Feedback | Group 4

Table of Contents

Milestone 1 Tasks

1. Problem Definition (you can learn more about it here)
2. Finalizing roles here
3. Schedule a call/meeting with me and Garo
4. Create a product roadmap and prioritized functionality (items)
5. Create a GitHub repository including `readme.md` and `.gitignore` (for Python) files
6. Create a virtual environment in the above repo and generate `requirements.txt` (ensure `venv` is ignored in git)
   - Create venv: `python -m venv venv`
   - Activate: `source venv/bin/activate`
   - Install: `fastapi`
   - Create `requirements.txt`: `pip freeze > requirements.txt`
   - Deactivate: `deactivate`
7. Push *Problem Definition, GitHub repo setup (readme.md and .gitignore), requirements.txt*
8. Prototype the UI using *Figma* or another similar tool
9. Create a private Slack channel in our Workspace and name it **Group {number}**
10. Install VS Code (also install the Project Manager extension)

Milestone 1 Feedback

Problem Definition | 10 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: 10/10

Roadmap | 10 points

I couldn't find the roadmap related staff.

Grade: 0/10

UI | 10 Points

Perfect

Grade: 5/10

Administrative Tasks | 5 points

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

Grade: 5/5

Technical Tasks | 5 points

- Proper `.gitignore` file is available for `Python`
- The `Requirments.txt` file is available with pre-installed packages, indicating that `venv` was created

Grade: 5/5

Grade

Final Grade: 30/40

Milestone 2 | Tasks

Product and Project Manager | 20 points

1. Install `mkdocs` package to start with the documentation (PSS will be available)
2. **Database schema:** Provide your product database structure (ERD)
3. Transform your project file structure according to the below tree.
4. Add roadmap
5. check all the bellow activities from your team and merge everything

```
PythonPackageProject/ #githhub repo
├── yourapplications/
│   ├── docker-compose.yaml
│   └── .env
│   └── service1/ #postgress
│       ├── .py files # if needed
│       └── Dockerfile # if needed
│    └── service2/ # pgadmin
│       ├── .py files # if needed
│       └── Dockerfile # if needed
|     └── service3/ # etl related
│       ├── .py files
│       └── requirments.txt
│       └── Dockerfile # if needed
|── example.ipynb # showing how it works
|── docs/ #this folder we need for documentation
|── .gitignore
├── README.md
├── LICENSE
```

Data Scientist and Data Analyst | 20 points

1. Create a new `git branch` and name it `ds`
2. Simulate the data if you need
3. Try to use the CRUD functionality done by DB Developer
4. Work on modeling part using simple models, conduct extra research
5. Push your works to respective branch
6. Create pull request for the Product Manager

Database Developer | 30 points

1. Create a new `git branch` and name it `db`
2. Create a DB and respective tables suggested by the Product Manager
3. Connect to SQL with Python
4. Push data from flat files to DB
5. Add extra `methods` that you might need throughout the project
6. Push your works to respective branch
7. Create pull request for the Product Manager

API Developer | 30 points

1. Create a new `git branch` and name it `back`
2. Create a new service and name it `back`
3. Communicate with the DB Developer and PM in order to design the API
4. You can create dummy endpoints in the beginning (PSS will be available)
5. The following endpoints must be available:
    1. GET
    2. POST
    3. PUT
    4. DELETE
6. Push your works to respective branch
7. Create pull request for the Product Manager

Front End Developer | 20

1. Create a new `git branch` and name it `front`
2. Create a container/service and name it `front`
3. Communicate with the PM in order to create the skeleton of the website.
4. Push your works to respective branch
5. Create pull request for the Product Manager

Milestone 2 | Feedback

Product and Project Manager | 20 Points

1. `MkDocs` is installed, and dummy documentation is present.
2. The file structure is **mostly correct**.
    1. What does it mean `yourapplications` ? You have named it already PythonPackageProject 😃
    2. There must be only one `docker-compose.yaml` file which should orchestrate the containers
3. The ERD seems **mostly correct**; however, the results table where the **p-values** should be stored is missing.
4. `docker-compose.yaml` is not complete
5. Merging has been done properly.

Grade: 10/20

Database Developer | 30 Points

From a database development perspective, everything has been done properly.

Grade: 30/30

Data Scientist and Data Analyst | 20 Points

Good job! Preliminary research was conducted 😊

It was expected to connect to the DB **directly** and fetch the data in order to prepere the final table and build a model.

Grade: 10/20

API Developer | 30 Points

From an API development perspective, everything has been done properly, except keeping extra docker-compose file

Grade: 25/30

Front End Developer | 20 Points

The skeleton of the website was created.

- No requirments.txt file
- No container/service

Grade: 10/20

Final Grade: 85/120

Milestone 3 | Tasks

Product and Project Manager | 40 Points

1. From the previous milestone, you must have:
   - Refactored the project file structure with services isolated.
   - Fix `docker-compose.yaml` and keep only on file accross services
2. Design all the endpoints required and share them with the Backend and Frontend teams:
   - Ensure the endpoints cover the functionality needed for the web application to work.
3. Support the Frontend Engineer in finalizing the UI (no need to connect with FastAPI within this milestone; this will be done in Milestone 4):
   - Research Streamlit components/elements.
   - Suggest appropriate elements.

Note: No need to reinvent, just stick with built-in Streamlit functionality.

Database Developer | 10 Points

1. Update the `database` tables based on the new `ERD` from the previous milestone.
2. Finalize the documentation using proper docstrings.
3. Push the final output to the respective **branch**.

Data Scientist | 20 Points

1. Build the final model.
2. Prepare the final output.
3. Push the output to the db
4. Push codes to the respective **branch**.

API Developer | 30 Points

1. Create **all** the required endpoints (coordinate with the Product Manager).
2. Create schemas using Pydantic:
   - **Response Models**: Define the structure of the return values.
   - **Documentation**: Add docstrings to all your endpoints.
3. Push the final output to the respective **branch**.

Frontend Developer | 20 Points

1. Build the final layouts of the app.
2. Communicate with the Product Manager for requirements.
3. Use Streamlit's built-in elements/components.
4. No need to connect with the endpoints; this will be required for the final version.
5. Push the final output to the respective **branch**.

Milestone 3 | Feedback

Milestone 3 | Feedback

Product/Project Manager

- Tasks from the previous milestone done
- The ednpoints seams complete
- UI is pretty good!

Grade: 40/40

Database Developer

- Documentation and table design are clear and well-executed.
- Functionality for database interaction is effectively implemented.

Grade: 10/10

Data Scientist

- Outputs are comprehensive and ready for integration.

Grade: 20/20

API Developer

- Comprehensive endpoint creation and good use of `Pydantic` schemas for response models.

Good job!

Grade: 30/30

Frontend Developer

- Good job!

Grade: 20/20

Grade: 120/120

Milestone 4 | Tasks

Final touches (30)

- Connect **Back End** with the **Front End**
- use `steamlit containers` to make your outputs more consistant

Documentation (30 points)

- Create comprehensive documentation using **MkDocs**.
- Each service (e.g., api, app, database, model) should have its own dedicated page with 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**.

README.md (25 points)

- The `README.md` must be as informative as possible. Include:
  - Weblinks:
    - **MkDocs**
    - **pgadmin**
    - **streamlit**
    - **swagger**
  - Steps for running the product (check my demo repo).
  - Swagger screenshot(s)
  - UI screenshot(s)

Repository Management (15 points)

- Clean up the repository to ensure it contains no extraneous files.