**QuickShip**

QuickShip is an **on-demand parcel delivery system**, similar to Rapido or Swiggy Genie, but specifically for sending and receiving parcels. It is a **backend-focused, company-level application** built using **Java Spring Boot and MySQL**, with full user management, delivery workflow, and payment handling.

The project allows **customers, delivery boys (riders), and admins** to interact in a seamless, automated way:

* **Customers** can book parcels for pickup and delivery.
* **Delivery boys** can accept requests, update delivery status, and manage their deliveries.
* **Admins** can monitor the system, track deliveries, and manage payments.

**How does this project work?**

1. **Customer books a parcel:**
   * Customer provides pickup and drop-off addresses, parcel details (weight, type).
   * The system calculates an estimated delivery fare automatically.
2. **System assigns a delivery boy:**
   * Based on availability and location (area-based), a delivery boy is assigned to the request.
   * Rider accepts the delivery request.
3. **Delivery workflow:**
   * Status moves through:  
     PENDING → ACCEPTED → PICKED\_UP → IN\_TRANSIT → DELIVERED
   * Both customer and admin can track the delivery status in real-time.
4. **Payment handling:**
   * Fare is calculated automatically based on distance and parcel weight.
   * Payment can be simulated as **Cash** or **Online**, and the system records it for both admin and rider earnings.
5. **Admin monitoring:**
   * Admin can view all deliveries, customers, and riders.
   * Admin can generate reports like total deliveries, total revenue, and active riders.

**How can we use this project?**

* **For Customers:** Easily send parcels anywhere in the service area.
* **For Delivery Boys:** Accept and manage deliveries efficiently, track earnings.
* **For Admins:** Monitor system performance, manage users, and ensure smooth operations.
* **For Recruiters / Interviewers:** Demonstrates full-stack backend development, workflow automation, security, and database management — all in a **company-level application**.

**Features**

**Customer Features**

1. **Sign Up / Login** – Register and log in securely.
2. **Profile Management** – Update name, phone, email, password.
3. **Book Parcel Delivery** – Provide pickup & drop addresses, parcel details (weight, type, remarks).
4. **Fare Calculation** – System calculates estimated fare automatically.
5. **Track Delivery Status** – View current status: PENDING → ACCEPTED → PICKED\_UP → IN\_TRANSIT → DELIVERED.
6. **View Booking History** – See all past and ongoing deliveries.
7. **Payment Simulation** – Cash or online payment record.
8. **Optional Feedback / Rating** – Rate the rider after delivery.

**2️⃣ Rider (Delivery Boy) Features**

1. **Sign Up / Login** – Register and log in securely.
2. **Profile Management** – Update personal details and vehicle info.
3. **View Available Deliveries** – See nearby delivery requests based on area/pin code.
4. **Accept Delivery** – Accept assigned delivery request.
5. **Update Delivery Status** – Move parcel through workflow: ACCEPTED → PICKED\_UP → IN\_TRANSIT → DELIVERED.
6. **Track Earnings** – View completed deliveries and payment/commission.
7. **Optional Notification Simulation** – Receive alerts when new delivery is assigned.

**3️⃣ Admin Features**

1. **Sign Up / Login** – Admin access with role-based security.
2. **User Management** – View and manage all customers and riders.
3. **Delivery Management** – View all parcel deliveries, status, and history.
4. **Payment Monitoring** – View all payments and manage records.
5. **Generate Reports / Analytics** – Total deliveries, revenue, active riders, etc.
6. **Optional Feature** – System configuration (e.g., fare rates, area zones).

**4️⃣ System / Backend Features**

1. **Role-Based Access Control** – JWT authentication and authorization.
2. **Automated Delivery Assignment** – Assign nearest available rider automatically.
3. **Workflow Management** – Track parcel status through predefined stages.
4. **Fare Calculation Logic** – Distance & weight-based pricing.
5. **Exception Handling & Validation** – Prevent errors and invalid inputs.
6. **Logging & Debugging** – Keep track of activities and errors for maintenance.
7. **Postman API Documentation** – Ready for testing and demonstration.
8. **Optional Advanced Features:**
   * Geolocation-based rider assignment
   * Ratings & Reviews
   * Weekly/Monthly Analytics
   * Notification simulation (Email/SMS)

**Technologies**

**Backend Technologies**

| **Technology** | **Purpose / Why We Use It** |
| --- | --- |
| **Java 17** | **Core programming language; modern features & stability for backend** |
| **Spring Boot** | **Framework for building robust, production-ready REST APIs quickly** |
| **Spring Security** | **Provides JWT-based authentication and role-based access control** |
| **Spring Data JPA / Hibernate** | **ORM for interacting with MySQL database efficiently** |
| **Maven / Gradle** | **Project management & dependency management** |

**2️⃣ Database**

| **Technology** | **Purpose / Why We Use It** |
| --- | --- |
| **MySQL** | **Relational database for storing users, parcels, payments, and system data** |
| **Workbench / CLI** | **For designing, testing, and managing database tables** |

**3️⃣ Frontend Technologies**

| **Technology** | **Purpose / Why We Use It** |
| --- | --- |
| **React.js *(optional)*** | **Build interactive UI for customers, riders, and admin** |
| **HTML / CSS / JavaScript** | **Basic structure, styling, and interactivity for web pages** |
| **Axios / Fetch API** | **To call backend REST APIs from frontend** |

***(If frontend is optional for your documentation/demo, we can keep it simple.)***

**4️⃣ Security & Authentication**

| **Technology** | **Purpose / Why We Use It** |
| --- | --- |
| **JWT (JSON Web Token)** | **Secure authentication & session management** |
| **BCrypt** | **Encrypt passwords for secure storage** |

**5️⃣ Testing & Documentation**

| **Technology** | **Purpose / Why We Use It** |
| --- | --- |
| **Postman** | **Test all REST APIs and generate collection for documentation** |
| **Swagger UI *(optional)*** | **Auto-generate API documentation** |
| **JUnit / Mockito *(optional)*** | **Unit testing backend logic** |

**6️⃣ Optional / Advanced Tools**

| **Technology** | **Purpose / Why We Use It** |
| --- | --- |
| **Google Maps API** | **Calculate distance between pickup & drop locations (for fare calculation)** |
| **Docker** | **Containerize backend for easy deployment** |
| **Heroku / Render / AWS** | **Deploy backend to cloud for live demo** |
| **Git & GitHub** | **Version control and portfolio showc** |