #this folder we need for documentation
├── .gitignore
├── requirments.txt
├── README.md
├── LICENSE
└── setup.py
```

### Data Scientist and Data Analyst | 20 points

1. Simulate the data if you need
2. Try to use the CRUD functionality done by DB Developer
3. Work on modeling part using simple models

```
from yourpackage.submodule2 import modelname
```

### Database Developer | 30 points

1. Create a DB and respective tables suggested by the Product Manager
2. Connect to SQL with Python
3. Push data from flat files to DB
4. Test the code provided [here](#) and complete the missing components
5. Add extra **methods** that you might need throughout the project:
  1. Communicate with PM and API Developer for custom functionality

```
from yourpackage.submodule1 import sqlinteractions
```

## API Developer | 30 points

1. Communicate with DB Developer and PM in order to design the API
2. You can create dummy endpoints in the beginning, then communicate with PM as well
3. The following endpoints must be available:
  1. GET
  2. POST
  3. UPDATE

Check out this [this repo](#).

```
from yourpackage.submodule2 import api
```

---

## Milestone 2 | Feedback

---

As promised M1 grade is updated!

I would recommend to add logger module or put it in utils.py and use instead of **print()**

## DataCamp

Done by everyone.

## Product and Project Manager | 40 points

1. The package is registered in Pypi
2. **mkdocs** package is in the requirements.txt. Also **index.md** is added
3. The schema is provided
4. Done: **good job!**
  1. **\_\_init\_\_.py** files are properly setup
  2. tests are provided
  3. **please remove the date from the package folder. Remember, when the user wants to read csv file using pandas, he/she is not putting it inside of the pandas package directory**

Grade: 40/40

## Data Scientist and Data Analyst | 20 points

- The data was successfully simulated/ingested
- modeling module was initiated and tested properly

Grade 20/20

## Database Developer | 30 points

- DB and schema was successfully implemented
- Connection between SQL and Python is available
- The Data is loaded
- Custom functions are available in `db_interactions.py` file
- I can see three db files 😊
- move tes Grade: 30/30

## API Developer | 30 Points

- `run.py` is working properly
- Requests:
  - POST request is available
  - GET request is available
  - PUT(update) request is not available

Grade: 30/30 Good Job!

M2 Grade: 120/120

---

## Milestone 3 | Tasks

---

### Remaining tasks from M2

- fix `__init__.py` files

## DataCamp

Complete the third chapter.

## Product and Project Manager | 30 points

1. Design the final endpoints:
  - the outputs you need for modeling
  - the outputs you need to analyze the study
2. Communicate the outputs with the team in order to help them create/modify final classes/methods, etc.
  - design query functions according to your needs

- design modeling components according to your needs
3. Create sample documentation using [mkdocs](#). Once you have the final version of a package, you'll update it. For now, push to GitHub the following:
- a selected template
  - index.md page1 and page2 with dummy content (though you are free to provide actual documentation as well)

## Data Scientist and Data Analyst | 30 points

- Create a model based on the Product Manager's requirements (or improve the existing file and ingest the output to DB)
- Insert the outcome into the respective SQL folder. (communicate with the Product Manager and DB developer in case you need extra table and/or functionality)
- Data Analyst must try to:
  - interpret the model
  - create custom visualizations
  - suggest/support Product Manager to make decisions about product's final design

## Database Developer | 30 points

- Based on the new/updated requirements, provide functionality in order to interact with the DB
  - API developer might need customer functionality for the final endpoints
  - Data Scientist/Analyst developer

## API Developer | 30 Points

- make your requests directly from the Database and update based on Product Manager's request
- Note: you can make endpoints to test the data as well `get_something()`.