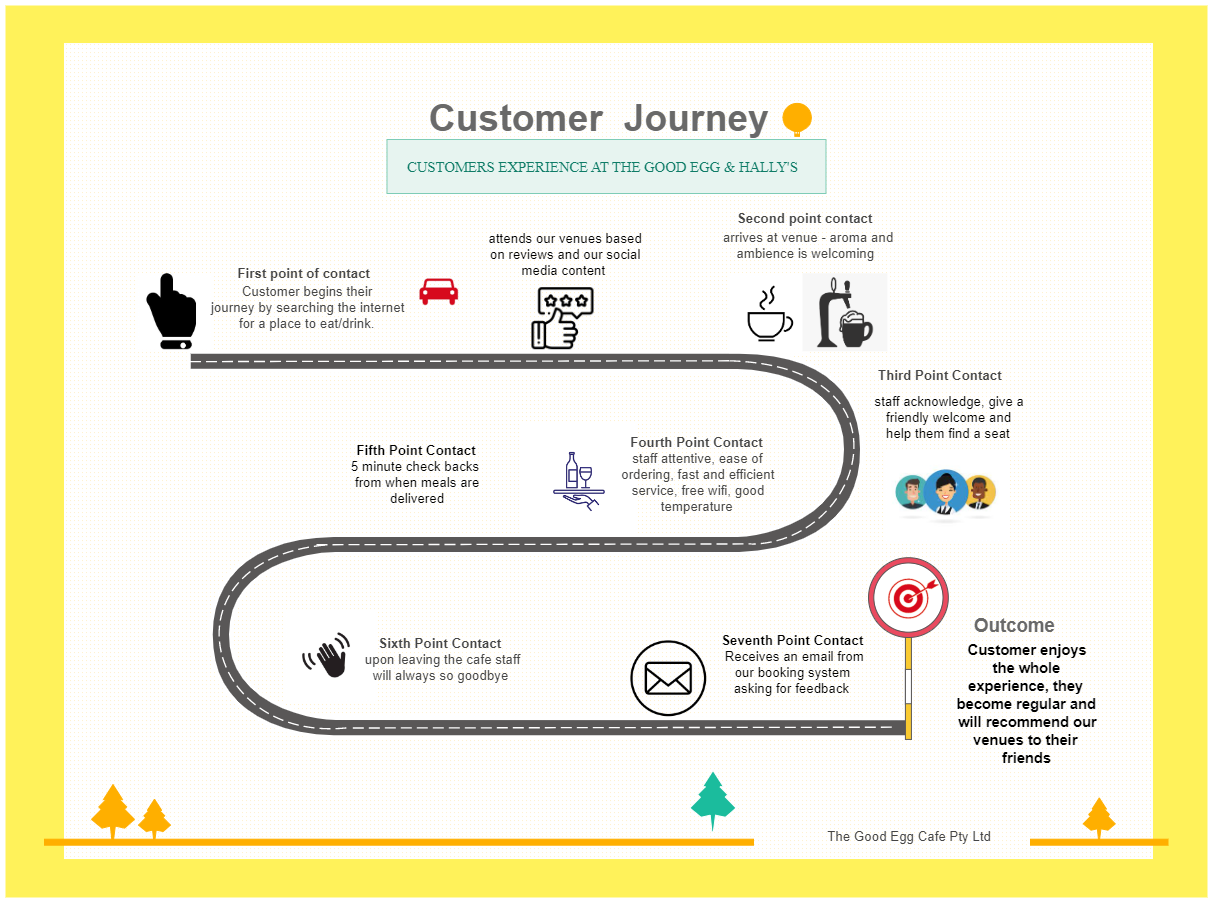
**Project Design Phase -||**

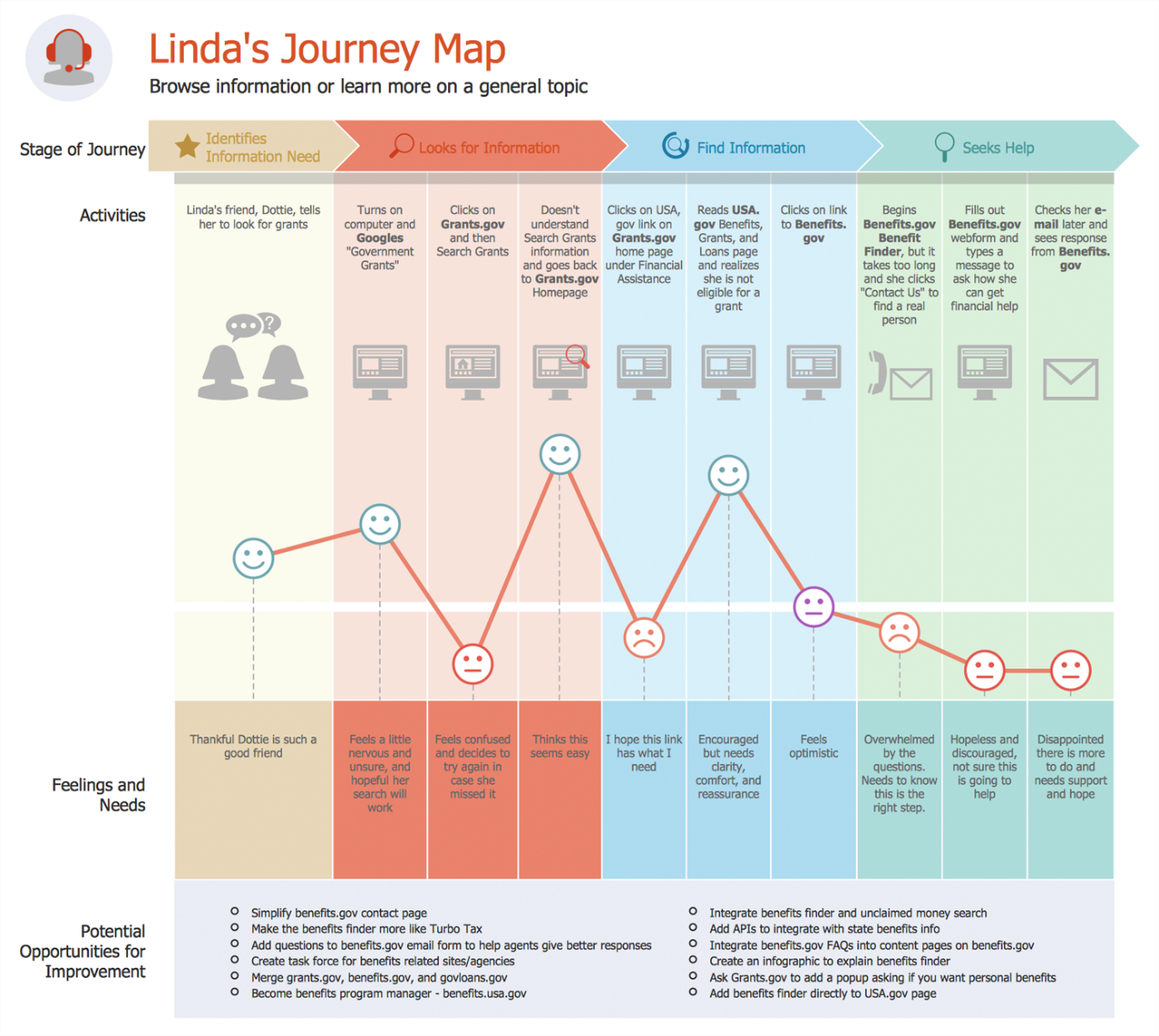
**Customer Journey Map**

|  |  |
| --- | --- |
| **Date** | 24 June2025 |
| **Team ID** | LTVIP2025TMID31613 |
| **Project Name** | Garage Management System |
| **Maximum Marks** |  |

**Customer Journey Map:**

A customer journey map in a garage management system helps visualize the entire experience a customer goes through—from the moment they consider bringing their vehicle in, all the way to post-service follow-up. It's a fantastic tool for spotting pain points, improving customer satisfaction, and streamlining internal operations.





**Essential Elements of a Customer Journey Map:**

1. **Customer Persona**

* A fictional but realistic profile of your ideal customer
* Includes demographics, goals, challenges, and behaviors

2. **Stages of the Journey**

Common stages might include:

* **Awareness**: Customer becomes aware of your product or service
* **Consideration**: Research and comparison phase
* **Purchase**: Final decision and transaction
* **Retention**: Continued use and loyalty-building
* **Advocacy**: Recommending your brand to others

3. **Customer Actions**

* What the customer does at each stage
* Examples: browsing your website, reading reviews, contacting support

4. **Emotions & Pain Points**

* Emotional highs and lows throughout the journey
* Frustrations, confusion, satisfaction, or delight

5. **Touchpoints**

* Interactions between customer and brand
* Can include emails, ads, app usage, in-store visits, social media, etc.

6. **Internal Processes**

* What happens behind the scenes at each stage
* Helpful for aligning internal teams with customer experience

7. **Opportunities for Improvement**

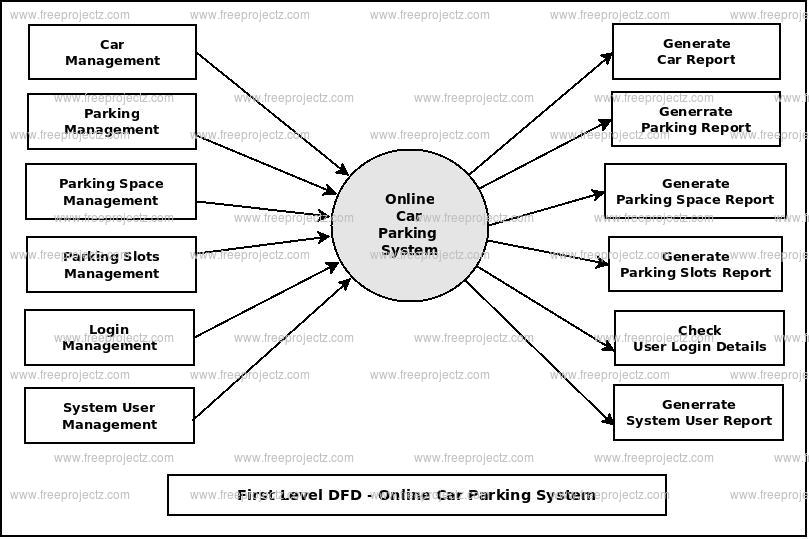
* Gaps or friction points you can enhance
* New strategies to optimize experience and reduce churn

**Project Design Phase -||**

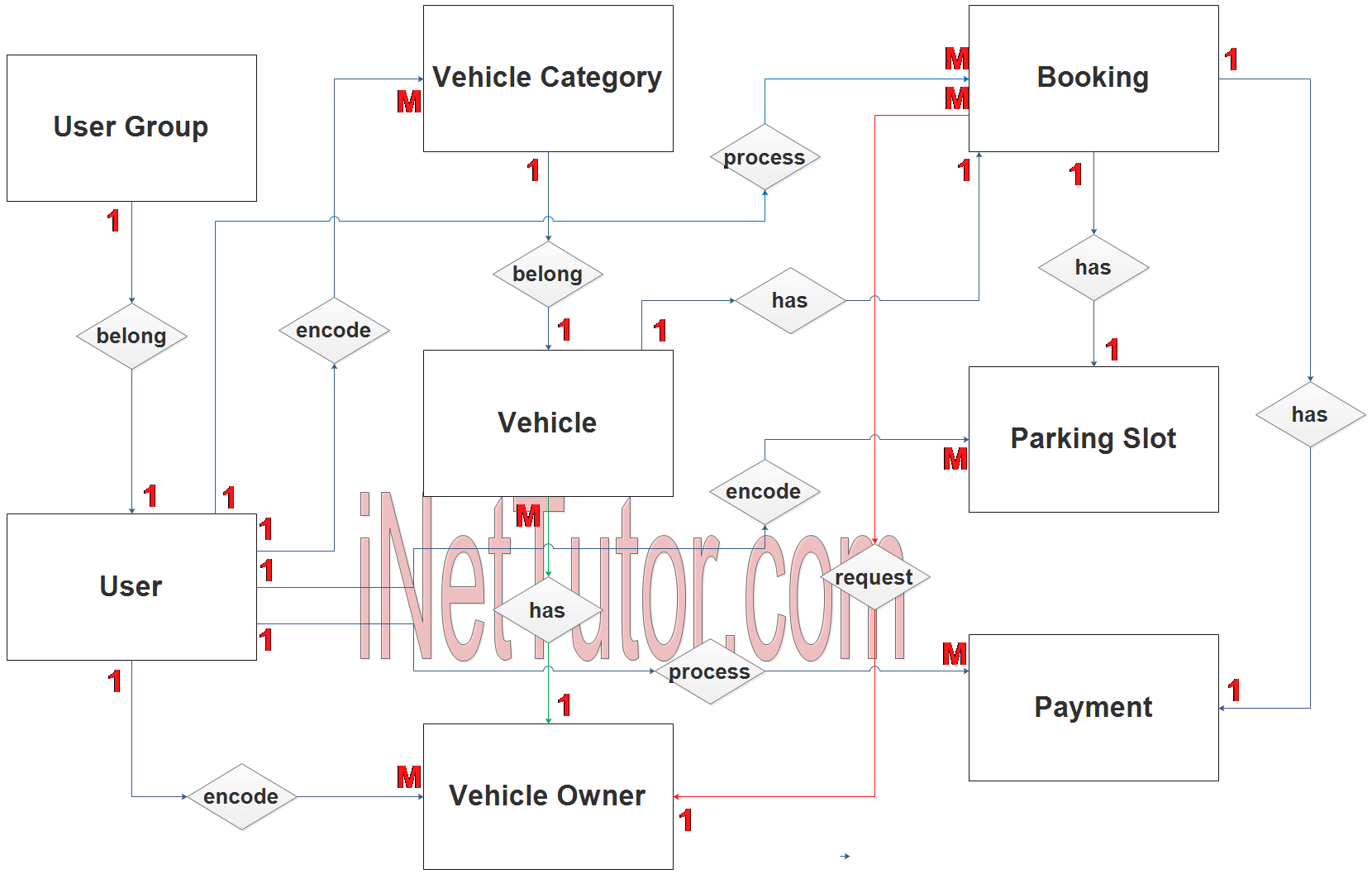
**Data Flow Diagram & User Stories**

|  |  |
| --- | --- |
| **Date** | 24 June 2025 |
| **Team ID** | LTVIP2025TMID31613 |
| **Project Name** | Garage Management System |
| **Maximum Marks** |  |

**Data Flow Diagram:**

****

**Example:**

**flow**

**User Stories:**

**Customer User Stories**

* *As a customer*, I want to book a service appointment online, so that I can avoid waiting at the garage.
* *As a customer*, I want to view my vehicle’s service history, so that I can track past repairs and maintenance.
* *As a customer*, I want to receive notifications when my vehicle is ready, so I can pick it up on time.

**Mechanic/Technician User Stories**

* *As a mechanic*, I want to see a list of assigned jobs, so I can plan my workday efficiently.
* *As a mechanic*, I want to update job status (e.g., In Progress, Completed), so the system reflects real-time progress.
* *As a mechanic*, I want to log parts used, so inventory is automatically updated.

**Admin/Manager User Stories**

* *As an admin*, I want to **add or remove services**, so the system reflects current offerings.
* *As an admin*, I want to **generate invoices**, so I can bill customers accurately.
* *As an admin*, I want to **track inventory levels**, so I can reorder parts before they run out.
* *As an admin*, I want to **view reports on service trends**, so I can make informed business decisions.

**System/User Interface Stories**

* *As a user*, I want to **log in securely**, so my data is protected.
* *As a user*, I want to **navigate the dashboard easily**, so I can find what I need without confusion.

**Project Design Phase -||**

**Solution Requirements(Functional & Non – functional)**

|  |  |
| --- | --- |
| **Date** | 24 June 2025 |
| **Team ID** | LTVIP2025TMID31613 |
| **Project Name** | Garage Management System |
| **Maximum Marks** |  |

**Functional Requirements:**

**1. Customer Management**

* Add, update, and delete customer profiles
* Track customer service history and preferences
* Send appointment reminders or service updates

**2. Vehicle Management**

* Register and manage vehicle details (make, model, VIN, etc.)
* Link vehicles to customer profiles
* Maintain service and repair history

**3. Appointment Scheduling**

* Allow customers or staff to book service appointments
* Display available time slots and technician availability
* Notify technicians and customers of upcoming appointments

**4. Service Management**

* Create and manage service orders and repair jobs
* Track job status and technician assignments
* Define service types and associated costs

**5. Inventory Management**

* Track spare parts and tools
* Manage stock levels and reordering alerts
* Associate parts with specific jobs

**6. Billing and Invoicing**

* Generate service estimates and final bills
* Apply discounts, taxes, and payment methods
* Maintain transaction history and receipts

**7. User Access Management**

* Role-based access for staff (mechanics, managers, etc.)
* Login and authentication for system use
* **Audit logs for activities**

**8. Reports and Analytics**

* Generate reports on daily jobs, revenue, and inventory
* Analyze trends in services and customer visits
* Export data for further analysis

**Non functional Requirements:**

**1. Performance**

* The system should respond to user actions within 2 seconds.
* It should handle multiple users (e.g. up to 100 concurrent logins) without performance degradation.

**2. Scalability**

* Capable of expanding to support additional garages, vehicles, or users as the business grows**.**

**3. Reliability**

* The system must be available 99.9% of the time with minimal downtime.
* It should recover automatically from minor faults without data loss**.**

**4. Usability**

* The interface should be intuitive, with minimal training required for new users.
* Consistent design and terminology throughout the system**.**

**5. Security**

* Role-based access control to restrict sensitive data.
* Data encryption during transmission and storage.
* Regular password updates and login authentication**.**

**6. Maintainability**

* Modular design to allow easy updates or bug fixes.
* Clear documentation for both users and developers**.**

**7. Portability**

* The system should be accessible across desktops, tablets, or mobile devices.
* Cross-platform compatibility (Windows, macOS, etc.).

**8. Compliance**

* Adheres to legal standards for data privacy and record keeping, such as GDPR or local regulations**.**

**Project Design Phase -||**

**Technology Stack(Architecture & Stack)**

|  |  |
| --- | --- |
| **Date** | 24 June 2025 |
| **Team ID** | LTVIP2025TMID31613 |
| **Project Name** | Garage Management System |
| **Maximum Marks** |  |

**Technical Architecture:**

**1. Presentation Layer (Frontend)**

* **Purpose:** Interface where users interact with the system (e.g. booking appointments, tracking jobs**).**
* **Technologies:** HTML, CSS, JavaScript, frameworks like React or Angular
* Devices Supported: Desktop browsers, tablets, possibly mobile apps

**2. Application Layer (Backend)**

* **Purpose:** Core business logic – handles job assignments, billing, customer profiles, etc.
* **Technologies:** Node.js, Python (Django/Flask), Java (Spring Boot), or .NET

** Responsibilities:**

* Processing user requests
* Enforcing business rules
* Managing notifications and workflows

**3. Data Layer (Database)**

* **Purpose:** Store and retrieve structured data for vehicles, customers, parts, services, and transactions

**Technologies:**

* Relational DBMS: PostgreSQL, MySQL, SQL Server
* NoSQL (optional for logs or analytics): MongoDB

**4. Integration Layer**

* **Purpose:** Enables communication with third-party tools or services
* **Examples:**
* SMS/email gateways for reminders
* Payment gateways
* Inventory suppliers
* APIs for vehicle diagnostics (if advanced)

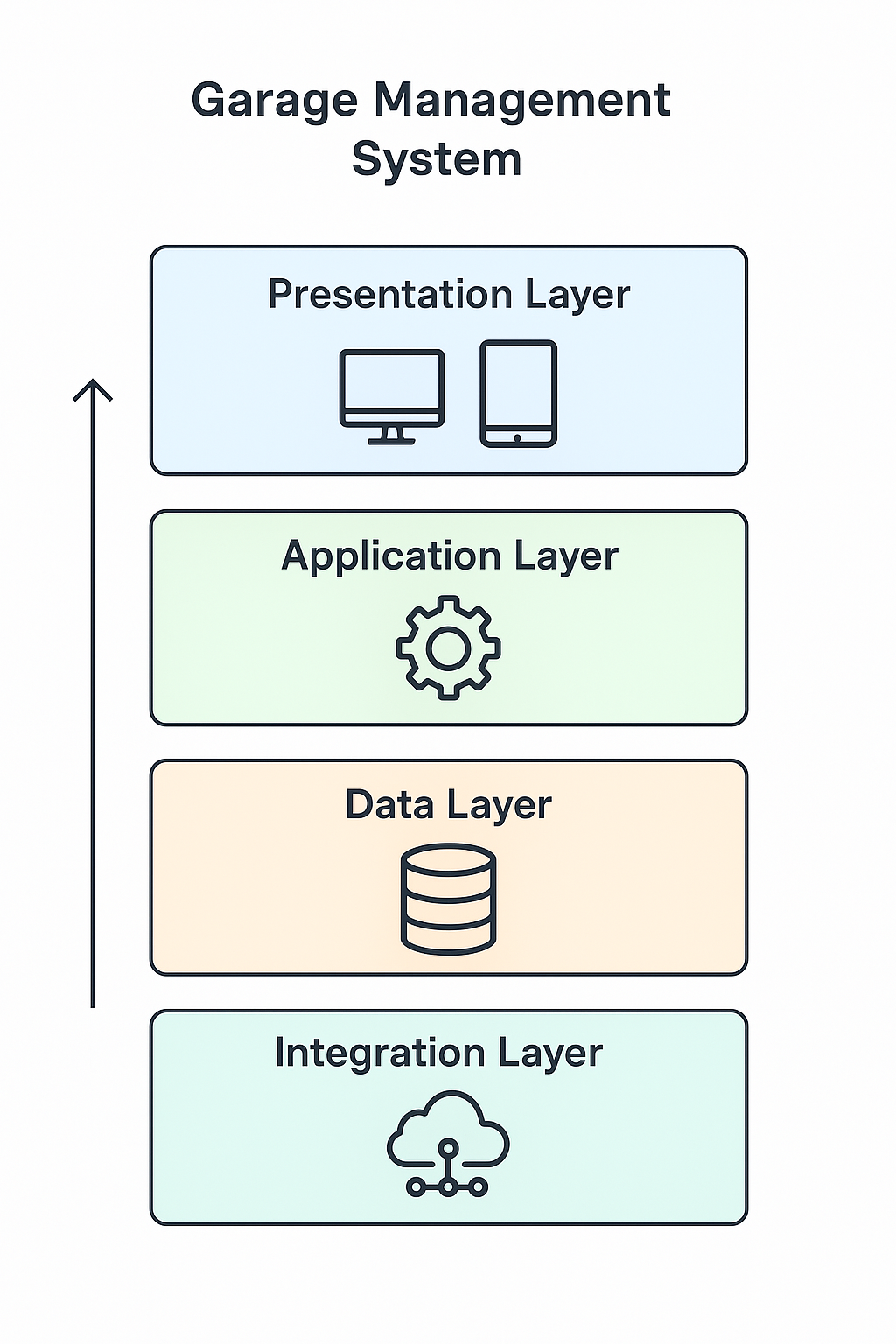
**5. Security Layer (Cross-cutting concern)**

* Role-based access control (RBAC)
* HTTPS/SSL encryption
* Authentication via OAuth2 or JWT

**6. Deployment & Infrastructure**

* **Deployment Options:** Cloud-hosted (e.g. AWS, Azure, GCP) or on-premise
* **Containerization:** Docker, Kubernetes for scalability
* **CI/CD:** Jenkins, GitHub Actions for automated builds and testing
* **Backup & Recovery Tools**

**Diagram:**

****

**Stack:**

**Frontend (Client-Side)**

**Where users interact with the system**

* Languages: HTML5, CSS3, JavaScript
* Frameworks: React.js or Angular
* Styling: Tailwind CSS or Bootstrap
* State Management: Redux (if using React**)**

**Backend (Server-Side Logic)**

**Where all business logic is handled**

* Languages: Node.js (JavaScript), Python (Django/Flask), or Java (Spring Boot)
* Frameworks/Platforms: Express (Node.js), Django (Python)
* API Architecture: REST or GraphQL
* Authentication: JWT (JSON Web Tokens) or OAuth2

**Database Layer**

Stores structured and unstructured data

* **Relational DB:** PostgreSQL or MySQL
* **NoSQL DB (optional):** MongoDB (for logs, analytics, flexible data models)
* **Caching:** Redis (for speeding up access to frequently used data)

**Infrastructure & Deployment**

Where and how the app runs

* **Containerization:** Docker
* **Orchestration:** Kubernetes (for scalability and fault tolerance)
* **Cloud Providers:** AWS (EC2, RDS, S3), Azure, or Google Cloud
* **CI/CD Pipelines:** GitHub Actions, GitLab CI, or Jenkins

**Security & Monitoring**

Protects data and maintains system health

* **Security Protocols:** HTTPS, TLS, CSRF/XSS protection libraries
* **Monitoring Tools:** Prometheus, Grafana, or Datadog
* **Logging:** ELK Stack (Elasticsearch, Logstash, Kibana) or Loki

**Optional Enhancements**

Adds value, especially in customer experience

* **Mobile App:** React Native or Flutter for cross-platform apps
* **Notification Services:** Firebase Cloud Messaging, Twilio (for SMS), or SendGrid (for emails)