**Vahan Bazar**

**Problem Statement Reference:**

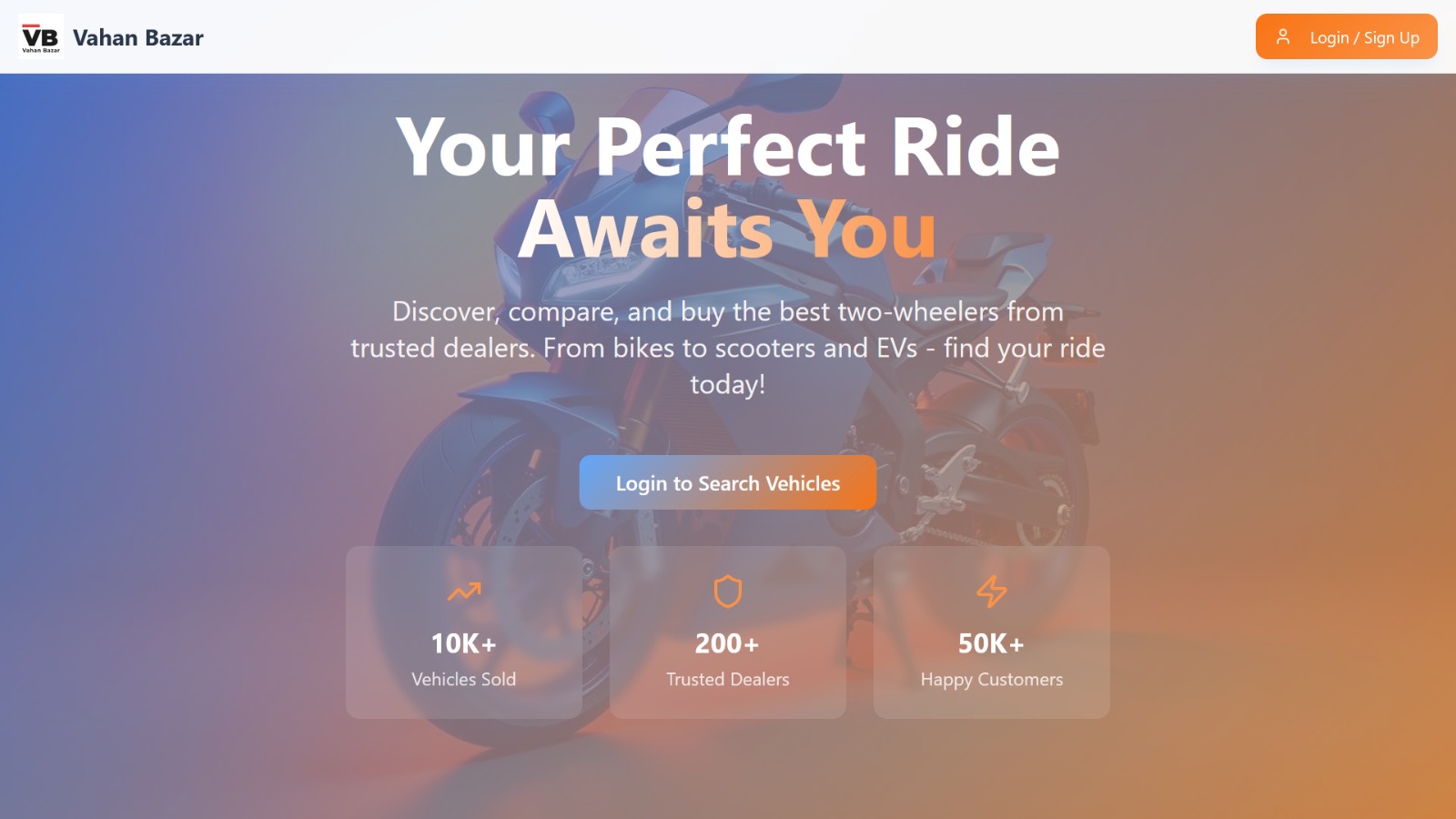
The following Problem Statement has been identified and adapted from the Vahan Bazar company for further analysis.

## Problem Statement Chosen:

The discovery and buying process for two-wheelers is fragmented, causing inconvenience to buyers and inefficiency for dealers.

## Reason to Choose the Problem Statement:

Today’s two-wheeler market is growing rapidly, yet customers struggle with scattered information across manufacturer sites, classifieds, and blogs, making it hard to compare, calculate affordability, or book test rides. Dealers, on the other hand, suffer from low online visibility and poor-quality leads. This gap highlights the need for a unified, trusted digital platform that simplifies research and improves dealer-buyer connections.



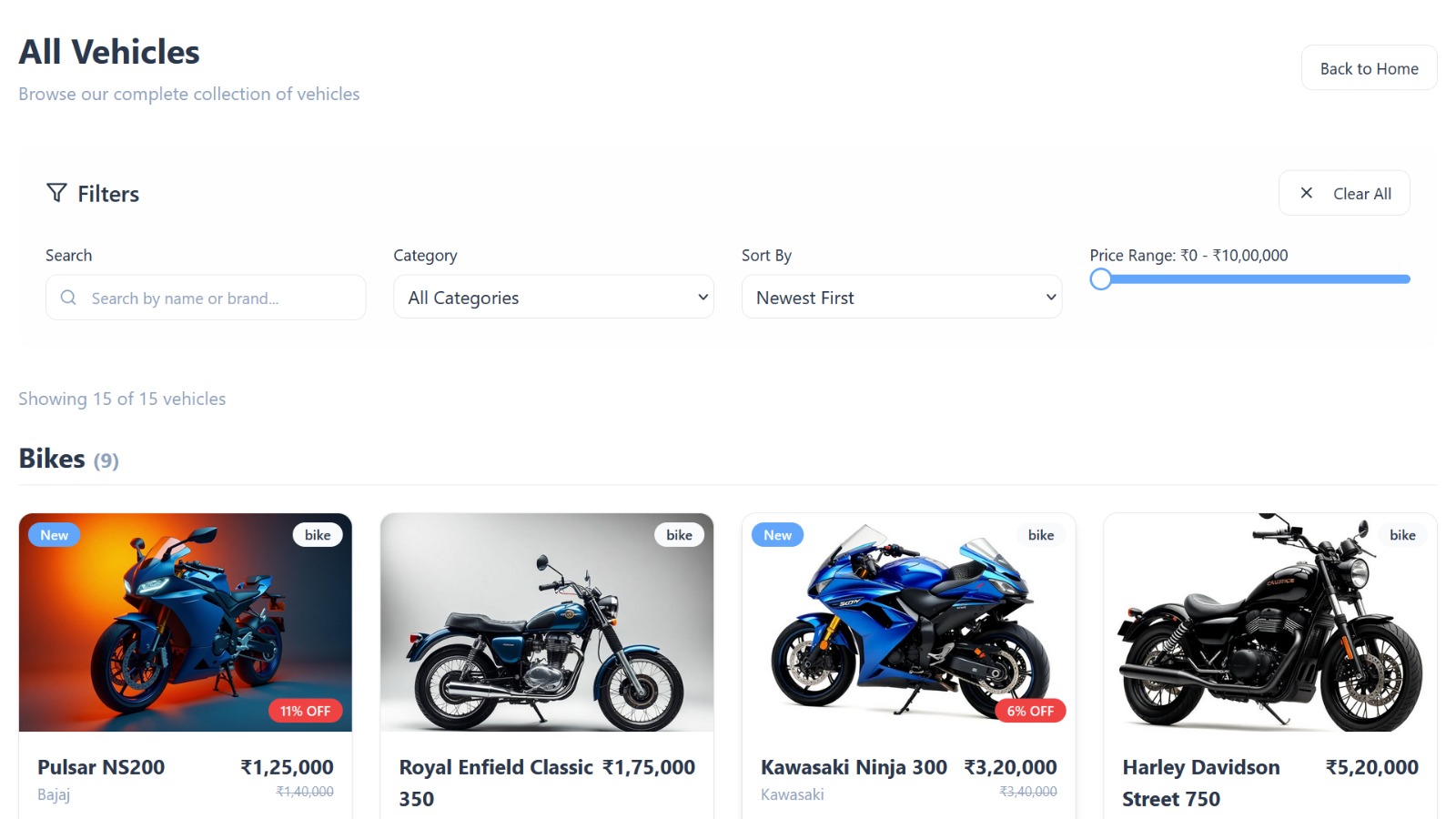
# b) Solution Overview

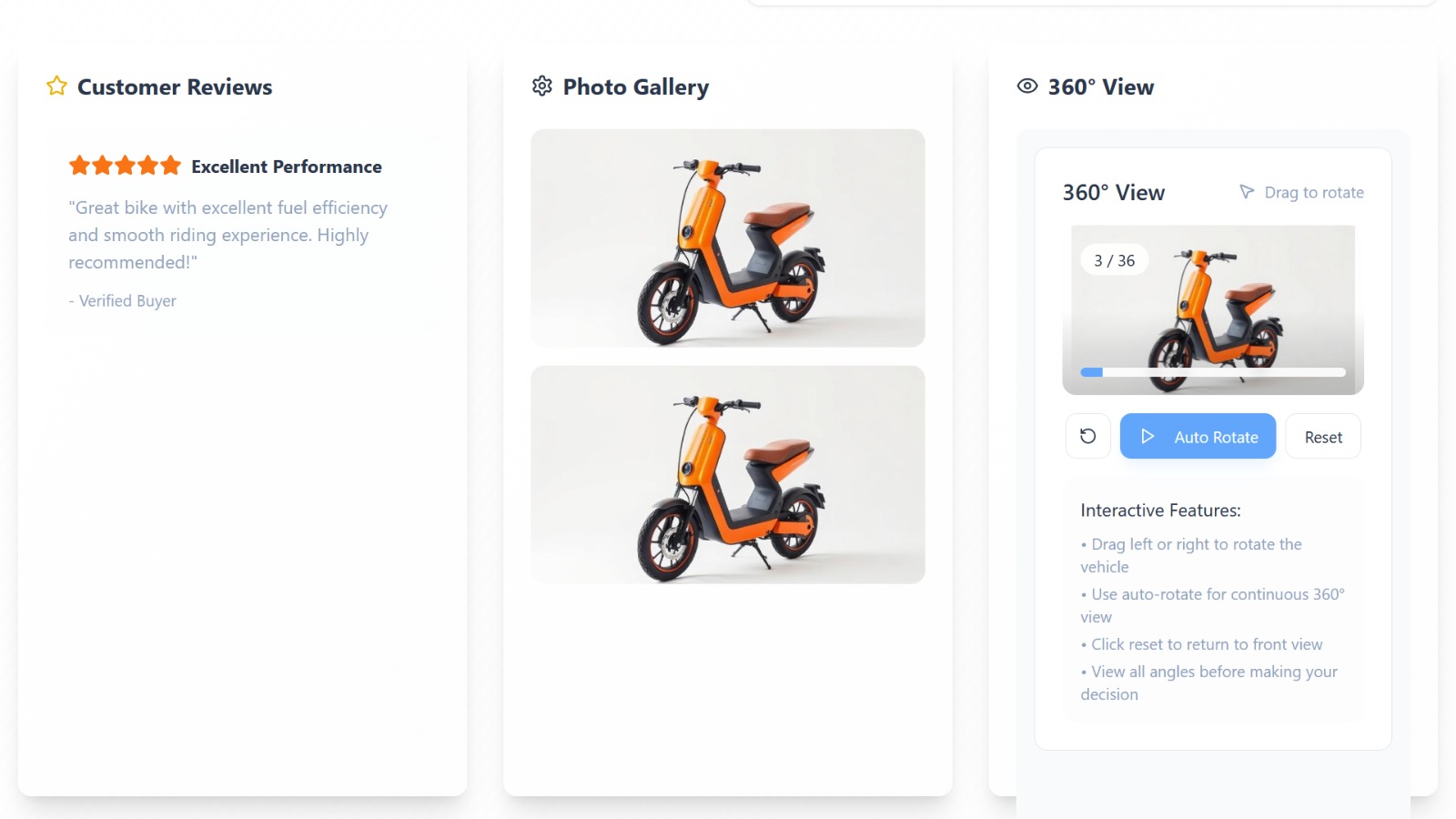
## Proposed Approach :

We propose a responsive, mobile-first Two-Wheeler Marketplace Web App that consolidates the entire journey—from discovery to booking and selling—on one platform, ensuring convenience for buyers and stronger reach for dealers.

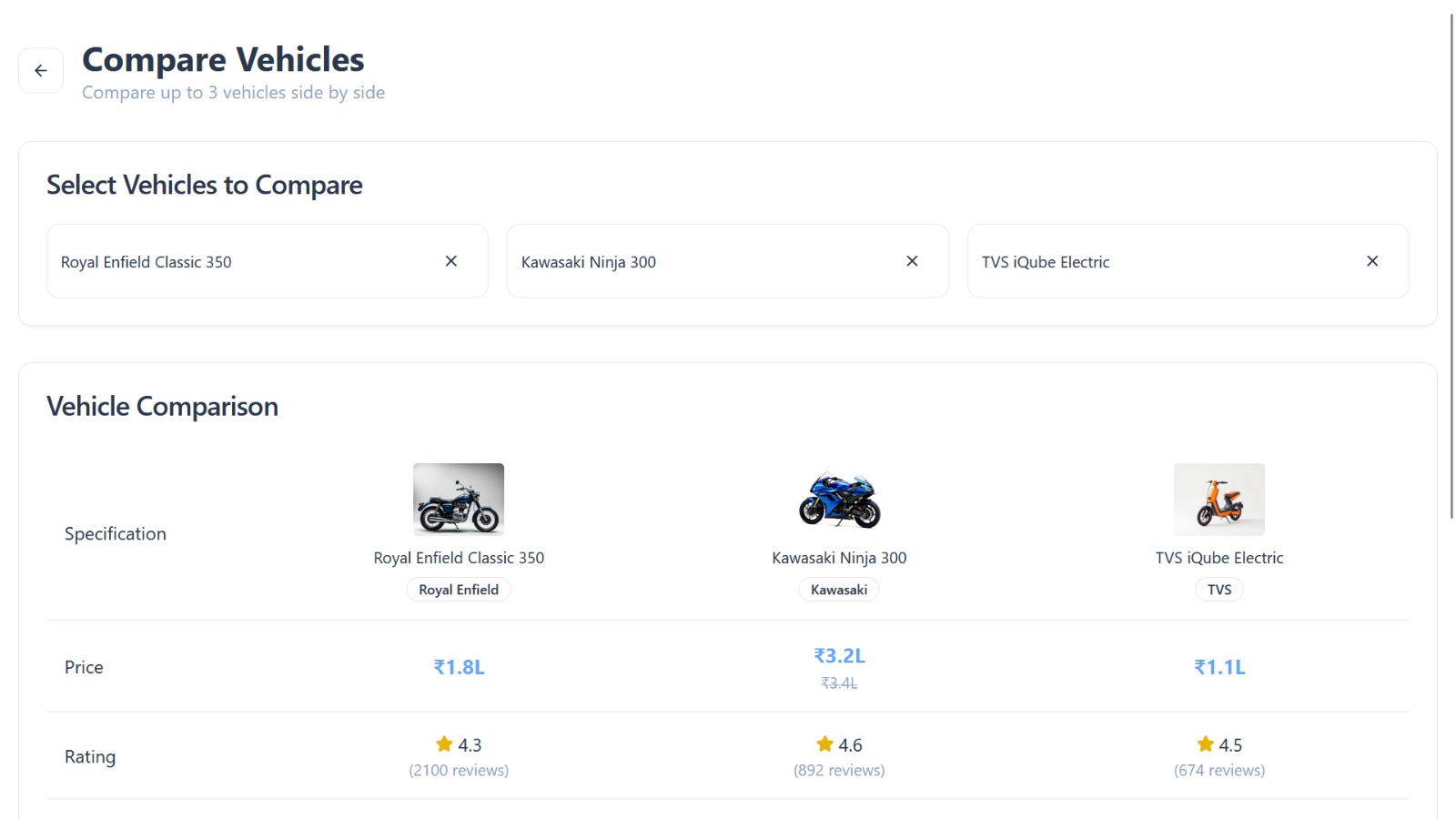
## Key Features / Modules:

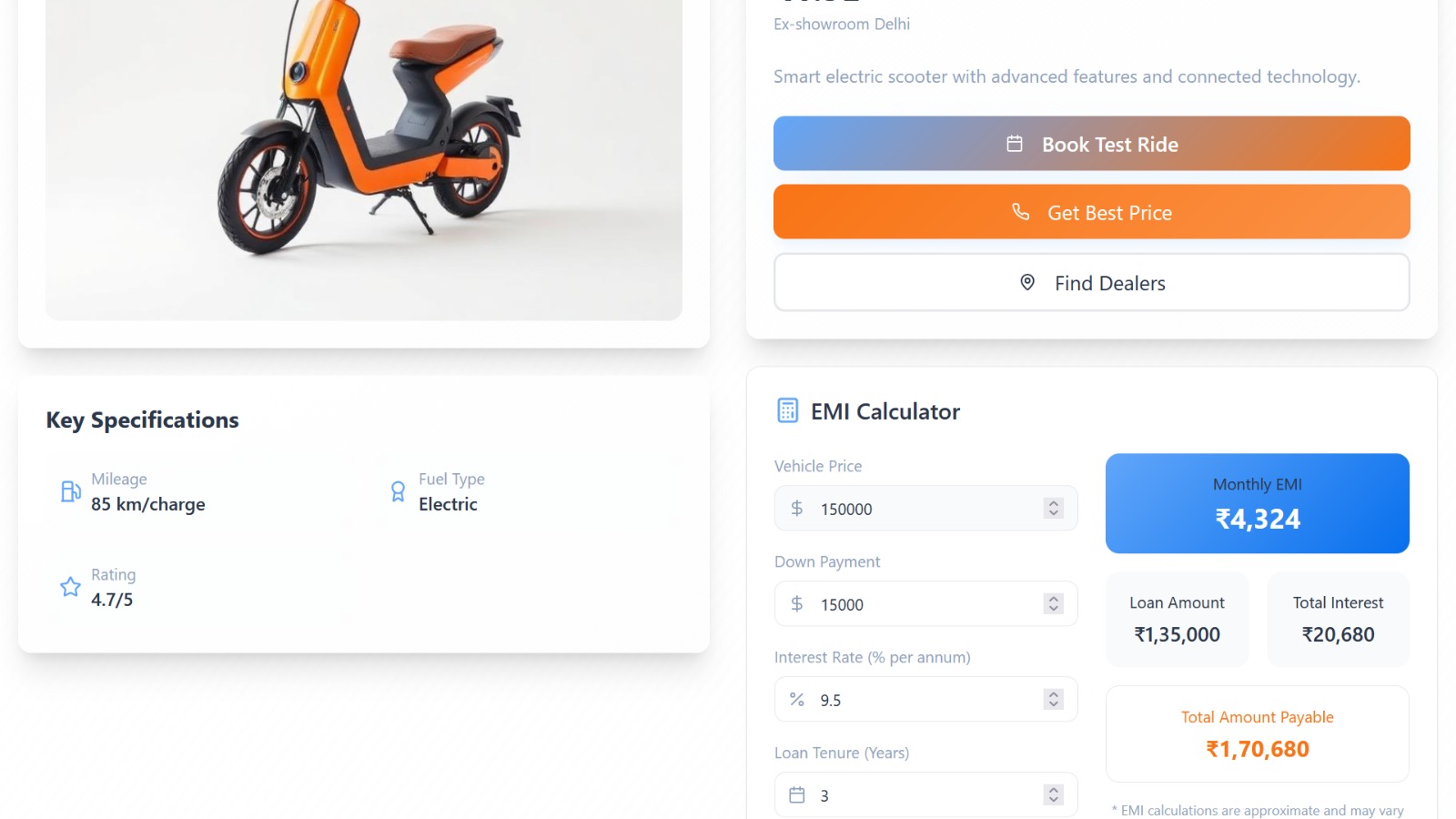
Advanced search & filters (brand, price, fuel type)

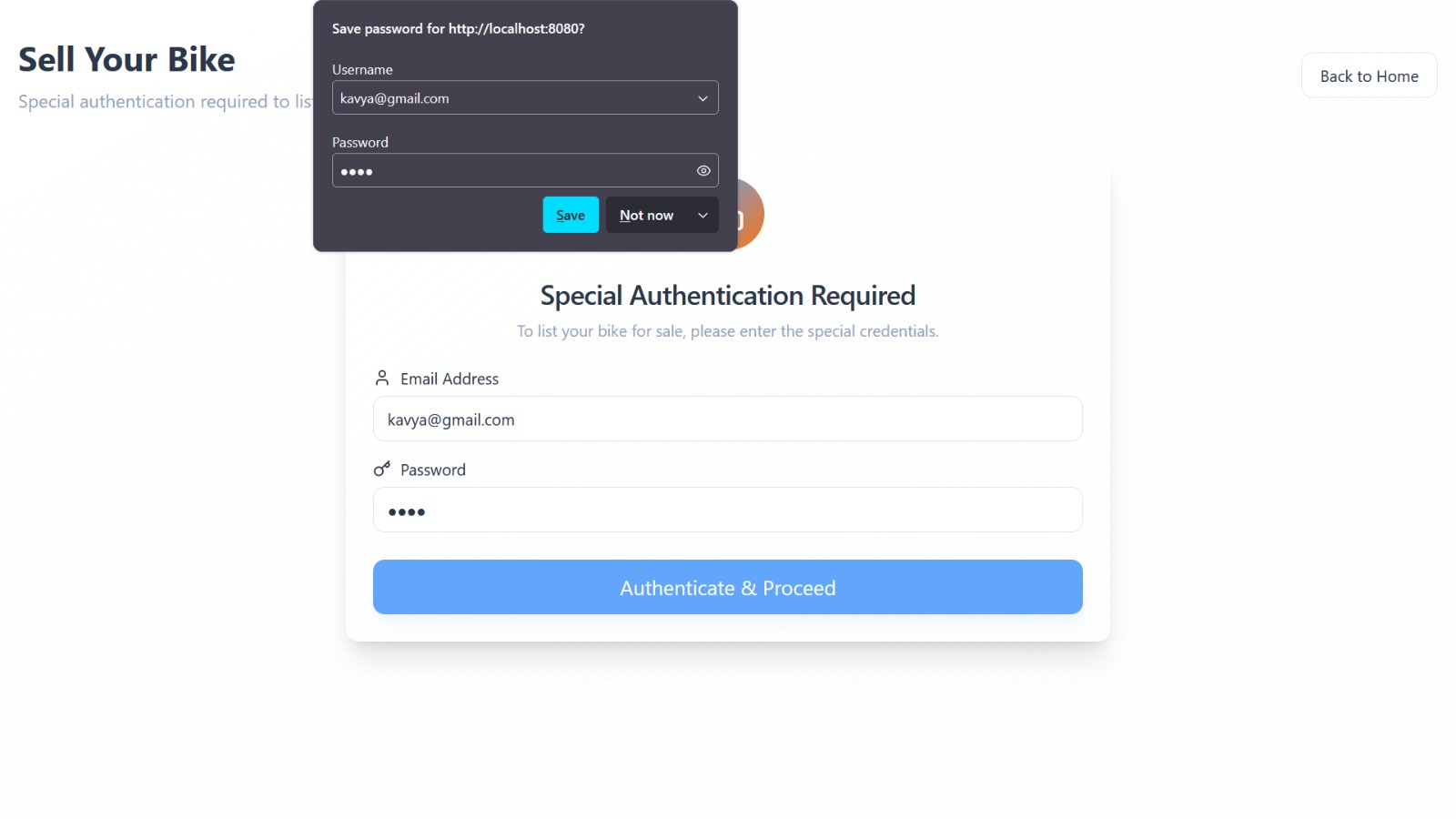
  
Model detail pages with images & specs

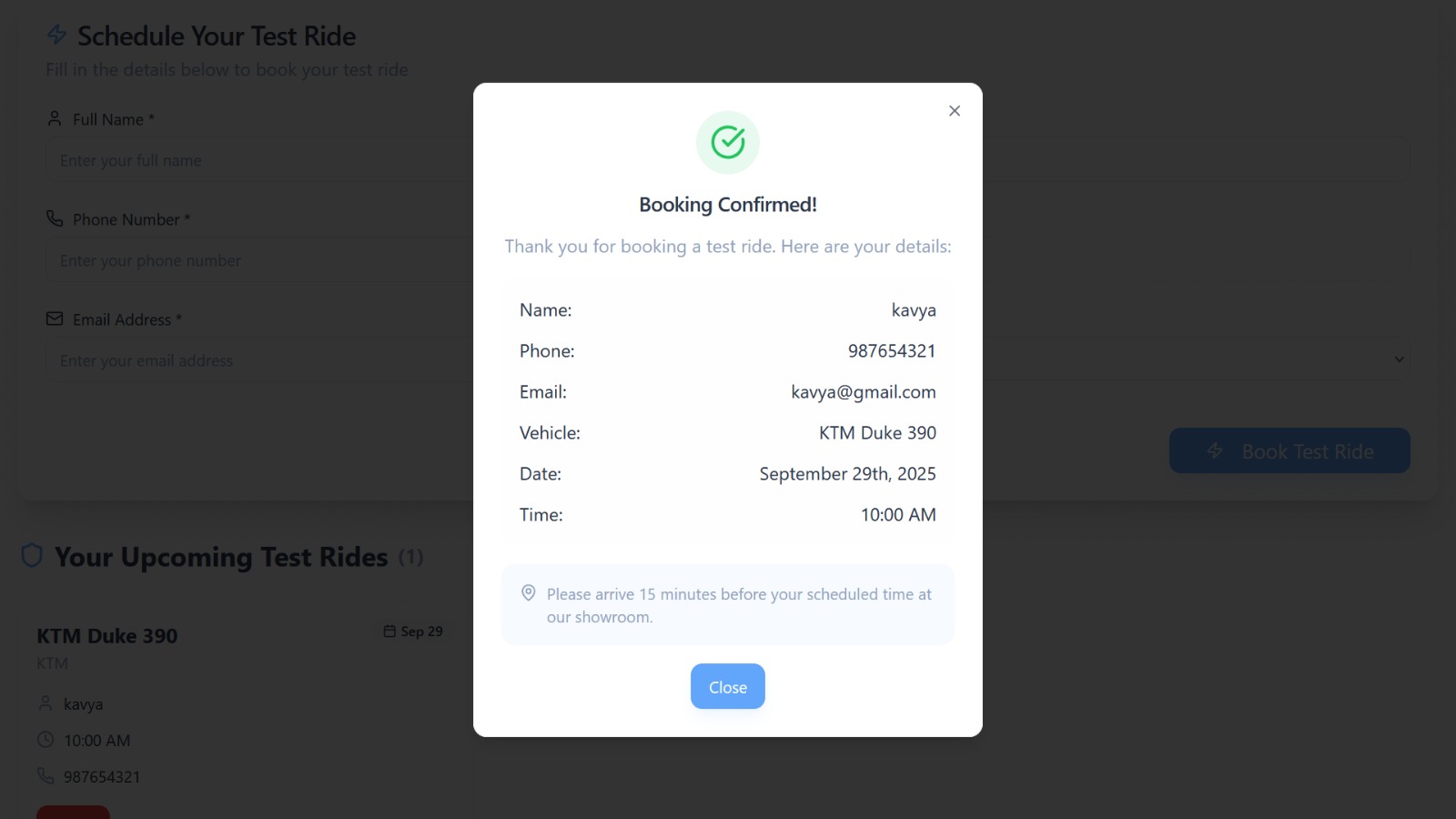


Side-by-side model comparison

  
EMI & running cost calculators

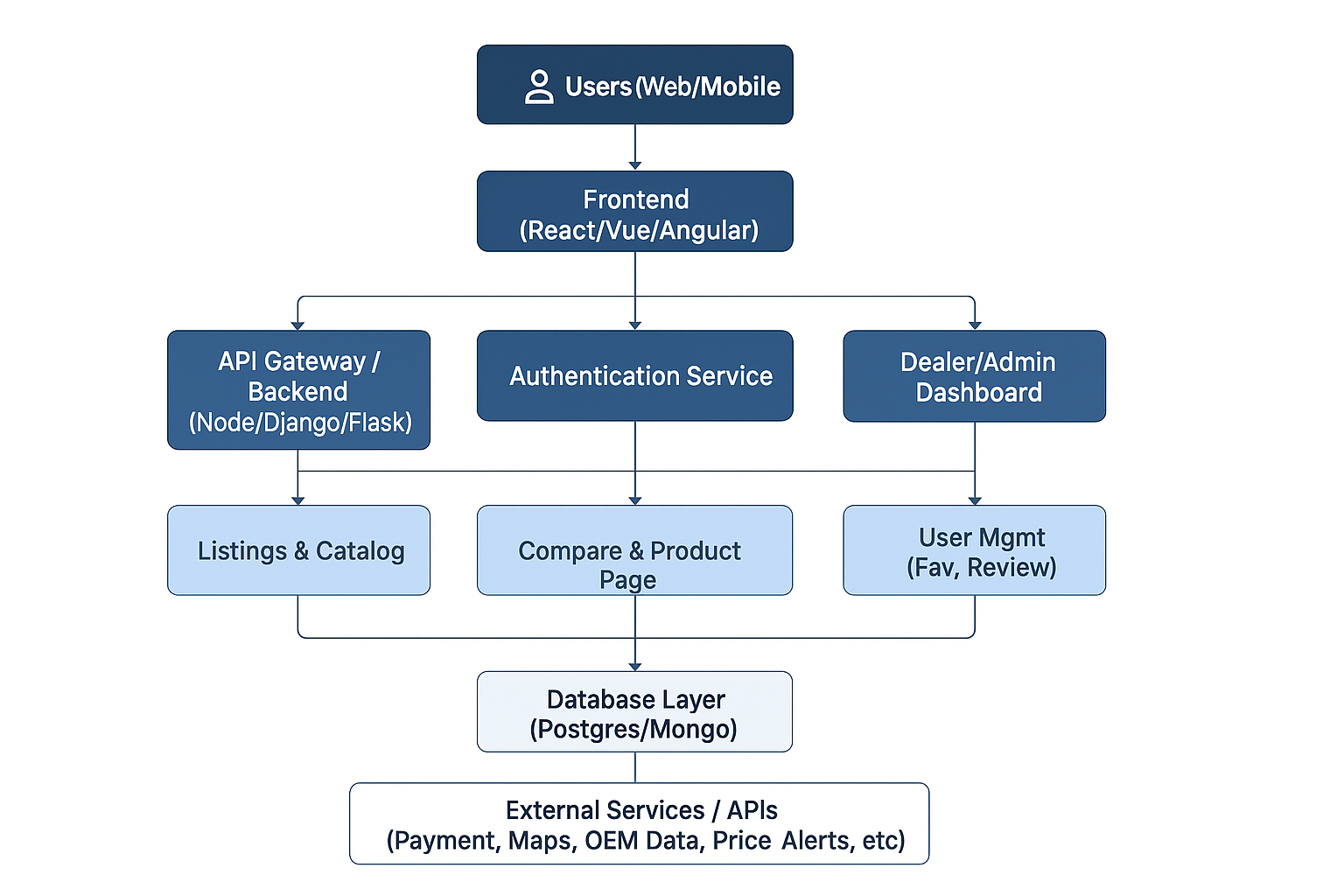
  
- Showroom locator + test ride booking  
- Verified used bike listings

- Dealer/seller dashboards  
- Admin panel with analytics & payments



# c) System Architecture

## Architecture Diagram / Workflow:



## Data Flow Explanation:

1. User browses/searches models → frontend sends request to backend.  
2. Backend queries database & APIs → results sent back to frontend.  
3. User compares, calculates EMI, or books test rides.  
4. Dealers notified via dashboards.  
5. All interactions stored for analytics & insights.

# d) Technology Stack

- Frontend: React.js + Tailwind CSS + Vue + Angular

- Backend: Node.js + Flask + Django

- Databases: MongoDB + PostgreSQL + SQLite  
- ML/AI Frameworks: TensorFlow / Scikit-learn (recommendation, pricing)  
- APIs / Libraries: Google Maps API, Payment Gateways, Chart.js/Recharts

# e) Algorithms & Models

- Algorithms Chosen: Recommendation system (for model suggestions), Price prediction model (for used bikes).  
- Reason: Personalization increases trust, while pricing ensures transparency.  
- Training & Testing: Datasets from manufacturer catalogs, dealer records, and historical pricing. Preprocessed and validated for accuracy.

# f) Data Handling

- Data Sources: Manufacturer APIs, dealership websites, classifieds datasets.  
- Preprocessing: Cleaning, normalization, VIN validation.  
- Storage & Pipeline: Cloud DB with indexing + ETL pipelines for continuous updates.

# g) Implementation Plan

1. Initial Setup: Configure environment (React + Node.js).  
2. Core Development: Search, filters, model pages, calculators.  
3. Integration & Testing: APIs, database, dashboards.  
4. Final Deployment: Cloud hosting (AWS/Azure) with CI/CD pipelines

**h) Performance & Validation**

**Performance**

Performance in the Vahan Bazar app can be improved with lazy loading, caching, pagination, and database indexing for faster searches. Optimized images via CDN, along with backend load balancing, auto-scaling, and monitoring tools like Prometheus or Grafana, ensure smooth handling of high traffic and reliable system performance.

**Validation**

Validation ensures secure and accurate user interactions by validating form inputs, sanitizing data to prevent attacks, verifying user roles with secure logins, and testing tools like EMI and fuel calculators for reliability.

# .i) Deployment & Scalability

The Vahan Bazar app can be hosted on cloud platforms using tools like Docker and Kubernetes. With CI/CD pipelines, updates will happen automatically. The app can grow smoothly with auto-scaling, load balancing, and microservices, so each feature can scale independently. For better speed and reliability, the database will use methods like replication and s

harding to handle heavy traffic.