**Project Overview:**

For my project, I'm going to build a flight management system that figures out the best ways to fly between cities like El Paso, Dallas, Phoenix, Atlanta, and Los Angeles. I need to make sure it can read flight data and user requests from files and then spit out the best routes based on how cheap or quick they are.  
  
**Tools and Libraries**:

Programming Language: Java (because it's what we've been using in class and I'm pretty comfortable with it).

Libraries: I’m going to use Java's LinkedList, Stack, and Arrays from java.util because that's all Prof. Khan said we could use. None of the stuff he said we are not allowed to use in his email like HashMaps or HashTables.

**Potential Challenges:**

Managing Data Structures: I think the tricky part will be using LinkedList and Stack efficiently since we can’t use more complex and convenient structures like HashMaps. I need to be careful about how I manage and search data so that it doesn't slow down or mess up.

File I/O: Reading and writing files has to be super careful to avoid messing up the data. Plus, I need to handle scenarios where the file might not be formatted right or missing.

Error Handling: Got to make sure the program doesn’t just crash if something unexpected happens. It should be able to tell the user what went wrong.

Performance: Since we might be dealing with loads of flight data, I need to make sure my code is snappy and doesn’t lag.

**Detailed Plan and Pseudocode:**

**Main Execution Flow:**

*Main method:*

*Start up FlightManager*

*Load flights from FlightData.txt*

*Process requests from RequestedFlightFile.txt*

*Output the outcomes*

**Loading Flight Data:**

*initiateFlights(file):*

*Open FlightData.txt*

*Read the number of routes*

*For each route:*

*Split the flight details*

*Update hubs for both origin and destination*

**Updating City Hubs and Routes:***updateCityHubs(origin, destination, fare, duration):*

*Find or make hub for origin*

*Find or make hub for destination*

*Link origin to destination with given fare and travel time* **Handling User Requests:**

*handleRequests(file):*

*Open RequestedFlightFile.txt*

*Read request count*

*For each request:*

*Get the request details*

*If there’s a format issue, log the error*

*Otherwise, find paths, sort them by cost or time*

*Build result string for each valid request*

*Save or display the results*

**Route Searching with DFS:**

*searchRoutes(start, finish):*

*Set up stack and paths list*

*Add start city to stack*

*While there’s something in the stack:*

*Take the top path off the stack*

*If it’s the destination, add to results*

*Otherwise, extend the path to connected cities not yet visited*

*Add these new paths back to the stack*

*Return all the paths found*