Development Plan
Software Engineering

Team 6, 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. It includes 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 roadmap for the project's progression.

1 Confidential Information?

[State whether your project has confidential information from industry, or not. If there is confidential information, point to the agreement you have in place. —SS]

[For most teams this section will just state that there is no confidential information to protect. —SS]

2 IP to Protect

[State whether there is IP to protect. If there is, point to the agreement. All students who are working on a project that requires an IP agreement are also required to sign the "Intellectual Property Guide Acknowledgement." —SS]

3 Copyright License

[What copyright license is your team adopting. Point to the license in your repo. —SS]

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 3 weeks, followed by in-person meetings later on. Meetings itself will be strutured 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
Informal Project Discussion: WhatsApp

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 2 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

GitHub Projects will be used as way to get a quick look at all tasks and deadlines.

Link: https://github.com/users/Sevhena/projects/2

[How will the project be scheduled? This is the big picture schedule, not details. You will need to reproduce information that is in the course outline for deadlines. —SS]

9 Proof of Concept Demonstration Plan

What is the main risk, or risks, for the success of your project? What will you demonstrate during your proof of concept demonstration to convince yourself that you will be able to overcome this risk?

10 Expected Technology

[What programming language or languages do you expect to use? What external libraries? What frameworks? What technologies. Are there major components of the implementation that you expect you will implement, despite the existence of libraries that provide the required functionality. For projects with machine learning, will you use pre-trained models, or be training your own model? —SS]

[The implementation decisions can, and likely will, change over the course of the project. The initial documentation should be written in an abstract way; it should be agnostic of the implementation choices, unless the implementation choices are project constraints. However, recording our initial thoughts on implementation helps understand the challenge level and feasibility of a project. It may also help with early identification of areas where project members will need to augment their training. —SS]

Topics to discuss include the following:

- Specific programming language
- Specific libraries
- Pre-trained models
- Specific linter tool (if appropriate)
- Specific unit testing framework
- Investigation of code coverage measuring tools
- Specific plans for Continuous Integration (CI), or an explanation that CI is not being done
- Specific performance measuring tools (like Valgrind), if appropriate
- Tools you will likely be using?

[git, GitHub and GitHub projects should be part of your technology. —SS]

11 Coding Standard

[What coding standard will you adopt? —SS]

Appendix — Reflection

[Not required for CAS 741 —SS]

The purpose of reflection questions is to give you a chance to assess your own learning and that of your group as a whole, and to find ways to improve in the future. Reflection is an important part of the learning process. Reflection is also an essential component of a successful software development process.

Reflections are most interesting and useful when they're honest, even if the stories they tell are imperfect. You will be marked based on your depth of thought and analysis, and not based on the content of the reflections themselves. Thus, for full marks we encourage you to answer openly and honestly and to avoid simply writing "what you think the evaluator wants to hear."

Please answer the following questions. Some questions can be answered on the team level, but where appropriate, each team member should write their own response:

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

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

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

Sevhena Walker

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 establish an action plan from the start, you allow yourself and your team to develop concrete goals and needs for your system. It is necessary to take a step back and analyze how exactly the components of your system are expected to interact with each other and what the end result is suppose to be.

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. It would be like trying to build a house while only ever looking at the next couple meters in front of you.

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

In CI/CD, there is a constant cycle of commiting code to a code base, and running automated tests on that piece of code before integrating it into the system. This constant and frequent cycle helps detect problems early leading toa more robust system. Furthermore, stress is taken away from the programmer when it comes to building and deploying software as it is all automated.

The flip side to this is, of course, that the developer must initially set up the CI/CD environment. All the automated tests must first be written, and they must be written *well*. The same goes for the build and deployment workflows. This setup can be complex and lengthy and most of all, it requires the developer to know how to do it. Furthermore, it must be *maintained*. Other risks include a serious over-reliance in automation which could potentially introduce security flaws that weren't thought of before.

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".

Appendix — Team Charter

[borrows from University of Portland Team Charter —SS]

External Goals

[What are your team's external goals for this project? These are not the goals related to the functionality or quality fo the project. These are the goals on what the team wishes to achieve with the project. Potential goals are to win a prize at the Capstone EXPO, or to have something to talk about in interviews, or to get an A+, etc. —SS]

Attendance

Expectations

[What are your team's expectations regarding meeting attendance (being on time, leaving early, missing meetings, etc.)? —SS]

Acceptable Excuse

[What constitutes an acceptable excuse for missing a meeting or a deadline? What types of excuses will not be considered acceptable? —SS]

In Case of Emergency

[What process will team members follow if they have an emergency and cannot attend a team meeting or complete their individual work promised for a team deliverable? —SS]

Accountability and Teamwork

Quality

[What are your team's expectations regarding the quality of team members' preparation for team meetings and the quality of the deliverables that members bring to the team? —SS]

Attitude

[What are your team's expectations regarding team members' ideas, interactions with the team, cooperation, attitudes, and anything else regarding team member contributions? Do you want to introduce a code of conduct? Do you want a conflict resolution plan? Can adopt existing codes of conduct. —SS]

Stay on Track

[What methods will be used to keep the team on track? How will your team ensure that members contribute as expected to the team and that the team performs as expected? How will your team reward members who do well and manage members whose performance is below expectations? What are the consequences for someone not contributing their fair share? —SS]

[You may wish to use the project management metrics collected for the TA and instructor for this. —SS]

[You can set target metrics for attendance, commits, etc. What are the consequences if someone doesn't hit their targets? Do they need to bring the coffee to the next team meeting? Does the team need to make an appointment with their TA, or the instructor? Are there incentives for reaching targets early? —SS]

Team Building

[How will you build team cohesion (fun time, group rituals, etc.)? —SS]

Decision Making

[How will you make decisions in your group? Consensus? Vote? How will you handle disagreements? —SS]