Development Plan
Software Engineering

Team 21, Alkalytics
Sumanya Gulati
Kate Min
Jennifer Ye
Jason Tran

Table 1: Revision History

Date	**Developer(s)**	**Change**
09-20-2024	Kate Min	Added Team Meeting Plan
09-21-2024	Kate Min	Added Team Communication Plan and Roles
09-22-2024	Kate Min	Added Team Charter, License Information, and Introduction
09-23-2024	Jason Tran	Added Workflow Plan, Project Decomposition and Scheduling, Expected Technology and Coding Standard
09-23-2024	Sumanya Gulati	Added Proof of Concept Demonstration Plan
09-24-2024	Sumanya Gulati	Added Appendix - Reflection
10-28-2024	Sumanya Gulati	Updated PoC Demo Plan
12-29-2024	Kate Min	Updated Team Charter

This document outlines Team Alkalytics' project development plan.

Administrative details such as copyright and Intellectual Property (IP) agreements, the team's meeting schedule, communication methods, and expected roles are thoroughly covered.

This is followed by details about the team's planned workflow, project deliverables, scheduling, a Proof of Concept (POC) demonstration plan, and an expected technology stack for the project and relevant coding standards.

The appendices include a reflection on the writing process of this document as well as a team charter that outlines a few ground rules and expectations for the team.

1 Confidential Information

There is no confidential information to protect with respect to this project.

2 IP to Protect

A formal IP agreement is not required for this project. The supervisors, as the clients, retain the right to use any code developed as part of this project at no cost. Each team member, hereafter referred to as the developer, retains ownership of their individual contributions and is free to use the code independently. However, the raw data provided to the developers is strictly for use within this project and must not be used outside of it. This agreement only applies to this team and does not extend to contributions made by external third parties.

3 Copyright License

The project will be adopting the standard MIT license, provided here.

4 Team Meeting Plan

This section provides detail on the team's internal and external meeting plans.

4.1 Weekly Meetings

The team will have weekly check-ins on Fridays from 3:30PM to 4:30PM, either in-person on campus or virtually, depending on member's availabilities. Virtual meetings will be hosted on Microsoft Teams. Additional meetings will be scheduled as needed.

The current meeting chair should open a GitHub issue prior to a meeting. During a meeting, the chair will go over each item on a prepared agenda and all members participate in the discussion. The notetaker will record meeting minutes and post them as a comment under the GitHub issue after the meeting, as well as a summary of task assignments.

4.2 Supervisor Meetings

The team will have biweekly or monthly meetings with the primary supervisor, Dr. Charles de Lannoy, depending on his availability. The team will use his Outlook calendar to arrange appropriate meeting dates. These meetings will be held either in his office or virtually on Microsoft Teams.

Weekly meetings with the secondary supervisor, Bassel Abdelkader will be on Tuesdays from 3:30PM to 4:30PM, as he works closely with the data that must be integrated into the project. These meetings will be virtual due to his schedule.

5 Team Communication Plan

The team uses an Instagram group chat for regular communication and a Microsoft Teams group chat for drafting emails and sharing resources.

Communication with supervisors will take place through email or scheduled meetings. The team liaison will be responsible for sending the emails and should Carbon Copy (CC) all team members.

All project-related communication will be done via GitHub issues. Each issue will track a specific task and assign team members. Every pull request should link to its corresponding issue(s), and feedback is expected from all team members when a pull request is created.

6 Team Member Roles

The team plans to rotate through the following roles between every stage, defined in Table 2:

- **Meeting chair**: The meeting chair should prepare an agenda for each meeting and open an issue for the meeting. The issue should contain an attendance tracker, the agenda, and other topics that may need to be discussed.
- **Notetaker**: The notetaker will record meeting minutes during a meeting and post it as a comment under the meeting issue, along with a summary of task assignments.

- **Primary reviewer for pull requests**: The primary reviewer will give the official "approval" for pull requests. All members should review pull requests and provide feedback, but the primary reviewer will make the final decision to ensure a smooth workflow.

Once implementation begins, all members are expected to contribute as developers. However, the team anticipates appointing technical leads based on the stage of development and the members' skillsets, described below:

- Sumanya has experience in data analytics and working with large datasets, and will also serve as the main point of communication between the team and the project's stakeholders.
- Kate has experience in web development and programming in Python, as well as proficiency in several Python libraries that may need to be used.
- Jason has experience in MongoDB and machine learning, as well as proficiency in full-stack development.
- Jennifer has experience working with large datasets and employing data visualization tools like PowerBI, along with a strong background in web development.

7 Workflow Plan

During each milestone, tasks will be delegated to the developers from the project board. To start developing, they will first create dedicated branches for their work and will develop independently, first by pulling changes from the main branch and then following the project's coding standards. Once their work is complete, they will submit a pull request for review, where it will be tested, reviewed by peers, and merged upon approval. This workflow ensures that development progresses smoothly, with ownership, and consistent quality.

7.1 Development with Git

Developers will adhere to the following convention and process.

7.1.1 Branches & Commits

Each project milestone will be managed using a single overviewing pre-production branch. Developers will do `git checkout -B "branch-name"` off this branch to create their respective dedicated work branch to make commits, ensuring that progress is isolated and well-documented for each task.

- **Branch Naming Convention**: `branchType/branchDesc`
  - Types:

        * `documentation`: new documentation
        * `frontend-feat`: new high-level frontend implementation
        * `backend-feat`: new high-level backend implementation
        * `improvement`: an upgrade to an existing feature
        * `bugfix`: a solution to an error
    - Example: `documentation/hazard-analysis`

- **Commit Naming Convention**: `commitType(directory): desc`
    - Types:
        * `feat`: new feature added
        * `fix`: a solution to an error
        * `imp`: an upgrade to an existing feature
        * `doc`: any documentation changes
    - Example: `fix(src): fix plot flickering during rerender`

7.1.2 Merging Strategy

Before a developer merges their individual branch into the milestone branch via a pull request, all commits within an overviewing directory will be squashed into a single commit. Creating a clean, consolidated history for the developer's branch. However, when merging the milestone (pre-production) branch into the main release, individual commits will be preserved to maintain ownership of changes. Throughout the project, developers will use rebase and merge to keep a clean Git history, resolving conflicts locally before merging.

- **Pull Requests**: Is a request to merge code changes from one branch into another. Each pull request requires the approval of at least two reviewers before merging.
    - **Code Review**: Reviewers must check for code quality, adherence to the coding standard, and functional correctness before approving a pull request. Any issues or discussions must be contained within the pull request for tracking.

7.1.3 Continuous Integration (CI)

The team use automated tests and linting for both frontend, backend, and documentation (LaTeX). These will run on each pull request upon creation and must pass before requesting review from other developers.

7.2 Issue Management

Bugs and complications are inevitable during the development process, so it is best practice to address them in a standardized manner. Upon discovering an issue, developers are expected to open an `issue` on GitHub Projects and follow the following procedure:

7.2.1 Template

Create an `issue` on the project board with a title, description, labels, links, and any relevant information. Furthermore, categorize the issue to the respective milestone and assign it accordingly.

- **Milestone**: Each stage is associated with a milestone. Issues will be added to the respective milestones based on the scope.
- **Assignee**: The developer responsible for addressing the issue.
- **Title**: A short, high-level description of the issue.
- **Description**: A detailed description of the issue.
- **Labels**: Categorize using label(s) (e.g., `bug`, `feature`, `enhancement`, and etc). Priority labels will indicate urgency and can be used at the team's discretion.
- **Links**: Link related issues if relevant to enable tracking.

8 Project Decomposition and Scheduling

The project is organized around key milestones with overviewing stages (refer to Table 2) in which the team will rotate roles and responsibilities to allow dynamic and fluid progression. This will constantly use the team's best strength and the rotation of the roles of team member will be at the team's discretion.

Stage	Milestone	Deadline
Stage 1	Problem Statement, POC Plan, Development Plan Requirements Document Revision 0	Sept 24 Oct 9
Stage 2	Hazard Analysis V&V Plan Revision 0	Oct 23 Nov 1
Stage 3	POC Demonstration	Nov 11 - 22
Stage 4	Design Document Revision 0	Jan 15
Stage 5	Revision 0 Demonstration	Feb 3 - 14
Stage 6	V&V Report Revision 0	Mar 7
Stage 7	Final Demonstration (Revision 1) EXPO Demonstration Final Documentation (Revision 1)	Mar 24 - 30 Apr TBD Apr 2

Table 2: Project Decomposition and Deadlines

Code development will start in Stage 2 once foundational requirements are concrete and will continue until the final demonstration.

8.1 GitHub Projects Usage

The project board will follow a Kanban style with columns for `To-Do`, `In Progress`, `Stuck`, `Under Review`, and `Done`. Tasks are divided into milestones and assigned to respective team members. Each task or issue will be represented as a card on the board, moving through the columns as work progresses.

9 Proof of Concept Demonstration Plan

Currently, all the data collected from the various experiments is downloaded from the apparatus in the form of Comma-Separated Values (CSV) files. These files are then manually copied onto an existing Excel template where faulty data is deleted and the rest of the data points are labelled, filtered and then analysed. As the number of experiments and the data collected from each experiment increase in size, this model of storing and analysing data has become unsustainable.

Based on the goals outlined in section 2 of the Problem Statement and Goals document, the following key risks have been identified that must be mitigated in the early development stage to demonstrate the feasibility of this project:

- **Data Migration**: As described in section 2.1, it is critical to ensure that all existing data is migrated to a scalable and extendable database without losing integrity, labelling or structuring.
- **Performance in Querying**: As described in section 2.2, it must be ensured that the database can efficiently handle queries to avoid bottlenecks that could affect system usability.

- **Parameter Comparability**: To accomplish the analysis of the consolidated datapoints, it is essential that the database can facilitate complex inter-parameter comparisons, as defined in section 2.3.
- **Basic Visualization**: To verify the accuracy of the data through graphs, all the parameters must be plotted with respect to time and an initial version of the graphs must be presented.

Since the remaining goals (2.4 to 2.7) are built upon the assumotion of a functioning database that allows efficient querying of the data, it must be demonstrated through the Proof of Concept (PoC) that the aforementioned risks can be mitigated and that the project can be completed on schedule.

The success of the PoC Demonstration and thus, the feasibility of the project will be determined based on the following criteria:

- Developing and implementing a migration algorithm for transferring CSV data files to the database.
- Ensuring that 100% of the existing data has been migrated without loss or error to the new database.
- Demonstrating that all existing inter-parameter comparisons in the Excel templates have been replicated in the database.

10 Expected Technology

Table 3 shows the expected technology for the development of this project.

Technology	**Purpose**
Visual Studio	Integrated development environment for developing code
GitHub	For hosting repositories, collaboration, and project tracking
Git	Version control system for managing changes and branches
TypeScript	Typed superset of JavaScript used for building front-end
Python	General-purpose programming language for backend
MongoDB	NoSQL database for managing and storing datasets

Table 3: Expected Technology Overview

The following, Table 4, shows potential but expected technology to be used and are subject to change according to the team's discretion at any stage of the development.

Technology	Purpose
React	JavaScript library for building dynamic user interfaces
Apollo Client	GraphQL client for managing queries and local state
Material-UI	Component library for creating user interfaces
GraphQL	Query language for APIs
Graphene	Python library for implementing GraphQL APIs
PyMongo	Python library for interacting with MongoDB
Pandas	Data analysis and manipulation library, used for CSV optimization
ESLint	JavaScript linting tool for enforcing coding standards
Flake8	Python linting tool for ensuring code standards

Table 4: Potential Technology Overview

11 Coding Standard

The team will adhere to the coding standard in Table 5 and the appropriate linter will be in use.

Category	Details
Frontend	Adhere to StandardJS for TypeScript coding standard.
Backend	Adhere to PEP 8 for Python coding standards.

Table 5: Coding Standards

Appendix — Reflection

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

   It is important to create a development plan prior to starting the project because it forces the team to have detailed conversations about the various aspects of the project.

   Although our team spent a considerable amount of time discussing our experiences and the technologies we thought would work the best for this project (taking into account the preferences of our supervisors), creating a detailed development plan compelled us to decide on not just the technologies but also the QA factors that we had not considered discussing such as branch and commit naming conventions, coding standards and more.

   Having a detailed conversation with our supervisors while coming up with a PoC Demonstration Plan allowed us to communicate our goals more precisely and set clear expectations in terms of timelines and first steps.

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

   So far, our team has only implemented CI for the documentation parts of the project. This means that all the checks created in GitHub Actions currently only relate to LaTeX syntax checks, pdf compilation and other such conflicts. We plan on incorporating linters and other checks in our CI process over the next couple of weeks.

   The initial process of figuring out how CI can be used for our project and setting it up was painstakingly arduous. The learning curve that came with adopting the CI checks seemed comepletely unnecessary at times especially because the checks take on average 10-12 minutes to complete.

   With that said, however, we do forsee the advantages of using CI. We have already run into a couple of issues with naming conventions of the commits and branches so having a linter in place that can flag style violations and auto-fix formatting issues to make the codebase consistent would be very useful.

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

   Our group did not face any conflicts or disputes. Most of our disagreements were related to writing styles (using third person as opposed to first person or avoiding repetition of words) and naming conventions (following

the naming convention for all the branch names and the commit messages).

To be fair, it is an exaggeration to classify these issues as 'disagreements' because for the naming convention (following the naming convention for all the branch names and the commit messages) issue, it was mostly the novelty of these naming conventions that caused confusion. Once Jason (who set up CI and suggested the convention style) explained how to do it in detail, we did not run into the issue again.

As for the incompatible writing styles - our team decided to review all the PRs over a 2-hour call where the writing style and formatting (indenting versus using newlines) discrepancies were brought up. All the team members were receptive to the feedback and worked together to incorporate the suggestions. In instances where we had to make decisions related to formatting, we held an informal vote and chose to proceed with the decided formatting style for all our documents to avoid confusion and maintain consistency.

Appendix — Team Charter [1]

External Goals

By the end of Capstone, our team wishes to deliver a project that is impactful and useful beyond graduation. We seek the personal satisfaction that comes from knowing the product could contribute to improvements in climate change. By achieving these goals, we thereby anticipate and hope for an excellent grade in the Capstone course.

Attendance

This section lays out ground rules and expectations regarding team member attendance.

Expectations

All members are expected to attend all weekly check-in meetings and meetings with supervisors to the best of their abilities and maintain punctuality. Full attendance at every Capstone lecture will not be required, as long as at least one member from the team attends. All forms of attendance (lectures, internal team meetings, supervisor meetings) will be logged through GitHub Issues and members are expected to maintain a 90% attendance rate throughout the course of the project.

If a meeting is held in person, team members should strive to attend in person when possible. If unable to attend in person, they should notify the team at least 4 hours in advance and join virtually. If a member is unable to attend a meeting entirely, they must inform the team of their absence at least 8 hours prior to the meeting (different expectations apply for emergencies).

Acceptable Excuse

It will not be acceptable for a member to miss a meeting or a deadline without prior communication. Members may miss up to three meetings per term with prior notice and a valid excuse, such as sickness, family responsibilities, religious observances, or previously arranged commitments that cannot be rescheduled. Missing more than three meetings without sufficient notice or an acceptable excuse will prompt an internal discussion about expectations and responsibilities. There will be no acceptable excuses for missing a deadline unless in the event of an emergency. The guidelines for such a situation are outlined in the following section.

In Case of Emergency

If a team member cannot meet a deadline or finish their work due to an emergency, they must communicate this as early as possible. The rest of the team

will discuss potential accommodations and/or redistribute tasks if necessary to ensure that all tasks are completed by the deadline. In the event of work redistribution, the team member who was unable to finish their work may be expected to take on additional tasks in the next phase to ensure a fair workload.

Accountability and Teamwork

This section covers expectations for the team's collaboration efforts.

Quality

All members should prepare for weekly check-ins and meetings by doing the following:

(a) Review the agenda prepared by the meeting chair in advance, and

(b) Prepare a short progress report and any questions/concerns that were not already mentioned in the agenda.

Each member is expected to put in consistent effort and deliver high-quality work that meets or exceeds the expectations described in the applicable rubric. Code should be well-structured, well-documented, and adhere to the coding standards outlined in Section 11 of this document.

Attitude

All members are encouraged to share ideas, ask for feedback/help, and collaborate harmoniously. Feedback should always be clear, concise, and respectful. We do not necessitate a Code of Conduct, but if issues begin to arise over the course of the project, we may consider standardizing one as part of a conflict resolution plan.

Stay on Track

Attendance and task assignments will be tracked using GitHub and GitHub Projects. We maintain a Kanban board to track attendance and tasks for deliverables as GitHub Issues, and other metrics will be collected by the methods provided in the projMngmnt folder of the project repository (e.g., performance metrics).

Evaluating contributions and performance solely on the number of commits or tickets closed may not be an accurate representation of each member's efforts. Therefore, a combination of additional factors—such as task complexity, code quality, and documentation quality—will be used to gauge each team member's contributions.

If uneven contributions, a decline in work quality, or other concerns become apparent, the team will take the following course of action:

1. Initial Discussion: The issue should be addressed with the team member through an internal conversation (during a team meeting or in the team's group chat). The goal is to understand the root of the problem, gather every member's perspective, and offer solutions for improvement.

2. Plan for Improvement: Clear expectations and/or suggestions will be decided with the team member to address the issue. These may include, but are not limited to, taking on additional responsibilities, adjusting workflows, or improving communication. Progress will be reviewed during the team's regular check-in meetings to ensure the issue is being properly addressed.

3. Teaching Assistant (TA) Involvement: If there is no improvement by the team's next major deadline after setting these expectations, the issue will be escalated to our assigned TA for further intervention.

4. Instructor Involvement: If there is still no improvement after the TA intervention, the issue will be escalated to the course instructor, in which the team member may be subject to disciplinary action, such as a grade adjustment.

Team Building

We plan on building team cohesion through our interactions outside of the Capstone project. As we all have similar class schedules, we have numerous opportunities to interact and work with each other in environments that differ from the Capstone project.

Decision Making

Decisions are to be made through a majority vote and should only be made after the team has had a thorough discussion. If there is a tie, we may approach our supervisors for their input if it is relevant for them.

References

[1] "Senior Capstone Design Team Charter 2018" [Online], Shiley School of Engineering, University of Portland, Oregon, 2018. Available: https://engineering.up.edu/industry_partnerships/files/team-charter.pdf