System Design for Software Engineering: PyERT

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

January 18, 2023

1 Revision History

Date	Version	Notes
January 12, 2023	1.0	Edited Reference Material, Introduction, Purpose, Scope
January 17, 2023	1.1	Edited Project Overview User Interface
January 18, 2023	1.2	Edited Timeline, Interface and Reflection

2 Reference Material

This section records information for easy reference.

2.1 Abbreviations and Acronyms

Acronym/Abbreviation	**Definition**
GIS	Geographic Information System
GERT	GIS-based Episode Reconstruction Toolkit
ArcGIS	A licensed GIS service used by GERT for data processing
PyERT	Python-based Episode Reconstruction Toolkit
Req.	Requirement
NFR	Non-Functional Requirement
CSV	Comma Separated Values
URL	Uniform Resource Locator
GPS	Global Positioning System
TUD	Time Use Diary: Survey data containing a chronological sequence of activities that respondents did over a time period
m/s	meters per second
SHP	shapefile: A data format for spreadsheet
LU	Land Use
PAL	Potential Activity Locations
CI/CD	Continuous Integration / Continuous Deployment
RCA	Route Choice Analysis
GeoJSON	A file format for encoding a variety of geographic data structures
geojson.io	an open-source web tool to convert, edit, and create GeoJSON files
GUI	Graphical User Interface

Contents

1 Revision History i

2 Reference Material ii
- 2.1 Abbreviations and Acronyms . . . ii

3 Introduction 1

4 Purpose 1

5 Scope 1
- 5.1 System Context Diagram . . . 1

6 Project Overview 2
- 6.1 Normal Behaviour . . . 2
- 6.2 Undesired Event Handling . . . 2
- 6.3 Component Diagram . . . 2
- 6.4 Connection Between Requirements and Design . . . 3
  - 6.4.1 Functional Requirements . . . 3
  - 6.4.2 Look and Feel Requirements . . . 5
  - 6.4.3 Usability and Humanity Requirements . . . 5
  - 6.4.4 Performance Requirements . . . 6
  - 6.4.5 Operational and Environmental Requirements . . . 6
  - 6.4.6 Maintainability and Support Requirements . . . 6
  - 6.4.7 Security Requirements . . . 7
  - 6.4.8 Legal Requirements . . . 7

7 System Variables 7

8 User Interfaces 8

9 Design of Hardware 8

10 Design of Electrical Components 8

11 Design of Communication Protocols 8

12 Timeline 9

A Interface 11

B Mechanical Hardware 12

C Electrical Components 12

D Communication Protocols 12

E Reflection 12

List of Tables

1 Timeline 9

List of Figures

1 **System Context Diagram** 1
2 **Component Diagram** 3
3 **Input** 11
4 **Stages** 11
5 **Output** 11
6 **Error Message** 11
7 **Warning Message** 11

3 Introduction

This is the System Design for the project PyERT. The project aims to re-implement the functionalities of GERT (Dalumpines and Scott, 2018) which uses ArcGIS Pro packages, with open-source packages and libraries, and remove any use of ArcGIS in GERT. The same as the original GERT toolkit, the purpose of the product system is to match GPS trajectories to transportation network for further analysis to the GPS data. Some other documentations for this project include the SRS (Ali et al., 2022c), the Hazard Analysis (Ali et al., 2022a), and the System Verification and Validation Plan (Ali et al., 2022d).

4 Purpose

The purpose of the documentation is to specify the scope of the system, to give an overview of the project which includes the normal behavior of the system, how the undesired even will be handled, a component diagram for the system and the correspondence between requirements specified in SRS(Ali et al., 2022c)and the design decisions, and to provide a sketch for the user interface of the product system.

5 Scope

The product system is designed for the supervisor of the project Dr. Antonio Paez and other potential users of the system to automatically match GPS trajectories to transportation network for further analysis of the GPS data. It includes the implementation of the system only.

5.1 System Context Diagram

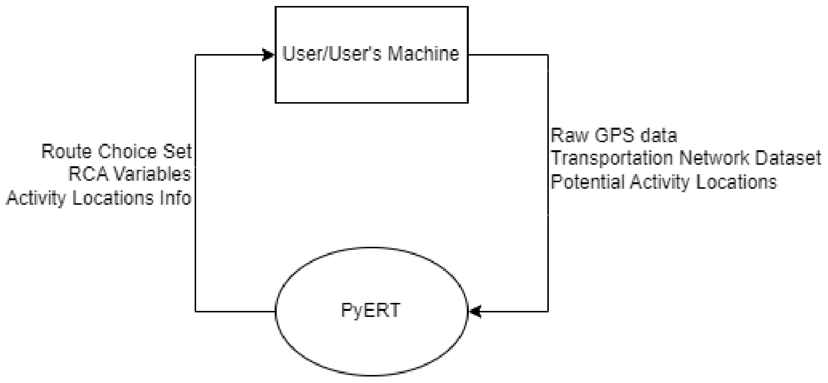


Figure 1: **System Context Diagram**

6 Project Overview

6.1 Normal Behaviour

The product is designed for the supervisor of the project Dr. Antonio Paez and other potential users who need a software tool to automatically match GPS trajectories to transportation network for further analysis of the GPS data. Users will use the product on their computers (laptop or desktop) which have Windows or Linux as operating system. The computer will store locally, the GPS data that needs to be matched in CSV format and the transportation network datasets the GPS data will be matched to in Geodatabase (.gdb) or Protocolbuffer Binary (.pbf) format. Users will use the product by first making a main function call in console. Then following the corresponding prompts in the console, the user will need to input the file paths to the GPS data, the network dataset and the output folder that the user wants to place the files generated by the product into. After gathering user's inputs, the product will start matching the GPS data and generate prompts in the console to inform the user about which specific stage the matching is currently on. Eventually, the product will generate a route choice SHP file, a CSV file that contains the RCA variables' values for the route choice, and a SHP file for activity locations' information if potential activity locations have been provided in the input network dataset, in the output folder that was specified by the user. The product will also generate two URLs in the console that the user can use to visualize the generated route choice and/or the activity locations on geojson.io.

6.2 Undesired Event Handling

To handle the undesired events, When an unexpected event occurs, the product should print a warning for the user in the console which will include a trace-back on the line of the code where the event occurred and a brief explanation if the reason can be determined, and then terminate the current use of the product as soon as possible. Terminating the current use of the product will ensure that any improper or erroneous data or format of input data will not accepted by the system. This will prevent further errors when the system attempts to read or modify corrupted or incorrect data. After terminating the current use of the product, users will need to making a main function call in console again and re-input the file paths to their GPS data and network dataset to be processed by the product again.

6.3 Component Diagram

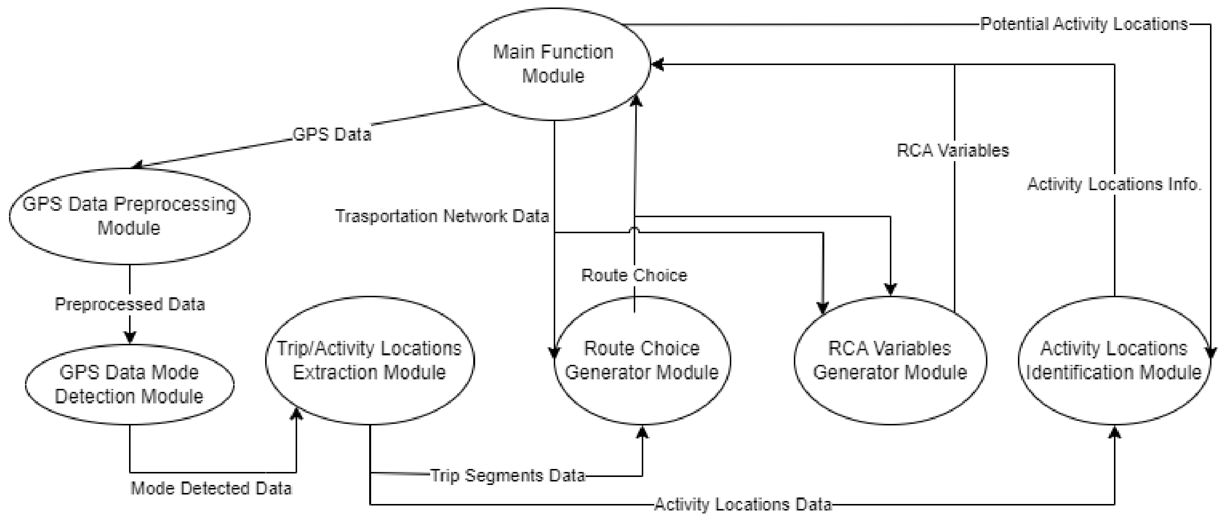

Main Function Module
Potential Activity Locations
GPS Data
RCA Variables
Activity Locations Info.
GPS Data Preprocessing Module
Trasportation Network Data
Route Choice
Preprocessed Data
GPS Data Mode Detection Module
Trip/Activity Locations Extraction Module
Route Choice Generator Module
RCA Variables Generator Module
Activity Locations Identification Module
Trip Segments Data
Mode Detected Data
Activity Locations Data

Figure 2: **Component Diagram**

6.4 Connection Between Requirements and Design

6.4.1 Functional Requirements

1. Req #1
   - To satisfy this requirement, a design decision the team will implement is to use the library Pandas to read and process CSV files, and use the library GeoPandas to read SHP files.
2. Req #2
   - To satisfy this requirement, a design decision the team will implement is to use the built in function in GeoPandas *drop_duplicates* that will remove any duplicate points.
3. Req #3
   - To satisfy this requirement, a design decision the team will implement is to use GeoPandas to divide GPS points and clusters
4. Req #4
   - To satisfy this requirement, a design decision the team will implement is to use the given GPS input data and define a stop point (stationary) as a row that has a value of *0* for the column *Speed_kmh* and the remaining rows will be trip (moving) points
5. Req #5

- To satisfy this requirement, a design decision the team will implement is to use GeoPandas to partition GPS trajectories into segments.

6. Req #6

   - To satisfy this requirement, a design decision the team will implement is to sort the GPS data by segments after partitioning it and from there extract each trip segment

7. Req #7

   - To satisfy this requirement, a design decision the team will implement is to first define each point as a stop point or trip point. Then from there the GPS trajectories will be partitioned using GeoPandas into segments. Next, sort the GPS Data by segments. To extract the points in stop episode we simply extract the points in the stop segments. To extract the end points of trip segments we simply go to the first point in each stop segment, look at the previous point and if it is a trip point then we know that is an end point of a trip segment

8. Req #8

   - To satisfy this requirement, a design decision the team will implement is to use Geopandas to generate a route for any trip segment which will then generate alternative routes for entire trip segments

9. Req #9

   - To satisfy this requirement, a design decision the team will implement is to use GeoPandas to generate SHP files for extracted activity locations.

10. Req #10

    - To satisfy this requirement, a design decision the team will implement is to display an error in the command line window with a brief description as to why there is a run-time error as well as create logs using *logging* to add traceability to the errors and provide a more detailed description for the error.

11. Req #11

    - To satisfy this requirement, a design decision the team will implement is to use GeoPandas to partition the GPS trajectories into segments. If a segment contains only stop points then it will be classified as a stop episode. If a segment contains only trip points then it will be classified as a travel episode. The team will extract each segment and put them into a separate list of GeoDataFrames, one for stop episodes and one for travel episodes. An example of what this would look like to access the first stop episode of the GPS trajectories is to define a list called *stop_episodes* and to call the first index of the list, *stop_episodes[0]*, which will output the GeoDataFrame of the first segment with only stop points, in other words the first stop episode

12. Req #12
    - To satisfy this requirement, a design decision the team will implement is to use Pandas to assign values to RCA variables in CSV format.

6.4.2 Look and Feel Requirements

1. LF1
    - To satisfy this requirement, a design decision the team will implement is to use the command line interface to prompt the user specific inputs and generate output files in directory who's path will be displayed for the user to easily access.
2. LF2
    - To satisfy this requirement, a design decision the team will implement is to create a progress bar on the command line as well as print out each specific stage the program is in.

6.4.3 Usability and Humanity Requirements

1. UH1
    - To satisfy this requirement, a design decision that will be implemented is to include all of the required Python modules for the program within the same folder, which can be uploaded online as an easily downloadable zip file.
2. UH2
    - To satisfy this requirement, a design decision that will be implemented is to include sample inputs/outputs within the program zip file, as well as provide descriptive error prompt messages when the program receives an unexpected inputs.
3. UH3
    - To satisfy this requirement, a design decision that will be implemented is to have a **help** flag that the user can type in the command line to display instructions on how to use the program.

6.4.4 Performance Requirements

1. PR1

    - To satisfy this requirement, a design decision that will be implemented is a timer function, which will output the time it took for the program to generate results after user input. This function will be used during testing.

2. PR2

    - To satisfy this requirement, a design decision that will be implemented is a conditional statement after the user input, which will check the amount of GPS points input, and proceed with the program if there are 50 million or less GPS points in the input, or provide an error message and exit the program otherwise.

3. PR3

    - To satisfy this requirement, a design decision that will be implemented is to avoid using any time-sensitive packages or libraries within the program.

6.4.5 Operational and Environmental Requirements

1. OE1

    - To satisfy this requirement, a design decision that will be implemented is to avoid using any libraries, packages, or code within the program that are restricted by an operating system.

2. OE2

    - To satisfy this requirement, a design decision that will be implemented is to avoid using any external packages within the code, and only use open source Python packages.

3. OE3

    - To satisfy this requirement, a design decision that will be implemented is to include all of the required Python packages within a 'requirements.txt' text file. These packages will be able to be installed using 'pip install -r requirements.txt'.

6.4.6 Maintainability and Support Requirements

1. MS1

    - To satisfy this requirement, a design decision that will be implemented is to have all of the source code for the program located on a GitHub repository, which will be modified any time there are changes to the program.

2. MS2
    - To satisfy this requirement, a design decision that will be implemented is that the GitHub repository the program is located on will have an option for developers to open tickets.
3. MS3
    - To satisfy this requirement, a design decision that will be implemented is to avoid using any packages, libraries, or code that depend on the user's current location within the program.

6.4.7 Security Requirements

1. SR1
    - To satisfy this requirement, a design decision that will be implemented is to avoid using any packages, libraries, or code that stores the user's personal information.
2. SR2
    - To satisfy this requirement, a design decision that will be implemented is to avoid using any packages, libraries, or code that uses files outside of the user provided inputs and files included in Python and the program directory.
3. SR3
    - To satisfy this requirement, a design decision that will be implemented is to push a Git commit with a descriptive message whenever a meaningful change is made to the source code.

6.4.8 Legal Requirements

1. LR1
    - To satisfy this requirement, a design decision the team will implement is to use free-use open source packages such as GeoPandas for the development of the product.

7 System Variables

N/A

8 User Interfaces

The product is designed to interact with the user through command-line prompts, which is the same as the existing implementation of GERT. Examples of the user interface can be found in Appendix A. As stated in the Problem Statement and Goals (Ali et al., 2022b) document, implementing the GUI would be considered a stretch goal for the current version of the system.

9 Design of Hardware

N/A

10 Design of Electrical Components

N/A

11 Design of Communication Protocols

N/A

12 Timeline

Modules	**Rev 0 Implementation Tasks**	**Deadline**	**Team Members Responsible**
Hardware-Hiding Module	Provided by OS	N/A	N/A
GPS Data Preprocessing Module	Implement all the functions included in the module	February 6, 2023	Jasper, Mike
GPS Data Mode Detection Module	Implement all the functions included in the module	February 6, 2023	Jasper, Mike
Trip Segments and Activity Locations Extraction Module	Implement all the functions included in the module	February 6, 2023	Jasper, Mike
Route Choice Set Generator Module	Implement all the functions included in the module	February 6, 2023	Mengtong, Hongzhao
Route Choice Analysis Variables Generator Module	Implement all the functions included in the module	February 6, 2023	Linqi, Hongzhao
Activity Locations Identification Module	Implement all the functions included in the module	February 6, 2023	Zabrain
Main Function Module	Implement all the functions included in the module	February 6, 2023	All team members
Dataframe Data Structure Module	Provided by external libraries	N/A	N/A
GeoDataframe Data Structure Module	Provided by external libraries	N/A	N/A
OSM Network Dataset Reader Module	Provided by external libraries	N/A	N/A
Plotting Module	Provided by external libraries	N/A	N/A

Table 1: Timeline

The Gantt Chart of the project timeline can be found here.

References

Zabrain Ali, Linqi Jiang, Jasper Leung, Mike Li, Mengtong Shi, and Hongzhao Tan. Hazard analysis for software engineering: Pyert, 2022a. URL https://github.com/paezha/PyERT-BLACK/blob/main/docs/HazardAnalysis/HazardAnalysis.pdf.

Zabrain Ali, Linqi Jiang, Jasper Leung, Mike Li, Mengtong Shi, and Hongzhao Tan. Problem statement and goals for software engineering: Pyert, 2022b. URL https://github.com/paezha/PyERT-BLACK/blob/main/docs/ProblemStatementAndGoals/ProblemStatement.pdf.

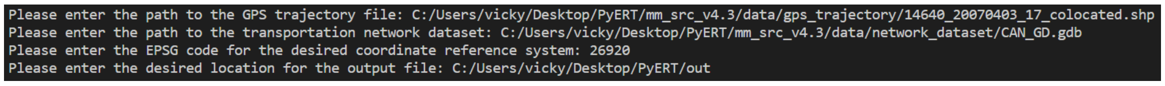
Zabrain Ali, Linqi Jiang, Jasper Leung, Mike Li, Mengtong Shi, and Hongzhao Tan. Software requirements specification for software engineering: Pyert, 2022c. URL https://github.com/paezha/PyERT-BLACK/blob/main/docs/SRS/SRS.pdf.

Zabrain Ali, Linqi Jiang, Jasper Leung, Mike Li, Mengtong Shi, and Hongzhao Tan. System verification and validation plan for software engineering: Pyert, 2022d. URL https://github.com/paezha/PyERT-BLACK/blob/main/docs/VnVPlan/VnVPlan.pdf.

Ron Dalumpines and Darren M. Scott. Gis-based episode reconstruction toolkit (gert): A transferable, modular, and scalable framework for automated extraction of activity episodes from gps data. *Travel Behaviour and Society*, 11:121–130, 2018. ISSN 2214-367X. doi: https://doi.org/10.1016/j.tbs.2017.04.001. URL https://www.sciencedirect.com/science/article/pii/S2214367X17300406.

A Interface

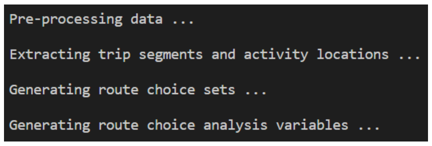
The system will ask for user inputs through command-line prompts as shown in Figure 3.



```
Please enter the path to the GPS trajectory file: C:/Users/vicky/Desktop/PyERT/mm_src_v4.3/data/gps_trajectory/14640_20070403_17_colocated.shp
Please enter the path to the transportation network dataset: C:/Users/vicky/Desktop/PyERT/mm_src_v4.3/data/network_dataset/CAN_GD.gdb
Please enter the EPSG code for the desired coordinate reference system: 26920
Please enter the desired location for the output file: C:/Users/vicky/Desktop/PyERT/out
```

Figure 3: **Input**

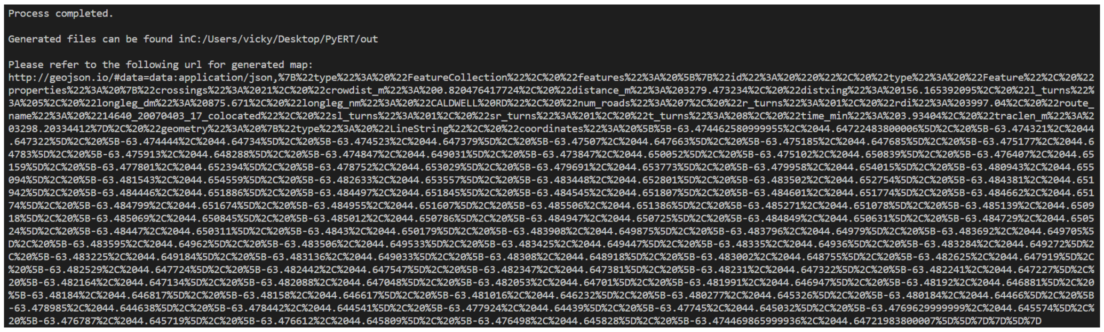
The system will tell the user which stage it is in as shown in Figure 4.



```
Pre-processing data ...

Extracting trip segments and activity locations ...

Generating route choice sets ...

Generating route choice analysis variables ...
```

Figure 4: **Stages**

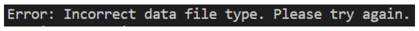
The system will show the user where the output files are and display the URL to the generated map as illustrated in Figure 5.



```
Process completed.

Generated files can be found inC:/Users/vicky/Desktop/PyERT/out

Please refer to the following url for generated map:
http://geojson.io/#data=data:application/json,%7B%22type%22%3A%20%22FeatureCollection%22%2C%20%22features%22%3A%20%5B%7B%22id%22%3A%20%220%22%2C%20%22type%22%3A%20%22Feature%22%2C%20%22
properties%22%3A%20%7B%22crossings%22%3A%2021%2C%20%22crowdist_m%22%3A%200.820476417724%2C%20%22distance_m%22%3A%203279.473234%2C%20%22distxing%22%3A%20156.165392095%2C%20%22l_turns%22%
3A%205%2C%20%22longleg_dm%22%3A%20875.671%2C%20%22longleg_nm%22%3A%20%22CALDWELL%20RD%22%2C%20%22num_roads%22%3A%207%2C%20%22r_turns%22%3A%201%2C%20%22rdi%22%3A%203997.04%2C%20%22route_
name%22%3A%20%2214640_20070403_17_colocated%22%2C%20%22sl_turns%22%3A%201%2C%20%22sr_turns%22%3A%201%2C%20%22t_turns%22%3A%208%2C%20%22time_min%22%3A%203.93404%2C%20%22traclen_m%22%3A%2
03298.20334412%7D%2C%20%22geometry%22%3A%20%7B%22type%22%3A%20%22LineString%22%2C%20%22coordinates%22%3A%20%5B%5B-63.47446258099995 5%2C%2044.6472248380000 6%5D%2C%20%5B-63.474321%2C%2044
.647322%5D%2C%20%5B-63.474444%2C%2044.64734%5D%2C%20%5B-63.474523%2C%2044.647379%5D%2C%20%5B-63.47507%2C%2044.647663%5D%2C%20%5B-63.475185%2C%2044.647685%5D%2C%20%5B-63.475177%2C%2044.6
4783%5D%2C%20%5B-63.475913%2C%2044.648288%5D%2C%20%5B-63.474847%2C%2044.649031%5D%2C%20%5B-63.473847%2C%2044.650052%5D%2C%20%5B-63.475102%2C%2044.650839%5D%2C%20%5B-63.476407%2C%2044.65
159%5D%2C%20%5B-63.477801%2C%2044.652394%5D%2C%20%5B-63.478752%2C%2044.653029%5D%2C%20%5B-63.479691%2C%2044.653773%5D%2C%20%5B-63.479958%2C%2044.654015%5D%2C%20%5B-63.480943%2C%2044.655
094%5D%2C%20%5B-63.481543%2C%2044.654559%5D%2C%20%5B-63.482633%2C%2044.653557%5D%2C%20%5B-63.483448%2C%2044.652801%5D%2C%20%5B-63.483502%2C%2044.652754%5D%2C%20%5B-63.484381%2C%2044.651
942%5D%2C%20%5B-63.484446%2C%2044.651886%5D%2C%20%5B-63.484497%2C%2044.651845%5D%2C%20%5B-63.484545%2C%2044.651807%5D%2C%20%5B-63.484601%2C%2044.651774%5D%2C%20%5B-63.484662%2C%2044.651
74%5D%2C%20%5B-63.484799%2C%2044.651674%5D%2C%20%5B-63.484955%2C%2044.651607%5D%2C%20%5B-63.485506%2C%2044.651386%5D%2C%20%5B-63.485271%2C%2044.651078%5D%2C%20%5B-63.485139%2C%2044.6509
18%5D%2C%20%5B-63.485069%2C%2044.650845%5D%2C%20%5B-63.485012%2C%2044.650786%5D%2C%20%5B-63.484947%2C%2044.650725%5D%2C%20%5B-63.484849%2C%2044.650631%5D%2C%20%5B-63.484729%2C%2044.6505
24%5D%2C%20%5B-63.48447%2C%2044.650311%5D%2C%20%5B-63.4843%2C%2044.650179%5D%2C%20%5B-63.483908%2C%2044.649875%5D%2C%20%5B-63.483796%2C%2044.64979%5D%2C%20%5B-63.483692%2C%2044.649705%5
D%2C%20%5B-63.483595%2C%2044.64962%5D%2C%20%5B-63.483506%2C%2044.649533%5D%2C%20%5B-63.483425%2C%2044.649447%5D%2C%20%5B-63.48335%2C%2044.64936%5D%2C%20%5B-63.483284%2C%2044.649272%5D%2
C%20%5B-63.483225%2C%2044.649184%5D%2C%20%5B-63.483136%2C%2044.649033%5D%2C%20%5B-63.48308%2C%2044.648918%5D%2C%20%5B-63.483002%2C%2044.648755%5D%2C%20%5B-63.482625%2C%2044.647919%5D%2C
%20%5B-63.482529%2C%2044.647724%5D%2C%20%5B-63.482442%2C%2044.647547%5D%2C%20%5B-63.482347%2C%2044.647381%5D%2C%20%5B-63.48231%2C%2044.647322%5D%2C%20%5B-63.482241%2C%2044.647227%5D%2C%
20%5B-63.482164%2C%2044.647134%5D%2C%20%5B-63.482088%2C%2044.647048%5D%2C%20%5B-63.482053%2C%2044.64701%5D%2C%20%5B-63.481991%2C%2044.646947%5D%2C%20%5B-63.48192%2C%2044.646881%5D%2C%20
%5B-63.48184%2C%2044.646817%5D%2C%20%5B-63.48158%2C%2044.646617%5D%2C%20%5B-63.481016%2C%2044.646232%5D%2C%20%5B-63.480277%2C%2044.645326%5D%2C%20%5B-63.480184%2C%2044.64466%5D%2C%20%5B
-63.478985%2C%2044.644638%5D%2C%20%5B-63.478442%2C%2044.644541%5D%2C%20%5B-63.477924%2C%2044.64439%5D%2C%20%5B-63.47745%2C%2044.645032%5D%2C%20%5B-63.476962999999%2C%2044.645574%5D%2C%
20%5B-63.476787%2C%2044.645719%5D%2C%20%5B-63.476612%2C%2044.645809%5D%2C%20%5B-63.476498%2C%2044.645828%5D%2C%20%5B-63.47446986599993 6%2C%2044.6472198380000 7%5D%5D%7D%7D%5D%7D
```

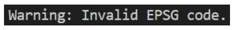
Figure 5: **Output**

The system will tell the user when an error occurs through error and warning messages, as shown in Figure 6 and 7, respectively.



```
Error: Incorrect data file type. Please try again.
```

Figure 6: **Error Message**



```
Warning: Invalid EPSG code.
```

Figure 7: **Warning Message**

B Mechanical Hardware

N/A

C Electrical Components

N/A

D Communication Protocols

N/A

E Reflection

The information in this section will be used to evaluate the team members on the graduate attribute of Problem Analysis and Design. Please answer the following questions:

1. What are the limitations of your solution? Put another way, given unlimited resources, what could you do to make the project better? (LO_ProbSolutions)

   One limitation of our solution is that it does not provide all of the same functionality as GERT. GERT provides an option to input Time Use Diaries (data sets which tracks person's activities throughout the day), and use those to create additional outputs. Since we were not able to obtain samples of what these Time Use Diaries, and also were limited by time, we were unable to implement this feature. If given unlimited time and resources, we would implement additional modules to process these diaries and generate additional outputs.

   Another limitation of our solution is that it does not have a GUI. The current proposed implementation will run using the command line. This is not as intuitive as a GUI, especially for the target audience of geography researchers/students who may not have experience with Python or the command line. If given unlimited resources and time, we would have created a Python application, which would not require any running of commands, and the user would be able to drag and drop inputs into the application window.

2. Give a brief overview of other design solutions you considered. What are the benefits and tradeoffs of those other designs compared with the chosen design? From all the potential options, why did you select documented design? (LO_Explores)

   One design solution we considered was to copy the design of the original GERT project, but replace all of the function calls which used the ArcPy package (a python package for ArcGIS, which required the ArcGIS License to be used) with our own code. The proposed plan was to use the ArcPy documentation and recreate all of the functions with GeoPandas/ similar python packages. The benefit of doing this would be that we wouldn't have to worry about designing our own software architecture, since we could just re-use the architecture from the original working GERT project. Also, replacing all of the code would ensure that our solution would provide the same functions as GERT. However, the tradeoff was that learning about and re-implementing the different functions provided by ArcPy would be too time consuming, compared to finding our own way to process the data.