Development Plan
Software Engineering

Team 4, EcoOptimizers

Nivetha Kuruparan
Sevhena Walker
Tanveer Brar
Mya Hussain
Ayushi Amin

Table 1: Revision History

Date	**Developer(s)**	**Change**
September 18th, 2024	All	Created first draft of document
September 23rd, 2024	All	Finalized document

This document outlines the development plan for improving the energy efficiency of engineered software through refactoring. Included are the details on intellectual property, team roles, workflow structure, and project scheduling. Additionally, the plan covers expected technologies, coding standards, and proof of concept demonstrations, providing a clear road map for the project's progression.

1 Confidential Information?

No confidential information to protect.

2 IP to Protect

The software and associated documentation files for this project are protected by copyright, can be accessed at this link(referred to as "License"). The License does not grant any party the rights to modify, merge, publish, distribute, sub license, or sell the software without explicit permission .

Unauthorized use, modification, or distribution of the software is prohibited and may result in legal action.

Permission is granted on a case-by-case basis, non-transferable and non-exclusive. No rights to exploit the software commercially or otherwise are granted without prior written consent from the copyright holders.

3 Copyright License

See LICENSE file in root of repository, can be accessed at this link.

4 Team Meeting Plan

The team will meet multiple times a week, once during Monday tutorial time and throughout the week as issues or concerns arise. The meetings will be conducted either online through a Teams meeting or in-person on campus. The team will hold an official meeting with the industry advisor once a week yet the time has not been decided. The meeting with the advisor will be online on Teams for the first three weeks, followed by in-person meetings later on. Meetings itself will be structured as follows:

1. Each member will take turns giving a short recap of work they have accomplished throughout the week.
2. Members will voice any concerns or issues they may be facing.
3. Team will form a discussion and make decisions for the project.

4. Any/all questions will be documented for the next meeting with the advisor.

The team will go ahead and use Issues on GitHub to add anything they may want to talk about in the next meeting as the week progresses in order to have some form of agenda for the next meeting.

5 Team Communication Plan

- **Issues**: GitHub
- **Meetings**: Microsoft Teams
- **Meetings (with advisor)**: Discord Group Chat
- **Informal Project Discussion**: WhatsApp, Discord Server
- **Formal Project Discussion**: Discord Server

6 Team Member Roles

As a team, we will all function as developers, sharing responsibilities for creating issues, coding, testing, and documentation in the early stages. Specific roles will be defined as the project evolves, allowing for flexibility and collaboration.

During our scheduled meetings (with the supervisor, within the team, etc.), we will follow an Agile Scrum structure, incorporating additional roles such as Scrum Master and Scribe. These roles will rotate weekly, with the Scrum Master responsible for organizing and leading the meetings, and the Scribe tasked with documenting key details. This approach ensures active participation and shared responsibility amongst team members.

We have chosen not to designate a team leader, as we all possess similar skills and knowledge. Instead, we aim to work collaboratively to resolve any challenges that arise.

7 Workflow Plan

The repository will contain two main persistent branches (branching off main) throughout the project: **dev** and **documentation**. These branches, that we will henceforth call "epic" branches, will be used for technical software development and documentation changes respectively.

The average workflow for the project will proceed as follows:

1. Pull changes from the appropriate **epic** branch
2. Create working branch from **epic** branch with the format [main contributor name]/[descriptive related to topic of changes]
3. Create sub-working-branch from previous branch if necessary
4. Commit changes with descriptive name
5. Create **unit tests** for said changes
6. Create pull request to merge changes from sub/working branch into working/epic branch
7. Wait for all tests run with **GitHub Actions** to pass
8. Wait for two approvals from teammates other than the one who created the Pull request
9. Merge changes into target branch (working or epic)

GitHub Issues will be used to track and manage bugs, feature requests, and development tasks. Team members can report issues, propose enhancements, and assign tasks to specific individuals. Each issue will be labelled (e.g., bug, enhancement, team-meeting) for easy categorisation, linked to relevant **milestones**, and tracked through **GitHub Project** boards. Comments and code references will be exploited to allow for collaboration and discussion on potential solutions. Issues will also be integrated with pull requests, so they'll automatically close once fixes are merged.

For situations where a certain type of issue is projected to be created at a steady frequency, templates will be created to facilitate their creation.

Milestones will be used to organize commits and pull requests for major deliverables. Once all working branches associated to the milestone have been merged into the appropriate epic branch, the team will go through the relevant prepared checklist to make sure that all requirements have been met. Once this is done, the epic branch will be merged into main as one pull request along with the checklist.

The **CI/CD** pipeline will be implemented via GitHub to improve automated testing as well as facilitate the feedback loop for the machine learning models used by the library. More detailed information will be added later.

8 Project Decomposition and Scheduling

The project is hosted on GitHub under the organization of Sustainable Systems and Methods (SSM) Lab and can be accessed at this link.

The team currently has one GitHub Project setup to serve as a visual tool not only to track deadlines for all tasks but also for accountability for assigned tasks. The project can include cards for:

- Issues for course related deliverables
- Lecture and Meeting logs
- Issues for project specific tasks(such as building a certain component).

Schedule

While the development of our system will be broken down into smaller features, the overall project will follow the following major deadlines.

Milestone	**Due Date**
Problem Statement, Proof of Concept, and Development Plan	September 24th, 2024
Requirements Document (Revision 0)	October 9th, 2024
Hazard Analysis (Revision 0)	October 23rd, 2024
Verification & Validation Plan (Revision 0)	November 1st, 2024
Proof of Concept	November 11th-22nd, 2024
Design Document (Revision 0)	January 15th, 2025
Project Demo (Revision 0)	February 3rd-14th, 2025
Final Demonstration	March 17th-30th, 2025
Final Documentation	April 2nd, 2025
Capstone EXPO	TBD

9 Proof of Concept Demonstration Plan

For context, our POC will consist of roughly the following steps:

1. Determine the code smells we want to address for energy saving.
   - List Comprehension in an `any` or `all` statement, Member Ignoring Method, Long Paramter List, Unused Imports, Long Message Chain, Unused Class Attributes and Variables
2. Determine the detectability of a specific code smell
   - These code smells are detectable using Pylint or manual AST parsing.
3. Determine the appropriate refactorings for a particular detected smell that results in decreased energy consumption.
   - We will use codecarbon to measure the emissions of a python code file. This step will involve various phases of trial and error as it is not a 1-1 trivial solution. There could be various refactorings possible

for a given situation that all result in different energy consumption levels. We want our tool to choose the most optimal refactoring possible. For our POC this can exist as an algorithm. For our final project we can attempt to implement a neural network to choose between refactorings. There are also prebuilt free to use libraries we can implement to perform simple refactorings.

4. Once we determine preset algorithms mapping detected smells to their appropriate refactorings, we want to then make those changes in the code, measure the energy consumption and test it against the original code ensuring it is less.

5. The code must then ensure that the original code functionallity is preserved. If it is not a different refactoring is required.

    - This can be done by testing the original test suite for the code against the new one. This original test suite can be a required argument for the user.

The following is a list of primary risks and how potential results from the POC could mitigate them.

1. The refactorings we propose may not reduce the code's energy consumption.

    - If this occurs the team will have to re-analyze the results, revisit refactoring strategies and conduct additional testing to ensure that the proper refactorings are occuring. Code exists in two forms, fully optimized, and not fully optimized. If it is the later, using a new strategy should work provided our team has the expertise to find the correct refactoring. If it is the former the correct use case of our tool is simply the result that no new optimal refactorings could be found. This is a valid result for some use cases.

2. Energy is decreased but functionality of the code is modified.

    - If this occurs then the refactoring we applied was incorrect for the given code. A new refactoring should occur. Finding valid refactorings can be implemented iteratively, it is okay for the code to get a couple wrong so long as the final answer is correct. Requiring the user to submit a test suite for the original code can ensure that the code is not modified beyond its purpose. We can also add error handling that lets the user be in charge of the refactorings by making them suggestions instead of absolutes, that allows for the software engineer to have more control over what their final code looks like.

Other smaller risks for our tool include:

- Integration challenges: Challenges adding our tool to people's CI/CD pipelines. What should that look like? How can we make it as accessible as possible?
  - Potential answer: If we make a plugin we should try and make it compatible with a widely used IDE.
- Software developers not wanting to adopt energy saving into their code bases. How can we make it as user friendly as possible?
  - Potential answer: Ease of use and the success of the tool will be the primary factors in it's use. For corectness we can ensure we have thorough testing to ensure the tool works as promised. For user experiences we can conduct small trials or focus groups seeing what people prefer.
- Code performance issues: A lot of code choices are tradeoffs. Are there situations in which we would be decreasing runtime or performance to achieve a better energy output? If so what consequences does this have?
  - Potential answer: Allowing the tool to act as a suggestion rather than a hard refactor and giving the developer the choice to undo changes could help the engineer have more control over the specifications that matter to them the most.

10 Expected Technology

This section contains a compilation of technologies that we expect to use for our project following an initial analysis. The choice of technology might change for one or more of the components if needed along the course of the project.

10.1 Languages

A diverse set of languages is being used to build different components of the project. Here is a detailed breakdown:

1. **Refactoring Library: Python**
   The library is being built in Python for the following reasons:
   - The language has well-established libraries like 'Rope' for refactoring and 'PyJoules' for energy analysis, which can be leveraged for the MVP.
   - The language is widely used in both research and industry, making the library adoptable by a broader community of developers.
2. **VS Plugin: TypeScript**
   Given the VS Code architecture (based on web technologies), the plugin needs to be built using JavaScript or TypeScript. The team has decided to pursue development in TypeScript given our experience in the language.

10.2 External Libraries/Customization

To build an MVP, the team is relying on multiple external libraries to handle the technical detail intensive work. Below is a list of external libraries that we are using along with their purpose. Note that their usage will be replaced with custom implementation once an MVP has been created and tested.

1. **Code Energy Calculation: PyJoules**
   PyJoules is a well-documented library that calculates the energy consumed on executing a given Python code base. Since the library calculates energy consumption at the hardware level, the calculations are very precise. This makes it an ideal choice for analyzing the energy impact of code refactorings.

2. **Inefficient Code Pattern Detection: PySmells**
   After reviewing PySmells documentation, it can be deduced that a majority of code smells can be detected and resolved using the library.

3. **Refactoring: Rope**
   Rope includes a wide range of refactorings, such as renaming, function extraction, and code restructuring. Some of these will be useful when refactoring for inefficient code patterns. Rope is an ideal choice as it simplifies the process of identifying and replacing these code patterns with more optimized versions.

10.3 DevOps Integration Framework

Since the goal is to integrate with GitHub, GitHub Actions is the natural choice for the CI/CD part of the project. By using GitHub Actions, the library can be configured to be automatically triggered whenever new code is pushed into a branch. This action can be integrated with test suites provided by the user to validate that functional behavior is retained past refactoring.

10.4 Machine Learning Model

The reinforcement learning model needs to be trained on specific refactoring techniques and developer feedback. Since the problem space is unique to this project, the model will be trained by the feedback received through the application of the library.

10.5 Machine Learning Framework

For the learning model, the team has decided to utilize the PyTorch framework for the following two reasons:

1. The framework is written in Python, so it's easier to integrate with the Python-based library.

2. The framework has a less steep learning curve compared to TensorFlow since it's syntactically similar to PyTorch.

10.6 Continuous Integration Plan

For automated testing and integration within the GitHub DevOps pipeline, the team has decided to utilize GitHub Actions since they have a wide variety of support articles for initial setup.

10.7 Additional Tools (for Development Support)

To ensure that the plugin, library, and reinforcement learning model run consistently across different environments, Docker containers will be used. Since these can package the entire application with its dependencies, there will be consistency in development, testing, and deployment environments.

10.8 Linter Tools

To ensure best practices, all team members will be using the following combination of linter tools for Python code:

1. **Mypy:** This tool enforces type annotations, which helps to detect type-related errors before runtime. Since Python is dynamically typed, Mypy enhances code reliability by making types explicit.

2. **Flake8:** This tool checks code for style guide enforcement (PEP 8) and can catch potential errors early in development, improving code quality and maintainability.

10.9 Code Smell Tools

To identify code smells during development, the team will include PySmells in the project. In addition to the fact that the tool covers a majority of Python code smells, the team was enthusiastic about the fact that the library itself will be used during implementation of the MVP.

10.10 Unit Testing

To ensure systematic testing of our growing codebase, the project will include unit tests written in `pytest`. We have chosen this framework as it helps to write scalable test cases, supports parameterized tests, and has easy integration with `coverage.py`, which is our code coverage measuring tool.

10.11 Code Coverage

`Coverage.py` has been chosen to measure the code coverage. This will be a useful tool to identify areas of the code base that need additional test coverage.

10.12 Performance Measurement

Performance measuring tools have been chosen to target common Python-specific code problems. Here is a list of tools the team plans to integrate into the project:

1. **PyTrace:** This tool will be used to trace execution and will be particularly useful in instances that might be prone to race conditions.
2. **cProfile:** This tool will be used to retrieve information on the time spent on each function during execution. This will be particularly useful to identify any performance bottlenecks.

11 Coding Standard

The team will include PEP 8 as the coding standard in the project. This standard is widely accepted across the Python community, therefore there is tons of resources available on initially setting it up. The team will have a useful tool to automate style enforcement and ensure consistency across the code base.

Appendix — Reflection

Mya Hussain Reflection

1. *Why is it important to create a development plan prior to starting the project?*

   Development plans act as good starting points by establishing clear objectives, recourse requirements, timelines and by assigning accountability within a project. This improves overall communication and understanding between members by solidifying the intent and requirements of a project. From my experience in industry, developers who don't have a good grasp of what they're developing and why they're developing it typically deliver products that either fail to meet the primary goal or prove useless to the user. In real life, you develop products for people with various backgrounds. People of different disciplines speak in different technical tongues. Development plans are critical in syncing many people from various disciplines towards a common goal. They can also further can divide the roles and responsibilities between the developers and later aid in making development decisions that push progress towards the primary goals defined in the document.

2. *What disagreements did your group have in this deliverable, if any, and how did you resolve them?*

   I would be lying if I were to say that my team had disagreements during the first week of working on the project. The easy answer to this question is to make up a disagreement of low significance and say we worked through it using open democratic discussion and compromise. The truth is that it is too early in the project for us to be disagreeing with each other. We're in our "honeymoon stage" where all members are excited to work on the project and are optimistic about the possible results we could achieve. Let us be in love during week one. We have many more weeks left to disagree.

Ayushi Amin Reflection

1. *Why is it important to create a development plan prior to starting the project?*

   A development plan is important because it serves as a clear road map for the project and ensures that all team members understand the the project scope, objectives and timeline. A clear plan helps to identify potential risks and areas of concern/challenges that can be caught at an early stage so a solution can be crafted early on. This also helps in resource allocation by ensuring that the necessary tools and budget are available when needed.

2. *What disagreements did your group have in this deliverable, if any, and how did you resolve them?*

   To be quite frank, our team did not have any disagreements during the first couple weeks of working together.

Sevhena Walker Reflection

1. *Why is it important to create a development plan prior to starting the project?*

   When starting a project with a large scope, it is easy to get lost in all the features you want to implement in the project. By establishing an action plan from the start, you allow yourself and your team to develop concrete goals and needs for your system. Taking a step back to analyze how the components of your system are expected to interact with each other and defining the intended end result is necessary.

   Without this planning stage, you are essentially going in blind with only a vague understanding and what you need to do. There is no way to organize task between team members efficiently either since specific components are sparsely detailed or non-existent. The project will be in a constant state of "debugging" you could say, constantly trying to figure out how this feature will work with the next and the next and so on. This would be like trying to build a house while only ever looking at the next couple meters in front of you.

2. *What disagreements did your group have in this deliverable, if any, and how did you resolve them?*

   I honestly cannot say that we had any disagreements as yet. Our team tried our best to discuss what our goals for the project were and everything was well communicated so that everyone was on the same page. We are at the beginning stages of a project that will implement technology is mostly new to us. We are excited, but also somewhat ignorant to some details which makes it hard to "disagree".

Tanveer Brar Reflection

1. *Why is it important to create a development plan prior to starting the project?*

   After having created one, I believe a development plan is important as it can be useful for the project in multiple ways. For one, it is a written record of the team's expected practices related to roles, communication, etc.(which ensures that everyone is on the same page with these practices).This can also be used by instructors as the source of truth if there are any conflicts in the project team.

   Secondly, with the pre-written format/prompts we were compelled to think about some logistics that we would otherwise have ignored(such as communication plan and coding standards). In short, it set a great starting note for the project so we were aware of our assumptions and were also compelled to think about less obvious aspects of project planning.

2. *What disagreements did your group have in this deliverable, if any, and how did you resolve them?*

   We haven't had any disagreements in our group for this deliverable. There were multiple instances where each of us had different ideas on how to approach a particular issue. After weighing in the pros and cons of each idea together, we ended up picking one of them unanimously. This was possible since we value each other's thought process and take a non partial approach when deliberating.

Nivetha Kuruparan Reflection

1. *Why is it important to create a development plan prior to starting the project?*

   A development plan is essential as it sets a structured foundation for the project, helping to clarify the team's objectives, roles, and processes before work begins. This ensures that everyone shares a common understanding of the project's scope and expectations, reducing the likelihood of miscommunication or misaligned efforts later on.

   A development plan is also important because it helps unify team members who have diverse technical skills, ensuring that everyone is working towards the same end goal. By laying out the project scope, objectives, and expectations clearly from the start, the plan serves as a guide that helps team members understand how their contributions fit into the bigger picture. This is important when skills vary across the team, as it prevents disagreements and promotes collaboration.

2. *What disagreements did your group have in this deliverable, if any, and how did you resolve them?*

   Our group did not encounter any disagreements during this deliverable. We had open communication from the start, which allowed us to share our ideas and expectations early on. If there were minor differences in opinions, we addressed them by discussing the pros and cons of each suggestion and collaboratively deciding what would work best for the project.

Group Answers

- *In your opinion, what are the advantages and disadvantages of using CI/CD?*

  CI/CD, or Continuous Integration/Continuous Deployment, is a process where continuous integration involves automatically integrating code changes from multiple contributors into a shared repository for every commit, and continuous deployment ensures the code base is deployable at any time. Key practices include automated testing and frequent commits, leading to better code quality as bugs and conflicts are caught early and fixed near their date of creation. Another advantage is faster time to market, as code is integrated and deployed more quickly, allowing teams to deliver features and fixes to users faster. This also improves collaboration through feedback, as features pushed to shared repositories can be reviewed by the team.

Disadvantages include the initial setup complexity of the CI/CD pipeline, which can be time-consuming and challenging for those new to the process. This requires ongoing maintenance and overhead to continuously commit, test, and merge code, diverting time from feature development. Additionally, the setup of automated tests and deployment workflows can be complex and requires knowledge of the process. Risks include over-reliance on automation, which could introduce security flaws. In other terms:
Advantages:

1. Faster development since changes would be pushed constantly as testing for new/fixed features would be successful
2. Shorter delivery times
3. Improved code quality (automated testing)
4. Easier change monitoring and rollback since there are five members and multiple branches and changes based on the member
5. Increased efficiency and productivity

Disadvantages

1. Requires strong discipline and commitment from the entire team and may be a learning curve for the team
2. May be challenging to implement for small teams since there is only five members
3. Initial setup and configuration can be complex and time-consuming since the team will be setting this up from scratch

Appendix — Team Charter

External Goals

Our team's primary goal is to learn something new and valuable that can be applied in the workforce, ensuring that this project enhances our practical skills. Additionally, we aim to create a project that we can confidently discuss in interviews, demonstrating our ability to work on real-world problems. While we focus on personal and professional growth, we also aim for an A+ as a nice-to-have achievement.

Attendance

Expectations

Our team expects full commitment to scheduled meetings, with everyone arriving on time and staying for the entire duration. If a team member cannot attend, they are expected to notify the group in advance and provide a valid reason, as well as organize an alternative meeting time, if all team members need to be present. Missing meetings without prior notice or frequently arriving late will be addressed by the team to prevent disruptions.

Acceptable Excuse

An acceptable excuse for missing a meeting or a deadline includes unforeseen emergencies, personal illness, family matters, or other significant personal obligations, as long as the team is informed in advance. Unacceptable excuses include vague or last-minute reasons such as simply forgetting or having conflicting non-essential plans, as these could impact the team's progress.

In Case of Emergency

In the event of an emergency that prevents a team member from attending a meeting or completing their assigned work for a deliverable, the team member must inform the team as soon as possible through the team's designated communication channel (either on WhatsApp or Discord). This will allow for adjustments to be made, such as redistributing tasks or rescheduling the meeting if necessary. For deliverables, if the emergency impacts a deadline, the team member should notify both the team and the professor promptly to ensure that any necessary arrangements are made without affecting the team's progress/grades.

Accountability and Teamwork

Quality

Our team has the following expectations regarding the quality of preparation for meetings and the deliverables brought to the team:

- **Meeting Preparation**:
  - Team members are expected to arrive at meetings fully prepared, having reviewed relevant materials and completed their assigned tasks in advance.
  - Each member should come ready to discuss their progress, share insights, and address any challenges they are facing.
  - Members should ensure that their updates are clear and concise, allowing meetings to stay focused and productive.
- **Deliverables Quality**:
  - All deliverables must meet the team's agreed-upon standards, demonstrating a high level of accuracy, thoroughness, and attention to detail.
  - Each deliverable should be carefully reviewed by each member before submission to avoid any errors or incomplete work.
  - Deliverables must align with the project's requirements and deadlines, ensuring they are both functional and meet the expected quality criteria.
- **Accountability and Feedback**:
  - Team members are responsible for completing their work to a high standard, communicating any issues early if they need assistance or more time.
  - Feedback on deliverables should be welcomed by all members, and revisions should be made promptly to improve the overall quality of the team's output.

By maintaining these expectations, our team will ensure that meetings are efficient and that all deliverables reflect a professional and high-quality standard.

Attitude

Our team has established the following **expectations** for team members' contributions, interactions, and cooperation to ensure a productive and respectful working environment:

- **Respectful Communication**: All team members are expected to listen to each other's ideas and provide constructive feedback. Communication should remain respectful, even in cases of disagreement.
- **Open Collaboration**: Each member is encouraged to share their ideas openly. Everyone should be willing to collaborate and help each other achieve team goals.

- **Accountability**: Team members are responsible for completing their tasks by the agreed-upon deadlines. If a member is struggling, they are expected to ask for help or communicate early.
- **Positive Attitude**: Maintaining a positive attitude, especially in challenging moments, is essential for team morale. Each member should encourage and support their teammates.
- **Commitment to Quality**: Every team member is expected to contribute to the project with their best effort, ensuring that the final product reflects high standards of quality.

We adopt the following **code of conduct** to guide behavior and interaction among team members:

- **Inclusivity**: Our team values diversity and is committed to creating an inclusive environment where everyone feels welcome and valued, regardless of background, experience, or opinion.
- **Professionalism**: Members will engage professionally, refraining from any inappropriate or offensive language or behavior. This applies to both in-person and online interactions.
- **Collaboration and Feedback**: We encourage constructive feedback and expect team members to accept and provide feedback in a way that helps everyone grow. Criticism should be focused on the work, not the individual.
- **No Tolerance for Harassment**: Harassment of any kind will not be tolerated. Any issues will be reported immediately and addressed in a structured manner.

To manage conflicts or disagreements that may arise during the project, we have a **conflict resolution plan** in place:

1. **Address the Issue Directly**: If a conflict arises, the involved members should first try to resolve the issue directly through a respectful discussion.
2. **Mediation by a Neutral Member**: If the conflict cannot be resolved, the team will appoint a neutral team member to act as a mediator to facilitate a discussion and find common ground.
3. **Escalation to Instructor/TA**: In the event that the conflict cannot be resolved within the team, the issue will be escalated to the instructor or TA for further guidance and resolution.
4. **Follow-Up and Monitoring**: After resolving the conflict, the team will continue to monitor the situation to ensure that the issue does not resurface and that team dynamics remain positive.

By adhering to these expectations, the code of conduct, and our conflict resolution plan, we aim to maintain a positive, collaborative, and respectful team environment.

Stay on Track

To keep our team on track, we will implement the following methods:

1. **Regular Check-ins and Progress Updates**: We will hold *weekly meetings* where each member will provide an update on their tasks and progress and any concerns or troubles they faced. These updates will help us identify issues early and adjust accordingly to stay on schedule.
2. **Performance Metrics**: We will track the following key metrics:
    - *Attendance* at meetings and check-ins will be documented through Issues on GitHub.
    - *Commits to the repository*, ensuring steady contributions.
    - *Task completion rates*, ensuring deadlines are met.
3. **Rewards for High Performers**: To encourage good performance, we will recognize and celebrate team members who meet or exceed expectations. Informal rewards may include public recognition during meetings or assigning leadership roles in future tasks.
4. **Managing Under performance**: If a team member's performance is below expectations:
    - We will start with a *team conversation* to understand any obstacles and offer support.
    - If under performance continues, consequences may include *more tasks* for milestone or in severe cases, a meeting with the TA or instructor.
5. **Consequences for Not Contributing**: If a team member does not contribute their fair share:
    - They may be assigned additional *tasks* to balance the workload.
    - In serious cases, the issue will be brought up to the TA or instructor.
6. **Incentives for Meeting Targets Early**: Members who consistently meet or exceed their targets will be rewarded with more desirable tasks as per their wants, such as leadership roles in key project components, helping to build their leadership experience. They will get first pick on tasks for the next team milestone.

Team Building

For team building events, the team has decided to have bi-weekly hangouts to bond and build relationships. The hangouts can attending on-campus events together, getting food or bubble tea on/off campus and more.

Decision Making

In our group, our primary way of making decisions will be through consensus. We believe that it is important to include everyone in the decision-making process so it can lead to better outcomes and strong group work. In certain situations where consensus cannot be reached, the group will take a vote and each member will have equal say and the decision will be based on the majority rule. We will make sure all group members had a chance to voice their opinions before making the final decision through consensus or a vote.

To Handle disagreements: The team will address each disagreement directly and respectfully.

1. Allow all team members to express their concerns and opinions without interruption, ensuring everyone feels heard.
2. Keep the focus of the discussion on the topic at hand rather than personal feelings.
3. When necessary, we may appoint a neutral party to facilitate the discussion and help guide it to a resolution.
4. If a resolution is not found or the disagreement persists after the resolution is found, we will aim to revisit our project goals and objectives to ensure that our decisions align with our common purpose.

By following these strategies, we aim to maintain a collaborative and positive team environment while effectively managing decisions and conflicts.