

# Grain Market Storage System



**Team Members:**

**Nishant-2401420032**

**Sakshi-2401420031**

**Group ID:**

**Y1-2024-25-G175**

# 1. Introduction

The **Grain Storage Management System** is a web-based platform designed to streamline grain storage bookings for farmers and storage management for warehouse owners. This system enables warehouse owners to add, update, and delete storage data while allowing farmers to book storage by specifying the quantity and other essential details.

## **2. Objective**

To create a simple and user-friendly grain storage management website.

To allow warehouse owners to manage available storage effectively.

To help farmers easily book storage space by specifying quantity and grain submission details.

### **3. Problem Statement**

In many rural and semi-urban areas, farmers face significant challenges in managing grain storage due to a lack of organized and accessible storage systems. Traditional methods often lead to grain spoilage, overbooking, lack of space transparency, and miscommunication between farmers and warehouse owners. There is a clear need for a centralized system where warehouse capacity can be managed effectively and farmers can book space reliably.

## 4. Proposed Solution

- To address these issues, we propose a web-based **Grain Storage Management System** built using Python Flask and SQLite. This platform allows:
- **Warehouse Owners** to add, update, and delete storage capacity details and monitor farmer bookings.
- **Farmers** to view available storage in real-time and book required quantities by providing relevant details such as name, contact, address, quantity in quintals, and date of submission.
- The system ensures accurate tracking by automatically updating available storage and validating booking dates.

This solution minimizes manual intervention, enhances storage visibility, and promotes efficient grain storage operations.

# 5. Methodology

## 1. Requirement Analysis

- Identified the needs of two primary users: Warehouse Owners and Farmers.
- Defined key functionalities like adding storage, booking storage, updating availability, and managing farmer data.

## 2. System Design

- Designed a clear architecture with three core sections:
- Home: Displays all storage data.
- Warehouse Section: Allows managing storage and viewing farmer bookings.
- Farmer Section: Enables farmers to book available storage with relevant details.
- Designed database schema for efficient storage and retrieval of data.

## 3. Front-End Development

- Used HTML, CSS, and Bootstrap for designing a responsive and clean user interface.
- Integrated navigation bar across all pages for easy access to features.
- All values are shown in quintals for clarity and standardization.

#### **4. Back-End Development**

- Built using Python 3.13 and Flask Framework.
- Flask routes handle actions like adding, updating, deleting storage, and booking storage.
- Data is stored and manipulated using SQLite database.
- 

#### **5. Database Integration**

- Created two tables:
- storage (for warehouse storage data)
- bookings (for farmer booking records)
- Linked bookings with storage records and implemented logic to automatically adjust availability.

#### **6. Validation and Logic**

- Implemented logic to:
- Prevent overbooking
- Ensure booking date is only on or a day before submission
- Restore storage availability when bookings are deleted

## **7. Testing and Debugging**

- Conducted extensive testing of all features:
- Adding/updating/deleting storage
- Booking flow with date and capacity validation
- UI responsiveness
- Database updates

## **8. Deployment & Submission**

- Organized the project structure as per academic guidelines
- Included:
- Presentation file
- Project video
- Source code in a separate folder (/src)
- README with instructions



## 6. Tools and Technologies Used

- **Programming Language:** Python 3.13
- **Web Framework:** Flask
- **Frontend:** HTML, CSS, Bootstrap
- **Database:** SQLite
- **IDE:** Visual Studio Code

# 7. System Features

## Home Section

- Displays current storage availability across all warehouses.
- All storage values are shown in **quintals**.

## Warehouse Section

- Add new storage with capacity (in quintals) and warehouse location.
- Update or delete existing storage entries.
- View farmer booking details.
- **Delete farmer booking** option, which automatically updates available storage.

# Farmer Section

- Book storage by selecting warehouse, specifying:
- Name
- Contact Number
- Address
- Quantity (in quintals)
- Date of Grain Submission
- Validation: Storage can only be booked **on the day of submission or one day before.**
- Automatically updates available storage upon booking

## 8. Project Workflow

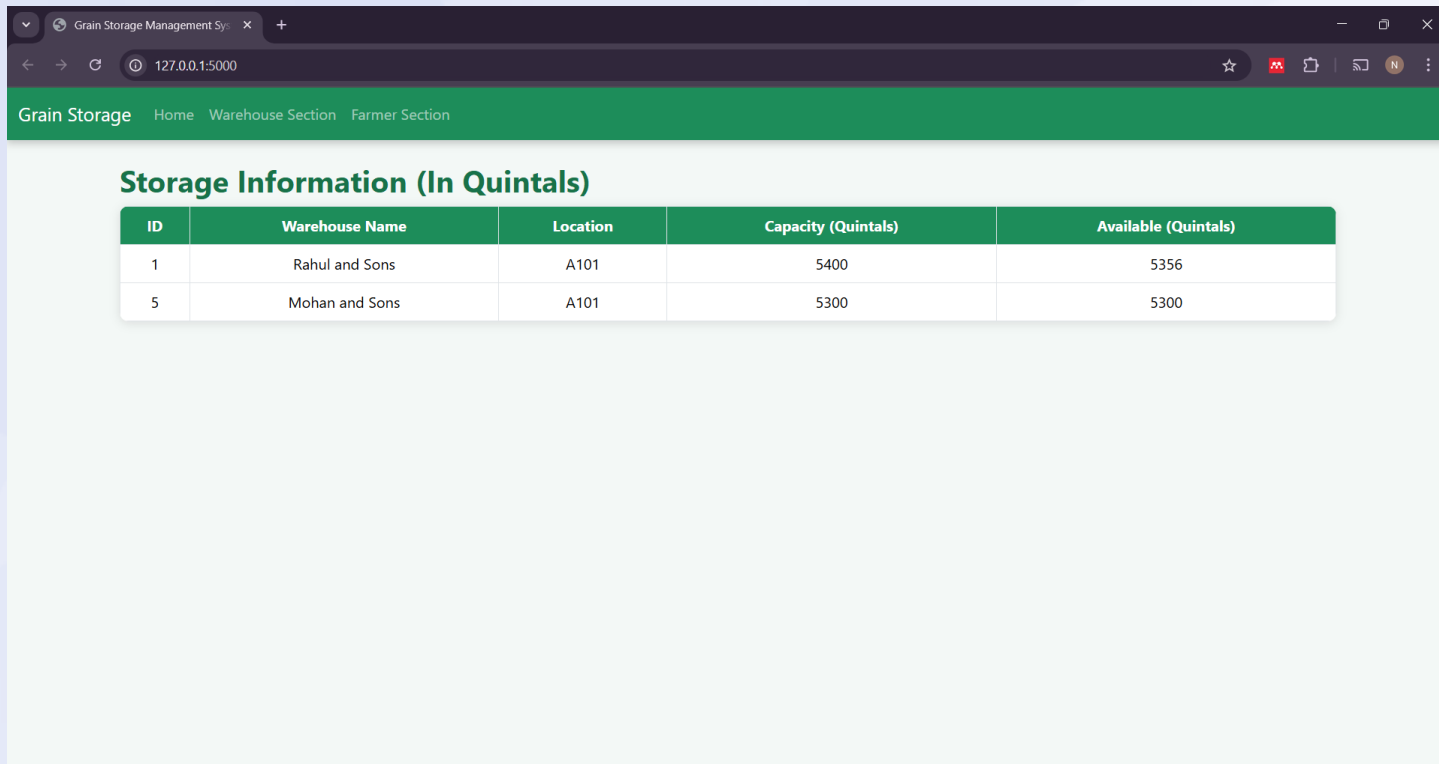
- **Homepage** fetches all storage records and shows available capacity.
- **Warehouse Owner** logs into their section to manage storage and view bookings.
- **Farmer** selects warehouse and books storage with necessary details.
- **System** checks validity of date, updates storage availability, and saves the booking.

## 9. Database Tables

- **storage:** Stores warehouse data (id, name, capacity, available, location)
- **bookings:** Stores farmer booking data (id, farmer\_name, contact, address, storage\_id, quantity, submission\_date)

# 10. Screenshots

## Home Menu



Storage Information (In Quintals)				
ID	Warehouse Name	Location	Capacity (Quintals)	Available (Quintals)
1	Rahul and Sons	A101	5400	5356
5	Mohan and Sons	A101	5300	5300

# Warehouse Section

Grain Storage Management System

127.0.0.1:5000/warehouse

Grain StorageHomeWarehouse SectionFarmer Section

### Manage Storage

Add Storage

### Update or Delete Storage

Update Storage

Delete Storage

### Farmer Bookings

Booking ID	Farmer Name	Contact Number	Address	Date of Submission	Storage Required (Quintals)	Warehouse	Action
3	Harkesh	9674854845	vill - ujina , nuh	2025-05-01	44	Rahul and Sons	Delete Booking

# Farmer Section

Grain Storage Management System

127.0.0.1:5000/farmer

Grain Storage Home Warehouse Section Farmer Section

## Book Storage

Book storage Only day before and on day of submission of grain.

Farmer Name

Contact Number

Address

dd-mm-yyyy

Storage Required (Quintals)

Select Warehouse

Book Storage



## **11. Conclusion**

This project successfully achieves the goal of a lightweight, functional grain storage system using Flask and SQLite. It simplifies the process for both warehouse owners and farmers and ensures real-time management of storage capacity.