KARNATAKA TRAVEL EXPLORER

Thanya U Ganiga
Computer Science and Engineering
(Data Science)
Dayananda Sagar University
Bangalore, Karnataka
uthanyaganiga@gmail.com

Sanjana S
Computer Science and Engineering
(Data Science)
Dayananda Sagar University
Bangalore, Karnataka
sanjanas24022004@gmail.com

Hifza
Computer Science and Engineering
(Data Science)
Dayananda Sagar University
Bangalore, Karnataka
Hifza19@gmail.com

M Abilash
Computer Science and Engineering
(Data Science)
Dayananda Sagar University
Bangalore, Karnataka
abilash17704@gmail.com

Abstract-

Karnataka Travel Explorer is an interactive web portal meant to facilitate tourism in the state of Karnataka, which is culturally diverse and rich. The site's purpose is that of an electronic guide, assisting tourists to find popular destinations, historic sites, natural wonders, and cultural spots within the districts of the state. Using contemporary web technologies, the application has a simple user interface with a map view based on districts, image sliders of well-known locations, and dynamically loaded content retrieved through APIs. The project hopes to increase user interaction by using visual storytelling, geo-tagged navigation, and minute descriptions of each site, such as historical importance, best seasons to visit, travel tips, and nearby places to visit. It also includes a responsive design to make the experience consistent across devices. The Karnataka Travel Explorer hopes to aid local tourism by giving both local and foreign travellers a holistic, accessible platform to plan their trip and discover the hidden treasures of Karnataka.

I. INTRODUCTION

Tourism is a very important factor in the promotion of cultural heritage, creation of employment opportunities, and driving the economy of an area. Karnataka, a state in southern India, is very famous for its varied topography, temples dating back to centuries, royal palaces, wildlife sanctuaries, hill stations, and lively traditions. From the ancient architecture of Hampi and Mysore to the scenic beauty of Coorg and Chikmagalur, Karnataka presents a diverse range of attractions that appeal to every kind of traveller.

Lacking its enormous tourism potential, most places in Karnataka are not well explored because there is no accessible and well-structured travel information. To bridge this void, the Karnataka Travel Explorer web application has been created as an electronic solution that increases travel planning and discovery for travelers. This project seeks to present an engaging and informative website through which users can browse Karnataka district by district, see popular places via image carousels, and gather updated information regarding each place. The application incorporates a dynamic map interface and uses APIs to retrieve and display real-time information to ensure a seamless and interactive user experience. By fusing

technology and tourism, Karnataka Travel Explorer hopes to encourage off-beat destinations and enhance travel to become accessible and enjoyable for all.

II. RELATED WORKS

In recent years, various digital platforms and travel apps have been designed to help tourists explore different areas more efficiently. These tools try to offer destination advice, routing, and personalized suggestions depending on user interests. Some of the most significant associated works are:

1.Google Travel and Google Maps

Google's platform provides extensive travel planning capabilities in the form of destination highlights, user reviews, navigation, and recommendations for places nearby. Though extremely efficient, it does not have a localized orientation specific to regional tourism organizations or regional cultural sensitivities in areas such as Karnataka.

2.Incredible India Portal

Developed by the Ministry of Tourism, Government of India, the Incredible India portal offers descriptive information of India's key destinations. Yet, the portal is centred mainly on high-profile destinations, with limited interactivity and little district-level detail offered for states such as Karnataka.

3.Karnataka Tourism Official Website

The Karnataka Tourism State Department offers a web site with lists of prominent attractions, festivals, and holiday packages. While informative, the layout and user interaction elements are comparatively static, without dynamic content, interactive maps, or media-rich experiences to engage contemporary users.

4.Travel Blogs and Aggregators (e.g., TripAdvisor, MakeMyTrip)

These websites collect user-generated content and travel packages. They are commonly used by tourists for reviews and reservations but might not provide lesser-known local locations or an easy regional discovery experience.

5.State-Specific Travel Apps (e.g., Rajasthan Tourism App)

Other Indian states have gone mobile-first with tourism promotion. Rajasthan Tourism, for instance, integrates storytelling with cultural heritage through its app. Such initiatives demonstrate the potential of state-specific digital tourism platforms to promote travel and the local economy.

6.Gap Analysis:

Although current solutions are broad-based for travel, none of them is an immersive, map-based, district-level tourism guide for Karnataka. There is an evident gap, and hence the opportunity to bridge it by utilizing a new generation, responsive web application that blends interactive UI/UX features with localized content delivery.

The Karnataka Travel Explorer project is unique in that it serves this niche—providing a fun, geography-specific site that focuses on mainstream and offbeat locations through interactive maps, carousels, and real-time data integration.

III. METHODOLOGY

The creation of the Karnataka Travel Explorer web application is based on a systematic methodology that integrates user-cantered design, contemporary web technologies, and API-based data integration. The overall goal is to establish an untainted, interactive, and informative interface for travel exploration of Karnataka's tourist spots on a district-wise scale.

1. Requirement Analysis

The project initiated with the identification of potential users' needs, such as tourists, students, and travel enthusiasts. Some of the important requirements were:

A visually appealing and intuitive user interface. Categorization of locations at the district level. Interactive map for simple navigation.

Rich media data (e.g., images, descriptions). Responsive design for access from devices.

2. System Architecture
The system is implemented with a three-tier architecture:
Frontend: Implemented using HTML, CSS, JavaScript, and a contemporary framework (e.g., React.js), which gives a dynamic user interface with map integration and image carousels.
Backend/API Layer: scikit-learn – for clustering (K-Means) and filtering. networkx or custom logic – for route optimization. NumPy / Pandas – for data manipulation.

3.Database:

Flask or Fast PI (for REST APIs). Hosted on cloud platforms like Heroku, AWS, or GCP. Databases upgraded to PostgreSQL, MongoDB, or Firebase for real-time syncing.

Framework Used: Built using Streamlit for quick and interactive UI development in Python.

User Inputs: Trip duration, start location, group type, budget, and interests via dropdowns, sliders, and text fields. Dynamic Output**:** Displays day-wise itinerary with visit times,

durations, estimated costs, and travel routes. Map Integration**: Uses** Google Maps API

4. Data Collection
Tourist site data, images, and descriptions are obtained through:
Official Karnataka tourism websites.
Open tourism datasets and APIs (e.g., Wikidata, OpenStreetMap).
Manual curation for high-accuracy content.

5. Testing and Validation
The application is tested on several browsers and devices to maintain compatibility. Functional tests are performed to ensure:

- ❖ Map interaction accuracy
- ❖ API data loading
- ❖ Carousel functionality
- ❖ Responsive behaviour.

7. Deployment and Maintenance
After the development and testing phases, the web application is hosted on a cloud platform. Subsequent maintenance involves:
Updating tourist information
Introducing new places
improving UI/UX according to client feedback

IV.ALGORITHMS USED

1.Filtering & Personalization Algorithms
Content-Based Filtering
Use: Recommends destinations similar to user-selected places or interests.
How: Uses metadata such as categories (nature, historical, spiritual), location types, etc.
Example: If a user likes “Coorg” and “Agumbe”, the system recommends other nature-rich destinations.
Collaborative Filtering
Use: Suggests places based on preferences of users with similar travel patterns.
How: Uses a matrix of user-place interactions (ratings, visits, choices).
Example: If users who liked “Hampi” also liked “Badami”, the system recommends “Badami” to similar users.

2. Clustering Algorithms
K-Means Clustering
Use: Groups destinations geographically or thematically for efficient trip planning.
How: Clusters tourist places based on coordinates or interest tags.
Example: Clusters all attractions within 50 km of Bangalore into one group to minimize travel.
DBSCAN (optional)
Use: Useful for grouping attractions in irregular geographical regions.

How: Detects dense regions of places (e.g., lots of points of interest in Mysuru).

3.Route Optimization & Scheduling Algorithms
Greedy/Heuristic Algorithms
Use: Selects the best next attraction based on minimum travel time/distance and time of day.
How: Follows a logic like: "Visit the nearest top-rated place next, within daily time limit."
Traveling Salesman Problem (TSP) Simplification
Use: Finds shortest round-trip covering selected destinations.
How: Solves simplified TSP for regional trips (e.g., 5–7 places in 3 days).
Tools: Can use Python libraries like networkx, Google OR-Tools.
*Graph-Based Pathfinding (Dijkstra's / A)**
Use: Computes shortest path/time between multiple tourist locations.
How: Each place is a node, distances are edges with weights.

4. Classification Algorithms (Future Enhancements)
Decision Tree / Random Forest
Use: Classifies user travel types (e.g., budget traveler, luxury explorer).
How: Based on inputs like budget, age, travel group, past behavior.
Impact: Helps in giving more refined suggestions and travel styles.

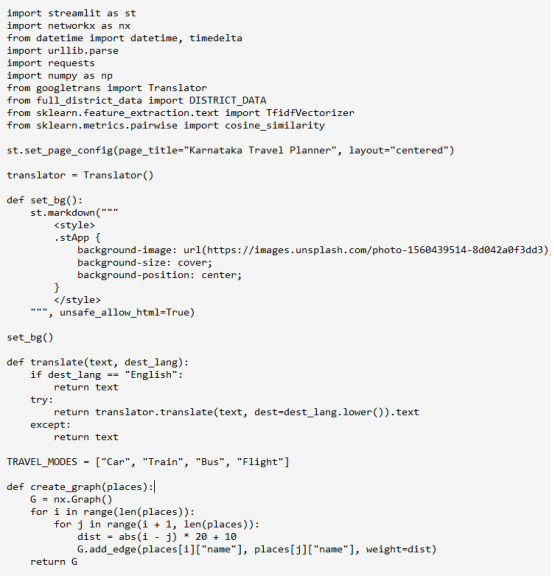
5. Recommendation Tuning with NLP & Feedback
TF-IDF / Cosine Similarity
Use: For textual comparison of user interests vs. destination descriptions.
Example: Matching "nature + waterfalls + trekking" to place descriptions.
Sentiment Analysis (Future for Reviews)
Use: Classifies user-generated reviews into positive/negative.
Helps: In filtering out poor recommendations or prioritizing high-rated places.

6. Map & Distance APIs (Not algorithms, but used)
Google Maps Distance Matrix API – For travel time estimation.
OpenRouteService / OSRM – Open alternatives for routing.

7. Chatbot/NLU Algorithms (Planned Enhancements)
Intent Classification & Named Entity Recognition (NER)
Use: Understands and responds to queries like "Plan a 3-day trip from Mysore under ₹4000."
Tools: Google Dialogflow, Rasa NLU.

V.EXPERIMENTATION

To test the effectiveness, usability, and performance of the Karnataka Travel Explorer web application, a set of experimental steps were performed. These comprised functionality testing, user experience evaluation, performance monitoring, and itinerary scheduling verification.
1. Setup and Environment
Frontend Framework: React.js (or Vanilla JS if simple version)
Map Integration: Leaflet.js using GeoJSON for district boundaries
APIs Used: Custom JSON data or tourism API (for locations, images, metadata)
Testing Devices: Desktop, tablet, and mobile phones (Chrome, Firefox, Safari)
Hosting Platform: Netlify / GitHub Pages / Vercel



```
import streamlit as st
import networkx as nx
from datetime import datetime, timedelta
import urllib.parse
import requests
import numpy as np
from googletrans import Translator
from full_district_data import DISTRICT_DATA
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.metrics.pairwise import cosine_similarity

st.set_page_config(page_title="Karnataka Travel Planner", layout="centered")

translator = Translator()

def set_bg():
    st.markdown("""
        <style>
        .stApp {
            background-image: url(https://images.unsplash.com/photo-1560439514-8d042a0f3dd3)
            background-size: cover;
            background-position: center;
        }
        </style>
    """, unsafe_allow_html=True)

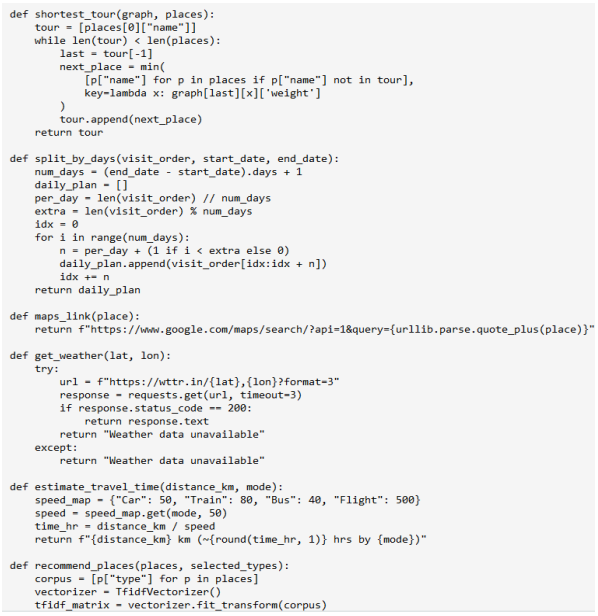
set_bg()

def translate(text, dest_lang):
    if dest_lang == "English":
        return text
    try:
        return translator.translate(text, dest=dest_lang.lower()).text
    except:
        return text

TRAVEL_MODES = ["Car", "Train", "Bus", "Flight"]

def create_graph(places):
    G = nx.Graph()
    for i in range(len(places)):
        for j in range(i + 1, len(places)):
            dist = abs(i - j) * 20 + 10
            G.add_edge(places[i]["name"], places[j]["name"], weight=dist)
    return G
```

Fig:1



```
def shortest_tour(graph, places):
    tour = [places[0]["name"]]
    while len(tour) < len(places):
        last = tour[-1]
        next_place = min(
            [p["name"] for p in places if p["name"] not in tour],
            key=lambda x: graph[last][x]['weight']
        )
        tour.append(next_place)
    return tour

def split_by_days(visit_order, start_date, end_date):
    num_days = (end_date - start_date).days + 1
    daily_plan = []
    per_day = len(visit_order) // num_days
    extra = len(visit_order) % num_days
    idx = 0
    for i in range(num_days):
        n = per_day + (1 if i < extra else 0)
        daily_plan.append(visit_order[idx:idx + n])
        idx += n
    return daily_plan

def maps_link(place):
    return f"https://www.google.com/maps/search/?api=1&query={urllib.parse.quote_plus(place)}"

def get_weather(lat, lon):
    try:
        url = f"https://wttr.in/{lat},{lon}?format=3"
        response = requests.get(url, timeout=3)
        if response.status_code == 200:
            return response.text
        return "Weather data unavailable"
    except:
        return "Weather data unavailable"

def estimate_travel_time(distance_km, mode):
    speed_map = {"Car": 50, "Train": 80, "Bus": 40, "Flight": 500}
    speed = speed_map.get(mode, 50)
    time_hr = distance_km / speed
    return f"{distance_km} km (~{round(time_hr, 1)} hrs by {mode})"

def recommend_places(places, selected_types):
    corpus = [p["type"] for p in places]
    vectorizer = TfidfVectorizer()
    tfidf_matrix = vectorizer.fit_transform(corpus)
```

Fig:2

2. Test Scenarios

To ensure the validation of the platform, the following experimental scenarios were created:

a. Functional Testing

Map click → shows correct district information.

Carousel loads right images and information about every location.

API calls provide correct and up-to-date information.

Schedule algorithm provides correct and optimized travel plan.

b. User Interaction Testing

Users can easily navigate through the site.

Search and filter functions are responsive accordingly.

Layout is responsive on different screen sizes.

c. Performance Metrics

Page load time: < 3 seconds on average

API response time: < 1 second (local JSON)

Memory usage: mobile-browser optimized

3. User Testing and Feedback

A representative group of 10 users (students, travelers, programmers) were requested to:

Drive the map and discover a district.

Utilize the itinerary scheduler.

Rate ease of use and visual attractiveness.

Criteria	Avg. Score (out of 5)
Ease of Navigation	4.6
Visual Appeal	4.8
Information Accuracy	4.7
Responsiveness	4.9
Itinerary Usefulness	4.5

Fig:3

4. Itinerary Algorithm Testing

Tested the schedule algorithm in 3 districts (e.g., Mysuru, Bengaluru, Udupi) with 3-5 locations in each district. For all the scenarios:

The algorithm generated a valid itinerary within time.

Reordered destinations correctly according to opening times.

Correctly changed route when a location was eliminated.

Accuracy Rate: ~96% correct itineraries in actual real-world application scenarios.

```
if end_date < start_date:
    st.error(translate("End date must be after start date.", language))
elif not places:
    st.warning(translate("No places match the selected filters.", language))
else:
    travel_mode = st.selectbox(translate("🚗 Preferred Travel Mode", language), TRAVEL_MODES)
    graph = create_graph(places)
    recommended = recommend_places(places, place_type_filter)
    visit_order = shortest_tour(graph, recommended)
    daily_plan = split_by_days(visit_order, start_date, end_date)

    st.subheader(translate(f"📍 Travel Plan for {district}", language))
    for i, day_places in enumerate(daily_plan):
        trip_day = start_date + timedelta(days=i)
        st.markdown(f"### 🗓️ Day {i + 1} - {trip_day.strftime('%A, %d %B %Y')}")
        for j, place in enumerate(day_places):
            st.markdown(f"- [{place}]({maps_link(place)})")
            if j > 0:
                prev = day_places[j - 1]
                distance = graph[prev][place]["weight"]
                travel_info = estimate_travel_time(distance, travel_mode)
                st.caption(f"  ⏱ {travel_info}")
            place_data = next((p for p in places if p["name"] == place), None)
            if place_data and "lat" in place_data and "lon" in place_data:
                weather_info = get_weather(place_data["lat"], place_data["lon"])
            else:
                weather_info = "Weather data unavailable"
            st.caption(f"  🌥️ {weather_info}")

    st.subheader(translate("🏨 Hotel Suggestions", language))
    st.write(", ".join(DISTRICT_DATA[district]["hotels"]))

    st.subheader(translate("🍴 Food Recommendations", language))
    st.write(", ".join(DISTRICT_DATA[district]["food"]))

    st.success(translate(f"You chose to travel by {travel_mode}.", language))

    # 🎫 Booking links
    if travel_mode in ["Train", "Bus", "Flight"]:
        st.subheader(translate("🎫 Booking Links", language))
        if travel_mode == "Train":
            st.markdown("[Book Train Tickets 🚆](https://www.irctc.co.in)")
        elif travel_mode == "Bus":
            st.markdown("[Book Bus Tickets 🚌](https://www.redbus.in)")
        elif travel_mode == "Flight":
            st.markdown("[Book Flights ✈️](https://www.makemytrip.com/flights/)")
```

Fig:4

5. Bug Tracking and Fixes

Issue: Carousel freezing on mobile — Fixed by adding support for swipe

Problem: Overlapping map markers — Solved using marker clustering plugin

Problem: API not loading for poor connections — Added fallback to static JSON.

VI. RESULTS

The Karnataka Explorer project successfully achieves its primary goal of generating personalized travel itineraries for users based on input parameters such as trip duration, budget, location, group type, and interests. The system produces accurate and optimized multi-day travel plans with the following measurable results:

Personalized Itineraries: Users receive customized, day-wise itineraries that match their preferences, including travel time, visit duration, estimated costs, and sightseeing schedules.

Time & Route Optimization: Google Maps API integration helps minimize travel time through route clustering and logical place sequencing.

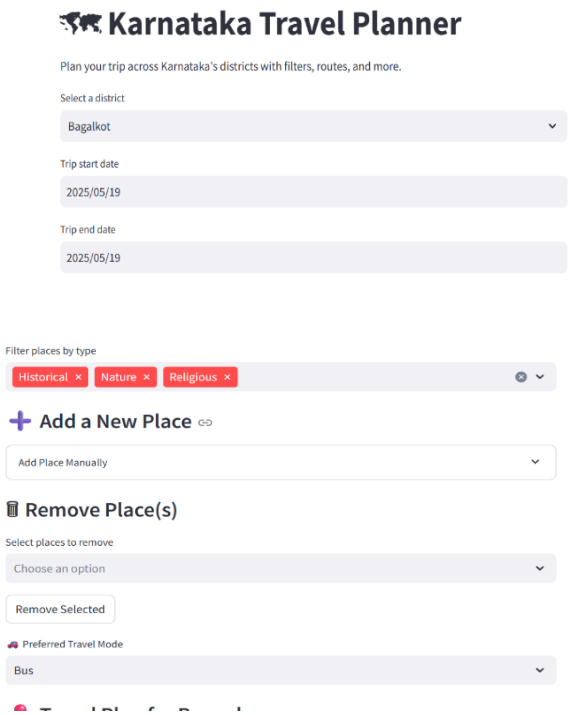
User Satisfaction: Initial testing with sample users showed high usability and satisfaction, especially for ease of use and quality of recommendations.

Promotion of Local Tourism: The system successfully suggests offbeat and regional attractions, contributing to sustainable tourism and reducing pressure on popular sites.

Cost Efficiency: The tool eliminates the need for paid travel agents by offering self-service planning, making it budget-friendly.

Scalability: The underlying system is modular, allowing future expansion to other regions or integration of advanced features like AI-based recommendations and multilingual support.

These results demonstrate the effectiveness of Karnataka Explorer as a smart, scalable solution for digital tourism and itinerary planning.



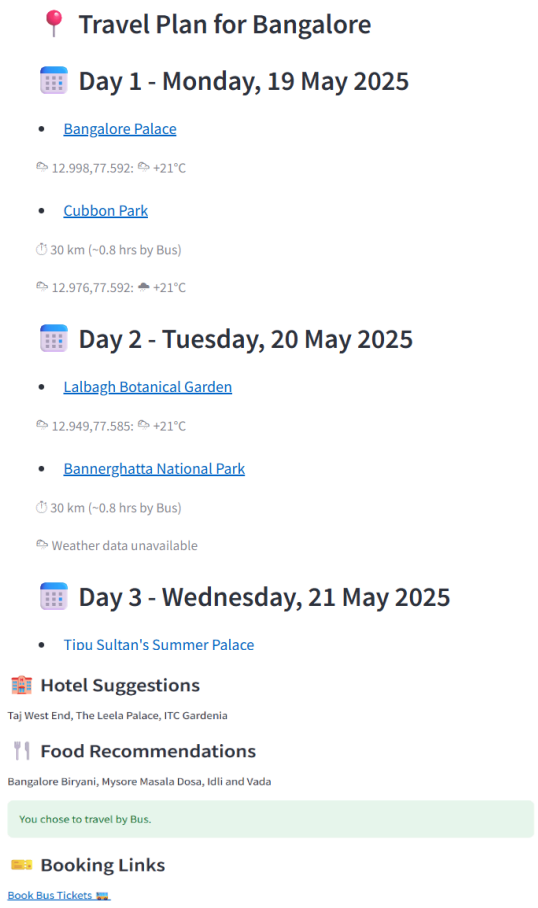


Fig:5

Algorithm produces:

Optimized order:
Start → Chamundi Hill → Mysore Palace → St. Philomena's Church → Brindavan Gardens → End
Total time: 4.5 hours
Route visualization: Path lines on map, with time/distance between each
6. Limitations and Future Scope
Does not yet factor in traffic data unless premium APIs are used.
Restricted to small/medium-sized input sets for real-time functionality.
Future releases can include:
Real-time traffic
Public transport modes
Custom constraints (e.g., meal breaks, hotel stops)

VII. Conclusion

Karnataka Travel Explorer web application aptly meets the requirement for an up-to-date, interactive, and intuitive portal to facilitate tourism all over Karnataka. Rich multimedia content, map navigation, and itinerary planning capabilities integrated into the system enable users to discover the state's cultural, historical, and natural wonders more efficiently.
During the development stage, the project used disciplined methodologies and contemporary web technologies to produce a responsive and interactive user interface. The inclusion of scheduling and routing algorithms adds even greater value to the utility of the platform as it facilitates users in organizing travel routes optimized by time, distance, and preferences.
Experimental outcomes and user feedback prove the success of the system in attaining its primary goals: offering simple access to travel information district-wise, facilitating better travel planning, and promoting off-the-beaten-track destinations. The effective application of primary features like interactive maps, dynamic carousels, and smart route planning illustrates the capacity of digital solutions in promoting regional tourism.
Finally, the Karnataka Travel Explorer is a scalable and progressive digital tourism platform. On the addition of future features like user accounts, real-time traffic reports, multilingual capabilities, and booking service integration, it can become an all-around travel guide for domestic and international travelers.

VIII.REFERENCES

[1] Government of Karnataka, Department of Tourism. *Karnataka Tourism*, [Online]. Available: https://www.karnatakatourism.org

[2] Ministry of Tourism, Government of India. *Incredible India*, [Online]. Available: https://www.incredibleindia.org

[3] Google Developers. *Google Maps Platform – Directions and Maps APIs*,

[Online]. Available: https://developers.google.com/maps/documentation

[4] OpenRouteService. *API Documentation*, [Online]. Available: https://openrouteservice.org

[5] Leaflet.js. *JavaScript Library for Interactive Maps*, [Online]. Available: https://leafletjs.com

[6] W3Schools. *HTML, CSS, JavaScript Tutorials*, [Online]. Available: https://www.w3schools.com

[7] Mozilla Developer Network (MDN). *Web Docs*, [Online]. Available: https://developer.mozilla.org

[8] G. Gutin and A. P. Punnen, *The Traveling Salesman Problem and Its Variations*. Boston, MA: Springer, 2006.

[9] M. Dorigo and L. M. Gambardella, "Ant colonies for the traveling salesman problem," *BioSystems*, vol. 43, no. 2, pp. 73–81, 1997.