

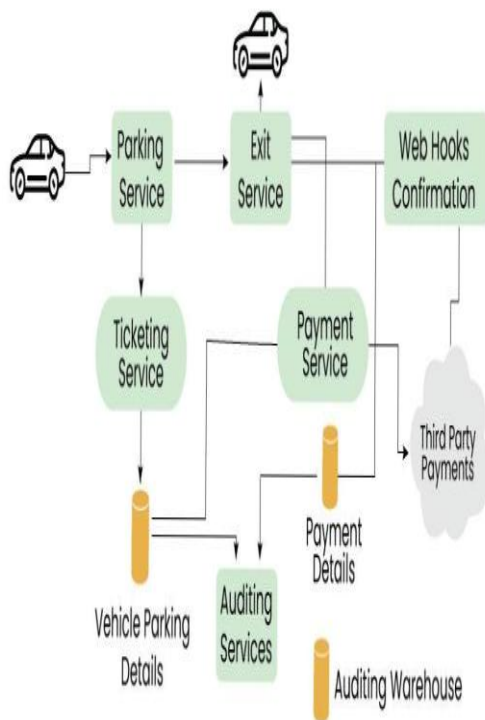
Technology Stack (Architecture & Stack)

| | |
|---------------|--------------------------|
| Date | 1/11/2025 |
| Team ID | NM2025TMID06645 |
| Project Name | Garage Management System |
| Maximum Marks | 4 Marks |

Technical Architecture:

The deliverable shall include the architectural diagram as below and the information as per Table 1 and Table 2.

Example: Garage Management system



Guidelines:

Include all the processes (as an application logic / technology block)

Provide infrastructural demarcation (Local / Cloud)

Indicate external interfaces (third-party APIs, payment gateways, etc.)

Indicate Data Storage components / services

Indicate interface to any IoT or Machine Learning models (if applicable)

Table 1: Components & Technologies

| S.No | Component | Description | Technology |
|------|------------------------|--|--------------------------------|
| 1 | User Interface | User and admin interact via web/mobile dashboard. | HTML, CSS, React / Flutter |
| 2 | Application Logic–1 | Handles vehicle registration, service booking, and scheduling. | Node.js / Python Flask |
| 3 | Application Logic–2 | Assigns mechanics automatically based on workload. | Backend Logic / Algorithm |
| 4 | Application Logic–3 | Sends notifications and updates to customers. | Twilio API / SMTP Email |
| 5 | Database | Stores vehicle, user, mechanic, and billing details. | MySQL / MongoDB |
| 6 | Cloud Database | Managed via cloud backend. | Firebase / AWS RDS |
| 7 | File Storage | Stores service records, invoices, and reports. | AWS S3 / Firebase Storage |
| 8 | External API–1 | (Optional) Payment Gateway integration | Razorpay / Stripe API |
| 9 | External API–2 | (Optional) Vehicle info lookup service | Vehicle API |
| 10 | Machine Learning Model | (Optional) Predicts service time or maintenance schedule. | Python ML Model (Scikit-learn) |

| | | | |
|----|---------------------------------|---------------------------------------|----------------------------|
| 11 | Infrastructure (Server / Cloud) | Hosted and managed on cloud platform. | AWS / Azure / Google Cloud |
|----|---------------------------------|---------------------------------------|----------------------------|

Table 2: Application Characteristics

| S.NO | Characteristics | Description | Technology |
|------|--------------------------|---|----------------------|
| 1 | Open-Source Frameworks | Uses open-source tools for front-end and backend development. | React, Node.js |
| 2 | Security Implementations | Role-based access control for admin, mechanic, and customers. | JWT Authentication |
| 3 | Scalable Architecture | Cloud-based scalable architecture for large data handling. | AWS / Firebase |
| 4 | Availability | High availability using cloud hosting and load balancing. | AWS EC2 / Cloud Run |
| 5 | Performance | Optimized using caching and asynchronous API calls. | Redis, Node.js Async |