Feedback | Group 4

Table of Contents

Milestone 1 | Tasks

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 items.

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

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 tree provided in the project scope.
4. Add a roadmap.
5. Check and merge all activities from your team.

Data Scientist and Data Analyst | 20 Points

1. Create a new `git branch` and name it `ds`.
2. Simulate the data if needed.
3. Use the CRUD functionality provided by the DB Developer.
4. Work on modeling using simple models and conduct additional research.
5. Push your work to the respective branch.
6. Create a pull request for the Product Manager.

Database Developer | 30 Points

1. Create a new `git branch` and name it `db`.
2. Create a database and respective tables suggested by the Product Manager.
3. Connect to SQL with Python.
4. Push data from flat files to the database.
5. Add extra `methods` that might be needed throughout the project.
6. Push your work to the respective branch.
7. Create a 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. Coordinate with the DB Developer and PM to design the API.
4. Create dummy endpoints initially.
5. Required endpoints:
    - GET
    - POST
    - PUT
    - DELETE
6. Push your work to the respective branch.
7. Create a pull request for the Product Manager.

Front End Developer | 20 Points

1. Create a new `git branch` and name it `front`.

2. Create a container/service and name it `front`.
3. Collaborate with the PM to create the skeleton of the website.
4. Push your work to the respective branch.
5. Create a 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**, but:
    - The folder name `yourapplications` seems redundant.
    - Only one `docker-compose.yaml` file should orchestrate the containers.
3. The ERD seems **mostly correct**, but the results table where **p-values** should be stored is missing.
4. `docker-compose.yaml` is incomplete.
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 😊 . However, it was expected to connect to the DB **directly** to fetch the data.

Grade: 10/20

API Developer | 30 Points

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

Grade: 25/30

Front End Developer | 20 Points

The skeleton of the website was created.

- Missing `requirments.txt` file.
- Missing container/service.

Grade: 10/20

Grade

Final Grade: 85/120

Milestone 3 | Tasks

Tasks

Product and Project Manager | 40 Points

1. Refactor the project file structure to isolate services.
2. Fix the `docker-compose.yaml` file to ensure a single file across all services.
3. Design all required endpoints and share them with the Backend and Frontend teams.

Database Developer | 10 Points

1. Update database tables based on the new ERD from the previous milestone.
2. Finalize documentation using proper docstrings.

Data Scientist | 20 Points

1. Build and finalize the model.
2. Push the final output to the database.

API Developer | 30 Points

1. Create all required endpoints, using Pydantic schemas.
2. Ensure the endpoints are fully documented.

Frontend Developer | 20 Points

1. Build and finalize the layouts for the application.
2. Collaborate with the PM to ensure the UI design meets expectations.

Milestone 3 | Feedback

Feedback

- Tasks are completed as expected.
- Excellent collaboration across teams.

Grade

Final Grade: 120/120

Milestone 4 Tasks

Tasks

Final Touches | 30 Points

1. Connect the Backend with the Frontend.
2. Use Streamlit containers for consistent outputs.

Documentation | 30 Points

1. Create comprehensive documentation using MkDocs.
2. Host the documentation on GitHub Pages.

README.md | 25 Points

1. Update the README with all required details:
    - Screenshots for Swagger and UI.
    - Instructions to run the application.

Repository Management | 15 Points

1. Clean up the repository, removing unnecessary files.

Milestone 4 Feedback

Feedback

- Documentation and README updates were **exemplary**.

Grade

Final Grade: 100/100

Demo

Excellent!

Final Grade: 20/20

Final Grade

Final Grade: 355/400