Feedback | Group 6

Table of Contents

Milestone 1

I renamed the docs folder to documents as you will need the `docs` folder for `documentation` with the `mkdocs` package.

Problem Definition | 20 points

The problem is defined correctly, and the structure is kept!

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

Grade: 20

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

Overall, you did a really great job during the M1. Keep it like that! Final Grade: 39/40

Milestone 2 | Tasks

Fix the problem statement from the first milestone.

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/ #githhub 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

```python
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

Terrific description in the `README.MD`. Make sure to provide your names in the `setup.py` file under the author field, instead of group 6 I would also recommend to keep `logger` module and use it instead of `print()` function

DataCamp

Done by everyone.

Product and Project Manager | 40 points

1. The package isregistered in `Pypi`
2. `mkdocs` package is not in the `requirments.txt`.
3. The schema is provided.
4. Partially done:
    - Note: you need to provide the references in the `__init__.py` files
    - From the project's point of view, there is no difference in how to insert the simulated data into SQL. However, I'd recommend first generating CSV files (let's say in the data folder: out of the package), then inserting them into the DB. From an application point of view, that makes sense, as the potential user of your package is going to provide either a csv/flat file or a db connection string.

Grade: 40/40

Data Scientist and Data Analyst | 20 points

- The data was successfully simulated/ingested
- modeling module was initiated 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

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
- Note: you can make endpoints to test the data as well `get_something().`

Milestone 3 | Feedback

Ramaining tasks from M2

All done!

Datacamp

Done by Everyone!

Product and Project Manager

- Final endpoints are provided
- Sample documentation is provided

Grade: 30/30

Data Scientist

- The "predictive model" is created
- Data Analytics is done

Grade: 30/30

Database Developer

All done!

30/30

API Developer

All done!

30/30

Good job! Grade: 120/120

Milestone 4 | Tasks

1. **Documentation 30 points**
    - Create comprehensive documentation using **MkDocs**.
    - Each module (e.g., API, database, logger, model) should have its own dedicated page within the documentation.
    - The first page should provide a high-level overview detailing the **Problem**, **Solution**, and **Expected Outcomes.**
    - Host the completed documentation on **GitHub Pages.**
2. **README.MD 25 points**
    - The README file is also going to be the first page description in pypi.org. So make sure to make it as informative as possible.
    - mkdocs weblink
    - steps using the package
    - API GET Requests (the links which are showing up in the swagger under the each endpoint)
    - put it in `setup.py` (in order to make it available on pypi)
3. **Requirements and Environments 15 points**
    - Develop at least two requirements.txt files to manage dependencies more effectively.
        - `package_requirement.txt`
        - `docs_requirements.txt`
    - Create two separate virtual environments
    - for the main package (excluding **ipykernel** or **notebook** and other not directly related packages)
    - building the documentation
4. **Repository Management 15 points**
    - Clean up the repository to ensure it contains no extraneous files.
    - Host the main package on PyPI.
5. **Demonstration Notebook: 15 points**
    - Provide an `example.ipynb` file **outside** of the main package.
    - This notebook should demonstrate at least two scenarios where the solution is applied effectively Grade: 30/30

Milestone 4 Feedback

Documentation

- The MkDocs weblink is missing.
- The docstrings are not properly written:
    - in the `sql_instersection.py` the data types of the arguments are missing

Grade: 27/30

README.MD

- Readme is included the `setup.py`
- The MkDocs weblink is provided, however the template wasn't the best 😃
- Readme is well designed

Grade: 25/25

Repository Management

Here everything is fine.

Grade: 15/15

Requirements and Environments

Here everything is fine.

Grade: 15/15

Demonstration Notebook

Perfect!

Grade: 15/15

M4 Grade: 100/100

Demo | 20 points

You need to introduce the product with 10 minutes.

The presentation format:

- **Slide 1:** The Problem
- **Slide 2:** Solution
- **Slide 3:** The problem solving methodology
- **Slide 4-5**: Demo
  - Anything you’d like to show
  - business case scenario 1
  - business case scenario 2

Demo Grade: 20/20

Final Grade

Grade: *399/400*