Development Plan
Software Engineering

Team #18, Gouda Engineers
Aidan Goodyer
Jeremy Orr
Leo Vugert
Nathan Perry
Tim Pokanai

Table 1: Revision History

Date	**Developer(s)**	**Change**
17 September 2025	Jeremy	Added 2, 3, 5, 6, 7
Date2	Name(s)	Description of changes
...	...	...

[Put your introductory blurb here. Often the blurb is a brief roadmap of what is contained in the report. —SS]

[Additional information on the development plan can be found in the lecture slides. —SS]

1 Confidential Information?

All data that will be used in any matter over the duration of this project is already publicly available on the Federated Research Data Repository (FRDR) thus no confidential information will be contained in this project.

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

2 IP to Protect

There is no IP to protect for the project. The data provided is openly avaiable to use. Likewise, our code will be open source. [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

The software that we will use will be MIT. The License is linked titled LICENSE and is at the root directory of our repository. [What copyright license is your team adopting. Point to the license in your repo. —SS]

4 Team Meeting Plan

Meetings will occur every Monday from 9:30 a.m. to 10:30 a.m. and every Friday from 12:30 p.m. to 1:30 p.m. Meetings will be in person in MDCL 2246 or virtual as needed. If extra meetings are needed, they will be scheduled on an ad-hoc basis with priority given to pre-established capstone timeslots.

Our group will plan to meet with the project supervisors once per week. This is simply a guideline and ad-hoc meetings will be added as needed. Additionally, all project documents will be forwarded to the supervisors to keep them in the loop and receive any feedback they want to provide.

Tim Pokanai is the group leader and thus will be responsible for leading team meetings. An agenda with the necessary discussion items will be collaboratively created ahead of time (likely through text).

A kanban board has been created on github for aiding in communication and work splitting. All action items for each deliverable will be added to the kanban board so that team members can claim sections they are working on.

5 Team Communication Plan

Our team will primarily communicate using Microsoft Teams and text messaging for quick, day-to-day coordination. For interactions with the professors associated with our project, we will schedule weekly or ad hoc meetings as needed. Email will be our main channel for formal communication with faculty.

To manage and divide work efficiently, we are utilizing GitHub Issues and a Kanban board. GitHub Issues allow us to assign tasks, track progress, and facilitate transparent communication among team members. The Kanban board complements this by providing a clear visual representation of task status and workflow. This will support project planning and execution.

[Issues on GitHub should be part of your communication plan. —SS]

6 Team Member Roles

Roles

Tim - Team Leader Oversees members, helps assign tasks, and ensures deadlines are met.

Nathan - Administrator Manages documentation, project logistics, and resources.

Leo - Meeting Chair Organizes and leads team meetings, sets agendas, and keeps discussions on track.

Jeremy / Aidan - Software Specalists Assist with development tasks such as CI/CD setup, code reviews, general project support, and note taking.

[You should identify the types of roles you anticipate, like notetaker, leader, meeting chair, reviewer. Assigning specific people to those roles is not necessary at this stage. In a student team the role of the individuals will likely change throughout the year. —SS]

7 Workflow Plan

- How will you be using git, including branches, pull requests, etc.?

  The team will use Git for version control. Branches will follow the convention: `<type-of-work>/<issue-name>`, where the types of work are:

  - documentation
  - feature
  - bugfix

Issue name will follow the name of the issue you are working on.

For pull requests, GitHub Actions will compile the code and run checks. A reviewer will always be required. Once a branch is merged, it will be deleted.

- How will you be managing issues, including template issues, issue classification, etc.?

  We will manage tasks and bugs using GitHub Issues. To ensure consistency and clarity, we will use issue templates for common types of issues such as feature requests, bug reports, and documentation tasks. Issues will be classified using labels (e.g., "bug", "enhancement", "in progress", "needs review") to track progress and priority. Each issue will be assigned to a team member and linked to relevant pull requests when applicable. This structured approach will help us maintain transparency, accountability, and effective collaboration throughout the development process.

- Use of CI/CD

  We will implement Githb actions for CI/CD. This will include automated checks to ensure that the code compiles correctly, passes all tests, and adheres to project standards before being merged into the main branch. By integrating CI/CD, we aim to catch issues early, maintain code quality, and streamline the deployment process.

8 Project Decomposition and Scheduling

- How will you be using GitHub projects?

  We will create GitHub issues for each task, and maintain a Kanban board for issues related to the current deliverable. The Kanban board will have 4–5 stages. For now, we will include stages: To Do, In Progress, Peer Review, and Completed. The main use of Github projects is to have good and clear project management.

- Include a link to your GitHub project

  OCD-Rat-Infrastructure on GitHub

[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?

The main risk arises when dealing with the integration and management of the public OCD rat data set with our application. The data set is quite large and heterogeneous, with approximately 20,000 trials worth of data contaning video recordings, trajectory data, and detailed metadata corresponding to the trials. Thus, the main risk stems from our system architecture and database design. If our overall architecture proves to be inadequate, we risk having poor performance in querying and loading data, as well as incorrect and inconsistent results in our natural language querying and when integrating related complex data types.

Our proof of concept will address these risks by demonstrating that data from a representative subset of the dataset, containing roughly 1000 trials, can be queried and visualized in a manner that is both efficient and user-friendly. We will specifically demonstrate that trial data can be filtered based on metadata corresponding to experimental variables, videos can be synchronized with their corresponding trajectories, all of which can be done through a simple Minimal Viable Product (MVP) user interface. Implementing these features on a small scale will demonstrate evidence that our design will perform and be consistent at scale, and most importantly will lay the foundations to upscale to the full data set while supporting natural language querying.

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]

A core module of the project is a web interface for researchers to query and interact with data from past experiments. The team anticipates to use some combination of modern web tools, including HTML, JavaScript, CSS, and a web framework such as React to facilitate the development of this module.

Another core component of the system is the database infrastructure needed to support the querying and storage of the experiment data. The team anticipates using a relational database, such as PostgreSQL or MySQL, to store and manage the data. One or many APIs may be needed to define an interface for interacting with the tools. The team plans to use a framework such as Java Spring Boot to implement these services.

Further goals include the implementation of data analysis and visualization tools. The team anticipates using Python for this purpose, as this is the lowest common denominator tool for many researchers. Another stakeholder request is the capability for natural language querying of data attributes over the trial datasets. The team anticipates using some combination of pre-trained language models like GPT-5 or equivalent to facilitate this task.

Testing is another important aspect of the project. As test frameworks are language specific, the team will select an appropriate test framework for the

language used. For frontend JavaScript, the team may use Jest. For backend in Java/Python, the team may use JUnit/PyTest in conjunction with other code quality and code coverage tools like SonarQube.

For CI/CD purposes, the team plans to use GitHub Actions to define workflows to build and test the code automatically on each commit and pull request.

[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]

Team members will follow the Google Style Guides for the respective programming languages used in the project. This will help ensure consistencty and define a single source of truth for language conventions. Linters will be used to enforce the standards defined in the style guides. All team members will be expected to run the linters locally before pushing code to the repository, as well as configure their linter to follow the defined ruleset.

Regarding software testing, the team will enforce a minimum of 80% code coverage for all new code added to the repository. This will be measured using a covererage tool integrated into the CI/CD pipeline. Pull requests will require passing tests and meeting the code coverage threshold before being merged into the main branch.

In addition to the above, all team members will be required to write clear documentation for new functions, classes, and modules, as well as clear commit messages and pull request descriptions. This must include the purpose and scope of the change, as well as an outline of the testing plan.

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?

Appendix — Team Charter

[borrows from University of Portland Team Charter —SS]

External Goals

1. To have an impressive project to put on a resume and to show off to potential employers. A tangible creation that showcases our abilities and expertise in the software engineering realm.

2. To create something that has genuine utility in the real world and that is used by people in the real world because it brings value to their lives. In other words, to create a legitimate engineering project.

3. To hone our skills in the software realm. From backend to frontend, databases to natural language processing, this project should make us feel like experts in our domain and provide the confidence that we can create a software project from end to end.

4. To acheive a good mark in the capstone course.

Attendance

Expectations

All members are expected to attend all scheduled meetings and be reasonably on time (5 or so minutes late). Leaving meetings early is acceptable as long as all planned action items have already been covered and their presence is no longer necessary. As university students, everyone is very busy and missing meetings from time to time is both understandable and acceptable, given that a sound reason is provided and given ahead of time. See below for acceptable excuses.

Acceptable Excuse

Acceptable excuses include any conflicts related to one's academic or professional career. Meetings for other classes, job interviews or needing to study for a midterm are examples of acceptable excuses, given that reasonable effort was made to prevent or reschedule conflicts (e.g. A team member neglecting to study for a midterm and then needing to skip a meeting to cram is not a reasonable effort). Personal emergencies are of course also acceptable excuses. Examples of unacceptable excuses are those which are both preventable and related to a subject that should not take priority over the capstone project. For example, a team member being too hungover despite knowing of the scheduled meeting or staying up too late playing video games.

In Case of Emergency

In the case of an emergency, members will not be expected to attend meetings and can be caught up after the fact once they become available. If the emergency results in them being unable to complete their individual work for a deliverable, the group will do their best to pick up the slack and deliver what needs to be done. Of course, this must be a reasonable task and based on the quantity of work and the amount of time until the deadline, a decision must be made if it is realistic to pick up the slack. If it is not, professors, TAs and supervisors (if relevant) must be contacted and informed of the situation and new expectations must be set.

Accountability and Teamwork

Quality

There is no specific requirement to formally prepare anything for meetings aside from a general agenda of what needs to be discussed which will be composed beforehand by the group as a collective. However, the quality of preparation should be that each member is aware of what will be discussed and has collected thoughts or familiarized themselves on the subject so they can meaningfully contribute to the discussion. If the action item specifically refers to a subject one group member is responsible for or has been extensively working on, they should be prepared to lead the disucssion.

Deliverable quality also does not have explicit requirements but it must pass three checks that will be done for each deliverable.

1. All code written must comply with the our groups coding standard.
2. All pull requests must hold up to the scrutiny of a reviewer that is responsible for critiquing the changes and provide feedback as needed (this cannot be the person who made the request).
3. In the case of written deliverables, all work must also hold up to a final document review done collectively by the group at the time of submission.

Attitude

All team members already have strong relationships with each other and thus we already have an implicit framework for interaction and cooperation. We all recognize each other as competent individuals so team members' ideas should and will be openly welcomed but also openly critiqued if a team member feels it is necessary to push back. Fortunately we have a comfort level that allows us to push back on ideas without fear of being made socially uncomfortable. This also includes bringing up any unmet expectations within the group. Our relationships and knowledge that each member is willing and able to do their part allows us to comfortably address anyone that is not doing their part.

As friends, we have also experienced conflict with each other in the past and although it is never enjoyable, we are comfortable with confrontation and are confident in our ability to resolve conflicts peacefully and thus a code of conduct or conflict resolution plan is not necessary.

Stay on Track

The main method to keep the team on track will be the kanban board in the projects of the repository. At the beginning of every deliverable/iteration, the kanban should be filled with all of the required action items for that deliverable/iteration. This gives visibility to all the things that need to be done and due to the assignment feature of the kanban, everyone can see exactly how much work each member is comitting to take on. This visibility ensures that each group member is comitting to complete a reasonable amount of work.

If a member does not contribute their fair share, the kanban will provide visibility to this and ideally, the team will be on top of them before the deliverable is due to avoid missing expectations. If a group member does not meet expectations, we do not believe there should be any specific punishments and we don't believe that any group member should be rewarded and thus encouraged to take on more than their fair share. We believe that talking to the member that did not meet expectations will be sufficient and if it is a continued pattern we will make the decision of whether or not to reach out to our capstone professor who will then provide potential consequences.

Note that our main performance metrics will be number of commits and number of kanban issues resolved although performance in these metrics will not necessarily have any material benefit for any specific group member. We are a team, we do not want to encourage forming a hierarchy based on material rewards and punishment.

Team Building

All members of the group already have relatively strong relationships with each other both inside and outside of class. Planned fun time or group rituals are not necessary as they will happen organically as a part of everyone's general social life.

Decision Making

Generally, decisions will be made by consensus. If there are disagreements, we will try to resolve them conversationally to see if a gap can be bridged and a solution devised. If it can not be resolved, we will resort to a vote in which all

team members will participate in and, as a group of five, the majority vote will be the direction the team moves in.