Table 1: Revision History

Date	**Developer(s)**	**Change**
Sept 24	Jasper Leung	Team Meeting Plan, Team Communication Plan
Sept 24	Hongzhao Tan	Technology
Sept 25	Mengtong Shi	Coding Standard
Sept 25	Zabrain Ali	Workflow Plan
Sept 25	Mike Li	POC Demonstration Plan
Sept 25	Linqi Jiang	Project Scheduling
Nov 13	Zabrain Ali	Modified POC Demonstration Plan
April 3	Mengtong Shi	Modified Technology, Coding Standard
April 5	Jasper Leung	Modified Document according to feedback from Revision 0

Development Plan
PyERT

Team 17, Track a Trace
Zabrain Ali
Linqi Jiang
Jasper Leung
Mike Li
Mengtong Shi
Hongzhao Tan

April 5, 2023

This document will go over the Development Plan of the project Match GPS Traces to a Transportation Network.

1 Team Meeting Plan

1.1 Overview

Track a Trace will hold scrum meetings weekly. All members will be required to attend these scrum meetings. Along with these scrum meetings, the team will organize at least one informal work session per week with a select number of members, where we will sync our progress on the tasks we are working on and work as a group to write documentation or develop code.

1.1.1 Scrum Meeting Structure

Due to inconsistent schedules, the time slots of these scrum meetings will vary. However, they will all be 15-30 minutes long and usually held online on Microsoft Teams. The Scrum Master will start the meeting. Team members will speak in order, stating what tasks they are planning to work on, currently working on, and have finished since the last scrum meeting. The team will give feedback to each other based on what they have said during their turn.

1.1.2 Work Session Structure

The time slots of these work sessions will vary based on the work that needs to be done. The general goal of these meetings is to have each participant of

the meeting state what they want to work on before starting the meeting, and after that, the team will come to a conclusion on what will be worked on during that work session. Once again, what happens during that work session will vary based on the work. After each work session, the team will collectively recap what was done during the work session. All work done will be documented by GitHub commits and code documentation.

2 Team Communication Plan

Track a Trace plans to hold in-person meetings, as mentioned above, but also plans to use online communication methods.

2.1 Microsoft Teams

Microsoft Teams will serve as the team's main form of online communication. Microsoft Teams provides file sharing, message logging, and real-time communication, which will be useful assets for the team. Team members will use the Teams messaging feature to update each other on what they will be working on next and to organize team meetings. Online meetings will be held on Teams as well.

2.2 Cell Phones

Cell phones will be used by the team to communicate with each other before meeting in person. Cell phones are more convenient to use than Teams on the go, and using having team members call and text each other will ensure that we are able to figure out how to meet ~~while we are~~ even if we are in transit.

2.3 GitHub

The team plans to utilize GitHub branches and pull requests, and through those, the team can give feedback on the changes that a member has made. This will allow for a more detailed review of the work that each member has done, as well as make it easier to pinpoint specific issues compared to using Microsoft Teams. Also, GitHub will be used to store all of the team's documentation.

3 Team Member Roles

- Jasper Leung
  - Scrum Master
  - Developer
  - Transport Mode Detection Expert: Will do research and understand how to detect different transport modes in a travel episode (e.g. stopping, walking, driving, etc.)

- Hongzhao Tan
  - Developer
  - Route Generation and Visualization technology: Will do research and understand how to generate and visualize routes from given sets of GPS points using open source Python packages
- Mengtong Shi
  - Developer
  - LaTeX Specialist: Will do research on how the group can best utilize LaTeX features for the group's documentation, and understand how to solve different LaTeX bugs that may arise
- Zabrain Ali
  - Developer
  - GeoPandas Expert: Will do research and thoroughly understand how to use the GeoPandas Python package, and assist other developers in using this package
- Mike Li
  - Developer
  - GERT Expert: Will read through the original code and documentation of GERT and understand the main functionalities
- Linqi Jiang
  - Developer
  - Activity Locations Detection Expert: Will do research and understand how to detect activity locations near a generated route

4 Workflow Plan

The process of the workflow will revolve around git. The first step of the workflow is to pull any changes from the master branch. The next step is to create a new branch ~~(i.e. feature, bugfix etc.)~~ from the master branch. These branch names will be named based on the feature being implemented in the branch (e.g. rev0-preprocessing-module, rev1-progressbar)). As code changes are made, commits with detailed messages will be made. Git tags will also be used to mark stable releases or the achievement of very important milestones. The next step is to add unit/integration tests to our code. After all the necessary code changes and tests are committed and pushed to a branch, the next step is to create a pull request to merge to the master branch. ~~We will use CI/CD for the software development of this project to run automated tests using GitHub Actions once a pull request has been made. Furthermore, all developers~~ All the developers

who have done work related to the content of the pull request will be added as a reviewer for each pull request, and all ~~developers~~ reviewers must approve the pull request before merging to the master branch. Once a branch has been merged to master by the approval of all reviewers, the branch will be deleted, and the workflow will repeat from the first step.

In the event there are any merge conflicts, here is the workflow to resolve them. First, we will identify where the merge conflicts occur and what developers were working on the same files. Next, developers will review the conflict together and make any necessary changes, whether that is combining both code changes, removing one developer change, or rewriting the file entirely. After editing the file, we will do a git add to stage the new merged file and commit it. Git will then create a new merge commit to finalize the merge.

5 Proof of Concept Demonstration Plan

~~The main risk for the success of this project is that ArcGIS might have a functionality that is too complex or not replicable with free Python tools. Since we are replacing ArcGIS functionality with free tools to make GERT free of licensing ArcGIS, any ArcGIS function that is irreplaceable would require licensing. Another risk would be if it would take too long to replace all the ArcGIS functions with free libraries and tools. As long as there exists a single function using ArcGIS in GERT, whether it's from the function being too complex to replace or if it's because we did not have enough time to replace it, a license needs to be bought, and the success of this project would be compromised. If during the Proof of Concept Demonstration, we demonstrate that all ArcGIS functionality a few methods in the source code can be replicated with free tools and libraries, and find a reasonable number of functions that require ArcGIS in GERT, we can convince ourselves that we can overcome the risks.~~

To be a successful project, the project should emulate the input, processing, and outputs of the GERT program. Inputs, data pre-processing and classification of GPS points should be relatively straightforward with Python packages such as GeoPandas (converting CSV files to GeoDataFrames, extracting/classifying info etc.). The main risk for the success of this project is that the main outputs of GERT, route generation and activity locations detection, may not be able to be replicated with free Python tools. In order to demonstrate that these obstacles can be overcome, the goal of this Proof of Concept is to write scripts in Python that show that these outputs can be recreated. There will be two Python scripts created: one script will show generate routes and visualize them from a set of GPS points. The other script will show that Activity Locations data can be extracted from the original file format, and activity locations near the generated route from the first script can be detected. Both of these scripts will only use open-source Python packages.

6 Technology

- Python ~~and R~~ will serve as the coding language~~s~~, because it offers many different free open source packages that can be used to recreate the features of GERT without the use of ArcGIS or any licensed software.
- OpenStreetMap, an open geographic database, will be used to obtain geographic data that the GPS points will be mapped to and also obtain activity and land use data. The database will be accessed using its free-to-use API
- GeoJSON, an open standard file format designed for visualizing geographical features, will be used as the main visualization tool for generated routes and activity locations. This is because it provides a free-to-use Python package and provides visualization that meets the group's standards.
- GeoPandas, an open-source project designed to make working with geospatial data in Python easier, will be used in the project. GeoPandas extends the datatypes used by pandas to allow spatial operations on geometric types. Geopandas will be used because it can be used alongside GeoJSON. Also, being able to use features from Pandas will be useful for data processing and sorting.
- Pylint will serve as the linter tool because it provides static code analysis for checking Python code written for the project, saving time by preventing the need for manual analysis.
- Pytest will serve as unit testing framework because it provides automated testing that can be used for the different Python modules that will be written for this project.
- GitHub will be used for version control because it is the technology that the group is experienced with, and it offers useful collaboration and version control features that will be useful for this project.
- ~~GitHub Actions will be used for Continuous Integration (CI) purposes. The main branch will be protected with branch protection rules, including:~~
  - ~~Require an approved pull request before merging~~
  - ~~Require status checks to pass before merging~~
  - ~~Require source branches to be up to date before merging~~
- Visual Studio Code~~, RStudio~~ and other potentially helpful IDEs will be used for implementation because they offer features useful for organizing code and increasing productivity during development.
- Latex and Overleaf will be used for editing and generating documentation because they produce professional-looking documentation and allow for easy collaboration between members.

7 Coding Standard

- The Python code will follow the PEP 8 style guide.
- Black will be used to format the Python code.
- pycodestyle (formerly pep8) will be used to check the Python code against the PEP 8 style guide.
- Pylint will be utilized as the linter and help enforce the coding standard.
- ~~The R code will follow Google's R Style Guide, which can be found here.~~ These coding standards will be followed to keep the code consistent, organized and easy to read.

8 Project Scheduling

- See Gantt chart at the following URL: Gantt chart