📄 Abstract

The rapid growth of urbanization has significantly increased the demand for organized and efficient home shifting services. However, traditional home relocation processes are often managed manually, resulting in a lack of pricing transparency, improper item management, delayed bookings, and inconsistent billing practices. Customers frequently face uncertainty regarding service costs, item handling, and payment procedures, which reduces trust and operational efficiency.

This project proposes the development of a **web-based Home Shifting Management System** designed to automate and streamline the booking and billing process. The system enables users to create accounts, select predefined household items, add custom items, and generate booking requests through an intuitive digital interface. One of the key features of the system is the implementation of a **weight-based dynamic pricing mechanism** for custom items. This ensures pricing transparency by calculating costs based on predefined weight slabs, eliminating the need for manual price approval by administrators.

The application follows a **three-tier architecture**, consisting of a React-based frontend, a Spring Boot REST API backend, and a relational database system for data persistence. The backend handles business logic such as booking management, item calculation, dynamic pricing, and total amount generation, while the frontend provides a user-friendly interface for interaction. The system also calculates and stores the final booking amount, including a configurable advance payment percentage to secure service confirmation.

By digitizing the booking process, automating price calculation, and maintaining structured data relationships between users, bookings, and items, the proposed system improves operational efficiency, ensures cost transparency, and enhances user trust. The project demonstrates practical implementation of modern full-stack development technologies and real-world system design principles, making it scalable for future enhancements such as payment gateway integration, administrative dashboards, and mobile application support.

📘 Introduction

In today’s fast-paced and rapidly urbanizing world, relocation has become a common necessity due to employment opportunities, education, business expansion, and lifestyle changes. The demand for home shifting services has significantly increased, especially in urban and semi-urban areas. However, despite the growing demand, many small and medium-scale home shifting businesses still operate using traditional manual methods for booking management, item estimation, pricing, and billing.

In conventional systems, customers typically communicate with service providers through phone calls or in-person visits to request quotations. The estimation process is often based on rough assumptions rather than structured item listings. This lack of systematic management leads to several challenges, including unclear pricing, inconsistent item documentation, calculation errors, booking mismanagement, and reduced customer trust. Furthermore, there is minimal transparency in how prices are determined, particularly when customers add customized or uncommon items for shifting.

With advancements in web technologies and digital platforms, there is a strong need to transform traditional home shifting services into an organized, transparent, and automated system. A web-based application can significantly improve efficiency by allowing customers to select predefined items, add custom items with estimated weights, and automatically calculate total charges using predefined pricing logic. Automation reduces dependency on manual intervention, minimizes human error, and enhances overall service reliability.

The Home Shifting Management System proposed in this project aims to address these challenges by providing a structured digital platform for booking and pricing management. The system integrates a modern frontend interface built using React, a backend developed with Spring Boot for handling business logic, and a relational database for maintaining structured data. The application introduces a weight-based pricing mechanism for custom items, ensuring pricing transparency and fairness. Additionally, the system calculates the final booking amount and manages advance payment requirements to confirm service requests.

By digitizing and automating the core processes of booking, item management, and billing, this project demonstrates how full-stack development technologies can be applied to solve real-world logistical problems effectively. The system is designed with scalability in mind, allowing future integration of features such as administrative dashboards, payment gateway support, live tracking, and mobile application deployment.

📌 Problem Statement

The traditional home shifting service process is largely manual, unstructured, and lacks pricing transparency. Customers typically communicate their shifting requirements through phone calls or informal discussions, where item estimation and pricing are determined without a standardized system. This approach leads to multiple operational and trust-related challenges.

One major issue is the absence of a structured item management system. Customers often provide a general list of items without categorization or predefined pricing references. This results in inconsistent quotations and confusion during final billing. There is no systematic way to distinguish between predefined household items and customized or uncommon items.

Another significant problem is handling custom items. When customers request transportation for items that are not part of predefined lists, pricing becomes subjective and dependent on manual evaluation by administrators. This creates several challenges:

- Lack of pricing transparency for users
- Dependency on admin approval for price finalization
- Delay in booking confirmation
- Possibility of inconsistent pricing decisions

Furthermore, there is no automated mechanism to calculate total booking amounts based on selected items and quantities. Manual calculations increase the risk of human error and billing disputes. The absence of a structured database design also makes it difficult to maintain relationships between users, bookings, and individual items.

Advance payment management is another issue. In traditional systems, advance amounts are either decided arbitrarily or not properly recorded, leading to financial tracking difficulties.

Therefore, there is a clear need for a structured digital system that:

- Maintains organized records of users and bookings
- Differentiates between predefined and custom items
- Implements transparent and automated pricing logic
- Calculates total booking amounts accurately
- Stores booking data in a relational database
- Reduces dependency on manual intervention

The proposed Home Shifting Management System aims to solve these problems by introducing a weight-based dynamic pricing mechanism for custom items, structured database relationships, and automated billing logic within a full-stack web application environment.

Challenges Faced and Solutions

During the design phase of the Home Shifting Management System, several architectural and structural challenges were encountered. These challenges were related to database modeling, pricing logic, data integrity, and system transparency. The following section outlines the key issues faced and the solutions implemented to resolve them.

---

1. Designing Proper Relationships Between User, Booking, and Items

Challenge:

Initially, it was unclear how to structure the relationships between users, bookings, and multiple items under a single booking. Since one user can create multiple bookings and each booking can contain multiple items, improper structuring could lead to data redundancy and integrity issues.

Solution:

A relational database structure with proper foreign key constraints was implemented:

- One User → Many Bookings
- One Booking → Many Booking Items

Foreign keys were introduced to maintain referential integrity and ensure structured data relationships.

---

2. Differentiating Between Predefined and Custom Items

Challenge:

The system needed to support two types of items:

- Predefined items with fixed pricing
- Custom items with dynamic pricing

Storing both types without distinction could create confusion in pricing logic and data storage.

Solution:

A boolean field `is_custom` was introduced in the `booking_items` table to clearly differentiate item types. Additionally, nullable fields such as `weight` and `rate_per_kg` were added to support weight-based pricing for custom items without affecting predefined items.

---

3. Handling Pricing for Custom Items

Challenge:

Initially, the idea of allowing administrators to assign prices manually for custom items was considered. However, this created multiple issues:

- Lack of pricing transparency for users
- Dependency on admin approval
- Delay in booking confirmation
- Uncertainty in final cost

Solution:

A weight-based dynamic pricing mechanism was implemented. The system calculates custom item cost automatically using predefined weight slabs and price-per-kilogram rules. This ensures:

- Transparency
- Automation
- Reduced manual intervention
- Immediate booking confirmation

---

4. Deciding Where to Store Final Booking Amount

Challenge:

There was confusion about whether the final booking amount should be calculated dynamically every time it is requested or stored permanently in the bookings table.

If calculated dynamically:

- Performance overhead
- Risk of historical data inconsistency if pricing rules change

Solution:

The `final_amount` field was stored in the `bookings` table at the time of booking confirmation. This preserves historical accuracy and ensures performance efficiency.

---

5. Avoiding Data Inconsistency When Prices Change

Challenge:

If predefined item prices were updated in the future, it could affect historical bookings if price references were fetched dynamically.

Solution:

At the time of booking creation, the item price is copied and stored inside the `booking_items` table. This ensures that past bookings remain unaffected even if item prices are updated later.

---

6. Ensuring Pricing Transparency

Challenge:

One of the core design goals was to eliminate hidden or manually adjustable pricing that could reduce user trust.

Solution:

All pricing calculations were designed to be automated and rule-based. The system ensures that:

- Predefined items use fixed prices
- Custom items use weight-based pricing slabs
- Total booking amount is calculated automatically
- Advance payment (percentage-based) is derived from the final amount

This design improves transparency and builds user confidence in the system.

Database Design

The database design of the Home Shifting Management System is structured using a relational database model to ensure data integrity, scalability, and efficient data retrieval. The system follows normalization principles to avoid redundancy and maintain clear relationships between entities such as users, bookings, and items.

The core objective of the database design is to:

- Maintain structured user records
- Store booking-level information separately
- Support multiple items under each booking
- Differentiate between predefined and custom items
- Preserve historical pricing accuracy

The database consists of the following main tables:

- Users
- Bookings
- Predefined_Items
- Booking_Items

---

1. Users Table

Purpose:

Stores registered user information and supports role-based access control for future enhancements (User/Admin).

Table: users

Column Name	Data Type	Description
id	BIGINT (PK)	Unique identifier for user
name	VARCHAR	Full name of user
email	VARCHAR (Unique)	User email address
password	VARCHAR	Encrypted password
phone	VARCHAR	Contact number
role	VARCHAR	USER or ADMIN

created_at	TIMESTAMP	Account creation timestamp

Relationship:

One user can create multiple bookings (One-to-Many).

---

2. Bookings Table

Purpose:

Stores booking-level details including scheduling, addresses, and final calculated amount.

Table: bookings

Column Name	Data Type	Description
id	BIGINT (PK)	Unique booking ID
user_id	BIGINT (FK)	References users(id)
booking_date	DATE	Scheduled shifting date
source_address	TEXT	Pickup location
destination_address	TEXT	Delivery location
final_amount	DECIMAL	Total calculated booking cost
advance_amount	DECIMAL	Calculated advance payment
status	VARCHAR	Booking status (PENDING, CONFIRMED, COMPLETED)
created_at	TIMESTAMP	Booking creation time

Relationship:

- Many bookings belong to one user.
- One booking contains multiple booking items.

The `final_amount` is stored at the time of booking confirmation to preserve historical pricing consistency.

---

3. Predefined_Items Table

Purpose:

Stores standard household items with fixed pricing.

Table: predefined_items

Column Name	Data Type	Description
id	BIGINT (PK)	Unique item ID
item_name	VARCHAR	Name of predefined item
price	DECIMAL	Fixed price per unit
created_at	TIMESTAMP	Record creation time

Usage:

When a user selects a predefined item, its price is copied into the booking_items table to avoid future pricing inconsistencies.

---

4. Booking_Items Table

Purpose:

Stores individual items associated with each booking, including both predefined and custom items.

Table: booking_items

Column Name	Data Type	Description
id	BIGINT (PK)	Unique record ID
booking_id	BIGINT (FK)	References bookings(id)
predefined_item_id	BIGINT (FK, Nullable)	References predefined_items(id)
item_name	VARCHAR	Item name
quantity	INT	Number of units
weight	DECIMAL (Nullable)	Weight in kg (for custom items)

rate_per_kg	DECIMAL (Nullable)	Rate applied for custom items
total_price	DECIMAL	Total price for this item
is_custom	BOOLEAN	True if custom item

Design Logic:

- If `is_custom = false`:
    - `predefined_item_id` is used
    - `weight` and `rate_per_kg` remain null
- If `is_custom = true`:
    - `predefined_item_id` is null
    - `weight` and `rate_per_kg` are used for dynamic pricing

This design ensures flexibility while maintaining data clarity.

Entity Relationship Overview

The overall relationship structure is as follows:

User (1) → (M) Bookings
Booking (1) → (M) Booking_Items
Predefined_Items (1) → (M) Booking_Items

This structure ensures:

- No data redundancy
- Clear ownership of bookings
- Flexible item handling
- Historical price preservation
- Future scalability for admin and pricing modules