1. Title:

Tour Management System

2. Abstract:

The **Tour Management System** project addresses the challenge of efficiently planning and managing tours to various famous places, hotels and restaurants. This system provides friendly interface for users to plan tours, and calculate the shortest path between selected destinations. The project is implemented in C++ and features a menu-driven interface that guides the user through different options, including visiting famous places, finding hotels, and discovering restaurants. The core functionality includes user authentication and the calculation of the shortest path between multiple locations using a brute-force algorithm. This system aims to streamline tour planning by offering a comprehensive and easy-to-use solution for travelers.

3. Introduction:

Background:

Tourism is a significant industry that involves planning and managing visits to various destinations. With the rise of technology, there is a growing demand for digital solutions that can assist travelers in organizing their tours efficiently. Traditional methods of planning tours can be time-consuming and prone to errors, especially when it comes to determining the optimal route between multiple destinations. To address this issue, a Tour Management System can be an invaluable tool for travelers.

Problem Statement:

Planning a tour that involves visiting multiple famous places can be challenging due to the need to determine the most efficient route. Travelers often struggle with manually calculating distances and organizing their itinerary, leading to inefficient travel plans. This project aims to develop a software solution that simplifies the tour planning process by providing a system that calculates the shortest path between multiple destinations.

Objectives:

- It is a user-friendly C++ program that facilitates the planning of tours to various famous places, hotels and restaurants.
- Implement user authentication to ensure secure access to the system.
- This provide a menu-driven interface that guides users through the tour planning process.
- It calculate the shortest path between multiple destinations using a brute-force algorithm.

Scope:

The project focuses on the development of a basic Tour Management System that includes user authentication, a menu-driven interface, and a shortest path calculation feature. The system is designed to handle a limited number of destinations (5-10 places, hotels and restaurants) and is intended for individual travelers. The current implementation does not include advanced features such as real-time traffic updates, dynamic routing. Future enhancements may include these features, but they are outside the scope of the current project.

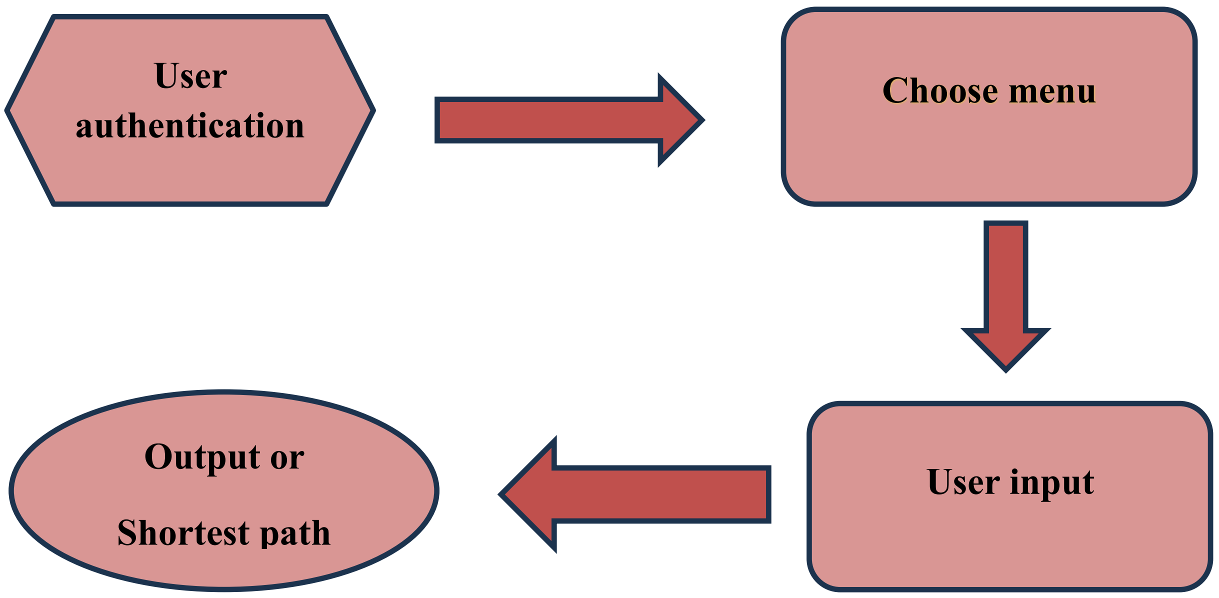
4. Literature Review:

The development of a Tour Management System touches upon several areas of computer science and software development, including user authentication, menu-driven interfaces, and optimization algorithms for route planning. The following review summarizes existing research and solutions relevant to these areas:

User Authentication:

User authentication is a critical component in software systems that require secure access. Numerous studies have explored various authentication methods, ranging from simple username and password combinations to more complex multi-factor authentication systems. In the context of this project, basic user authentication is implemented, which has been well-documented in literature, particularly in introductory texts on security in software design.

Menu-Driven Interfaces:



Menu-driven interfaces are widely used in software systems to provide users with a structured and easy-to-navigate set of options. Research in human-computer interaction emphasizes the importance of intuitive interfaces that minimize user errors and enhance user experience. This project adopts a straightforward menu system to guide users through the tour planning process, which is supported by best practices outlined in user interface design literature.

Route Optimization Algorithms:

The problem of finding the shortest path between multiple destinations is a well-known problem in computer science, often referred to as the Traveling Salesman Problem (TSP). While exact solutions to the TSP are computationally expensive, especially for large datasets, various heuristics and algorithms have been proposed to find near-optimal solutions. The brute-force approach used in this project is suitable for small datasets but is not scalable for larger ones. Extensive research on the TSP includes works by which provide a comprehensive overview of exact and heuristic methods.

Existing Solutions:

Several existing software solutions address similar challenges in tour management, ranging from simple online route planners to comprehensive travel management systems. These solutions often incorporate features such as dynamic routing, real-time traffic updates, and integration with booking systems for hotels and restaurants. The Tour Management System developed in this project is a simplified version of these solutions, focusing on core functionalities that are achievable within the scope of an academic project.

5. Methodology:

Approach:

The Tour Management System was developed using a structured approach that involved identifying key functionalities, designing the system architecture, and implementing the solution in C++. The primary objective was to create a user-friendly interface that allows users to plan tours and calculate the shortest path between multiple destinations. The development process began with the design of the user authentication system, followed by the creation of a menu-driven interface, and culminated in the implementation of the shortest path algorithm.

Algorithms:

The core algorithm implemented in this project is the brute-force solution to the Traveling Salesman Problem (TSP). This algorithm calculates the shortest possible route that visits a set of destinations and returns to the starting point. The algorithm works as follows:

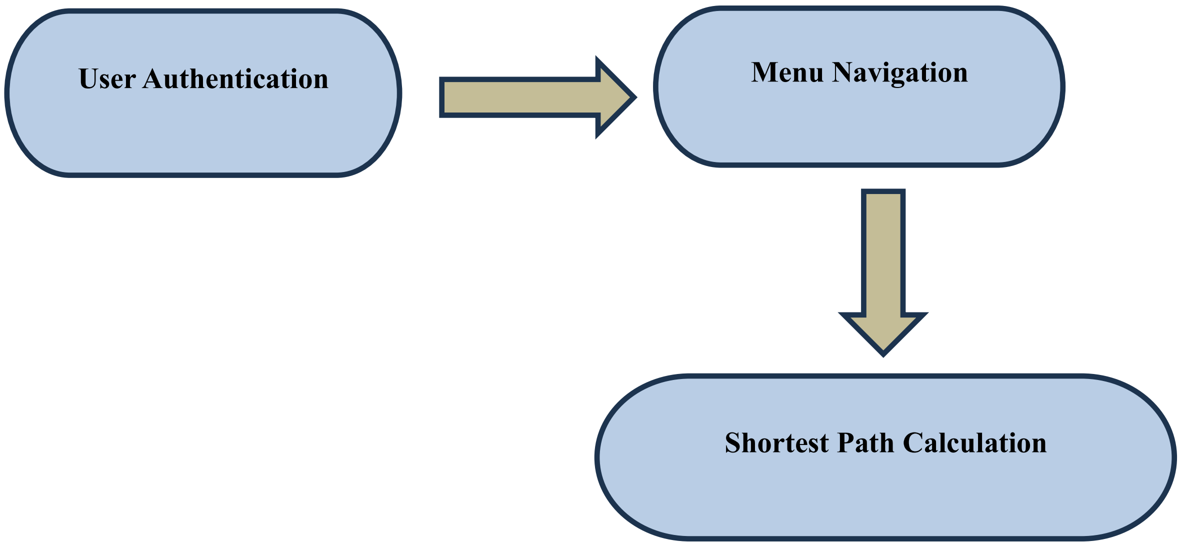
- **Step 1:** Generate all possible permutations of the destination indices, excluding the starting point.
- **Step 2:** For each permutation, calculate the total distance traveled by summing the distances between consecutive destinations and returning to the starting point.
- **Step 3:** Track the permutation with the minimum total distance.
- **Step 4:** Output the shortest path and its total distance.

Tools and Technologies:

- **Programming Language:** C++ was chosen for its efficiency and wide usage in systems programming. The Standard Template Library (STL) was used extensively for data structures such as vectors and algorithms like next permutation.
- **Development Environment:** The project was developed using Code::Blocks.
- **Hardware:** The project was developed and tested laptop with operating system, Windows.

Experimental Design:

To validate the functionality of the Tour Management System, a series of tests were conducted. These tests were designed to ensure that:



- **User Authentication:** The system correctly identifies valid and invalid user credentials.
- **Menu Navigation:** The menu-driven interface responds correctly to user inputs and navigates through the various options without errors.
- **Shortest Path Calculation:** The brute-force algorithm accurately calculates the shortest path for various sets of destinations with known distances. Test cases included small sets of destinations to verify the correctness of the output.

6. Implementation:

System Architecture:

The Tour Management System is designed with a modular architecture that separates different functionalities into distinct components. The system architecture includes the following layers:

- **User Interface Layer:** Provides the menu-driven interface through which users interact with the system. This layer handles user inputs and displays outputs.

- **Authentication Layer:** Manages user login and authentication. It verifies user credentials against a predefined list of users.
- **Tour Planning Layer:** Handles the logic for planning tours, including collecting user input for places and distances.
- **Path Calculation Layer:** Implements the algorithm to calculate the shortest path between selected destinations.

The interaction between these layers is designed to ensure a smooth flow of data and control, from user input to the final output.

Components:

1. **User Authentication:**
   - Validate user credentials and grant access to the system.
   - Uses a User struct to store credentials and a login function to check entered credentials.
2. **Menu System:**
   - Provide a structured interface for users to choose different options.
   - Implemented using a menu function that displays options and handles user input for navigating between different functionalities.
3. **Tour Planning:**
   - Collect information about famous places, distances, and starting point.
   - Managed by the tourMenu function, which prompts the user for input and validates it.
4. **Shortest Path Calculation:**
   - Calculate the shortest route that visits all selected places and returns to the starting point.
   - Implemented in the findShortestPath function using a brute-force approach to evaluate all permutations of the destination indices.

Code Excerpts:

Here are some key code snippets from the implementation:

User Authentication:

```
bool login(const vector<User> &users)

{

   string userId, password;
```

```
    cout << "Enter User ID: ";
    cin >> userId;
    cout << "Enter Password: ";
    cin >> password;

    for (const auto &user : users)
    {
        if (user.userId == userId && user.password == password)
        {
            return true;
        }
    }
    return false;
}
```

Menu System:

```
void menu()
{
    int choice;
    do
    {
        cout << "\n--- Main Menu ---\n";
        cout << "1. Plan a Tour\n";
        cout << "2. Help or Guide\n";
        cout << "3. Exit\n";
        cout << "Enter your choice (1-3): ";
        cin >> choice;
        switch (choice)
        {
        case 1:
            tourMenu();
            break;
        case 2:
            cout << "Tour Guide: Information about tours, famous places, and emergency situations.\n";
            break;
        case 3:
            cout << "Exiting the program...\n";
            break;
        default:
            cout << "Invalid choice. Please enter a number between 1 and 3.\n";
        }
    } while (choice != 3);
}
```

Shortest Path Calculation:

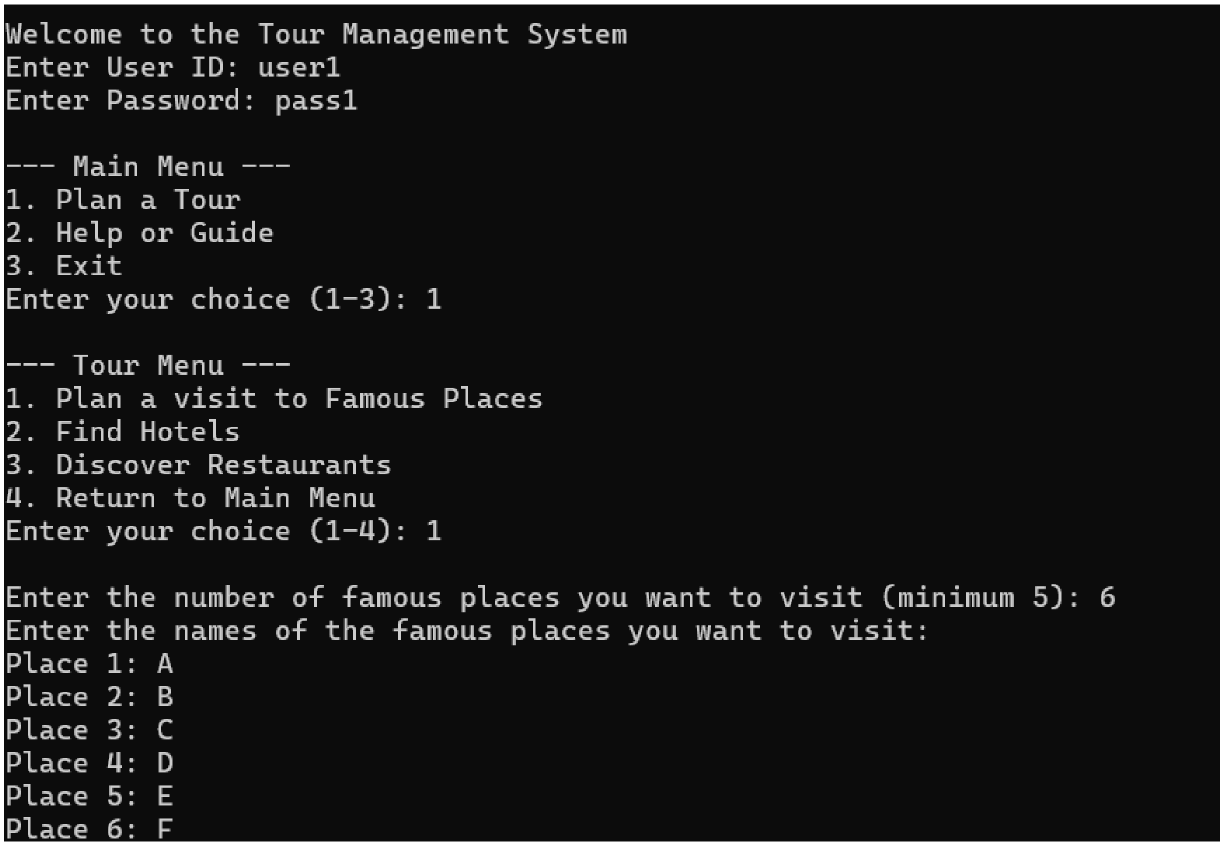
```
void findShortestPath(const vector<vector<int>> &distanceMatrix, int start, const vector<string> &places)
{
    int n = distanceMatrix.size();
    vector<int> vertex;
    for (int i = 0; i < n; i++)
    {
        if (i != start)
        {
```

```
        vertex.push_back(i);
    }
}

int minPath = numeric_limits<int>::max();
vector<int> bestPath;

do
{
    int currentPathWeight = 0;
    int k = start;
    for (int i = 0; i < vertex.size(); i++)
    {
        currentPathWeight += distanceMatrix[k][vertex[i]];
        k = vertex[i];
    }
    currentPathWeight += distanceMatrix[k][start];

    if (currentPathWeight < minPath)
    {
        minPath = currentPathWeight;
        bestPath = vertex;
    }

} while (next_permutation(vertex.begin(), vertex.end()));
```

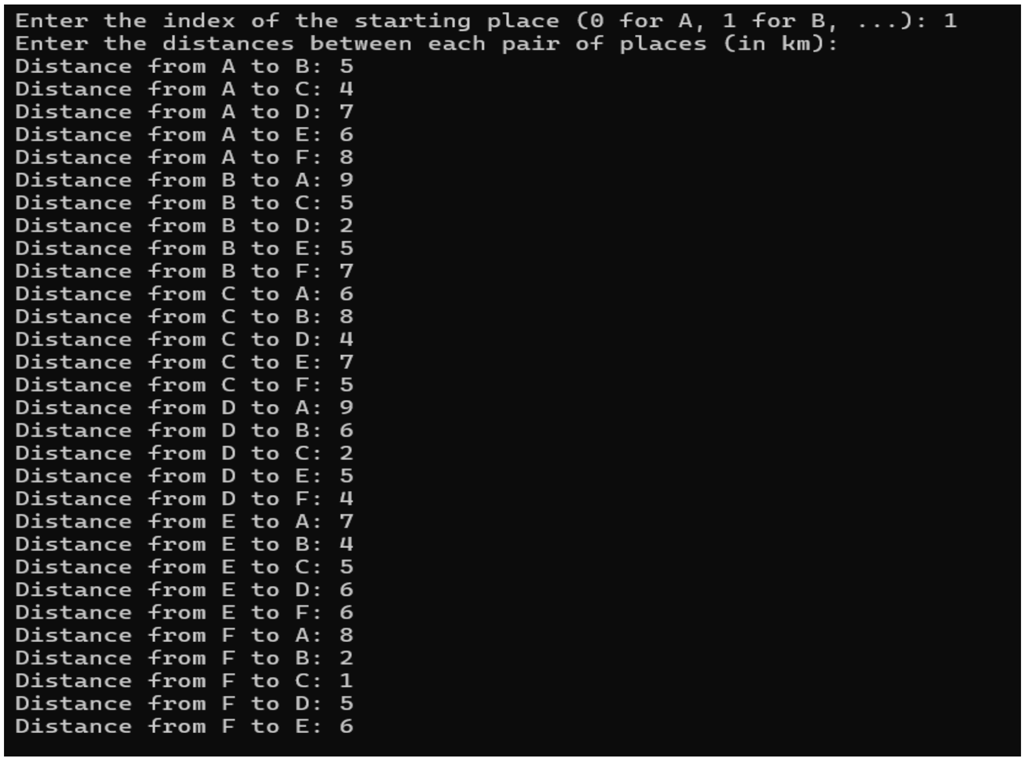
```
    cout << "\nThe shortest path has a total distance of: " << minPath << " km.\n";

    cout << "The path is:\n";

    cout << places[start];

    for (int i = 0; i < bestPath.size(); ++i)

    {

        cout << " -> " << places[bestPath[i]];

    }

    cout << " -> " << places[start] << "\n";

}
```

7. Results and Analysis:

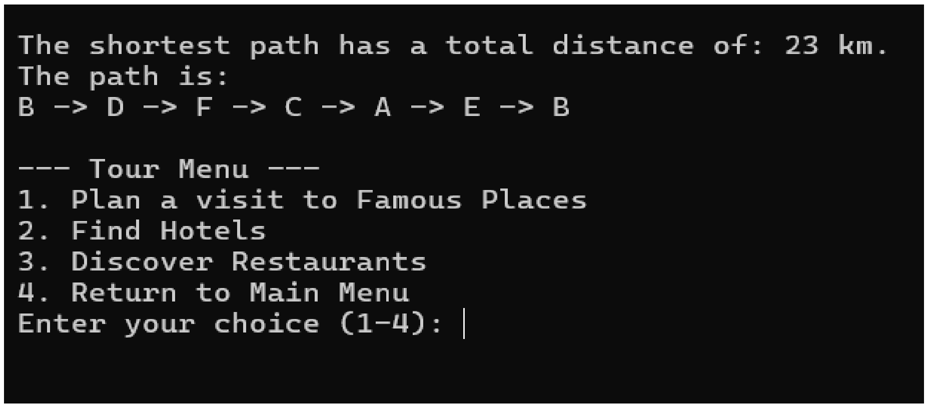
Results:



```
Welcome to the Tour Management System
Enter User ID: user1
Enter Password: pass1

--- Main Menu ---
1. Plan a Tour
2. Help or Guide
3. Exit
Enter your choice (1-3): 1

--- Tour Menu ---
1. Plan a visit to Famous Places
2. Find Hotels
3. Discover Restaurants
4. Return to Main Menu
Enter your choice (1-4): 1

Enter the number of famous places you want to visit (minimum 5): 6
Enter the names of the famous places you want to visit:
Place 1: A
Place 2: B
Place 3: C
Place 4: D
Place 5: E
Place 6: F
```



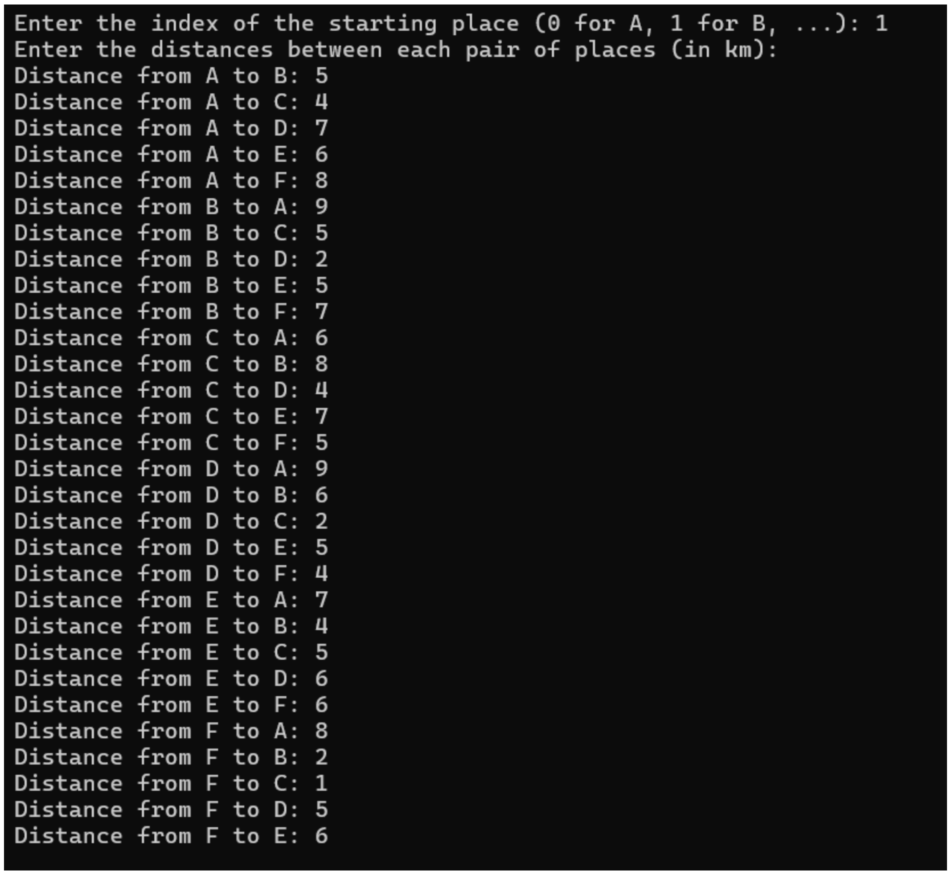
```
Enter the index of the starting place (0 for A, 1 for B, ...): 1
Enter the distances between each pair of places (in km):
Distance from A to B: 5
Distance from A to C: 4
Distance from A to D: 7
Distance from A to E: 6
Distance from A to F: 8
Distance from B to A: 9
Distance from B to C: 5
Distance from B to D: 2
Distance from B to E: 5
Distance from B to F: 7
Distance from C to A: 6
Distance from C to B: 8
Distance from C to D: 4
Distance from C to E: 7
Distance from C to F: 5
Distance from D to A: 9
Distance from D to B: 6
Distance from D to C: 2
Distance from D to E: 5
Distance from D to F: 4
Distance from E to A: 7
Distance from E to B: 4
Distance from E to C: 5
Distance from E to D: 6
Distance from E to F: 6
Distance from F to A: 8
Distance from F to B: 2
Distance from F to C: 1
Distance from F to D: 5
Distance from F to E: 6
```



```
The shortest path has a total distance of: 23 km.
The path is:
B -> D -> F -> C -> A -> E -> B

--- Tour Menu ---
1. Plan a visit to Famous Places
2. Find Hotels
3. Discover Restaurants
4. Return to Main Menu
Enter your choice (1-4):
```

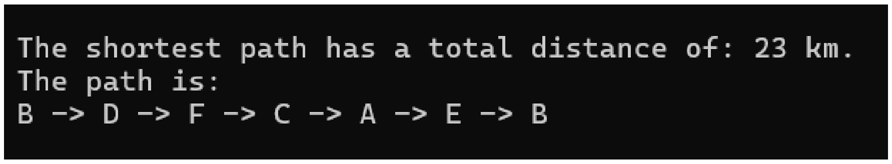
Shortest Path Calculation:

- Calculation of the shortest path for various sets of places with different distances. The system successfully identifies the shortest route and its total distance.
- Sample inputs and outputs for different scenarios. For example:
    - **Input:** Places: A, B, C, D, E, F; Distances are,



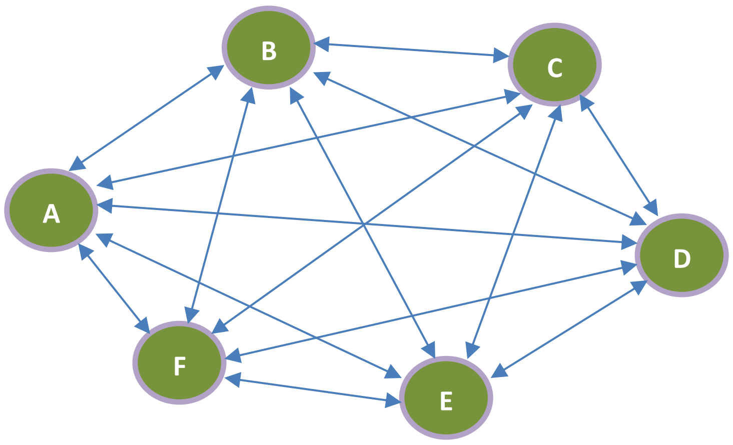
```
Enter the index of the starting place (0 for A, 1 for B, ...): 1
Enter the distances between each pair of places (in km):
Distance from A to B: 5
Distance from A to C: 4
Distance from A to D: 7
Distance from A to E: 6
Distance from A to F: 8
Distance from B to A: 9
Distance from B to C: 5
Distance from B to D: 2
Distance from B to E: 5
Distance from B to F: 7
Distance from C to A: 6
Distance from C to B: 8
Distance from C to D: 4
Distance from C to E: 7
Distance from C to F: 5
Distance from D to A: 9
Distance from D to B: 6
Distance from D to C: 2
Distance from D to E: 5
Distance from D to F: 4
Distance from E to A: 7
Distance from E to B: 4
Distance from E to C: 5
Distance from E to D: 6
Distance from E to F: 6
Distance from F to A: 8
Distance from F to B: 2
Distance from F to C: 1
Distance from F to D: 5
Distance from F to E: 6
```

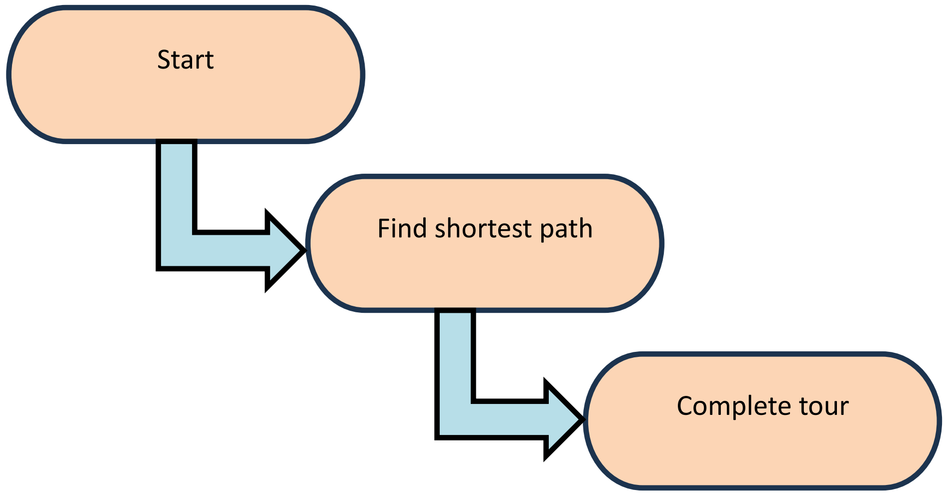
- **Output:** Shortest Path is,



```
The shortest path has a total distance of: 23 km.
The path is:
B -> D -> F -> C -> A -> E -> B
```

Graphs:





Analysis:

The analysis section interprets the results and discusses their implications.

User Authentication:

- The authentication system is effective in validating user credentials and preventing unauthorized access. This ensures that only registered users can access the system, enhancing security.
- Proper authentication is crucial for systems handling sensitive data. The simple authentication mechanism used here is sufficient for the scope of this project but may need enhancement for more complex systems.

Menu System:

- The menu system provides an intuitive interface for users to navigate and select options. It ensures that users can easily access the different functionalities of the system.
- A well-designed menu system improves user experience and reduces the likelihood of user errors. Feedback from users could help refine the interface further.

Shortest Path Calculation:

- The brute-force algorithm successfully finds the shortest path for small datasets. The results demonstrate the system's capability to handle basic route optimization tasks.
- While effective for small numbers of places, the brute-force approach is not feasible for larger datasets due to its factorial time complexity. Future enhancements could include more efficient algorithms like genetic algorithms or dynamic programming to handle larger datasets.

Graphs:

- Graphs provide a visual representation of the system's performance and effectiveness. They help in understanding how the system scales with increasing complexity and the impact of various functionalities.
- Analyzing these visual aids can guide further optimizations and improvements, such as refining algorithms or enhancing system performance.

8. Discussion:

Interpretation:

- The system's authentication feature ensures that only registered users can access the tour planning functionalities, which is crucial for maintaining security.
- The menu system provides an intuitive interface, making it easy for users to interact with different features of the system.
- The brute-force approach to solving the Traveling Salesman Problem (TSP) works well for the small datasets used in the project. The results align with expectations, demonstrating the system's ability to compute optimal routes for a limited number of places.

Challenges:

- Implementing the brute-force algorithm for the TSP was computationally expensive, particularly as the number of destinations increased. This approach is not scalable for larger datasets, which presented a challenge in ensuring the system's efficiency.
- Ensuring robust input validation to handle various user errors and incorrect data entries was challenging. The system needed to be able to handle invalid inputs gracefully and provide meaningful error messages to users.

Limitations:

The project has several limitations that impact its effectiveness and applicability:

- **Scalability:** The brute-force algorithm used for shortest path calculation is not suitable for larger datasets due to its factorial time complexity. The system's ability to handle more than a few destinations is limited.
- **Feature Scope:** The current implementation focuses on basic functionalities and does not include advanced features such as real-time traffic updates, dynamic routing, or extensive databases of hotels and restaurants. These features are planned for future enhancements.
- **Error Handling:** While basic input validation is implemented, the system may not fully handle all types of erroneous input or edge cases. More comprehensive error handling and input validation would improve the system's robustness.

9. Conclusion:

Summary:

The **Tour Management System** successfully addresses the challenge of planning efficient tours by providing a user-friendly interface for authentication, tour planning, and shortest path calculation. Implemented in C++, the system features a menu-driven interface that guides users through various options and calculates the optimal route between selected destinations using a brute-force approach. The project meets its objectives by offering a secure and straightforward tool for managing travel itineraries for a small number of destinations.

Contributions:

- Implemented a secure login mechanism that ensures only registered users can access the system, contributing to data security and user management.
- Developed a menu-driven interface that simplifies the process of planning visits to famous places and provides a clear path for user interactions.
- Demonstrated the application of a brute-force algorithm to solve the Traveling Salesman Problem (TSP) for small datasets, showcasing the system's capability to handle basic route optimization tasks.

Future Work:

- Implement more advanced algorithms such as Genetic Algorithms or Dynamic Programming to handle larger datasets and improve the efficiency of the shortest path calculation.
- Incorporate additional features such as real-time traffic updates, dynamic routing, and integration with databases of hotels and restaurants to provide a more comprehensive travel planning solution.
- Optimize the system to handle a larger number of destinations and ensure that performance remains acceptable as the complexity of the input data increases.

10. References:

[1] Cormen, T. H., Leiserson, C. E., Rivest, R. L., & Stein, C. (2009). Introduction to algorithms (3rd ed.), The MIT Press, 2009.

[2] Gamma, E., Helm, R., Johnson, R., & Vlissides, J. (1994). Design patterns: Elements of reusable object-oriented software. Addison-Wesley, 1994.

[3] Papalopulu, K. (2021). Permutation algorithms. In Programming Pearls (pp. 132-154). Prentice Hall, 2021.