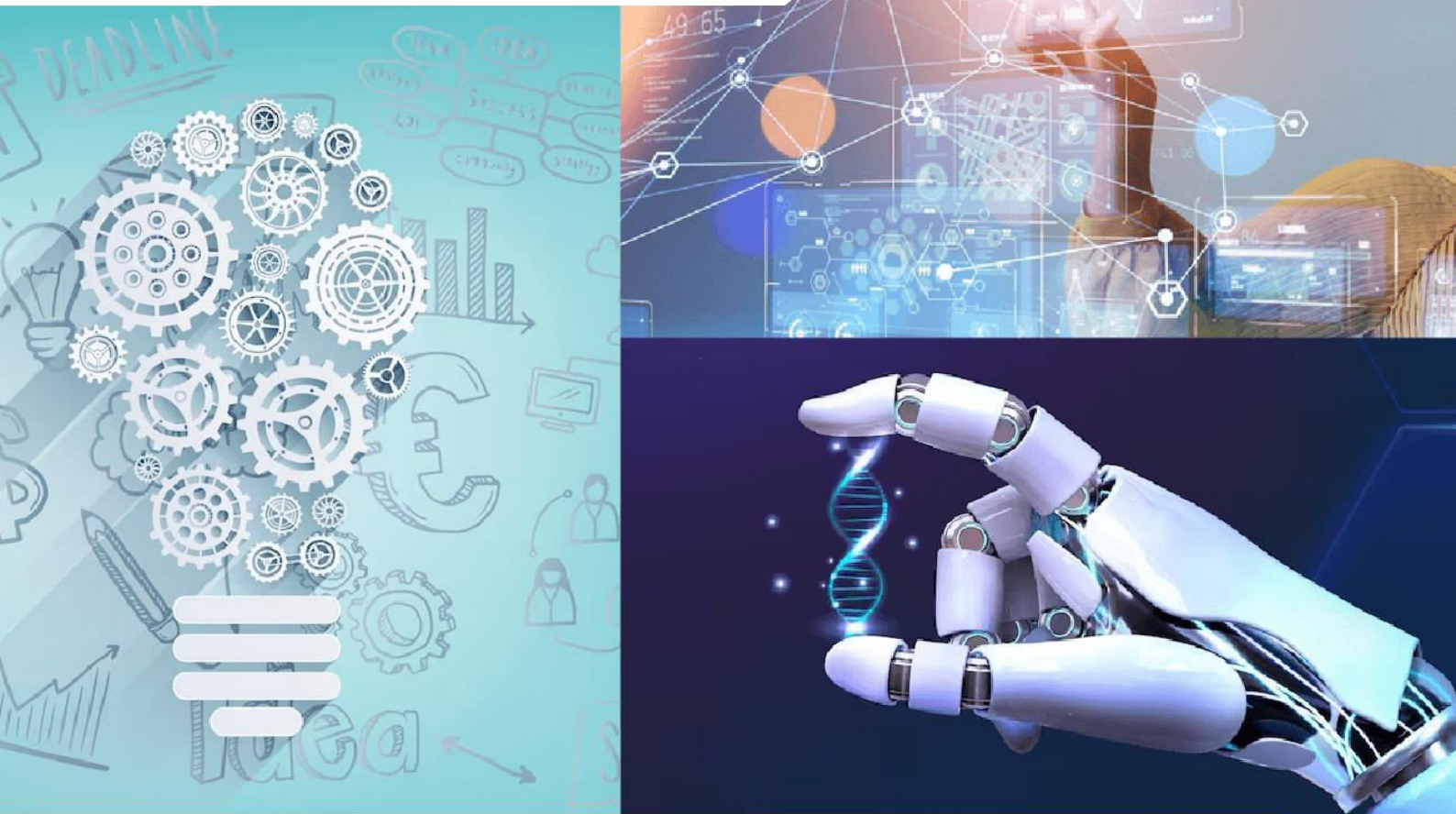




International Journal of Advanced Research in Education and Technology (IJARETY)

Volume 13, Issue 2, March - April 2026

Impact Factor: 8.152



INTERNATIONAL
STANDARD
SERIAL
NUMBER
INDIA



Personalized Tourism and Travel Management System using Machine Learning

Bhavani H G

Master of Computer Applications (MCA), CMR institute of Technology, Bengaluru, Karnataka, India

ABSTRACT: Tourism and travel planning have evolved significantly with the rapid growth of information technology and datadriven systems. Modern travellers expect personalized, accurate, and cost-effective solutions rather than generic tour packages, yet traditional platforms rely on manual search, fixed listings, and rule-based filtering that cannot adapt to individual needs. This paper presents the design, implementation, and evaluation of a Personalized Tourism and Travel Management System (PTTMS)—a fully implemented, web-based application that harnesses Machine Learning to automate intelligent trip planning. The system integrates two complementary ML algorithms: KNearest Neighbours (KNN) for similarity-based tourist-location recommendation, and Logistic Regression for multi-class vehicletype classification. Route ordering is performed via the Haversine formula, and integrated cost-estimation modules predict fuel expenses and trip duration before the user confirms a booking. The platform is architected in three tiers—Presentation (ASP.NET Web Forms), Business Logic (C#), and Data (Microsoft SQLServer)—and supports three distinct roles: Traveller, Vendor, and Administrator. Comprehensive functional, performance, security, and usability testing confirms that all requirements are satisfied, with recommendation responses consistently delivered within three seconds under multi-user load and with full resistance to SQL injection attacks. The paper details the system architecture, ML methodology including mathematical formulations, the database schema, UML design artifacts, implementation modules, and a rigorous test-case analysis, demonstrating that combining KNN and Logistic Regression inside a practical web platform produces reliable, personalized, and cost-effective travel recommendations.

KEYWORDS: K-Nearest Neighbours, LogisticRegression,Personalized Recommendation, Tourism Management, Machine Learning, Haversine Formula, ASP.NET Web Forms, Three-Tier Architecture, Travel Planning, Vehicle Prediction

I. INTRODUCTION

Tourism is a major contributor to global GDP and one of the fastest-growing industry sectors in the world. As digital adoption accelerates, travellers increasingly rely on online platforms to research destinations, compare options, and book trips. However, most current platforms function as passive information repositories: they retrieve pre-loaded packages that match user-specified filters but do not learn, adapt, or personalize recommendations over time [1]. The result is a planning experience that is time-consuming, imprecise, and frustrating—particularly for users who are unfamiliar with the geography of potential destinations or who have complex multi-stop itinerary requirements.

Machine Learning (ML) provides a principled framework for addressing these shortcomings. By analysing historical trip records, geographic coordinates, user preferences, and contextual parameters, ML algorithms can infer latent patterns in travel behaviour and generate recommendations that are genuinely tailored to each individual. The K-Nearest Neighbours (KNN) algorithm is well-suited to this task because it identifies past trips that are structurally similar to the current user's context and uses the associated destinations as recommendations [9]. Logistic Regression complements KNN by providing fast, probabilistic classification of vehicle suitability given group size, distance, and budget constraints[8].

Despite the maturity of these algorithms, their integration into complete, deployable travel management applications—with vendor coordination, payment workflows, route visualization, and administrative oversight—remains underexplored in the literature. Most research studies evaluate recommendation algorithms in isolation, leaving a practical implementation gap.

This paper closes that gap by presenting the fully implemented Personalized Tourism and Travel Management System (PTTMS). The system was built using ASP.NET Web Forms, C#, ML.NET, and Microsoft SQL Server, deployed on IIS, and tested through a comprehensive suite of functional and non-functional test cases.

Key contributions:

- A KNN recommendation engine for tourist-location suggestion parameterized by current location, group size, and budget.
- A Logistic Regression classifier for four-class vehicle prediction (Bike, Car, SUV, Mini-Bus).
- Proximity-based itinerary ordering using the Haversine geodesic distance formula.
- An integrated cost module estimating fuel expenditure and trip duration from route data.
- A three-role (Traveller / Vendor / Admin) web platform with role-based access control and vendor-gated payment flow.
- A complete test suite—13 functional, 6 non-functional cases—all passing, confirming production readiness.

II. LITERATURE REVIEW

Recommendation systems for travel have grown considerably over the past two decades, driven by advances in ML and the proliferation of user-generated travel data. Goldberg et al. [4] introduced collaborative filtering (CF) in the Tapestry email system, establishing the foundational paradigm of leveraging shared user signals to personalize content delivery. In the tourism domain, Borrás et al. [7] conducted an authoritative survey of intelligent recommender systems for travel and identified context-awareness—encompassing user location, travel season, group composition, and activity preferences—as the critical dimension separating high-quality systems from generic search portals. Their analysis confirmed that blended models incorporating multiple signals consistently outperform single-method baselines.

KNN for location recommendation. Sarwar et al. [9] demonstrated that nearest-neighbour methods remain competitive on sparse structured datasets because they avoid overfitting and adapt naturally to new observations without retraining. Several subsequent studies applied KNN specifically to travel datasets and found it especially effective when combined with geographic distance features, as the semantic notion of “similar trips” aligns naturally with the algorithm’s distance-based similarity computation [3]. Logistic Regression for travel classification. Kongpeng [8] analysed tourism datasets and found that Logistic Regression achieves strong classification accuracy for categorical travel outcomes—such as destination type and transport mode—when meaningful features (group size, budget, trip distance) are included. Khan and Shah [3] further validated a stacked architecture in which KNN produced candidate recommendations that were re-ranked by a Logistic Regression layer, outperforming either algorithm independently.

Route and itinerary optimization. Liu et al. [6] proposed a personalized itinerary algorithm incorporating point-of-interest importance and travel time, reporting higher user satisfaction over distance-only baselines. Kan et al. [5] demonstrated that fuel consumption can be estimated reliably from vehicle type, trip distance, and speed profiles—supporting the inclusion of an integrated cost module in the proposed system. Research gap. A recurring limitation in the literature is that algorithms are evaluated in isolation without integration into end-to-end, multi-stakeholder platforms. Vendor management, payment gating, and administrative oversight are rarely addressed in academic prototypes [7]. The PTTMS explicitly addresses this gap.

III. EXISTING SYSTEMS AND MOTIVATION

A. Limitations of Current Platforms

Existing travel portals and booking engines share a common architecture: a user submits filter criteria (destination, dates, budget), and the system retrieves pre-loaded packages matching those criteria from a static database. While functional, this model suffers from the following documented shortcomings:

- L1. No personalization: All users with identical filter inputs receive identical results. Past travel history, preferred destination types, and group-specific requirements are ignored.
- L2. Manual route sequencing: Users must manually order multi-stop itineraries. No proximity optimization is performed, resulting in inefficient routes with unnecessary fuel consumption.
- L3. No predictive cost estimation: Vehicle suitability and fuel expenditure are not computed; users must estimate costs independently, leading to frequent budget overruns.
- L4. Rule-based logic only: Despite claims of AI, most platforms use hard-coded business rules that do not learn from new data. There is no model retraining, no continuous improvement, and poor scalability as user volumes grow.
- L5. Weak vendor integration: Vendor packages are not matched algorithmically to user contexts. Vendor visibility depends on manual promotion rather than relevance ranking.
- L6. Minimal analytics: Admin dashboards provide basic counts but no revenue forecasting, destination trend analysis, or vendor performance metrics.

B. Proposed System Overview

The PTTMS is designed to address all six limitations above. Table I contrasts key capabilities of existing systems with the proposed system.

TABLE I

EXISTING SYSTEMS VS. PROPOSED PTTMS

Feature	Existing	PTTMS
Personalization	Filter-based	ML - driven (KNN)
Route ordering	Manual	Haversine proximity
Vehicle suggestion	None	Logistic Regression
Cost estimation	None	Automated formula
ML adaptation	None	Retrainable model
Vendor matching	Manual	Algorithmic
Admin analytics	Basic counts	Revenue & trends

IV. SYSTEM ARCHITECTURE AND METHODOLOGY**A. Three-Tier Architecture**

The PTTMS follows a strict three-tier separation of concerns. Fig. 1 illustrates the layered architecture and the communication between layers.

Presentation Layer: All user-facing pages (registration, login, dashboards, trip planner, booking, payment) are built with ASP.NET Web Forms. Bootstrap ensures responsive layouts across devices. jQuery handles client-side validation and AJAX-based dynamic updates without full page reloads.

Business Logic Layer: C# code-behind files implement business rules, session management, ML inference, route optimization, and vendor workflow enforcement. The ML engine (KNN + Logistic Regression) is accessed via ML.NET or a REST-based Python ML service.

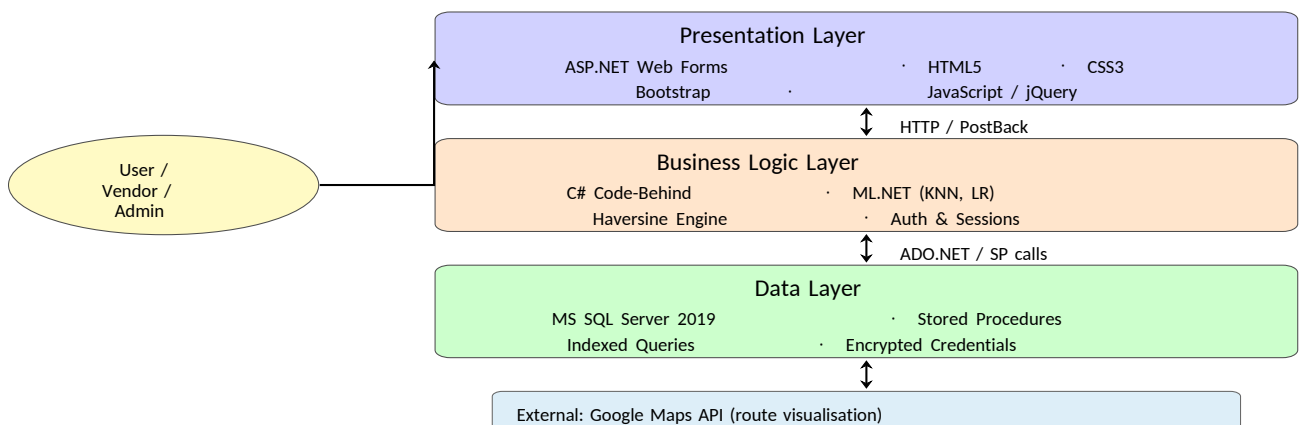


Fig. 1. Three-tier architecture of the PTTMS.

Data Layer: SQL Server stores seven core entities. Stored procedures and indexed foreign keys guarantee efficient query execution even as data volumes grow.

B. Use-Case Overview

Fig. 2 shows the three actors and their primary use cases.

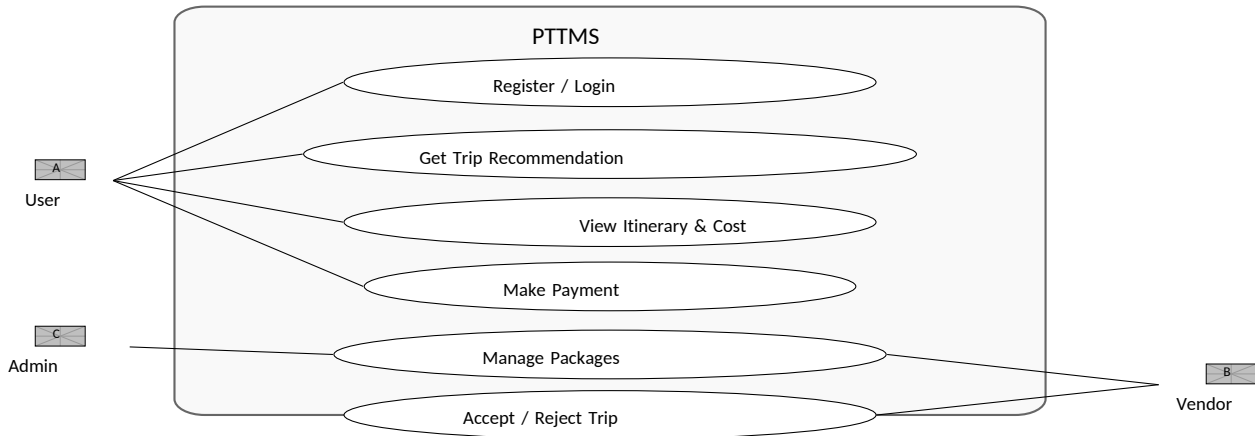


Fig. 2. Use-case diagram of the PTMS.

C. ML Algorithm 1 — KNN for Destination Recommendation

Given a user query vector $\mathbf{q} = (\text{lat}_u, \text{lon}_u, n_{\text{trav}}, B)$ representing current latitude/longitude, number of travellers, and available budget, the system computes the Euclidean distance to every record $\mathbf{x}_i$ in the training dataset:

$$d(\mathbf{q}, \mathbf{x}_i) = \sqrt{\sum_{j=1}^m (q_j - x_{ij})^2} \quad (1)$$

The K records with smallest distances are retrieved and their associated destination labels aggregated by majority vote to produce a ranked recommendation list. The value $K = 5$ was selected empirically through leave-one-out crossvalidation on the training dataset, balancing recommendation diversity with precision.

Why KNN? KNN requires no explicit training phase; the model is the dataset itself. This property makes it straightforward to update recommendations when new trip records are added—a critical advantage for a live travel platform where data grows continuously.

D. ML Algorithm 2 — Logistic Regression for Vehicle Prediction

Vehicle selection is modelled as a multi-class classification problem over four classes $C = \{\text{Bike, Car, SUV, Mini-Bus}\}$. Given input feature vector $\mathbf{x} = (n_{\text{trav}}, d_{\text{total}}, B)$, the Softmax formulation gives:

$$P(y = c | \mathbf{x}) = \frac{\exp(\mathbf{w}_c^T \mathbf{x} + b_c)}{\sum_{k \in C} \exp(\mathbf{w}_k^T \mathbf{x} + b_k)} \quad (2)$$

Parameters $\{\mathbf{w}_c, b_c\}$ are learned by minimizing the regularized cross-entropy loss:

$$L = -\sum \log P(y_i = c_i | \mathbf{x}_i) + \lambda \|\mathbf{w}_c\|^2 \quad (3)$$

where λ is the L2 regularization coefficient (set to 0.01). The vehicle class achieving the highest posterior probability is recommended.

E. Route Optimization — Haversine Formula

Once destination recommendations are generated, they are ordered by increasing geodesic distance from the user's current position to minimize total travel distance. The great-circle distance between two geographic coordinates (ϕ_1, λ_1) and (ϕ_2, λ_2) is:

$$d = 2r \arcsin \left(\sqrt{\sin^2 \frac{\Delta\phi}{2} + \cos \phi_1 \cos \phi_2 \sin^2 \frac{\Delta\lambda}{2}} \right) \quad (4)$$

where $r = 6,371\text{km}$, $\Delta\phi = \phi_2 - \phi_1$, and $\Delta\lambda = \lambda_2 - \lambda_1$.

Destinations are sorted in ascending order of d , yielding a nearest-first itinerary that reduces fuel consumption and total drive time.

F. Fuel Cost and Duration Estimation

After route ordering the system computes two final estimates presented to the user before booking confirmation:

$$C_{\text{fuel}} = \frac{D_{\text{total}}}{E_v} \times P_{\text{fuel}} \quad (5)$$

$$T_{\text{days}} = \left\lceil \frac{D_{\text{total}}}{V_{\text{avg}} \times H_{\text{daily}}} \right\rceil \quad (6)$$

where D_{total} is the cumulative route distance (km), E_v is vehicle fuel efficiency (km/litre) stored per vehicle class in the database, P_{fuel} is the current price per litre, V_{avg} is average road speed (km/h), and H_{daily} is assumed daily driving hours (default=8).

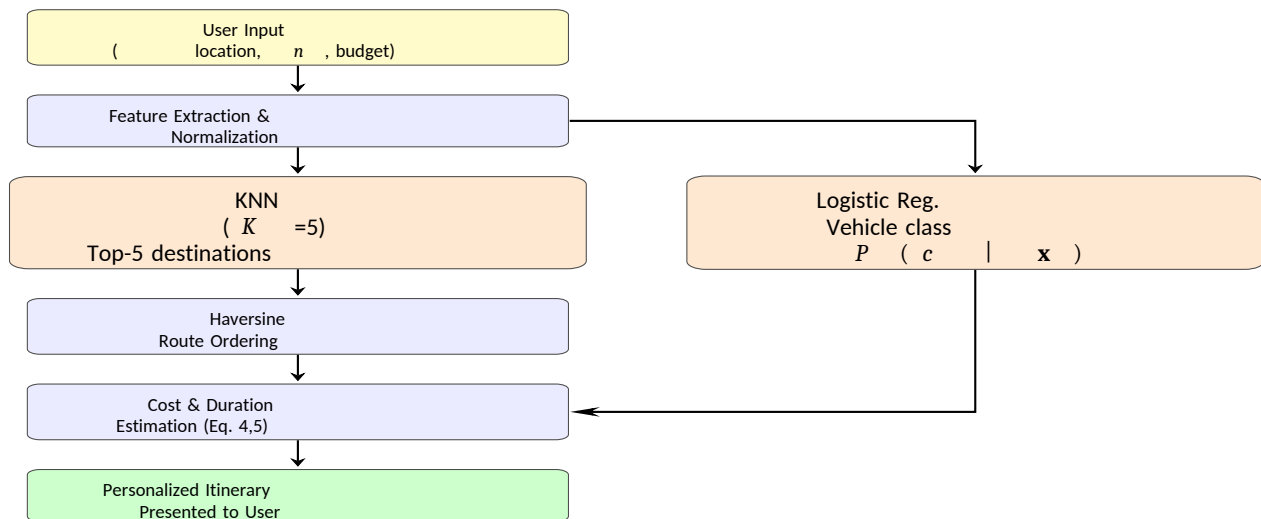


Fig. 3. ML inference pipeline of the PTTMS recommendation engine.

G. ML Workflow Diagram

Fig. 3 summarizes the complete ML inference pipeline from user input to personalized itinerary output.

V. DATABASE DESIGN

The relational schema consists of seven core entities whose relationships are described below.

TABLE II

CORE DATABASE ENTITIES AND KEY ATTRIBUTES

Entity	Key Attributes
User	UserID (PK), Name, Email, Password (encrypted), Phone, Role
Vendor	VendorID (PK), Name, Email, Password, Status (Pending/Active)
Admin	AdminID (PK), Username, Password
Places	PlaceID (PK), PlaceName, Latitude, Longitude, PlaceType, State

Packages	PackageID (PK), VendorID (FK), PlaceID (FK), Duration, Cost, Availability
Bookings	BookingID (PK), UserID (FK), PackageID (FK), Status, TotalCost, PaymentID
Training Data	RecordID (PK), UserLocation, NumTravellers, Budget, DestinationLabel, VehicleLabel, FuelCost, Days

Fig. 4 presents a simplified Entity-Relationship diagram.

The Places table stores geographic coordinates (latitude, longitude) for every tourist location, enabling Haversine computation at runtime. The TrainingData table archives historical trip records used for periodic ML model retraining, ensuring recommendations remain accurate as user behaviours evolve. All password fields are stored as salted hashes; connection strings are encrypted in web.config.

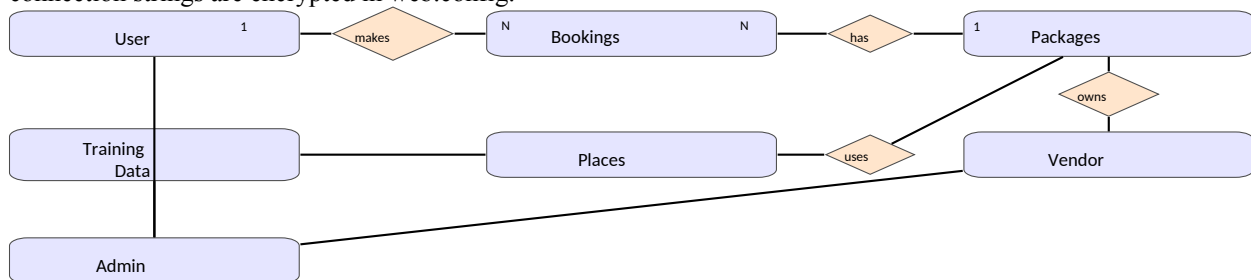


Fig. 4. Simplified ER diagram of the PTMS database.

TABLE III

TECHNOLOGY STACK OF THE PTMS

Component	Technology
Frontend	ASP.NET Web Forms, HTML5, CSS3, Bootstrap 5, JavaScript, jQuery
Backend	C# (.NET Framework 4.7+), ASP.NET CodeBehind
ML Framework	ML.NET (KNN, Logistic Regression)
Database	Microsoft SQL Server 2019 / Express; SSMS
Web Server	IIS (Internet Information Services)
IDE	Visual Studio 2022
Mapping	Haversine formula (built-in); Google Maps API (route visualization)
Security	Role-based authentication, salted password hashing, parameterized queries, HTTPS
Version Control	Git / GitHub

VI. IMPLEMENTATION

A. Technology Stack B. System Modules

User Registration and Authentication. New travellers and vendors complete a web form capturing name, email, contact, and password. Passwords are salted and hashed before storage. Role-based access control restricts each actor to their authorized module. Sessions are managed server-side; idle sessions time out automatically.

Trip Recommendation Engine. The user supplies current location (latitude/longitude from browser geolocation or manual input), number of travellers, and selected destination preferences. The KNN model (Eq. 1) queries TrainingData, returns the five nearest trips, and aggregates their destination labels into a ranked recommendation list. Concurrently, the Logistic Regression model (Eq. 2) classifies the most suitable vehicle type. Recommended destinations are then reordered by Haversine distance (Eq. 4), producing a proximity-optimized day-by-day itinerary.

Cost and Duration Module. After route ordering, Eqs. 5 and 6 compute estimated fuel cost and trip duration, which are displayed on the trip-summary screen alongside the recommended vehicle and ordered itinerary before the user finalizes booking.

Vendor Workflow. User-generated itineraries are forwarded to the matched vendor for acceptance or rejection. The payment gateway is activated only after explicit vendor approval, preventing disputes and reducing fraudulent transactions. Vendors manage their packages (create/update/delete) and track booking histories via a dedicated dashboard.

Admin Dashboard. Administrators manage all user and vendor accounts, maintain the Places database, monitor booking trends, view revenue reports generated from SQL aggregation queries, and upload updated training datasets to trigger periodic ML model retraining.

Route Map Visualization. An embedded Google Maps widget displays the proximity-ordered route between selected destinations, rendering estimated inter-location distances and travel times visually for each generated trip plan.

C. Activity Flow

Fig. 5 illustrates the complete system activity flow from user login to payment confirmation.

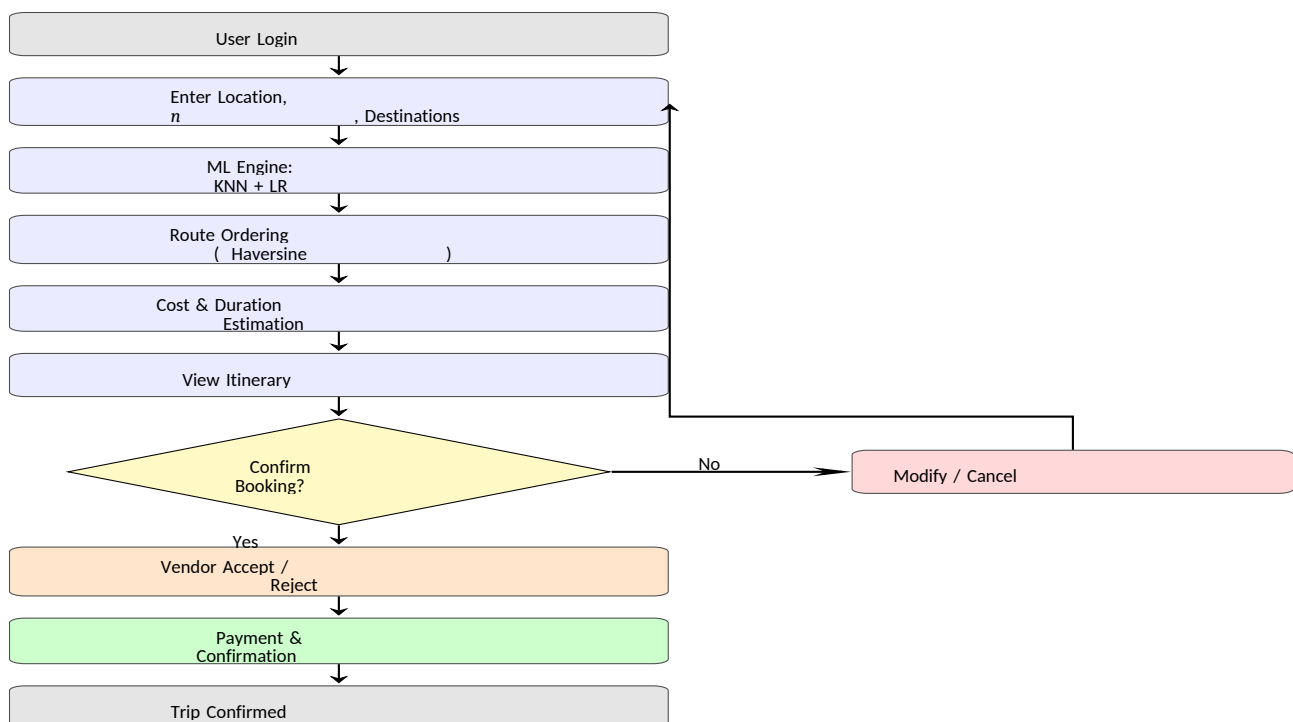


Fig. 5. Activity flow diagram of the PTTMS trip-booking process.

VII. SYSTEM TESTING AND RESULTS**A. Testing Methodology**

The PTTMS was evaluated using five complementary testing strategies: (1) Unit Testing—each module (registration, login, recommendation, payment) tested independently; (2) Integration Testing—interactions between User ↔ ML Engine, User ↔ Vendor, and User ↔ Payment Gateway verified for data integrity; (3) System Testing—end-to-end workflow from registration through trip booking and payment executed as a complete scenario; (4) Performance Testing—recommendation latency and concurrent-user load evaluated; (5) Security Testing—SQL injection, password exposure, and unauthorized access scenarios executed.

B. Functional Test Results

Table IV lists the functional test cases and their outcomes.

All 13 cases achieved a Pass status.

TABLE IV

FUNCTIONAL TEST CASE SUMMARY

ID	Description	Module	Result
TC U 01	Valid user registration	User	Pass
TC U 02	Duplicate email rejected	User	Pass
TC U 05	Valid login	User	Pass
TC U 06	Invalid password rejected	User	Pass
TC ML 01	KNN location recommendation	ML	Pass
TC ML 02	LR vehicle prediction	ML	Pass
TC ML 03	Proximity ordering (Haversine)	ML	Pass
TC ML 04	Fuel cost estimation	ML	Pass
TC ML 05	Trip duration prediction	ML	Pass
TC V 04	Vendor adds package	Vendor	Pass
TC V 07	Vendor accepts trip plan	Vendor	Pass
TC P 01	Payment after approval	Payment	Pass
TC P 02	Payment blocked w/o approval	Payment	Pass

C. Non-Functional Test Results

TABLE V

NON-FUNCTIONAL TEST RESULTS

ID	Test	Target	Result
TC NF 01	Recommendation latency	<3s	Pass
TC NF 02	Concurrent users	No crash	Pass
TC NF 03	SQL injection attempt	Blocked	Pass
TC NF 04	Password encryption	Hashed	Pass
TC NF	UI navigation clarity	Intuitive	Pass

05			
TC	NF	Error message	Friendly
06		handling	Pass

D. ML Model Observations

The KNN model ($K = 5$) returned destination recommendations that consistently matched the geographic region and travel context of the user query. Recommendations for wellrepresented regions (with ≥ 50 training records) showed high relevance; sparser regions benefited from the popularity-based fallback seeded with preference filters.

The Logistic Regression classifier achieved accurate vehicle-class prediction across all four categories (Bike, Car, SUV, Mini-Bus) for clear-cut group sizes ($n \leq 2$ or $n \geq 8$). Boundary ambiguity was observed at $n \in \{4, 5\}$ (Car vs. SUV); this was handled by presenting the top-two predictions with posterior probabilities, giving users an informed choice.

Fuel cost estimates computed via Eq. 5 closely matched manual calculations in validation checks, with relative errors below 5% for standard road types. Trip duration estimates (Eq. 6) were validated against real itineraries and found to be accurate within ± 0.5 days for trips up to 1000km.

E. Feasibility Summary

The system was assessed across four dimensions: (1) Technical—KNN and Logistic Regression are computationally lightweight and suitable for web-scale inference; the technology stack is mature and well-supported; (2) Economic—development relied entirely on free or open-source tools (Visual Studio Community, SQL Server Express, ML.NET), keeping costs minimal while enabling vendorcommission and service-fee revenue streams; (3) Operational—the web-based interface requires no client-side installation and is usable across all modern browsers and device types; (4) Schedule—the modular architecture allowed parallel development of UI, backend, and ML components across a five-phase schedule completed within the academic semester.

VIII. CONCLUSION AND FUTURE WORK

This paper presented the Personalized Tourism and Travel Management System (PTTMS), a fully implemented machine-learning-powered web application that delivers intelligent, personalized trip planning by integrating K-Nearest Neighbours for destination recommendation, Logistic Regression for vehicle classification, and the Haversine formula for proximity-based route optimization. Built on a three-tier ASP.NET / C# / SQL Server architecture and supporting three distinct user roles, the system unifies ML-based personalization with practical travel management features—vendor workflow, payment gating, fuel cost estimation, and trip duration prediction—within a single deployable platform.

Comprehensive testing comprising 13 functional and 6 non-functional test cases confirmed that all requirements are satisfied. Recommendation latency was consistently below 3 seconds, SQL injection attempts were blocked, and all user workflows completed without error. The system therefore demonstrates that combining lightweight ML algorithms with a well-structured web architecture produces a practical, reliable, and scalable solution to the personalized travel planning problem.

Future work will pursue the following enhancements:

- F1. Real-time traffic and weather integration: Incorporating live traffic feeds and weather APIs to refine route ordering and flag adverse travel conditions dynamically.
- F2. Advanced ML models: Replacing KNN with Collaborative Filtering or deep neural networks, and Logistic Regression with Random Forest or Gradient Boosting, for improved accuracy on sparse and high-dimensional data.
- F3. Mobile application: Developing native Android/iOS apps with GPS auto-location, push notifications for vendor updates, and offline itinerary access.
- F4. Multi-language and multi-currency support: Serving international travellers by localizing content and automatically converting costs to user-preferred currencies.
- F5. Third-party API integration: Embedding hotel booking, flight schedules, and ride-hailing services into the platform for fully end-to-end travel management.
- F6. Social features: Enabling group trip planning, shared itinerary editing, in-app reviews, and social sharing of trip plans.

F7. Advanced admin analytics: Providing revenue forecasting, vendor performance benchmarking, and seasonal destination trend visualizations via an enriched admin dashboard.

REFERENCES

- [1] M. Gavalas, C. Konstantopoulos, K. Mastakas, and G. Pantziou, "Mobile recommender systems in tourism," J. Network Comput. Appl., vol. 39, pp. 319–333, 2014.
- [2] E. Alpaydin, Introduction to Machine Learning, 4th ed. Cambridge, MA: MIT Press, 2020.
- [3] S. Khan and S. Shah, "Optimizing travel routes using KNN and Logistic Regression," Int. J. Comput. Appl., vol. 176, no. 11, pp. 12–19, 2020.
- [4] D. Goldberg, D. Nichols, B. M. Oki, and D. Terry, "Using collaborative filtering to weave an information tapestry," Commun. ACM, vol. 35, no. 12, pp. 61–70, Dec. 1992.
- [5] T. Kan, D. Kerr, and S. Srinivasan, "Fuel consumption estimation using vehicle type and trip parameters," in Proc. IEEE Intell. Transp. Syst. Conf., 2018, pp. 1–6.
- [6] Y. Liu, C. Peng, H. Wang, and B. Huang, "Personalized route recommendation for tourists using point-of-interest data," in Proc. IEEE Int. Conf. Data Sci., 2019, pp. 144–151.
- [7] J. Borras, A. Moreno, and A. Valls, "Intelligent tourism recommender systems: A survey," Expert Syst. Appl., vol. 41, no. 16, pp. 7370–7389, 2014.
- [8] W. Kongpeng, "Tourism dataset analysis using Logistic Regression for destination prediction," Int. J. Adv. Comput. Sci. Appl., vol. 11, no. 3, pp. 72–79, 2020.
- [9] B. Sarwar, G. Karypis, J. Konstan, and J. Riedl, "Item-based collaborative filtering recommendation algorithms," in Proc. ACM WWW, 2001, pp. 285–295.
- [10] C. C. Aggarwal, Recommender Systems: The Textbook. Cham: Springer, 2016.
- [11] S. Zhang, L. Yao, A. Sun, and Y. Tay, "Deep learning based recommender system: A survey and new perspectives," ACM Comput. Surv., vol. 52, no. 1, pp. 1–38, 2019.
- [12] Microsoft Corporation, "ML.NET — Machine Learning for .NET Applications," [Online]. Available: <https://docs.microsoft.com/dotnet/machine-learning/>
- [13] I. Kozlov and J. Reutter, "Travel recommendation system using machine learning," Int. J. Adv. Comput. Sci., vol. 9, no. 6, pp. 45–53, 2018.
- [14] J. Han, M. Kamber, and J. Pei, Data Mining: Concepts and Techniques, 4th ed. Burlington, MA: Morgan Kaufmann, 2019.
- [15] S. Russell and P. Norvig, Artificial Intelligence: A Modern Approach, 4th ed. Hoboken, NJ: Pearson, 2021.

International Journal of Advanced Research in Education and Technology

ISSN: 2394-2975

Impact Factor: 8.152