**VAHAN BAZAR - RIDE THE FUTURE: BUY, SELL, TEST**

**OBJECTIVE:**

Our objective is to build a seamless and reliable digital marketplace that simplifies the process of discovering, comparing, and purchasing two-wheelers. The platform is designed to empower users with transparent information, financial planning tools, and easy access to dealerships, while also creating a space for sellers and showrooms to showcase their vehicles.

In addition to new vehicles, **Vahan Bazar also enables users to buy and sell used two-wheelers**, ensuring trusted listings, transparent pricing, and smooth transactions. By integrating browsing, comparison, EMI estimation, test-ride booking, and pre-owned bike sales into a single solution, the goal is to deliver a convenient and trustworthy experience for both buyers and sellers in the two-wheeler ecosystem.

**PROBLEM STATEMENT REFERENCE:**

**a) Problem Statement Chosen:**

Empower vehicle dealers and change the way vehicles are searched online.

**b) Reason to choose the problem statement:**

* The two-wheeler market is vast and rapidly growing, yet buyers lack a **centralized and reliable digital platform** to explore and compare options seamlessly.
* Existing platforms lack an integrated marketplace that unites browsing, financial tools, and dealer connectivity.
* Dealers lack a smart platform to manage inventory and reach buyers effectively.
* Customers want **transparent information** on pricing, mileage, and EMI options, but most platforms do not provide these tools in one place.
* Rising demand for **EVs alongside petrol/diesel vehicles** creates the need for a platform that allows easy cross-comparison.

**SOLUTION OVERVIEW:**

**a) Proposed Approach:**

We propose a full-stack two-wheeler marketplace web app that streamlines the vehicle buying and selling process. It combines browsing, comparison, intelligent finance tools, AI-Powered recommendations, Price prediction for used bikes and test-ride booking into a single, user-friendly platform connecting buyers, sellers, and dealerships.

**b) Key Features/Modules:**

1. Listings with filters (brand, price, mileage, fuel type)
2. Detailed product pages with specs & offers.
3. Side-by-side comparison of models.
4. EMI & fuel cost calculators.
5. Used bikes marketplace with AI-powered price prediction.
6. Showroom directory & test ride booking.
7. Upcoming launches with dates & specs.
8. AI chatbot for queries.
9. Secure buying & payment transactions.

**SYSTEM ARCHITECTURE:**

1. **Architecture diagram:**

**BACK END  
BUSINESS LOGIC  
APIs  
AUTHENTICATIONN**

**EXTERNAL APIs**  
  
**FUEL PRICE API  
LOCATION API**

**FRONT END**

**AI/ML ENGINE**  
  
**RECOMMENDATIONS  
PRICE PREDICTIONS  
CHATBOT**

1. **Data flow explanation:**

* **User Interaction** – A buyer logs in and interacts with the frontend to browse or compare vehicles.
* **Request to Backend** – Frontend sends API requests (e.g., “fetch vehicles by brand” or “calculate EMI”).
* **Processing in Backend** – Backend handles authentication, business logic (filtering, comparison, EMI formula), and booking flows.
* **Database Operations** – Backend queries or updates the database (fetch vehicle details, store booking request, update user profile).
* **Integration Layer** – For showroom locator or notifications, backend calls external APIs (Google Maps, email/SMS service).
* **Response Back** – Processed results (vehicle lists, EMI results, booking confirmation) are sent back to the frontend for display.

**TECHNOLOGY STACK:**

**Backend**: Node.js with Express (REST APIs, authentication, booking management)

**Frontend**: Next.js (responsive UI, browsing, filtering, comparisons) and tailwind css for design and layout.

**Databases:** MongoDB (NoSQL, scalable for vehicle data and users)

**ML/AI Frameworks**

* scikit-learn / TensorFlow Lite (for simple recommendation engine like “best vehicle for your budget/needs”)
* Gemini API (for chatbot or natural language vehicle search)

**APIs / Libraries**

* Google Maps API (showroom locator, distance calculation)
* Email & SMS APIs (SendGrid/Twilio for booking confirmations)
* Axios (frontend-backend communication)
* Chart.js / Recharts (for EMI visualization, fuel vs EV cost comparison)

**ALGORITHMS AND MODELS:**

**a) Algorithm(s) Chosen**

* **Recommendation Engine:** Hybrid (Collaborative + Content-Based Filtering).
* **Used Bike Price Prediction:** Regression Model (XGBoost & Random Forest).
* **Chatbot:** NLP model (Gemini API).

**b) Reason for Choice**

* Collaborative filtering captures user preference patterns.
* Regression ensures accurate pricing for used bikes.
* NLP-based chatbot improves customer engagement.

**c) Model Training & Testing Approach**

* Dataset split into **train (80%) / test (20%)**.
* Evaluation with **MAE (Mean Absolute Error)** for price prediction.
* Recommendation accuracy measured with **Precision and Recall.**

**DATA HANDLING:**

**a) Data Sources Used (APIs / Datasets)**

* **Vehicle Details:** Public datasets or dealer-provided CSVs for used bikes (brand, model, specs, price, fuel type).
* **Showroom Locations:** Google Maps API or dealer location datasets.

**b) Pre-Processing methods:**

* **Data Cleaning:** Remove duplicates, handle missing fields (e.g., price or mileage).
* **Normalization / Scaling:** Standardize numerical fields (price, mileage) for calculations and optional ML models.
* Text preprocessing for chatbot.
* **Validation:** Ensure data integrity for frontend display (e.g., correct vehicle images, specs, and showroom info).

**c) Storage / Pipeline Setup:**

* MongoDB collections for users, vehicles, bookings, dealers.
* Separate **Used Bikes DB** with predicted price field.
* Data pipeline for continuous updates (ETL for bike launches).

**IMPLEMENTATION PLAN**

**a) Initial Setup & Environment**

* Set up version control with **GitHub** for collaboration.
* Configure **frontend environment**: Next.js with necessary libraries (Axios, Tailwind).
* Configure **backend environment**: Node.js with Express.
* Set up **database**: MongoDB.
* Integrate development tools: Postman for API testing, VS Code.

**b) Core Module Development:**

* Vehicle listings, filters, and product page.
* EMI & fuel cost calculators.
* Used bike posting & viewing.
* AI models for recommendations & price predictions.

**c) Integration & Testing**

* Connect frontend with backend APIs using Axios.
* Connect AI services to backend.
* Test API endpoints with Postman.
* Validate data flow: vehicle search → comparison → booking → confirmation.
* Unit testing of modules (filtering, EMI calculation, booking).

**d)Final Deployment-ready Build**

* Deploy frontend (Vercel).
* Deploy backend (Render)
* MongoDB Atlas for cloud database.

### ****PERFORMANCE & VALIDATION****

**a) Evaluation Metrics**

* **Response Time:** Measure the speed of vehicle search, filtering, and comparison APIs.
* **Accuracy of Results:** Ensure filters and comparison modules return correct vehicles and specifications.
* **EMI & Cost Calculations:** Validate financial tools with sample inputs for correctness.
* **Booking Flow Reliability:** Confirm test ride booking is properly recorded and notifications are sent.
* **Precision & Recall:** Evaluate relevance of suggested vehicles to user preferences.

**b) Testing Strategy**

1. **Unit Testing:**
   * Test individual frontend components (search bar, filter dropdowns, EMI form).
   * Test backend functions (EMI calculation, booking API, authentication).
2. **Integration Testing:**
   * Verify end-to-end flow: User searches → compares → calculates EMI → books a test ride.
   * Ensure frontend and backend communication works correctly.
3. **Data Validation:**
   * Check data consistency in the database (vehicle details, bookings, dealer info).
   * Validate API responses against expected results.
4. **Performance Testing:**
   * Simulate multiple users searching and booking simultaneously to check system load.
5. **User Acceptance Testing:**
   * Optional quick user testing to ensure interface is intuitive and features work as intended.

### ****DEPLOYMENT & SCALABILITY****

**a). Deployment Plan**

* **Frontend Deployment:**
  + Build Next.js application and deploy on **Vercel.**
* **Backend Deployment:**
  + Deploy backend Node.js server on **Render.**
* **Database Deployment:**
  + Use **MongoDB Atlas** for production-ready reliability.
* **API & External Services:**
  + Integrate Google Maps API for showroom locations.
  + Configure Email & SMS API for booking notifications.
* **Environment & Configuration:**
  + Set environment variables securely for API keys, DB connections, and credentials.
  + Enable HTTPS and CORS for secure client-server communication.

**b). Scalability Considerations**

* **Backend:**
  + Stateless APIs to allow horizontal scaling (adding more server instances).
  + Load balancing using cloud provider tools.
* **Database:**
  + Optimize queries and indexing for faster search & filter operations.
* **Frontend:**
  + Lazy loading of images and components to improve performance on large listings.

**CONCLUSION:**

 Vahan Bazar provides an **all-in-one intelligent solution** to streamline the two-wheeler marketplace. With Vahan Bazar, we can show a complete user journey: searching for a bike → comparing → calculating EMI → booking a test ride. This makes the solution tangible and easy to evaluate.