Santiago Hernández Saavedra A00382538
Juan Jose Barrera A00394876
Sergio Florez Sanabria A00396046

ENGINEERING DESIGN METHOD

1). **Problem identification**

General problem:

The airline Fly-Fast has a serious problem, there is a high degree of dissatisfaction from its customers due to the routes offered. Passengers claim that the offered stopover system is ineffective and negatively affects the flight experience making the routes quite long, uncomfortable and expensive. This has led many users to move to other airlines because, according to them, they offer a better transport service. The staff of the airline does not have very clear how to solve this situation, so we have hired and asked for a technological solution to solve this problem.

Specific symptoms or problems:

1. **Shortfall in evaluating shorter routes:** The airline needs to identify shorter routes for each destination, considering factors such as distance between destinations, possible stopovers and flight times.

2. **Inability to identify user needs and preferences:** The airline must understand the needs and preferences of its users in terms of factors such as flight duration, cost, availability and convenience of schedules, to be able to offer options that meet those needs.

3. **Lack of information in a clear and accessible way:** The airline needs to present information on shorter routes in a way that is easy for its customers to understand and access, so they can make informed decisions about their route selection.

4. **Shortfall in assessment and optimisation of route information:** The airline should regularly review information on shorter routes to ensure that it remains accurate and relevant to users and make adjustments as needed to improve the customer experience.

2). Collection of necessary information:

Santiago Hernández Saavedra A00382538
Juan Jose Barrera A00394876
Sergio Florez Sanabria A00396046

It is pertinent to investigate similar problems and analyze the way in which they were solved. Below are some sources of information.

- **"How Delta Air Lines Is Winning Customer Satisfaction With Big Data".** This Forbes article talks about how Delta Air Lines is using technology and data analysis to improve the customer experience. In particular, the "Shop by Schedule" tool is mentioned to help users find more convenient routes.

  Taken from:

  https://d3.harvard.edu/platform-digit/submission/big-data-takes-flight-at-deltaair-lines/#:~:text=By%20bringing%20data%20to%20what,engage%20customers%20and%20generate%20loyalty.

  "Emirates Launches AI-Powered Travel Assistant Tool". This CNBC article describes how Emirates has launched an artificial intelligence-based travel planning tool, which uses travel data and personal preferences to provide route and fare recommendations to users.

  Taken from:

  https://www.marketingdive.com/news/emirates-vacations-digital-display-ads-integrate-chatbots-that-give-travel/518243/

- **"How United Airlines Uses AI to Improve the Customer Experience".** This VentureBeat article describes how United Airlines is using artificial intelligence to improve the customer experience throughout the journey, from planning to arrival. A travel planning tool called "ConnectionSaver" is mentioned to help users find more convenient routes in case of flight delays or cancellations.

  Taken from:

  https://www.cio.com/article/419642/united-airlines-gives-employees-the-digital-tools-to-make-customers-happy.html

In addition to the information found, there are several concepts that can help us in understanding and analyzing the problem, such as:

- **Stopover [In flight]:** Stopover flights are those in which the aircraft always makes a detour in a city before landing at the final destination, that stop is made for passengers who are going to disembark at that destination, and for those heading for the final destination to board. https://www.egali.com.co/blog/vuelos-con-escala/

- **Customer support:** Satisfaction is a measure of how the products supplied and the services provided by a company, meet or exceed customer expectations. https://www.itaerea.es/atencion-cliente-aeropuertos

- **Route:** A route is the route you take to get from one place to another. In the context of airlines, a route refers to the sequence of destinations that are visited on a flight.

  https://www.beetrack.com/es/blog/ruta-de-transporte-dise%C3%B1arla

- **User experience:** User experience refers to a user's perception of interacting with a product or service. In the context of airlines, user experience can be influenced by factors such as ease of booking flights, quality of service on board and clear and accurate information about routes and schedules.

  https://www.questionpro.com/blog/es/importancia-de-la-experiencia-del-usuario/

- **Travel planning:** Travel planning refers to the process of organizing and booking trip details, including destinations, dates, schedules, accommodations, and activities. In the context of airlines, travel planning can be a complex process that requires accurate and up-to-date information about routes and flight options.

  https://www.euskadi.eus/contenidos/informacion/8005/eu_2594/adjuntos/docu5_04.pdf

Algorithms definitions:

- **Dijkstra:** The Dijkstra algorithm uses a strategy known as "uniform cost search" to gradually scan the nodes of the graph and determine the shortest path from a given source node to all other achievable nodes. The algorithm is very efficient and is widely used in network and transport applications, such as packet routing on the Internet and navigation systems.

- **Floyd - Warshall:** Unlike the Dijkstra algorithm, which finds the shortest path between a pair of specific nodes, the Floyd-Warshall algorithm finds the shortest path between all the node pairs in the graph. This means that it can be used to calculate the shortest distances between all nodes in a single algorithm run, making it efficient for problems where you need to know the minimum distances between all nodes.

- **BFS:** It is a search algorithm used to explore or traverse data structures, such as graphs or trees. It is characterized by traversing the nodes horizontally, first scanning all neighboring nodes before moving to the next nodes. The BFS algorithm starts at a source node and visits all of its neighboring nodes before advancing to the next node level. Then, scan all the neighbors of the newly visited nodes and continue this process until all the nodes reachable from the source node have been visited or the target node has been found (if you are looking for a specific target).

- **Prim:** The objective of the Prim algorithm is to find a subset of edges that forms a tree that includes all nodes of the graph and has the minimum total weight. This tree is known as minimum expansion tree or minimum generator tree.

- **Kruskal:** The objective of the Kruskal algorithm is to find a subset of edges that forms a tree that includes all nodes of the graph and has the minimum total weight. This tree is known as minimum expansion tree or minimum generator tree.

3) Search for creative solutions.

Taking into account the graph algorithms that we saw in class. These are some of the possible solutions to attack the problem of minimizing travel time and having the minimum cost, depending on the client's need.

1. **Dijkstra y Floyd-Warshall**: To use Dijkstra's algorithm, we must build a graph that represents all the possible routes between the destinations offered by Fly Fast. We will assign weights to the edges that indicate the cost or duration of each path. We then run Dijkstra's algorithm from the desired origin airport to find the shortest path to all other airports. This will give us the necessary information to select the most efficient routes in terms of time and cost. In the case of the Floyd-Warshall algorithm, we build an adjacency matrix that represents all the possible routes between the destinations offered by Fly Fast. We assign weights to cells in the matrix that indicate the cost or duration of each route. We then run the Floyd-Warshall algorithm on the matrix to find the shortest distance between all pairs of airports. This will allow us to select the optimal routes in terms of time and cost for passengers.

2. **Dijkstra and BFS:** Apply Dijkstra's algorithm to find the shortest path from the source to the target, returning a list that represents, in order, the vertices to be reached. This involves assigning weights or distances to the edges of the graph and calculating the

shortest distances using Dijkstra's algorithm. Use the results obtained from the previous step to identify the shortest and most efficient routes between destinations.

Already with the list of airports that represents the fastest route, the distance of the entire trip will be calculated with the BFS algorithm.

3. **Prim y Kruskal**: Prim's algorithm is used to find a minimum spanning tree in a weighted and undirected graph. In this case, a graph could be built where the nodes represent the airports and the edges represent the possible routes between them. Assigns weights to edges based on cost or travel time. Then, run Prim's algorithm to find the minimum spanning tree that connects all the airports. This will give you a set of efficient and profitable routes for Fly Fast passengers. On the other hand, Kruskal's algorithm is also used to find a minimum spanning tree in a weighted and undirected graph. Following a similar approach, we can build a graph where the nodes are the airports and the edges represent the routes between them with their respective weights. Run Kruskal's algorithm to select a subset of edges that form a minimum spanning tree connecting all airports. This tree will provide you with the most efficient and profitable routes for Fly Fast passengers.

4). Transition of ideas to preliminary designs.

Alternative #3 discarded.

Knowing the options available to solve the problem, the option of implementing **Prim y Kruskal**, since although Prim and Kruskal's algorithms are effective in finding efficient and profitable routes in route selection problems, there are some considerations that could hinder their applicability in the case of Fly Fast. These algorithms do not take into account additional restrictions such as flight schedules or connections between routes, which limits their ability to generate practical solutions in a dynamic context and in real time. To address these challenges, an additional approach would be required to incorporate Fly Fast-specific information and restrictions, in order to offer suitable routes and meet passenger needs more accurately and efficiently. In the implementation part, the Prim and Kruskal algorithms can present difficulties. These difficulties include handling complex data structures, optimizing performance on large graphs, adapting to special cases, and properly handling real-time updates. These challenges require an additional level of knowledge and experience to implement the algorithms effectively.

Which leaves us with alternatives 1 and 2, which help for the following reasons:

Possible options:

Alternative #1 Dijkstra y Floyd-Warshall

Santiago Hernández Saavedra A00382538
Juan Jose Barrera A00394876
Sergio Florez Sanabria A00396046

The combination of the Dijkstra and Floyd-Warshall algorithms offers a comprehensive solution.

Dijkstra's algorithm is useful for finding the shortest path between two specific airports. We can apply this algorithm to determine the most efficient route from one origin airport to all other airports. This will help us select routes that minimize total travel time and avoid unnecessary layovers. Dijkstra focuses on a single source, allowing routes to be analyzed individually and ensuring time optimization.

On the other hand, the Floyd-Warshall algorithm is used to find the shortest distance between all airport pairs in one step. This algorithm considers all the possible paths between the airports and provides the necessary information to evaluate the most profitable routes in terms of costs. Floyd-Warshall takes a global view and allows us to consider all possible origin and destination combinations, helping us select the most economical routes and avoid costly options.

By combining both algorithms, we can obtain a complete solution to the problem of selecting suitable paths. We can use the Floyd-Warshall algorithm to analyze all combinations of origin and destination, obtaining the shortest distance and the associated cost between each pair of airports. We can then use Dijkstra's algorithm to find the shortest route from the origin airport to all other airports. This combination allows us to identify the optimal routes in terms of time and cost, ensuring efficient and cost-effective route selection for passengers.

In summary, the combination of the Dijkstra and Floyd-Warshall algorithms provides us with a comprehensive solution to the Fly Fast route selection problem. Dijkstra helps us to optimize travel time, avoiding unnecessary stopovers, while Floyd-Warshall allows us to select the cheapest routes in terms of cost. This combination ensures efficient and cost-effective route selection.

Alternative #2 Dijkstra and BFS

The Dijkstra and BFS algorithms can be combined in the following way to solve the problem:

Apply Dijkstra's algorithm to find the shortest path from the source to the target. This involves assigning weights or distances to the edges of the graph and calculating the shortest distances using Dijkstra's algorithm.

Use the results obtained from the previous step to identify the shortest and most efficient routes between destinations. These routes can be presented as options for users, considering factors such as flight duration, cost, and schedule convenience.

Use the BFS to calculate the total distance, with the list of airports taken from the Dijkstra showing the most efficient route at that factor

Santiago Hernández Saavedra A00382538
Juan Jose Barrera A00394876
Sergio Florez Sanabria A00396046

It is important to note that the construction of the graph can be computationally expensive if there are many destinations and connections. In that case, we can apply optimizations such as limiting the scan to a maximum number of scales or using heuristics to guide the search and reduce the scan space.

5) Evaluation and selection of the best solution:

In the evaluation and selection process, the following criteria will be taken into account:
ease of implementation, efficiency of the algorithm in search of better routes, spatial efficiency, temporal efficiency and flexibility and adaptability.

System of criteria and qualifications:

Criterion A. ease of implementation:
[1] It is very complex to implement the algorithm.
[2] It is neither very complex nor very easy to implement the algorithm.
[1] It is easy to implement the algorithm Criterion

B. Efficiency of the algorithm.

[1] Algorithms are prone to easily avoidable failures.
[2] Algorithms are prone to self-inflicted failures.
[3] Algorithms almost never fail in almost any situation.

Criterion C. Spatial efficiency.

[1] The algorithm requires a lot of memory space.
[2] The algorithm requires the necessary space in memory.
[3] The algorithm requires very little memory space.

Criterion D. Time efficiency.

[1] The algorithm requires a lot of execution time depending on the size of the input.
[2] The algorithm requires the necessary execution time depending on the input.
[3] The algorithm requires little execution time depending on the size of the input.

Criterion E. Flexibility and adaptability.
[1] The algorithm is very inflexible and adaptable.
[2] The algorithm is moderately flexible and adaptable.
[3] The algorithm is very flexible and adaptable

Santiago Hernández Saavedra A00382538
Juan Jose Barrera A00394876
Sergio Florez Sanabria A00396046

Evaluation system:

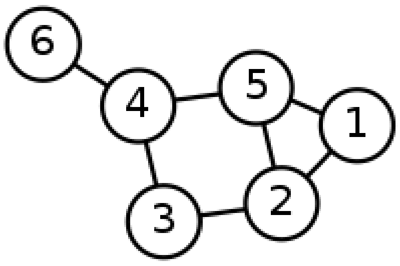
	Criterion A	**Criterion B**	**Criterion C**	**Criterion D**	**Criterion E**	**Total**
Dijkstra y Floyd-Warshall	1	2	3	2	2	10
Dijsktra and BFS	3	1	3	1	3	11

Selection:

Alternative 2, which uses the Dijkstra and BFS algorithms, stands out as the best option. It is easy to implement, efficient in terms of time and space, and flexible in its application. Dijkstra finds short paths between two nodes, while BFS is useful for exhaustively traversing graphs. That is, this alternative meets all the criteria evaluated, which makes it the ideal choice to solve the problem posed.

TAD´S

TAD Grafo



Abstract Object: Graph
Graph={Vertexes = <vertexes>, Edges<edges>}

{Inv:
***vertexes > 0**
***A graph must had at least a vertex.**
}

Santiago Hernández Saavedra A00382538
Juan Jose Barrera A00394876
Sergio Florez Sanabria A00396046

Primitive Operations:

Modifiers:

- **AddVertex: Graph X Vertex -> Graph**
- **AddEdge: Graph X Vertex1 X Vertex2 X Weight -> Graph**
- **DeleteVertex: Graph X (T valueVertex) ->Graph**
- **DeleteEdge: Graph X Edge –> Graph**

Analyzers:

- **BFS: Graph X VertexSource -> ArrayList<Vertexes>:**
- **DFS: Graph X VertexSource -> ArrayList<Vertexes>:**
- **Dijkstra: Graph X VertexSource X VertexDestiny -> ArrayList<Vertexes>**
- **Floyd-Warshall: ArrayList<Vertexes>**
- **Prim: Graph -> Arraylist<Vertexes>**
- **Kruskal: Graph ->Arraylist<Vertexes>**

AddVertex
It adds a new vertex in the graph.
{*pre*: {{Graph:...} ⋀ v} ⋀ v ∊ Vertex}
{*post*: Graph.vertexes = new v}

AddEdge
It adds an edge between two vertexes.
{*pre*: {Graph…}, {v1 ≠Nil, v2 ≠Nil}, {e between v1 ⋀ v2 = Nil } ⋀ v1, v2 ∊ Vertex ⋀ e ∊ Edge}
{*post*: Graph.edges = new e }

DeleteVertex
It deletes a vertex of the graph.
{*pre*:{{Graph:...} ⋀ v} ⋀ v ∊ Vertex}
{*post*: *remove* Graph.vertexes.v}

DeleteEdge
It deletes an edge of the graph.

Santiago Hernández Saavedra A00382538
Juan Jose Barrera A00394876
Sergio Florez Sanabria A00396046

{*pre*:{{Graph:...} ∧ e} ∧ e ∊ Edge}
{*post*: *remove* Graph.vertexes.e}

BFS
It explores a graph starting on a vertex and carries on with all its neighbors successively.
{*pre*: :{{Graph:...} ∧ vSource} ∧ vSource ∊ Vertex}
{*post*:{BF tree}}

DFS
It explores all the graph uniformly starting in the first vertex.
{*pre*: :{{Graph:...} ∧ vSource} ∧ vSource ∊ Vertex}
{*post*:{DF forest}}

Dijkstra
It returns the shortest path between a vertex and another.
{*pre*: :{{Graph:...}, v ∧ v'} ∧ v and v' ∊ Vertex}
{*post*: Path between v ∧ v' with less weight}

Floyd-Warshall
It returns the shortest path between all the pairs of vertexes.
{*pre*:{{Graph:...} ∧ vSource} ∧ vSource ∊ Vertex}
{*post*: Shortest distance between all pairs of vertices}

Prim
It returns the minimum spanning tree of a graph using the adjacent nodes with the least weight of the vertexes that have been already discovered.

Santiago Hernández Saavedra A00382538
Juan Jose Barrera A00394876
Sergio Florez Sanabria A00396046

{pre:{Graph:...}
{$post$: Minimum spanning tree of the graph}

Kruskal
It returns the minimum spanning tree of a graph using the edges with the least weight in all the graph.
{pre:{Graph:...}
{$post$: Minimum spanning tree of the graph}

THE PROBLEM TO SOLVE

1). Problematic situation:

The Fly Fast airline has hired us to put an end to a problem that has affected its reputation, its economy and its image towards its loyal customers. The problem that Fly Fast faces regarding the selection of suitable routes is a matter of great importance. This airline is dedicated to providing an efficient and high-quality service to its users, and the selection of routes is a critical aspect in customer satisfaction, since the customer is our main source of income.

The problem is that several airline employees do not know or distinguish all the multiple routes that are available, and this causes that appropriate routes to have not been selected, this can affect passengers in various aspects. Firstly, if routes with unnecessary layovers are selected, the total travel time of passengers may be increased and cause them unnecessary inconvenience. Second, if expensive routes are selected, this can negatively affect customers who are looking to save money and may opt for other airlines.

Fly Fast is committed to continually improving its service and ensuring that its users receive the best possible care. That is why we have been contracted to develop a system that can provide efficient and cost-effective routes for our passengers.

This problem can affect the reputation and profitability of Fly Fast, since users can opt for other travel options if they are not satisfied with the route options offered. That is why it is crucial to find an effective and efficient solution to this problem.

2) Analysis of the problem and functional requirements.

Santiago Hernández Saavedra A00382538
Juan Jose Barrera A00394876
Sergio Florez Sanabria A00396046

Client	**Airline Fly Fast**
User	*Fly-Fast* Tripulation
Functional requirements	R1. Allow the user to enter new destination cities R2. Allow the user to enter new connections or routes between cities R3. Allow the user to choose their priority between time or cost. R4. Show the user the route that takes the least time R5. Show the user the route with the lowest price.
Context of the problem	The problem is that several airline employees do not know or distinguish all the multiple routes that are available, and this causes that appropriate routes to have not been selected, this can affect passengers in various aspects. Firstly, if routes with unnecessary layovers are selected, the total travel time of passengers may be increased and cause them unnecessary inconvenience. Second, if expensive routes are selected, this can negatively affect customers who are looking to save money and may opt for other airlines.

R1.

Name or identifier	Allow the user to enter new destination cities
Summary	Allows you to add a new vertex to the graph. The vertex represents a destination city.

Santiago Hernández Saavedra A00382538
Juan Jose Barrera A00394876
Sergio Florez Sanabria A00396046

Inputs	**Input name**	**Datatype**	**Selection or repetition condition**
	city	String	that the data is different from null
General activities necessary to obtain the results	The required data must be valid.		
Result or Postcondition	Successful vertex creation.		
Outputs	**Output Name**	**Datatype**	**Selection or repetition condition**
	Msg	String	

R2.

Name or identifier	Allow the user to enter new connections or routes between cities

Santiago Hernández Saavedra A00382538
Juan Jose Barrera A00394876
Sergio Florez Sanabria A00396046

Summary	This requirement allows the system to create connections between two existing cities.		
Inputs	**Input name**	**Datatype**	**Selection or repetition condition**
	city1	String	That the city exists
	city2	String	That the city exists
	distance	double	That is not null or not equal to a type double
General activities necessary to obtain the results	That all the data is filled in correctly to avoid exceptions or unexpected results.		
Result or Postcondition	The creation of a new edge that symbolizes the journey from one city to another		
Outputs	**Output Name**	**Datatype**	**Selection or repetition condition**

	msg	String	

R3.

Name or identifier	Allow the user to choose their priority between time or cost.		
Summary	The user is asked to know if he prefers better time or better cost for his flight, thus knowing which alternative would seem better.		
Inputs	**Input name**	**Datatype**	**Selection or repetition condition**
	option	int	That is of type int, not null and represents a valid option.
	city1	String	That it is a city that exists in the system

	city2	String	That it is a city that exists in the system
General activities necessary to obtain the results	The user must mark a valid option to solve their problem.		
Result or Postcondition	The user will be shown the ideal path according to their requests.		
Outputs	**Output Name**	**Datatype**	**Selection or repetition condition**
	msg	String	

R4.

Name or identifier	Show the user the route that takes the least time

Santiago Hernández Saavedra A00382538
Juan Jose Barrera A00394876
Sergio Florez Sanabria A00396046

Summary	If the user wants to know the route that takes the least time, then check this option.		
Inputs	**Input name**	**Datatype**	**Selection or repetition condition**
General activities necessary to obtain the results	The user chose this option, and the method that finds the best path is called.		
Result or Postcondition	A message that shows you which is the best route, for the trip requested by our client.		
Outputs	**Output Name**	**Datatype**	**Selection or repetition condition**
	msg	String	

R5.

Santiago Hernández Saavedra A00382538
Juan Jose Barrera A00394876
Sergio Florez Sanabria A00396046

Name or identifier	Show the user the route with the lowest price.		
Summary	The user chooses the option that shows the route with the lowest price		
Inputs	**Input name**	**Datatype**	**Selection or repetition condition**
General activities necessary to obtain the results	The user chose this option and will be shown the cheapest route to reach their destination.		
Result or Postcondition	a message that shows you which is the most economical route to reach your destination.		
Outputs	**Output Name**	**Datatype**	**Selection or repetition condition**
	msg	String	