To give an idea of what is expected in the project here are some of the requirements and guidelines for grading.
Does the code run and produce reasonable/correct output (-10%)
Is the dataset on which it is operating of a reasonable size (1000s of elements/rows/nodes/other units. A benefit of submitting a proposal is that you will know if your dataset is appropriately sized) (-10%)
Is the implementation of good complexity (150+ lines of code, split in modules, not including tests) (-10%)
Does the code have tests of good quality (-10%)
Is there a good write-up describing what the project does, how to run it, what the output looks like etc. (-10%)
What is the quality of the coding (variable name selection, split of functionality in reusable functions, good use of iterators and language features (i.e. enums, structs, methods, ect) (-10%)
Did you use git properly with multiple checkins as you were implementing your project (-10%)
As long as something is turned in even if it doesn't meet the above requirements it will get a base grade of 30% so make sure you submit one!

You are welcome to discuss your project with others but you are expected to submit it individually. Crediting anyone who helped you with ideas or debugging is a must!
In this project you must use Rust. You could analyze a graph dataset using one of the various graph algorithms taught in class or some other kind of dataset using one of the DS techniques taught in class.
When submitting the proposal make sure you mention the following:
1. What data set are you planning to use and why it is interesting to you?
2. What is the problem you are solving or the question you are asking? Show that you have thought about it and share your insights.

3. What are the steps/components needed to accomplish the project? Specify the milestones with approximate dates you will accomplish them and how you plan to test the individual components.
Project Ideas and implementation

Since the class is heavy on graph algorithms here are some possible ideas of projects you can pursue in that space. Do not feel obligated to adopt one of those though you are welcome to. We will consider extra credit for original ideas.
Six degrees of separation: What is the usual distance between pairs of vertices in your graph? Is the answer very different for this versus another graph? (We will cover some algorithms that can be used for computing distances between vertices later in the class. They are called Breadth–First Search and shortest Paths)
Degree distributions: What is the distribution of vertex degrees in your graph? What if you look at the number of neighbors at distance 2? Often people believe that such a distribution should be a power–law distribution (look up online what it means) for social networks and similar graphs. How well does the distribution you see fit this assumption (and how are you going to evaluate it)? Degree of a vertex is the number of other vertices it is connected to.
How often are friends of my friends my friends? This is very generic question, but can you find two vertices who are friends with most similar or most dissimilar sets of connections? What is the right measure of similarity? This could involve searching for established measures of this type and or coming with your own.
Graph clustering and partitioning: Take a network in which the identity of nodes is meaningful. Try to find k best representatives of all the graph vertices. How satisfied are you with the selection, given your previous beliets about this network?
Densest subgraph: (The vocabulary here may not all be obvious, but I'm happy to answer all ques- tions you may have.) Let V be

the set of vertices of the graph. For any V′ ⊆ V, we write E(V′) to denote the subset of edges of the graph that have both endpoints in V ′. The density of a subgraph induced by V ′ is |E(V′)|/|V′|
In the densest subgraph problem, the goal is to find a subset V ′ ⊆ V that maximizes the density.
This problem can be solved exactly, but this is too complicated for this class, I think (but feel free to prove me wrong). Instead there is a relatively simple algorithm that gives a 2-approximation, i.e., a subset V ′ with density at least half the maximum possible. See, for instance, the top of page 5 in https://www.cs.umd.edu/~samir/grant/ICALP09.pdf
Your goal could be to implement this algorithm and apply it to some graph with meaningful vertices. Are the results as expected? What happens when you look at specific subgraphs (say, people living in Massachusetts, or college students)? Are the results surprising?
Centrality measures: Compute select centrality measures for your graph. Do the highest centrality vertices match your your intuition? What happens when you look at some meaningful subgraphs. Is this the case for them?
A few random links that discuss centrality measures:
– https://en.wikipedia.org/wiki/Centrality
– https://towardsdatascience.com/notes-on-graph-theory-centrality-measurements-e37d2e49550a
In particular, look at closeness and betweenness centralities as something that is relatively easy to compute.

Link to data set- https://www.kaggle.com/datasets/shubhambathwal/flight-price-prediction?resource=download

When submitting the proposal make sure you mention the following:

1. **What data set are you planning to use and why it is interesting to you?**

Flight Price Prediction

Objective
This project aims to predict flight ticket prices using historical flight booking data from the "Ease My Trip" website. A linear regression model is employed to make predictions based on various features such as duration, source city, destination city, and class.

Features
- **Airline**: The airline company (6 categories).
- **Flight**: Flight code (categorical).
- **Source City**: The city from which the flight departs (6 unique cities).
- **Departure Time**: Categorized time of day (6 labels).
- **Stops**: Number of stops between source and destination (3 values).
- **Arrival Time**: Categorized arrival time (6 labels).
- **Destination City**: The city where the flight lands (6 unique cities).
- **Class**: Economy or Business (2 values).
- **Duration**: Flight duration in hours (continuous).
- **Days Left**: Number of days left until the flight (derived feature).
- **Price**: Target variable (ticket price).

Running the Project
1. Clone the repository: `git clone <repo-url>`
2. Install dependencies: `cargo install`
3. Run the project: `cargo run`
4. Run tests: `cargo test`

2. What is the problem you are solving or the question you are asking? Show that you have thought about it and share your insights.

1. Degrees of separations

The output of the `degrees_of_separation` function will be the number of **steps** required to travel between two cities. If no direct or indirect connection exists, the function will return `None`.

For example:

- **Degree of separation between Mumbai and Hyderabad**: This should return `1` because there is a direct flight between the two.
- **Degree of separation between Mumbai and Chennai**: This should return `2` because there is no direct flight, but there is an indirect route via Hyderabad.
- **Degree of separation between Mumbai and a non-existent city**: This should return `None`, indicating that no path exists.

2. Regressions:

3. Centrality measures: