Technical Architecture:

The proposed *Garage Management System* will be developed as a **web-based application** that helps manage customer registrations, vehicle service tracking, mechanic job assignments, and billing in an efficient and digital manner.

The architecture includes **three major layers** —

1. **Presentation Layer (Frontend)** for user interaction,
2. **Application Layer (Backend)** for business logic and data processing, and
3. **Database Layer (Storage)** for managing data securely.

External APIs such as payment gateways and notification services are integrated to enhance functionality. The system is deployed on a cloud-based infrastructure for scalability and accessibility.

Example: Centralized garage management platform accessible to customers, managers, and mechanics via web and mobile devices.

Reference: https://developer.ibm.com/patterns/ai-powered-backend-system-for-order-processingduringpandemics/

S.No	Component	Description	Technology
1	**User Interface**	Customers, managers, and mechanics interact through a responsive web portal.	HTML5, CSS3, Bootstrap 5, JavaScript
2	**Application Logic – 1**	Handles customer registration and service booking workflows.	Node.js / Express.js
3	**Application Logic – 2**	Assigns service jobs to mechanics and tracks job progress.	RESTful APIs
4	**Application Logic – 3**	Generates automated invoices and sends status notifications.	Python (Flask) / Twilio API
5	**Database**	Stores details of customers, vehicles, services, and billing records.	MySQL / PostgreSQL
6	**Cloud Database**	Cloud-hosted database for high availability and data backup.	AWS RDS / Firebase
7	**File Storage**	Stores service receipts, reports, and customer feedback files.	AWS S3 / Cloud Storage
8	**External API – 1**	SMS and email notification integration for service updates.	Twilio / SendGrid API
9	**External API – 2**	Payment gateway for online bill payments.	Razorpay / PayPal API
10	**Machine Learning Model**	Predictive maintenance suggestion (optional future enhancement).	TensorFlow / Scikit-learn
11	**Infrastructure (Server / Cloud)**	Hosted and managed on scalable cloud services.	AWS EC2 / Google Cloud Platform

Table – 2: Application Characteristics

S.No	Characteristics	Description	Technology
	Open-Source	Uses open-source frameworks for	

1	**Frameworks**	Node.js, Bootstrap, React flexibility and cost-effectiveness.
2	**Security**	Implements role-based access control and JWT Authentication,
	Implementations	encrypted data storage. HTTPS
3	**Scalable Architecture**	Easily expandable for multiple garage branches and users. Microservices Cloud Load Balancing,
4	**Availability**	System hosted on a cloud server ensures 24/7 uptime. AWS Cloud / Azure
		Optimized database queries and API Redis / Indexed DB

5	**Performance**	caching for faster response. Queries	
6	**Maintainability**	Modular structure enables easy updates and maintenance.	MVC Framework (Express / React)
		Supports third-party APIs for payments	

7 **Integration** and communication. REST / JSON APIs