Aircraft Route Optimization with Genetic Algorithm

The provided Flask application is a sophisticated route optimizer that employs genetic algorithms to enhance flight routes based on user-input starting locations and aircraft types. This application integrates key libraries and components to facilitate its functionality.

Components & Libraries Used:

Flask: For web application creation.

render_template, request, jsonify, redirect, url_for: For web requests, template rendering, and redirection.

random, numpy, pandas: Facilitating random value generation, numerical calculations, and data manipulation.

CSRFProtect, FlaskForm, SelectField, SubmitField: Necessary for form creation and guarding against CSRF attacks.

Flask Application Setup:

The Flask application is established and secured with a secret key for encryption. Additionally, CSRF protection is enforced for enhanced security.

Form Setup:

The application reads data from the airportDataset.csv file to populate a list of cities and configures the RoutingForm class. This form incorporates fields for selecting the starting location and aircraft type.

Route Definitions:

Home Route (/): Renders the primary form (index.html) enabling user input for the starting location and aircraft type.

Optimize Route (/optimize_route): Processes POST requests, reads datasets containing airport and weather information, applies genetic algorithms to optimize flight routes, and simulates weather changes.

Route Optimizer (/route_optimizer): Handles form submissions, validates input, and returns a JSON response containing the submitted starting location and aircraft type.

Best Route Map (/best_route_map), Show Results (/show_results): Displays the optimized route and associated information.

Optimization and Simulation:

The optimize_route function executes the optimization process using genetic algorithms. It generates random routes, calculates fitness scores, performs crossover, mutation, and computes total distance and fuel consumption.

Weather simulation involves modifying wind speed and direction over multiple time steps to simulate realistic weather changes.

Application Flow:

User Input: Users access the homepage and input the starting location and aircraft type.

Processing: Upon form submission, the application processes the data using genetic algorithms and weather simulation.

Results Display: The optimized route and relevant details are exhibited on the results page (results.html).

Route Visualization: Users can view the best route map and additional details on respective routes (best_route_map.html).

Conclusion:

This Flask application excels in handling form submissions, employing genetic algorithms for optimization, and simulating weather changes to ascertain the most efficient flight route. Leveraging a blend of web development, optimization algorithms, and data simulation, it successfully addresses practical challenges in aviation.