Aircraft Routing

Team J: Dalton Bealer, Jeremy Bell, Brenden Guillen, Joe Panthalani, Jonathan Thornton

1. Project Overview
    a. Primary Objective
        i. Develop a program that, given a user defined graph with specified weights, will return the optimal route(s) for a plane to take while meeting certain criteria (all nodes visited, cannot run out of fuel mid flight, etc.)
    b. Goals
        i. Develop a user interface to easily interact with the server. GUI will ideally allow the user to select locations that need to be visited, then will find the best route when the user submits. The user will also be able to select which algorithm to use.
        ii. Implement multiple different algorithms for the user to use.
        iii. (stretch goal): Create an AI model to generate the path(s)
        iv. Create cost algorithms to use for optimizing routes
        v. (stretch goal): multiple routes/algorithms calculated simultaneously using multithreading
    c. Project Boundaries
        i. Included
            1. Project will only handle .csv files
            2. Find routes for a single plane
            3. (stretch goal): Add more planes to path finding
            4. Planes will visit every node
            5. (stretch goal): Planes can be directed to only *need* to visit a set of nodes
    d. Excluded
            1. Will not handle real-time updates to locations (will have to rerun the algorithms to “update” it)
            2. Will not check for hazards (ie, other planes or inclement weather)

2. Understanding Your Users and Stakeholders
    1. Who are your primary users?

        Boeing airplane route designers who need to see the cost and benefits of different routes for their airplanes.
    2. What problems are you solving for them?

        We are trying to create algorithms that give the optimal path for airplane flight designers so they can choose which flight path that their airplanes will take if they know which locations the airplane must visit.
    3. What are their technical capabilities?

        They are capable of going to our website and using the interface in order to enter different locations that the plane needs to visit and then selecting an algorithm to decide how the plane will visit all these points.
    4. What are their key needs and pain points?

A key need is finding an algorithm that will create a path, given a set of points. The path displayed should also display the cost related to using this path (distance travelled, fuel used, etc). Our website should be able to take this input which contains the location points as a .csv file.

Stakeholder Analysis

1. List all project stakeholders (professors, clients, end-users)
    a. Boeing, pilots, passengers
2. Document their expectations and requirements
    a. Boeing wants to find the cheapest routes for their pilots to take
    b. Pilots and passengers expect to have a fast flight
3. Note any conflicts between different stakeholder needs
    a. The cheapest routes that boeing's wants may not be the fastest

3. Functional Requirements

Detail what your system needs to do:

Feature ID: F001

Name: Website

Description: A front end web application where the user goes in order to access the application

Priority: High

User Story: As a user I want to be able to go to a website so I can enter in locations that the plane must fly through. I should also be able to choose an algorithm among a selection which will decide how the planes will fly through the locations.

Acceptance criteria: Website is able to be accessed publicly and when the user goes to it everything should be displayed properly (location graph, algorithm selection, flight path).

Feature ID: F002

Name: Algorithm API

Description: After selecting an algorithm among a drop down selection of algorithms, the website should connect to our back-end application to create the flight pathway based on that algorithm.

Priority: High

User Story: Given location points have been entered into the web application or when a proper file is uploaded, and when an algorithm is selected from a dropdown, then the flight path will be displayed based on that algorithm meaning the web application will connect to the back end application to find the best path using that algorithm.

Acceptance Criteria: Done when the website is able to connect to the back end application in order to run the back end app based on the location points and algorithm selected in the front end web app.

Feature ID: F003

Name: .CSV File Input

Description: On the web application the user should be able to input the locations that the plane must visit by inputting a .csv file into the website

Priority: High

User Story: When the user wants to enter location points that the plane must visit, the user is able to upload a csv file through the website and the website will then display these points on a graph.
Acceptance Criteria: Finished when the user is able to upload their csv file through our web application and the locations will be displayed on the website. The location points should also be stored and able to be accessed by our back end application.

Feature ID: F004
Name: Backend Algorithm
Description: Given the location points and a selected algorithm, our back end application should find an optimal flight path based on the selected algorithm and then send this path back to the website application.
Priority: High
User Story: When the user enters the algorithm type and locations, the back end application should be called so it can find the optimal path through the locations.
Acceptance Criteria: Done when the back end algorithm can find an optimal path given a set of points based on the selected algorithm. The found path should be sent back to the web application to be displayed when finished as well.

Feature ID: F005
Name: Image Creation Algorithm
Description: Given a graph with or without a path, will create an image version of the graph and path if included for the user to see.
Priority: Low
User Story: When the user requests an image of either their input graph, or the output graph with generated path, the application should run this algorithm and either save it as an image and/or display the graph on the GUI
Acceptance Criteria: Complete when the algorithm can create an accurate and complete version of an input graph when told to do so, and either display it via GUI or save it as an image for the user to open later. If an incorrect graph/form is given, the output should either include what it can do and display a warning, or just display an error to the user.

Feature ID: F006
Name: Interactive Graph Maker
Description: Gives the user the option to create the input graph directly on our website using interactive GUI methods.
Priority: Medium
User Story: As a user, I want to be able to create graphs directly on the website using a GUI so that I don't have to create my own csv file to upload.
Acceptance Criteria: Complete when the user can generate any connected, loopless, graph with no multi-edges within the bounds of system memory using solely the GUI.

Feature ID: F007
Name: GUI Graph to csv Converter
Description: Converts the user's GUI generated graph to a csv file that the backend can handle.

Priority: Medium
User Story: As a dev, I want the backend to handle the same file type for simplicity while also giving the user the option to use a GUI.
Acceptance Criteria: Complete when the converter can correctly generate a csv file that maps to the GUI graph.

Acceptance Criteria: [List specific conditions that must be met]
System Behaviors

1. Describe how your system responds to user actions
2. Define input requirements and expected outputs
3. Specify error handling procedures

4. Technical Requirements
Specify how your system will be built:

Development Specifications

1. Programming languages and frameworks
    a. Python programming language
    b. Javascript/HTML/CSS
    c. Flask, React, NetworkX/PlantUML
    d. Pytest for writing test cases.
2. Database requirements
    a. None
3. API integrations needed
    a. None
4. Development tools and environments
    a. Using Fedora Linux in a virtual machine to keep all operating system issues the same for all group members.

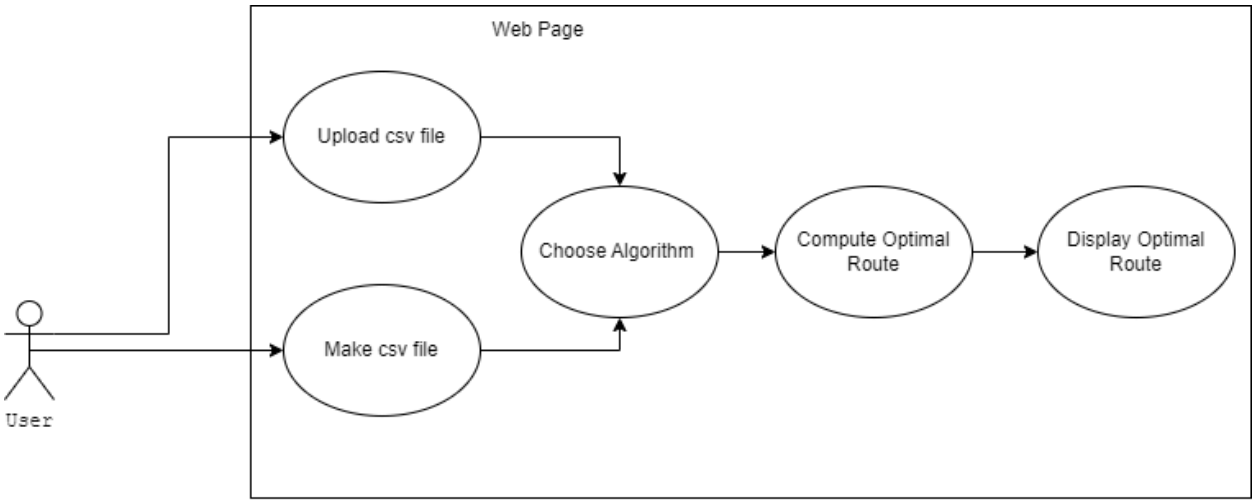
Quality Requirements

1. Performance expectations (response times, load capacity)
    a. Time should be the main performance expectation, with possible computational/memory expectations (ie might be run on a small computer rather than a big server)
2. Security requirements
    a. None, this will not store any data inside about the flight paths, nor will it have a “user login” page
3. Reliability standards
    a. Should be able to create the similar “priced” routes given the same input and algorithm used within a certain time frame
4. Compatibility requirements
    a. Should be compatible will all linux machines given they are running/can run a certain version of python and other add ons (to be determined what those exact versions will be)

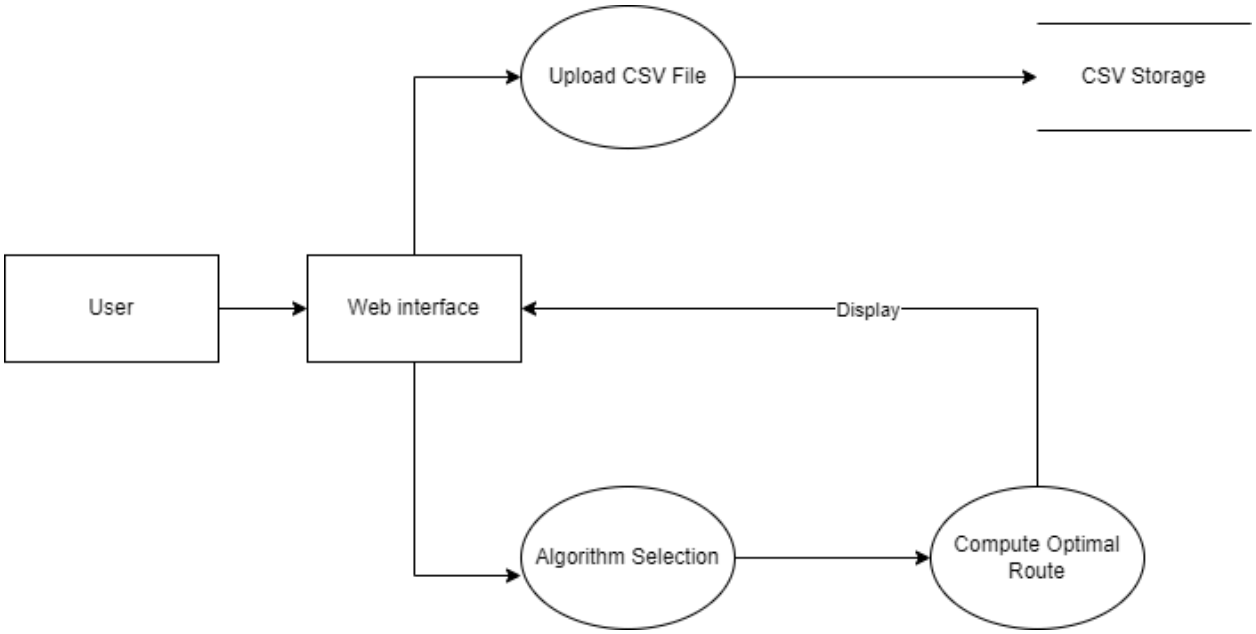
5. Documentation and Diagrams

Create visual representations of your system:
Required Diagrams

1. Use Case Diagram
    - Show system interactions with users
    - Include all major functionalities



2. Data Flow Diagram
    - Illustrate how data moves through your system
    - Show major processes and data stores



3. Entity-Relationship Diagram (if using a database)
    - Show data structure and relationships
    - Include attribute lists for each entity
      Not using a database, unneeded

6. Project Planning Timeline
Organize your development timeline around the planned sprints

S1: Research relevant algorithms, start to structure backend, start structuring frontend
S2: Implement frontend file upload, backend should start handling csv files, implement a backend algorithm
S3: Implement another backend algorithm, implement frontend dropdown algorithm selector
S4: Implement visual graph creator so the user can more easily interpret the optimal path
S5: Implement GUI graph creator and GUI graph to csv converter.
S6: Prettify frontend, debug, refactor code, potentially host site with AWS.

7. Validation Strategy

Ensure your requirements are solid:

Review Process

- Self-review the documentation: done
- Peer review by team members: done
- Professor/advisor review: not done
- Stakeholder review (if applicable): not done
- Final adjustments based on feedback: not done